Jacek Piechota

HTML5 canvas 2d

Wprowadzenie

Vollys
Follow the green dragon ...

Redaktor Naczelny: dr Jacek Piechota

Projekt okładki: Maciej Krzywicki

Adres wyłącznie do korespondencji:

Vollys sp. z o.o.
ul. Marszałkowska 34/50 m. 9
00-554 Waszawa
email: vollys@vollys.pl

ISBN 978-83-7681-011-9

Wstęp

Książka jest poświęcona językowi JavaScript Canvas z kontekstem 2d. Opisuje kompletnie podstawy interfejsu i aktualne możliwości języka. Jest książką przeznaczoną dla początkujących programistów.
Obok informacji dotyczącej JavaScript zawarto też sporą porcję informacji w rozdziałach takich jak ‘Przypomnienie z matematyki’, ‘Równania prostej’, ‘Wektory’, ‘Macierze’, ‘Transformacje’ i ‘Krzywe Beziera’, które pozwalają przypomnieć sobie lub poznać podstawy działania funkcji interfejsu.
W drugim tomie książki znajdą się bardziej zaawansowane zagadnienia dla średnio zaawansowanych i zaawansowanych programistów

Znacznik ‘canvas’

Czym jest canvas?

Canvas jest elementem (znacznikiem) HTML dodanym w HTML5.
Canvas zawiera bitmapę złożoną z pikseli (context). Przy użyciu JavaScript można na niej rysować i wykonywać operacje na całości obrazu lub poszczególnych pikselach.
Zawartość canvas jest widoczna jeśli przeglądarka obsługuje ten element.
Jeśli przeglądarka nie obsługuje tego elementu widziana jest treść umieszczona między znacznikiem początkowym, a końcowym.
Canvas podobnie jak inne elementy HTML posiada atrybuty i zdarzenia.

Przegląd elementu HTMLCanvasElement

Właściwości

Stan canvas				
Właściwość	**Opis**	**Wartość lub klasa obiektu**	**Wartość domyślna**	**Przykład**
width	Opisuje szerokość canvas	unsigned long	300px	
height	Opisuje wysokość canvas	unsigned long	150px	

Funkcje

Funkcje ogólne			
Typ zwracany	**Nazwa**	**Opis**	**Przykład**
DOMString	toDataURL([type], [args...])	Zwraca dane w postaci łańcucha tekstowego, który powstał w wyniku zakodowania informacji o obrazie przy użyciu kodowania Base64. DOMString type - oznacza typ MIME "image.png" (domyślne) albo "image/jpeg". Jeśli type="image/jpeg" to "args..." może być liczbą oznaczającą 'dokładność obrazka' od 0.0 - 1.0. Obrazek zapamiętywany jest w rozdzielczości 96dpi.	
void	toBlob(callback, [type],[args...])	callback - funkcja wywołania zwrotnego przyjmująca obiekt blob jako jedyny argument. type - opcjonalne - DOMString podający format obrazka w typie MIME: "image/png" - domyślne albo "image/jpeg". Jeżeli drugi argument jest "image/jpeg" to możemy podać args, np. liczbę od 0.0 do	

		1.0 oznaczającą 'dokładność' obrazka. Obrazek podawany jest w rozdzielczości 96dpi.	
RenderingContext	getContext(id)	Zwraca kontekst, na którym można wykonywać wszystkie operacje DOMString id - dopuszczalne wartości '2d' albo 'webGL'.	

Umieszczanie canvas na stronie

Canvas umieszczamy na stronie podobnie jak inne znaczniki HTML. Pomiędzy znacznikiem początkowym i końcowym umieszczamy treść, która będzie widoczna wyłącznie wtedy, gdy przeglądarka klienta nie obsługuje elementu canvas.

Listing

```
<canvas id="canvas" width="300" height="200"
style="border: 1px solid #238E3E;">Zawartość możesz zobaczyć w
przeglądarce obsługującej element <canvas>
z kontekstem "2d"</canvas>
```

Pozycja skryptu JavaScript względem canvas

Skrypt umieszcza się na stronie na dwa sposoby:

Sposób 1

Skrypt umieszczony jest na stronie w elemencie body (bezpośrednio) po elemencie, którego dotyczy.

Listing

```
<body>
<canvas id="canvas" width="300" height="300" style="border: 1px solid
#238E3E;">Zawartość możesz zobaczyć w przeglądarce obsługującej element <canvas>
z kontekstem "2d"</canvas>
<script type="text/javascript">
</script>
</body>
```

Sposób 2

Skrypt umieszcza się w elemencie head a canvas w elemencie body.

Listing

```
<head>
<meta charset="utf-8">
```

```
<script type="text/javascript">
      window.onload = function(){

      };
      </script>
</head>
```

Listing

```
<body>
<canvas id="canvas" width="300" height="300"
             style="border: 1px solid #238E3E;">Zawartość możesz zobaczyć w
             przeglądarce obsługującej element <canvas>
             z kontekstem "2d"</canvas>
</body>
```

Skrypt w oddzielnym pliku

Oczywiście skrypt możemy też umieścić w zewnętrznym pliku, a w elemencie head podać szczegóły dotyczące lokalizacji skryptu.

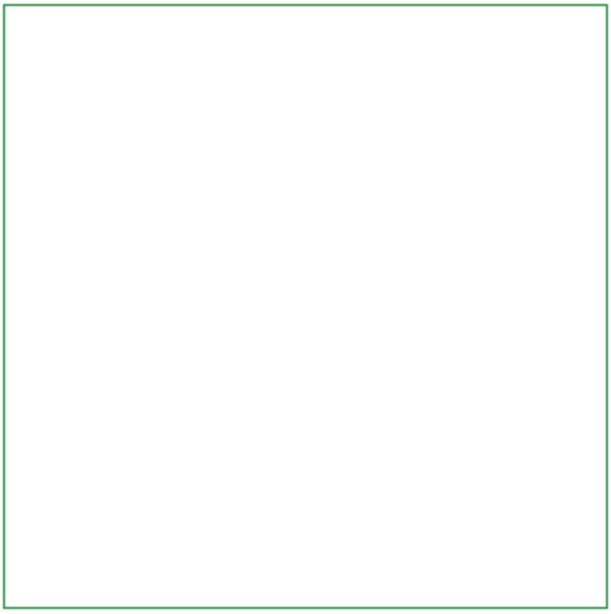
Listing

```
<head>
<script src="scripts/canvas.js"></script>
</head>
```

Pobieranie znacznika canvas w kodzie

Znacznik pobieramy używając w skrypcie zapisu `document.getElementById("canvas")` , gdzie 'canvas' jest nazwą elementu canvas.

Element canvas:



Listing

```
<canvas id="canvas" width="300" height="300"
      style="border: 1px solid #238E3E;">Zawartość możesz zobaczyć w
      przeglądarce obsługującej element <canvas>
      z kontekstem "2d"</canvas>
```

Pobranie w skrypcie:

Listing

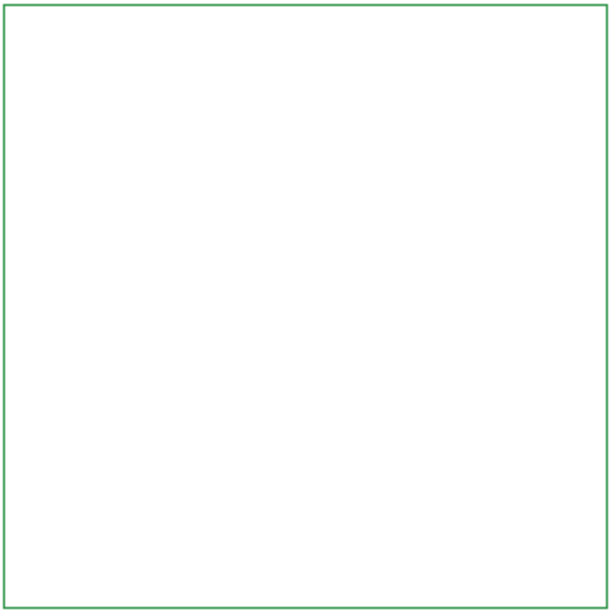
```
var cv = document.getElementById("canvas");
```

Kontekst ‘2d’

Pobieranie kontekstu "2d"

Istnieją dwa konteksty:

- 2d
- webGL, którego tutaj nie omawiamy



Aby pobrać kontekst w skrypcie umieszczmy zapis:

Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
```

Wszystkie operacje rysowania i inne wykonywane są na canvas, a kontekst dostarcza narzędzi programistycznych JavaScript do rysowania.

Interface CanvasRenderingContext2D - przegląd

```
CanvasRenderingContext2D implements CanvasDrawingStyles
CanvasRenderingContext2D implements CanvasPathMethods
```

Zwrotna referencja do canvas:

```
readonly attribute HTMLCanvasElement canvas;
```

Właściwości

Stan kontekstu

Właściwość	Opis	Wartość lub klasa obiektu	Wartość domyślna	Przykład
globalAlpha	Przezroczystość obowiązująca dla wszystkich konturów i wypełnień	dowolna liczba rzeczywista	1.0	
globalCompositeOperation	Określa obowiązującą zasadę składania kolorów dla wszystkich figur i obrazów wyświetlanych na canvas.	DOMString	"source-over"	

Style i kolory

Właściwość	Opis	Wartość lub klasa obiektu	Wartość domyślna	Przykład
strokeStyle	Określa kolor lub styl obowiązujący dla konturów. Kolor lub styl jest zastosowany po wywołaniu funkcji stroke()	DOMString CanvasGradient CanvasPattern	"black"	
fillStyle	Określa kolor lub styl obowiązujący dla wypełnień. Kolor lub styl jest zastosowany po wywołaniu funkcji fill()	DOMString CanvasGradient CanvasPattern	"black"	

Cienie

Właściwość	Opis	Wartość lub klasa obiektu	Wartość domyślna	Przykład
shadowOffsetX	Określa przesunięcie cienia względem figury w osi X	dowolna liczba rzeczywista	0	
shadowOffsetY	Określa przesunięcie cienia względem figury w osi Y	dowolna liczba rzeczywista	0	
shadowBlur	Określa wielkość (szerokość) cienia	dowolna liczba rzeczywista	0	
shadowColor	Określa kolor cienia	DOMString (kolor CSS)	"black"	

Zobacz właściwości z implementowanych interfejsów: CanvasDrawingStyles oraz CanvasPathMethods.

Funkcje

Stan kontekstu

Typ	Nazwa	Opis	Przykład

zwracany		
void	save()	Przenosi aktualny stan kontekstu na stos
void	restore()	Pobiera ostatni zapisany stan kontekstu ze stosu. Pobrany stan staje się stanem canvas

Style i kolory

Typ zwracany	Nazwa	Parametry	Opis	Przykład
CanvasGradient	createLinearGradient(x0,y0,x1,y1)	x,y - są dowolnymi liczbami rzeczywistymi	Tworzy gradient liniowy od punktu (x0,y0) do punktu (x1,y1).	
CanvasGradient	createRadialGradient(x0,y0,r0,x1,y1,r1)	x,y,r - są dowolnymi liczbami rzeczywistymi	Tworzy gradient radialny rozpięty między między dwoma kołami o podanych położeniach środków i podanych promieniach.	
Canvas Pattern	createPattern(element, powtórzenie)	element: HTMLImageElement image | HTMLCanvasElement canvas | HTMLVideoElement video powtórzenie: DOMString	Tworzy deseń (teksturę). Podstawą deseniu może być obrazek, canvas lub wideo. Dopuszczalne wartości da powtórzenia to: 'repeat', 'no-repeat',' repeat-x' i	

'repeat-y'.

Transformacje

Typ zwracany	Nazwa	Parametry	Opis	Przykład
void	scale(x,y)	x - dowolna liczba rzeczywista - określa zmianę wymiarów wzdłuż osi X y - dowolna liczba rzeczywista - określa zmianę wymiarów wzdłuż osi Y	Dodaje skalowanie do aktualnej macierzy transformacji	
void	rotate(angle)	angle - dowolna liczba rzeczywista oznaczająca kąt w radianach	Dodaje obrót w kierunku ruchu wskazówek zegara, o podany kąt do aktualnej macierzy transformacji	
void	translate(x,y)	x - dowolna liczba rzeczywista - określa przesunięcie wzdłuż osi X y - dowolna liczba rzeczywista - określa przesunięcie obiektu wzdłuż osi Y	Dodaje przesunięcie do aktualnej macierzy transformacji	
void	transform(a,b,c,d,e,f)	a,b,c,d,e,f - dowolne liczby rzeczywiste	Mnoży aktualną macierz transformacji przez podaną macierz	
void	setTransform(a,b,c,d,e,f)	a,b,c,d,e,f - dowolne liczby rzeczywiste	Zastępuje aktualną macierz transformacji przez podaną transformację	

Prostokąty

Typ zwracany	Nazwa	Parametry	Opis	Przykład
void	clearRect(x,y,w,h)	x,y,w,h - dowolne liczby rzeczywiste określające kolejno położenie lewego górnego rogu prostokąta	Czyści piksele wybranego prostokąta	

		oraz jego szerokość i wysokość		
void	fillRect(x,y,w,h)	x,y,w,h - dowolne liczby rzeczywiste określające kolejno położenie lewego górnego rogu prostokąta oraz jego szerokość i wysokość	Rysuje wypełnienie prostokąta o podanych parametrach (nie wymaga polecenia fill())	
void	strokeRect(x,y,w,h)	x,y,w,h - dowolne liczby rzeczywiste określające kolejno położenie lewego górnego rogu prostokąta oraz jego szerokość i wysokość	Rysuje kontur prostokąta o podanych parametrach (nie wymaga polecenia stroke())	

Scieżki

Typ zwracany	Nazwa	Parametry	Opis	Przykład
void	beginPath()	-	Rozpoczyna nową ścieżkę (ustawia liczbę podścieżek na 0). Jakiekolwiek ścieżki ustanowione ale nie narysowane nie zostaną wyświetlone.	
void	fill()	-	Wypełnia wszystkie otwarte podścieżki i zakańcza je.	
void	stroke()	-	Odrysowuje wszystkie kontury aktualnej ścieżki.	
void	drawFocusIfNeeded(element)	Element element	?	
void	clip()	-	Tworzy nowy region przycinania przez utworzenie iloczynu aktualnego obszaru przycinania i powierzcni zdefiniowanej przez aktualną ścieżkę.	
boolean	isPointInPath(x,y)	x,y- dowolne liczby rzeczywiste	Określa czy punkt o podanych współrzędnych (x,y) znajduje się w obrębie ścieżki.	

Teksty

Typ zwracany	Nazwa	Parametry	Opis	Przykład
void	fillText(text,x,y, maxWidth)	DOMString text x,y,maxWidth - dowolne liczby rzeczywiste maxWidth - jest elementem opcjonalnym	Wypełnia podany tekst i wyświetla go w miejscu o podanych współrzędnych.	
void	strokeTekst(text,x,y, maxWidth)	DOMString text x,y, maxWidth - dowolne liczby rzeczywiste maxWidth jest elementem opcjonalnym	Konturuje podany tekst i wyświetla go w miejscu o podanych współrzędnych.	
TextMetrics	measureText(text)	DOMString text	Wykonuje pomiar podanego tekstu.	

Rysowanie obrazów

Typ zwracany	Nazwa	Parametry	Opis	Przykład
void	drawImage(image, x,y)	x,y - dowolne liczby rzeczywiste element: HTMLImageElement | HTMLCanvasElement | HTMLVideoElement	Odrysowuje pobrany element w punkcie o współrzędnych (x,y).	
void	drawImage(image, x,y,w,h)	x,y,w,h - dowolne liczby rzeczywiste element: HTMLImageElement | HTMLCanvasElement | HTMLVideoElement	Odrysowuje pobrany element w punkcie o współrzędnych (x,y). Narysowany element ma podaną szerokość i wysokość.	
void	drawImage(image, sx,sy,sw,sh, x,y,w,h)	sx,sy,sw,sh,x,y,w,h - dowolne liczby rzeczywiste element: HTMLImageElement | HTMLCanvasElement | HTMLVideoElement	Odrysowyje pobrany z obrazka prostokąt o współrzędnych (sx,sy,sw,s). Obrazek jest rysowany w prostokącie o wymiarach (x,y,w,h).	

Obszary kolizji

Typ zwracany	Nazwa	Parametry	Opis	Przykład
void	addHitRegion(options)	HitRegionOptions options	Dodaje obszar kolizji o podanych opcjach options	
void	removeHitRegion(id)	DOMString id	Usuwa obszar kolizji o podanym id	
void	clearHitRegions();	-	Usuwa obszary kolizji	

Operacje na pikselach obrazu

Typ zwracany	Nazwa	Parametry	Opis	Przykład
ImageData	createImageData(sw, sh)	sw,sh - dowolne liczby rzeczywiste	Tworzy nowy (pusty) obiekt ImageData o podanej szerokości i wysokości.	
ImageData	createImageData(data)	ImageData data	Tworzy nowy (pusty) obiekt ImageData o szerokości i wysokości takiej jak w obiekcie data.	
ImageData	getImageData(x,y,w,h)	x,y,w,h - liczba rzeczywista	Zwraca obiekt ImageData zawierający dane zawarte w prostokącie o wymiarach i położeniu (x,y,w,h).	
void	putImageData(data,x,y[,dx,dy,dw,dh])	ImageData data x,y,dx,dy,dw,dh - liczby rzeczywiste dx,dy,dw,dh - są opcjonalne	Wstawia podany obiekt data w punkcie o współrzędnych (x,y). W przypadku podania również parametrów opcjonalnych w punkcie (x,y) wstawia	

				wycinek danych zawarty w prostokącie (dx,dy,dw,dh) w danych 'data'.

Zobacz funkcje z implementowanych interfejsów: CanvasDrawingStyles oraz CanvasPathMethods.

interface CanvasDrawingStyle

Właściwości

Linie: złączenia i zakończenia

Właściwość	Opis	Wartość lub klasa obiektu	Wartość domyślna	Przykład
lineWidth	Określa grubość linii	dowolna liczba rzeczywista	1	
lineCap	Określa kształt zakończenia linii	DOMString. Dozwolone wartości to: "butt", "round" i "square"	"butt"	
lineJoin	Określa sposób wyświetlania złączeń linii (narożników)	DOMString. DOzwolone wartości to: "round", "bevel" i "miter".	"miter"	
miterLimit	Określa maksymalną długość, o którą linia może być przedłużona jeśli 'lineJoin' jest usatwione na "miter'. Jeśli długość niezbędna do połączenia jest większa niż podana wartość - połączenie nie zostanie wykonane.	dowolna liczba rzeczywista	10	

Linie przerywane

Właściwość	Opis	Wartość lub klasa obiektu	Wartość domyślna	Przykład
lineDashOffset	Określa odstęp między segmentami linii przerywanej	dowolna liczba rzeczywista		

Tekst

Właściwość	Opis	Wartość lub klasa obiektu	Wartość domyślna	Przykład

font	Ustawia atrybuty czcionki	DOMString.. Składnia zgodna z zasadami CSS.	"10px sans-serif"
textAlign	Ustawia wyrównywanie tekstu	DOMString. Dozwolone wartości: "start", "end", "left", "right", "center".	"start"
textBaseLine	Ustawia linię podstawową dla tekstu	DOMString. Dozwolone wartości to: "top", "hanging", "middle", "alphabetic", "ideographic", "bottom".	"alphabetic"

Funkcje

Linie przerywane

Typ zwracany	Nazwa	Parametry	Opis	Przykład
void	setLineDash(segments)	segments: tablica dowolnych liczb rzeczywistych. Domyślnie lista jest pusta	Określa liczbę i wymiary odcinków tworzących linię przerywaną. Po wyczerpaniu listy segmenty powtarzają się od początku listy.	
lista liczb rzeczywistych	getLineDash()	zwraca listę wymiarów ustawionych dla linii przerywanej		

interface CanvasPathMethods

Funkcje

Scieżki

Typ zwracany	Nazwa	Parametry	Opis	Przykład
void	closePath()	-	Zamyka ostatnią podścieżkę i tworzy nową, której pierwszym punktem jest ostatni punkt ścieżki zamkniętej	
void	moveTo(x,y)	x,y - dowolna liczba rzeczywista	Tworzy nową podścieżkę, z dodanym pierwszym punktem (x,y)	
void	lineTo(x,y)	x,y - dowolna liczba rzeczywista	Rysuje linię od ostatniego punktu	

			podścieżki do punktu (x,y).
void	quadraticCurveTo(cpx,cpy,x,y)	cpx,cpy,x,y- dowolne liczby rzeczywiste	Rysuje kwadratową krzywą Béziera od ostatniego punktu podścieżki do punktu (x,y) z punktem kontrolnym (cpx,cpy).
void	bezierCurveTo(cpx1,cpy1,cpx2,cpy2,x,y)	cpx1,cpy1,cpx2,cpy2,x,y - dowolne liczby rzeczywiste	Rysuje sześcienną krzywą Béziera od ostatniego punktu podścieżki do punktu (x,y) z uwzględnieniem punktów kontrolnych (cpx1,cpy1) oraz (cpx2, cpy2).
void	arcTo(x1,y1,x2,y2,r)	x1,y1,x2,y2,r - dowolne liczby rzeczywiste	Rysuje łuk o promieniu r od punktu (x1,y1). Punkt (x2,y2) nie jest rysowany. Służy do wyznaczenia trzeciego punktu (x3,y3), który będzie punktem końcowym łuku. Luk jest rozpięty między punktem (x1,y1) a (x3,y3).
void	arc(x,y,r,startAngle,endAngle,[anticlckwise])	x,y,r,startAngle,endAngle - dowolne liczby rzeczywiste boolean anticlockwise - opcjonalny, domyślnie 'false'.	rysuje łuk koła o środku (x,y) i promieniu r, od kąta startAngle, do kąta endAngle w kierunku w kierunku ruchu wskazówek zegara (anticlockwise='false') lub w

			kierunku przeciwnym do ruchu wskazówek zegara (anticlockwise='true').
void	rect(x,y,w,h)	x,y,w,h - dowolne liczby rzeczywiste	Tworzy nową podścieżkę i rysuje prostokąt w punkcie (x,y), szerokości w i wysokości h.

Interface CanvasGradient

Funkcje

Linie przerywane

Typ zwracany	Nazwa	Parametry	Opis	Przykład
void	addColorStop(offset, color)	offset - liczba rzeczywista DOMString color	Dodaje kolor color do gradientu w położeniu względnym offset.	

Interface CanvasPattern

Interfejs pusty

Interface TextMetrics

Właściwości

Metryka tekstu

Właściwość	Opis	Wartość lub klasa obiektu	Wartość domyślna	Przykład
width	Podaje szerokość tekstu	double - tylko do odczytu	-	

Interface ImageData

ImageData

Właściwość	Opis	Wartość lub klasa obiektu	Wartość domyślna	Przykład
width	szerokość danych	unsigned long - tylko do odczytu	-	
height	wysokość danych	unsigned long - tylko do odczytu	-	

data	Tabela zawierająca dane	Uint8ClampedArray	-

Rozszerzenia

Właściwości

Brak.

Funkcje

Linie przerywane

Typ zwracany	Nazwa	Parametry	Opis	Przykład
void	ellipse(x, y, radiusX, radiusY, rotation, startAngle, endAngle, anticlockwise);	x, y, radiusX, radiusY, rotation, startAngle,endAngle - dowolne liczby rzeczywiste. Trzy ostatnie kąty podane w radianach. boolean anticlockwise	Rysuje elipsę, o środku (x,y), promieniu radiusX w osi X, promieniu radiusY w osiY, o kącie rotacji rotation w radianach, od kąta początkowy startAngle, do kąta końcowego endAngle, w kierunku przeciwnym do ruchu zegara(anticlockwise=true) albo zgodnie z ruchem wskazówek zegara (anticlockwise=false)	

System współrzędnych

Omówienie znajduje się w Dodatku 1.

Linie proste

Podstawy teoretyczne dotyczącej prostej i operacji na prostych znajdziesz w Dodatku 2.

Rysowanie linii

Rysowanie odbywa się w kontekście na ścieżkach. Istnieje jedna domyślna ścieżka na której możemy rysować.

Nową ścieżkę dodajemy instrukcją `ctx.beginPath()`. Rozpoczęcie nowej ścieżki zakańcza poprzenią ścieżkę.

Aby narysować linię 'głowicę rysującą' przenosimy do punktu początkowego linii używając polecenia `moveTo()`, a następnie kreślimy linię od tego punktu używając polecenia `lineTo()`.

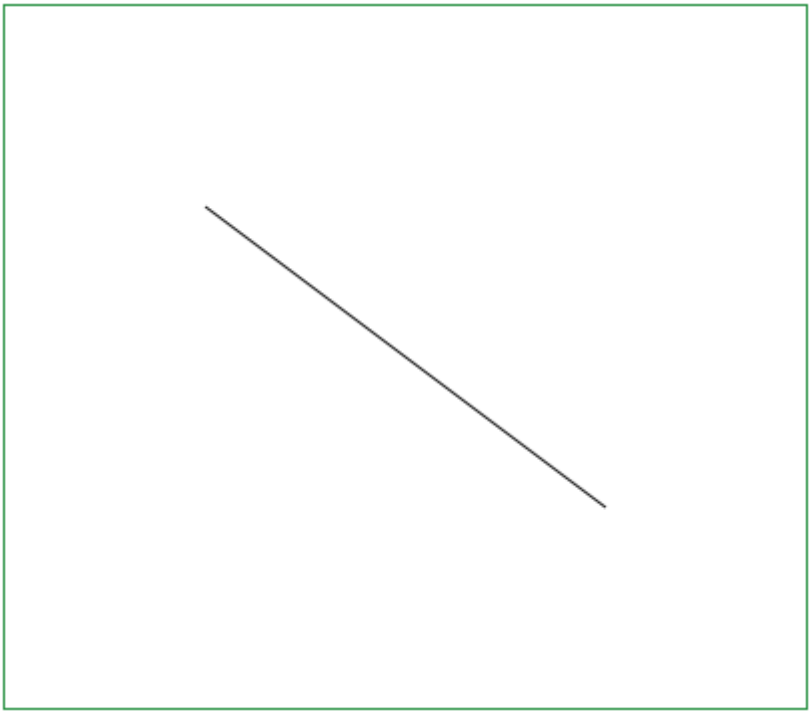
Linia jest rysowana w pamięci. Nie jest jeszcze widoczna.

Aby uwidocznić linię wydajemy polecenie `stroke()` które służy do rysowania konturów.

Ponieważ nie ustawiliśmy dla kontekstu żadnych właściwości użyte są domyślne właściwości:

- kolor - 'black' (czarny)
- szerokość (grubość) linii - '1px' (1 piksel)

Sposób ich użycia podamy dalej w tekście.



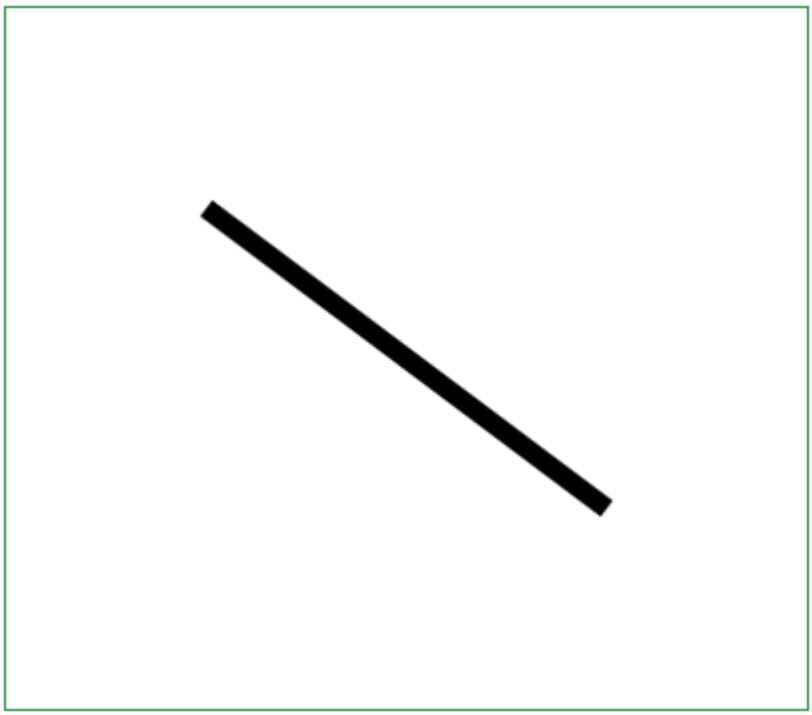
Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
ctx.beginPath();
ctx.moveTo(100, 100);
ctx.lineTo(300, 250);
ctx.stroke();
```

Szerokość linii

Szerokość linii jest ustawiana dla kontekstu. Dopóki nie zostanie zmieniona obowiązuje dla wszystkich ścieżek.

Szerokość ustawiamy ustawiając właściwość `lineWidth` podając liczbę pikseli.



Listing

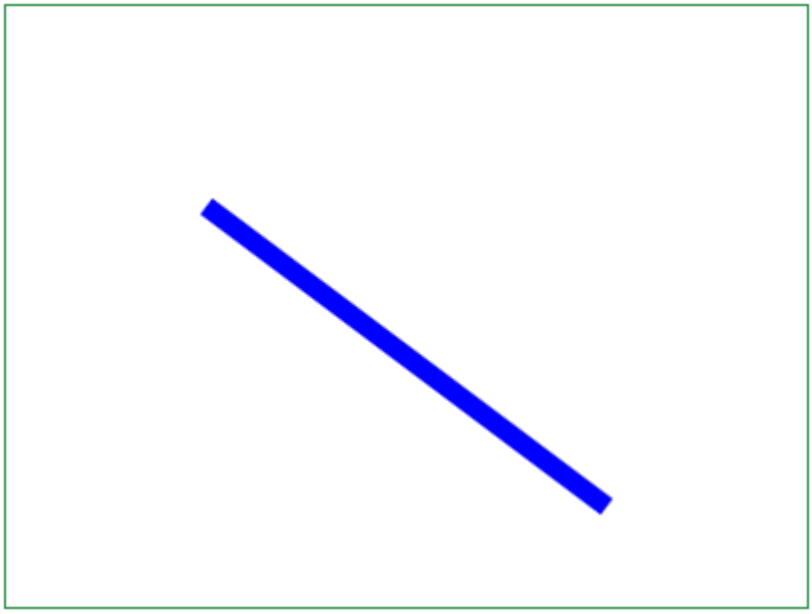
```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
ctx.beginPath();
ctx.moveTo(100, 100);
ctx.lineTo(300, 250);
ctx.lineWidth = 10;
ctx.stroke();
```

Kolor linii

Ustawiając właściwość `strokeStyle` ustawiamy kolor konturu dla kontekstu (nie tylko dla linii).

Każdy kontur będzie rysowany przy użyciu tego koloru aż do jego zmiany tym samym poleceniem.

Więcej o kolorach i sposobach ich podawania powiemy dalej w tekście.

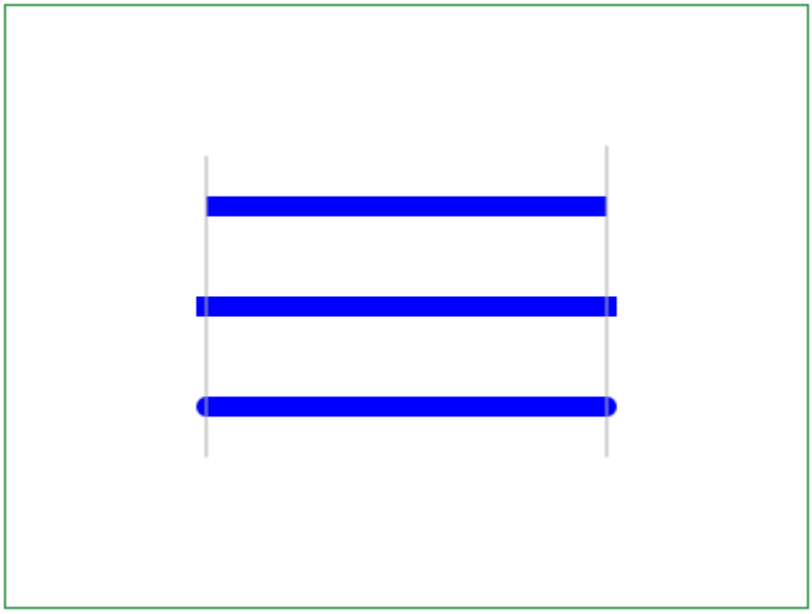


Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
ctx.beginPath();
ctx.moveTo(100, 100);
ctx.lineTo(300, 250);
ctx.lineWidth = 10;
ctx.strokeStyle = "#0000ff";
ctx.stroke()
```

Zakończenia linii

Właściwość `lineCap` służy do określenia kształtu zakończenia linii. Właściwośc ta jest ustawiana dla kontekstu i obowiązuje aż do zmiany.Linie na canvas mogą mieć trzy zakończenia:



Pierwsze zakończenie określane jest jako 'butt'.

Drugie zakończenie jest określane jako 'square'.

Trzecie zakończenie jest określane jako 'round'.

Sposób zakończenia linii jest szczególnie istotny przy łączeniu linii.

Listing

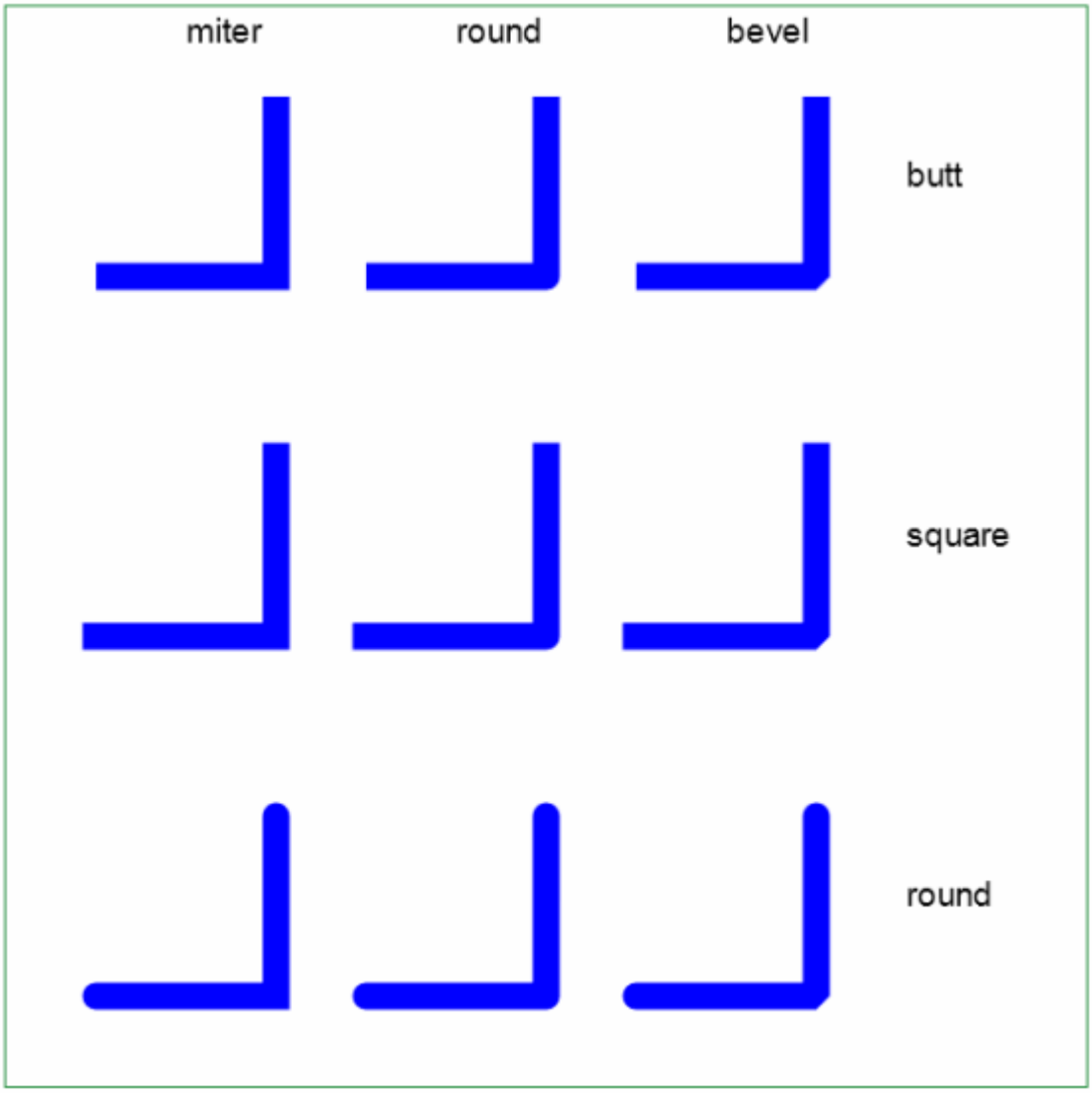
```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
//-
 ctx.beginPath();
 ctx.moveTo(100, 100);
 ctx.lineTo(300, 100);
 ctx.lineWidth = 10;
 ctx.strokeStyle = "#0000ff";
 ctx.lineCap = "butt";
 ctx.stroke();
//-
 ctx.beginPath();
 ctx.moveTo(100, 150);
 ctx.lineTo(300, 150);
 ctx.lineWidth = 10;
 ctx.strokeStyle = "#0000ff";
 ctx.lineCap = "square";
 ctx.stroke();
//-
 ctx.beginPath();
 ctx.moveTo(100, 200);
 ctx.lineTo(300, 200);
 ctx.lineWidth = 10;
 ctx.strokeStyle = "#0000ff";
 ctx.lineCap = "round";
 ctx.stroke();
```

Złączenia linii

Właściwość `lineJoin` jest ustawiana dla kontekstu i obowiązuje aż do zmiany.

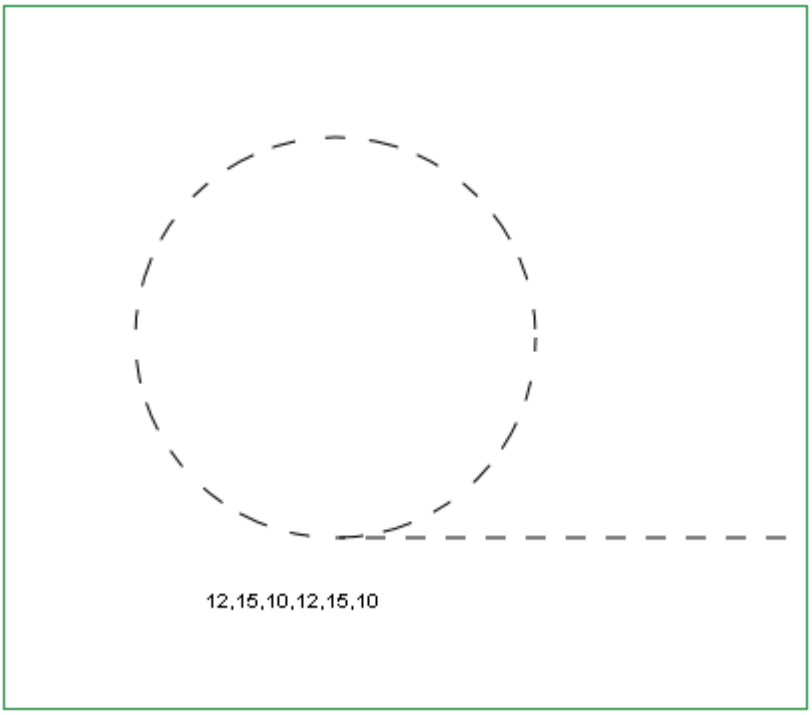
Istnieją trzy sposoby łączenia linii okreśone jako 'miter', 'round' i 'bevel'.

Istnieje 9 możliwości wzajemnego użycia właściwości lineCap i lineJoin. W skrypcie pokazujemy tylko jedną z nich.



Linia przerywana

W momencie pisania (grudzień 2014) nie działa w przeglądarce Safari.



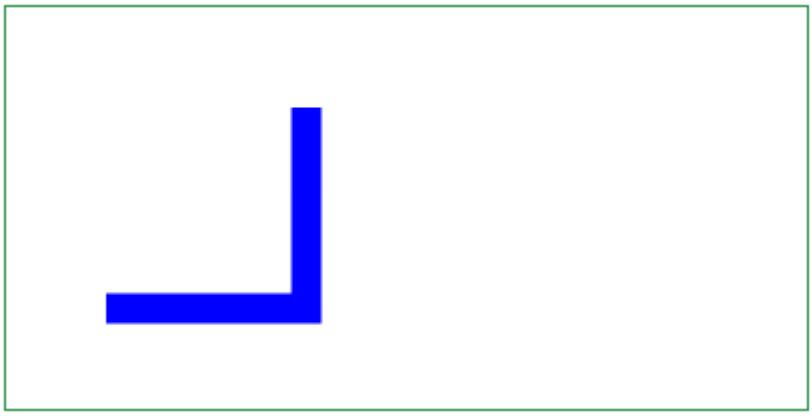
Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
ctx.setLineDash([12,15,10]);
ctx.lineDashOffset = 5;
ctx.arc(165,165,100,0,2*Math.PI, false);
ctx.stroke();
var a = ctx.getLineDash();
ctx.fillStyle="black";
ctx.fillText(a, 100, 300);
ctx.beginPath();
ctx.setLineDash([10,10,10]);
ctx.moveTo(165,265);
ctx.lineTo(400, 265);
ctx.stroke();
```

Ścieżki

Ścieżka domyślna

Dla kontekastu istnieje jedna domyślna ścieżka, na której można rysować

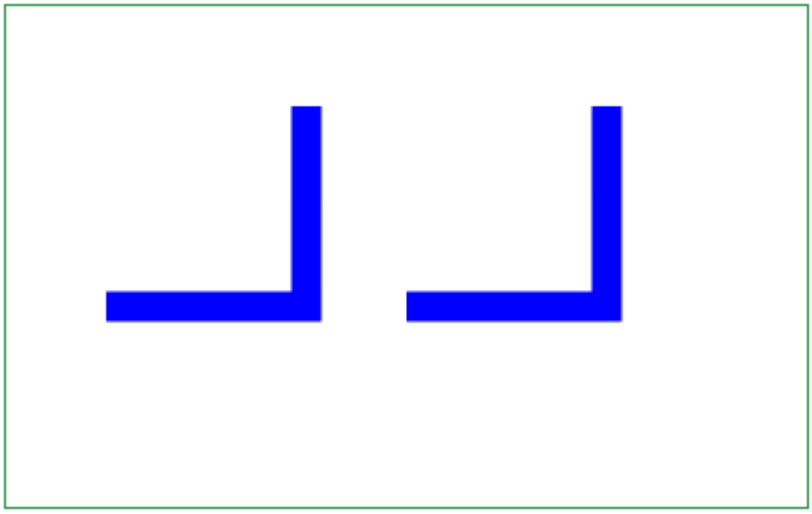


Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
ctx.moveTo(50, 150);
ctx.strokeStyle = "#0000ff";
ctx.lineTo(150, 150);
ctx.lineTo(150, 50);
ctx.lineWidth = 15;
ctx.stroke();
```

Otwieranie nowej ścieżki

Nową ścieżkę otwieramy po wydaniu polecenia `beginPath()`. Oznacza to automatyczne zamknięcie każdej poprzedniej ścieżki, a więc i ścieżki domyślnej. Nie zmienia to ustawionych właściwości.



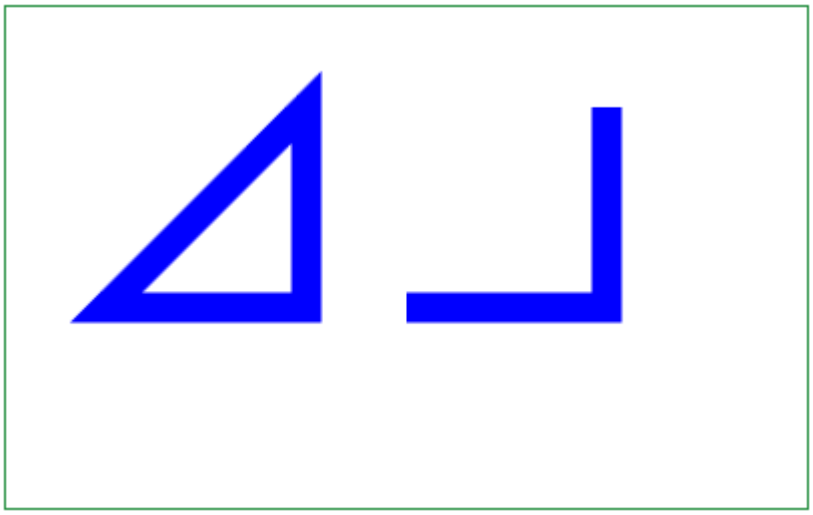
Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
//-
ctx.moveTo(50, 150);
ctx.strokeStyle = "#0000ff";
ctx.lineTo(150, 150);
ctx.lineTo(150, 50);
ctx.lineWidth = 15;
ctx.stroke();
```

```
//-
ctx.beginPath();
ctx.moveTo(200, 150);
ctx.strokeStyle = "#0000ff";
ctx.lineTo(300, 150);
ctx.lineTo(300, 50);
ctx.lineWidth = 15;
ctx.stroke();
```

Zamykanie ścieżki

Polecenie `closePath()` zamyka bieżącą operację rysowania. Po wydaniu tego polecenia automatycznie dorysowywana jest linia pomiędzy ostatnim punktem bieżącej ścieżki, a jej punktem początkowym.



Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
//-
ctx.moveTo(50, 150);
ctx.strokeStyle = "#0000ff";
ctx.lineTo(150, 150);
ctx.lineTo(150, 50);
ctx.lineWidth = 15;
ctx.closePath();
ctx.stroke();
//-
ctx.beginPath();
ctx.moveTo(200, 150);
ctx.strokeStyle = "#0000ff";
ctx.lineTo(300, 150);
ctx.lineTo(300, 50);
ctx.lineWidth = 15;
ctx.stroke();
```

Jak widzimy nie ma potrzeby dorysowywania trzeciej linii. Jest ona rysowana automatycznie.

Gotowe kształty

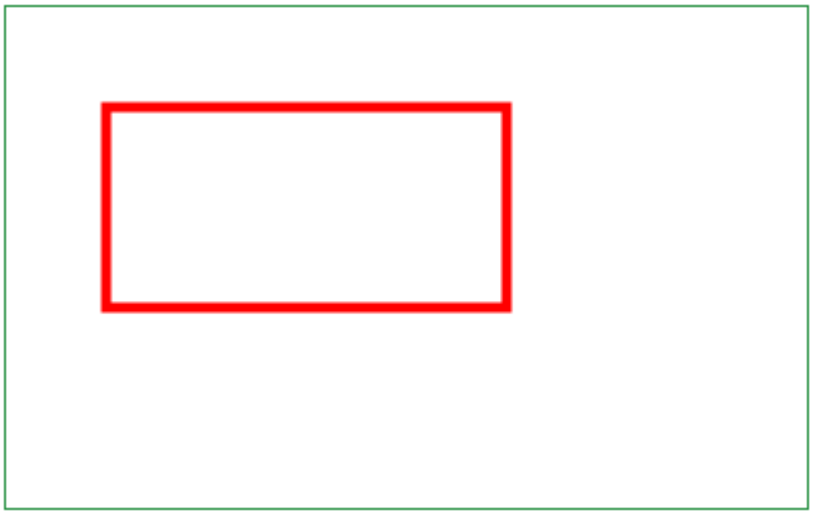
Prostokąt

Kontur

Prostokąt rysujemy w pamięci poleceniem `rect(x,y,w,h)`.

Podajemy współrzędne lewego górnego rogu (x,y) oraz szerokość i wysokość prostokąta (w,h). Gdy użyjemy polecenia `stroke()` uzyskamy kontur prostokąta.

Polecenie `stroke()` jest poleceniem uniwersalnym, czyli możemy narysować w pamięci wiele figur i wykreślić ich kontury jednym poleceniem `stroke()`.



Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
ctx.beginPath();
ctx.rect(50, 50, 200, 100);
ctx.lineWidth = 5;
ctx.strokeStyle = "#ff0000";
ctx.stroke();
```

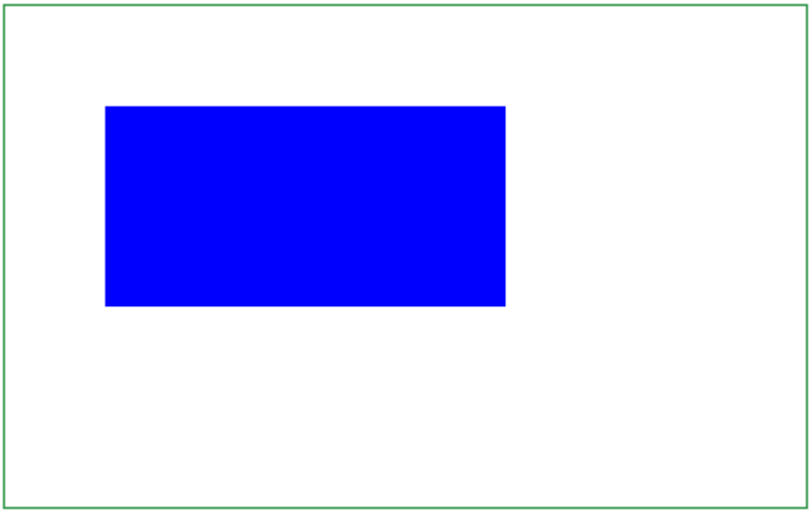
Wypełnienie

Jeżeli po narysowaniu w pamięci prostokąta, ustawimy właściwość `fillStyle` i użyjemy polecenia `fill()` narysujemy prostokąt bez obrzeża, zawierający tylko wypełnienie.

Właściwość `fillStyle` określa rodzaj wypełnienia, którym może być:

- kolor
- deseń
- gradient

Powiemy o tym w dalszym ciągu tekstu. Teraz użyliśmy koloru.

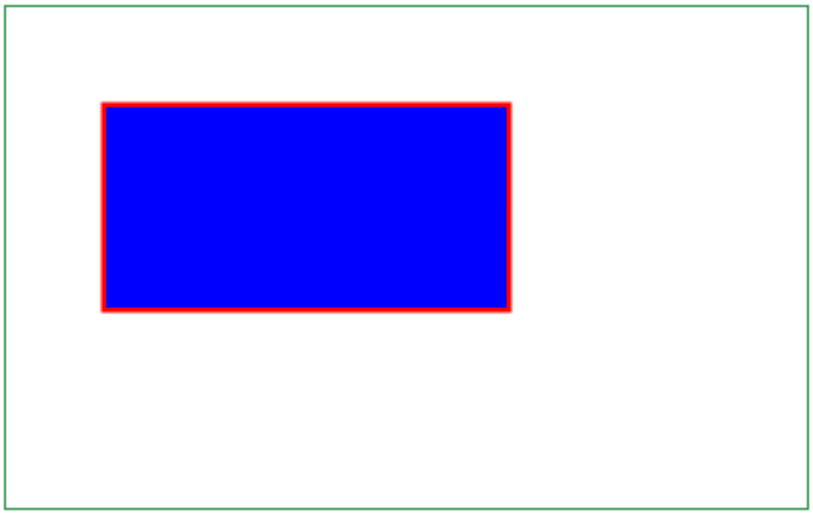


Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
ctx.beginPath();
ctx.rect(50, 50, 200, 100);
ctx.fillStyle = "#0000ff";
ctx.fill();
```

Kontur i wypełnienie (1)

Jeżeli narysujemy prostokąt i użyjemy polecenia `stroke()` i polecenia `fill()` wówczas zobaczymy figurę posiadającą zarówno kontur jak i wypełnienie.

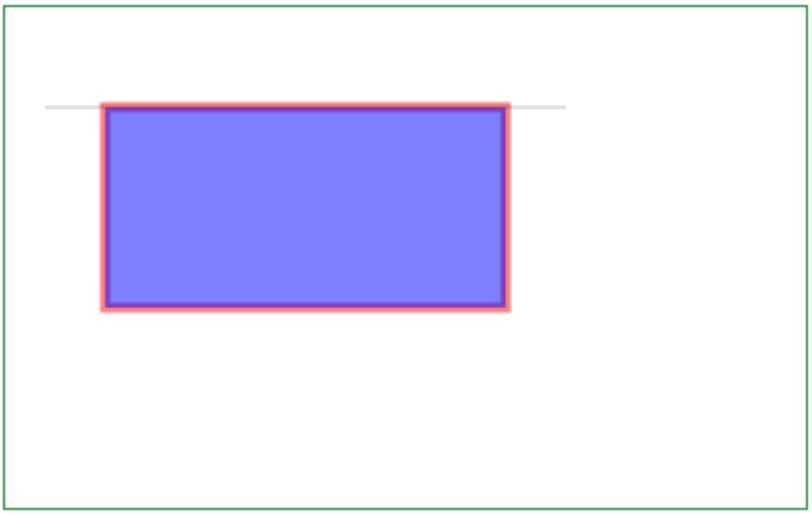


Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
ctx.beginPath();
ctx.rect(50, 50, 200, 100);
ctx.lineWidth = 5;
ctx.strokeStyle = "#ff0000";
ctx.stroke();
ctx.fillStyle = "#0000ff";
ctx.fill();
```

Kontur i wypełnienie (2)

Jeżeli wyświetlamy kontur i wypełnienie to wypełnienie jest dokładnie wielkości podanego prostokąta. Kontur połową grubości mieści się w obrębie tego prostokąta, a druga połowa wystaje ponad wymiary tego prostokąta. Dlatego najlepiej jest podawać najpierw polecenie `stroke()` , potem polecenie `fill()` .



Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
ctx.beginPath();
ctx.rect(50, 50, 200, 100);
ctx.lineWidth = 5;
ctx.globalAlpha = 0.5;
ctx.strokeStyle = "#ff0000";
ctx.stroke();
ctx.fillStyle = "#0000ff";
ctx.fill();
```

Łuk koła i wycinek koła (1)

Łuk rysujemy używając polecenia `arc(`x_0`, `y_0`, r, `α_s`, `α_e`, counterClockwise)`,

gdzie:

x_0 i y_0 oznaczają współrzędne środka koła, którego łuk lub wycinek rysujemy

r - to promień koła, którego łuk lub wycinek rysujemy.

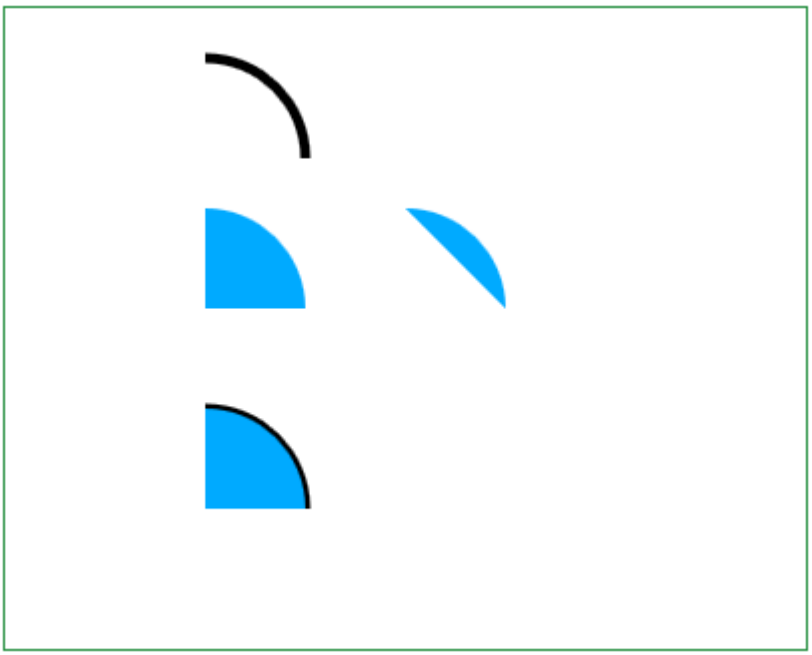
α_s – to kąt początkowy łuku

α_e - to kąt końcowy łuku

`counterClockwise` - kierunek rysowania. Są dwie możliwości:

- true - gdy kierunek rysowania jest przeciwny do ruchu wskazówek zegara

- false - gdy kierunek rysowania jest zgodny z ruchem wskazówek zegara



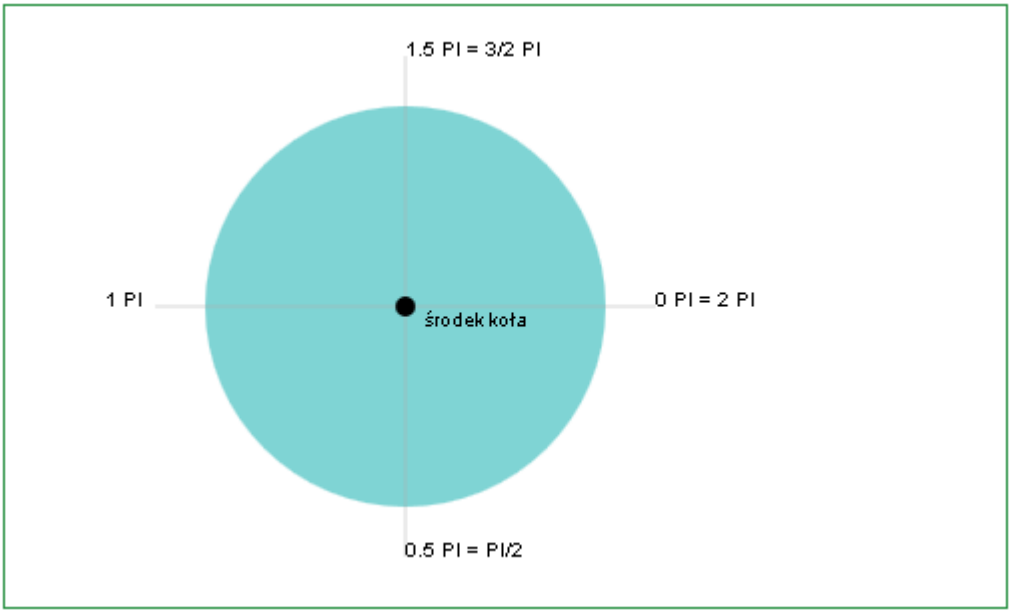
Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
ctx.beginPath();
var x = 100;
var y = 75;
var radius = 50;
var startAngle = 1.5*Math.PI;
var endAngle = 0;
var counterClockwise = false;
ctx.arc(x, y, radius, startAngle, endAngle, counterClockwise);
ctx.lineWidth = 5;
ctx.strokeStyle = "black";
ctx.stroke();
//-
ctx.beginPath();
var x = 100;
var y = 150;
var radius = 50;
var startAngle = 1.5*Math.PI;
var endAngle = 0;
var counterClockwise = false;
ctx.arc(x, y, radius, startAngle, endAngle, counterClockwise);
ctx.lineTo(x,y);
ctx.closePath();
ctx.fillStyle = "#00aaff";
ctx.fill();
//-
ctx.beginPath();
var x = 100;
var y = 250;
var radius = 50;
var startAngle = 1.5*Math.PI;
var endAngle = 0;
var counterClockwise = false;
ctx.arc(x, y, radius, startAngle, endAngle, counterClockwise);
ctx.lineWidth = 5;
ctx.strokeStyle = "black";
ctx.stroke();
```

```
ctx.lineTo(x,y);
ctx.closePath();
ctx.fillStyle = "#00aaff";
ctx.fill();
//-
ctx.beginPath();
var x = 200;
var y = 150;
var radius = 50;
var startAngle = 1.5*Math.PI;
var endAngle = 0;
var counterClockwise = false;
ctx.arc(x, y, radius, startAngle, endAngle, counterClockwise);
ctx.closePath();
ctx.fillStyle = "#00aaff";
ctx.fill();
```

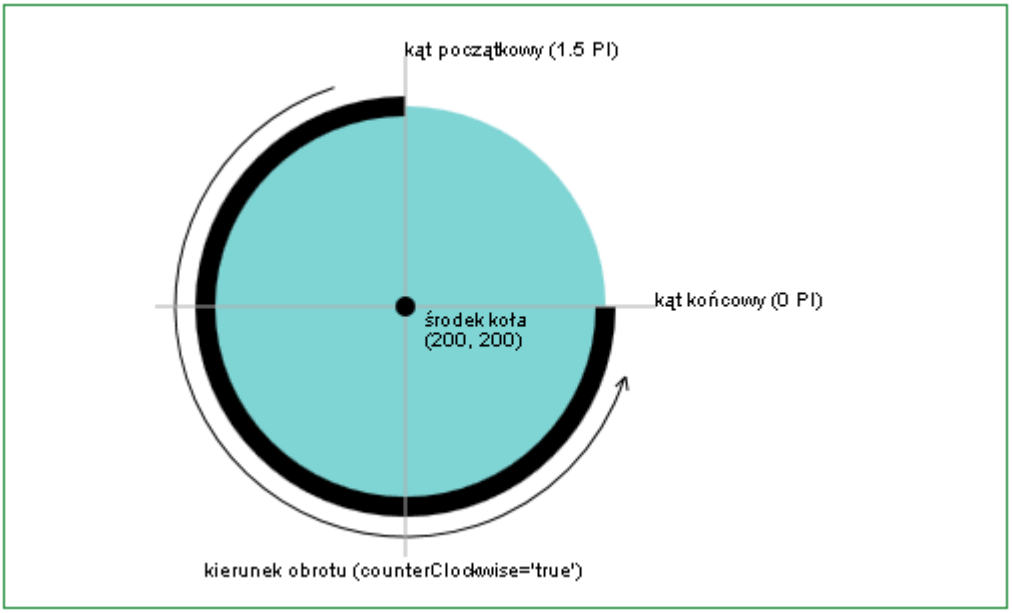
Objaśnienie 1

Kąt początkowy łuku:



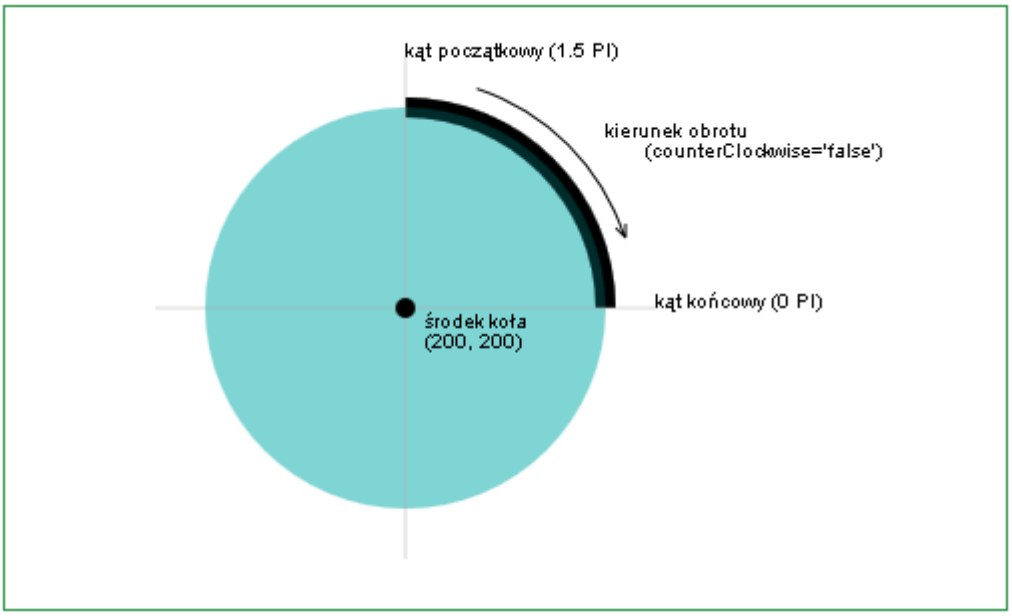
Objaśnienie 2

Kierunek przeciwny do kierunku ruchu wskazówek zegara:

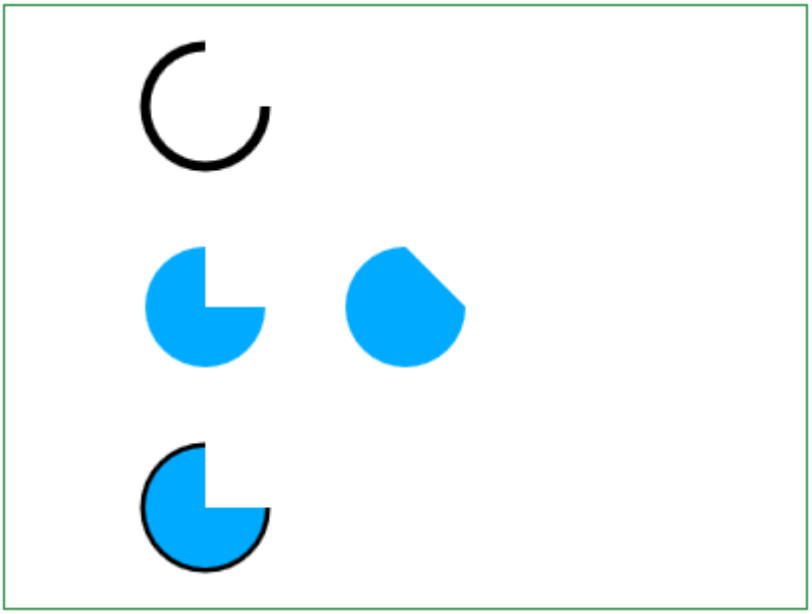


Objaśnienie 3

Kierunek ruchu zgodny z kierunkiem ruchu wskazówek zegara:



Łuk koła i wycinek koła (2)



Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
ctx.beginPath();
var x = 100;
var y = 50;
var radius = 30;
var startAngle = 1.5*Math.PI;
var endAngle = 0;
var counterClockwise = true;
ctx.arc(x, y, radius, startAngle, endAngle, counterClockwise);
ctx.lineWidth = 5;
ctx.strokeStyle = "black";
ctx.stroke();
//-
ctx.beginPath();
var x = 100;
var y = 150;
var radius = 30;
var startAngle = 1.5*Math.PI;
var endAngle = 0;
var counterClockwise = true;
ctx.arc(x, y, radius, startAngle, endAngle, counterClockwise);
ctx.lineTo(x,y);
ctx.closePath();
ctx.fillStyle = "#00aaff";
ctx.fill();
//-
ctx.beginPath();
var x = 100;
var y = 250;
var radius = 30;
var startAngle = 1.5*Math.PI;
var endAngle = 0;
var counterClockwise = true;
ctx.arc(x, y, radius, startAngle, endAngle, counterClockwise);
ctx.lineWidth = 5;
ctx.strokeStyle = "black";
ctx.stroke();
ctx.lineTo(x,y);
ctx.closePath();
```

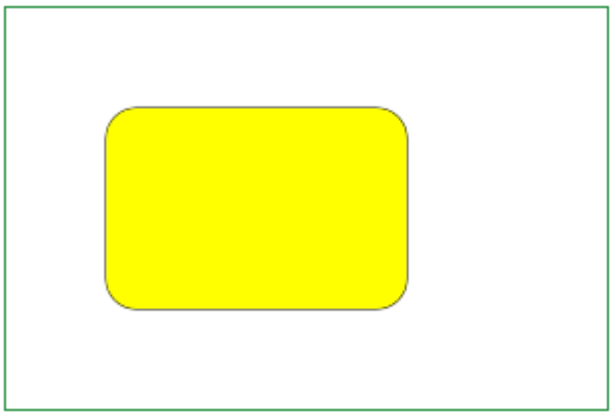
```
ctx.fillStyle = "#00aaff";
ctx.fill();
//-
ctx.beginPath();
var x = 200;
var y = 150;
var radius = 30;
var startAngle = 1.5*Math.PI;
var endAngle = 0;
var counterClockwise = true;
ctx.arc(x, y, radius, startAngle, endAngle, counterClockwise);
ctx.closePath();
ctx.fillStyle = "#00aaff";
ctx.fill();
```

Zaokrąglanie rogów prostokąta

Prostokąt zaokrąglamy używając polecenia `arcTo(`x_0`,`y_0`,`x_1`,`y_1`, r)`

W poleceniu podajemy:

- punkt początkowy łuku (x_0,y_0)
- punkt końcowy łuku(x_1,y_1)
- r - promień łuku



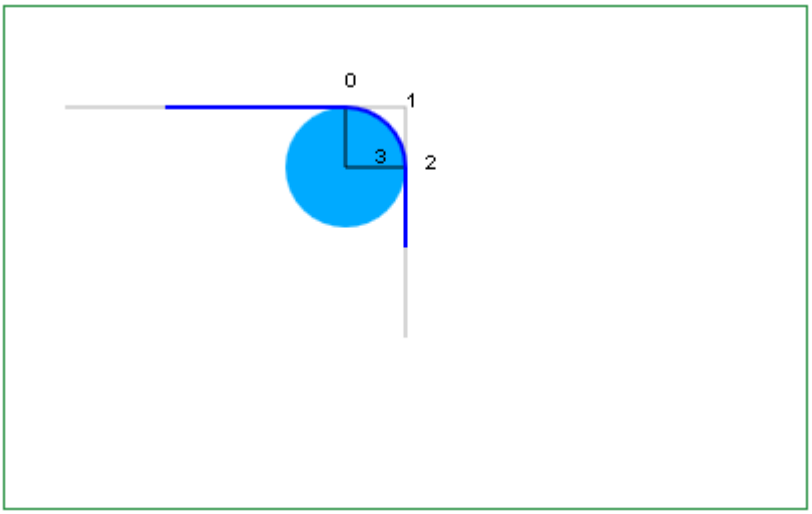
Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var leftX = 50;
var topY = 50;
var width = 150;
var height = 100;
var cornerRadius = 15;
ctx.beginPath();
ctx.moveTo(leftX + cornerRadius, topY);
ctx.lineTo(leftX+ width - cornerRadius, topY);
ctx.arcTo(leftX + width, topY, leftX + width, topY + cornerRadius,
             cornerRadius);
ctx.lineTo(leftX + width, topY + height - cornerRadius);
ctx.arcTo(leftX + width, topY + height, leftX + width - cornerRadius,
             topY + height, cornerRadius);
ctx.lineTo(leftX + cornerRadius, topY + height);
ctx.arcTo(leftX, topY + height, leftX, topY + height - cornerRadius,
                    cornerRadius);
ctx.lineTo(leftX, topY + cornerRadius);
ctx.arcTo(leftX, topY, leftX + cornerRadius, topY, cornerRadius);
ctx.strokeStyle = "black";
```

```
ctx.stroke();
ctx.fillStyle = "yellow";
ctx.fill();
```

Objaśnienie arcTo()

-



Punkt 0 jest punktem końcowym ścieżki i punktem początkowym łuku o współrzędnych (x0, y0).

Punkt 1 jest punktem formującym łuk o współrzędnych (x1, y1).

Punkt 2 jest punktem końcowym łuku i punktem początkowym ścieżki o wspłrzędnych (x2, y2).

Punkt 3 jest promieniem łuku.

Radius nie może być ujemny.

Jeżeli punkty 0,1 i 2 nie pokrywają się oraz jeżeli wszystkie trzy punkty nie leżą na jednej linii to łuk tworzony przez metodę jest najkrótszym łukiem obwodu koła o promieniu 3, które ma:

- jeden punkt styczny z prostą zaczynającą się w punkcie 1 i przechodzącą przez punkt 0 biegnącą do nieskończoności
- drugi (inny) punkt styczny z prostą zaczynającą się w punkcie 1 i przechodzącą przez punkt 2 i biegnącą do nieskończoności

W metodzie arcTo() podajemy współrzędne punktu 1 i 2 oraz wartość 3.

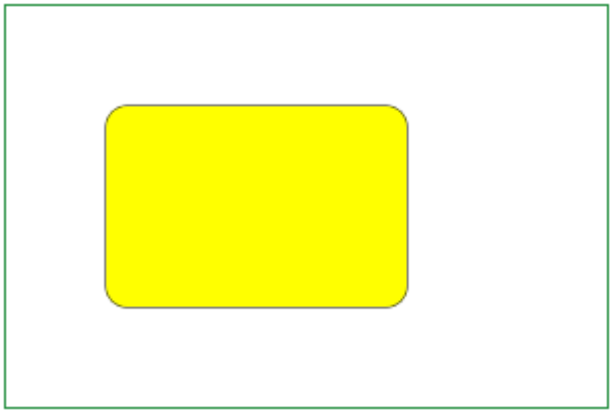
Metoda jest dośc trudna do stosowania. Dlatego stosuje się ją przede wszystkim do zaokrąglania brzegów prostokątów. W takim przypadku:

- odległość od 0 do 1 wynosi 3
- odległość od 1 do 2 wynosi 3

W przypadku innych łuków czy łączenia ścieżek znacznie łatwiej (a więc i lepiej) jest użyć metody arc. Oczywiście również w opisywanym przypadku można by użyć metody arc.

Zaokrąglanie rogów prostokąta - tworzenie funkcji

Tworzymy funkcje rysujące prostokąt z zaokrąglonymi rogami.

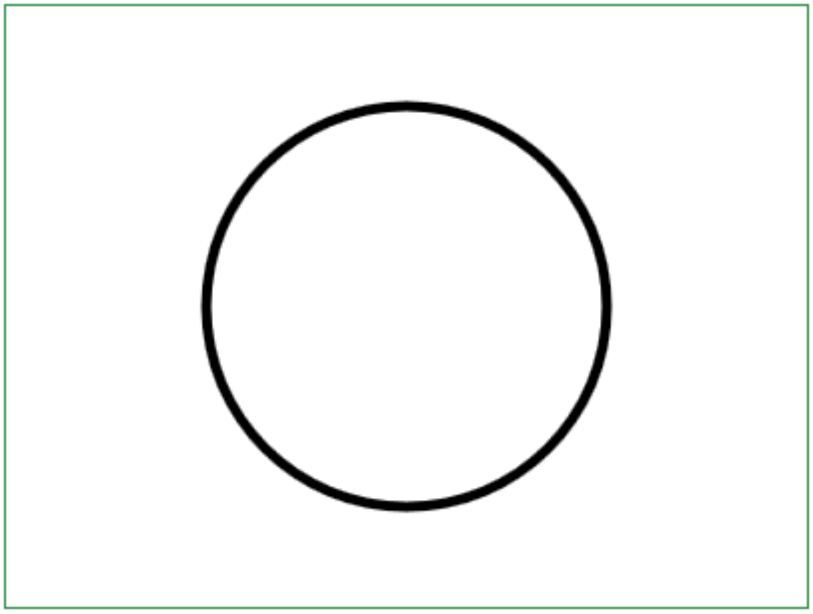


Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
function roundRect(leftX, topY, width, height, cornerRadius) {
       ctx.moveTo(leftX + cornerRadius, topY);
       ctx.lineTo(width - cornerRadius, topY);
       ctx.arcTo(leftX + width, topY, leftX + width, topY + cornerRadius,
                    cornerRadius);
       ctx.lineTo(leftX + width, topY + height - cornerRadius);
       ctx.arcTo(leftX + width, topY + height, leftX + width
                    - cornerRadius, topY + height, cornerRadius);
       ctx.lineTo(leftX + cornerRadius, topY + height);
       ctx.arcTo(leftX, topY + height, leftX,
                    topY + height - cornerRadius, cornerRadius);
       ctx.lineTo(leftX, topY + cornerRadius);
       ctx.arcTo(leftX, topY, leftX + cornerRadius, topY, cornerRadius);
};
function strokeRoundRect(leftX, topY, width, height, cornerRadius,
              strokeStyle) {
       ctx.beginPath();
       roundRect(leftX, topY, width, height, cornerRadius);
       ctx.strokeStyle = strokeStyle;
       ctx.stroke();
}
function fillRoundRect(leftX, topY, width, height, cornerRadius,
              fillStyle) {
       ctx.beginPath();
       roundRect(leftX, topY, width, height, cornerRadius);
       ctx.fillStyle = fillStyle;
       ctx.fill();
}
strokeRoundRect(50, 50, 150, 100, 10, "black");
fillRoundRect(50, 50, 150, 100, 10, "yellow");
```

Okrąg

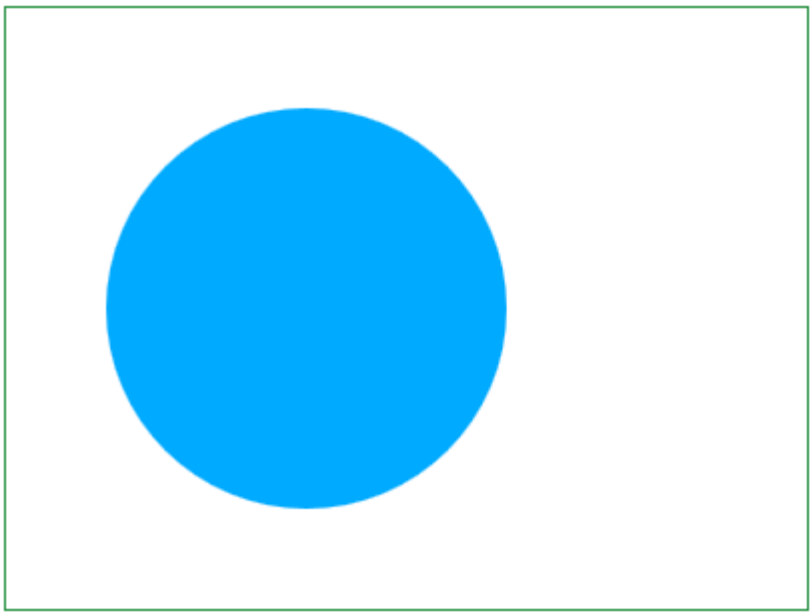
Okrąg rysujemy używając polecenia `arc()`. Jedyną różnicą jest to, że podajemy jeden z kątów jako 0, a drugi jako 360° (2* Math.PI)



Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
ctx.beginPath();
var x = 200;
var y= 150;
var radius= 100;
var startAngle = 2*Math.PI;
var endAngle = 0;
var counterClockwise = false;
ctx.lineWidth = 5;
ctx.arc(x, y, radius, startAngle, endAngle, counterClockwise);
ctx.strokeStyle = "#000000";
ctx.stroke();
```

Koło



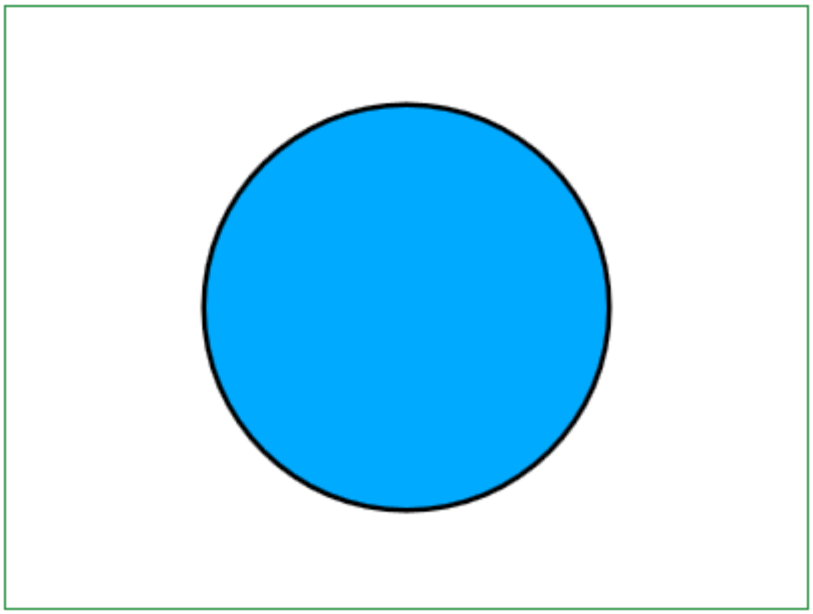
Koło rysujemy podobnie jak okrąg, ale używamy polecenia `fill()`.

Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
ctx.beginPath();
var x = 150;
var y= 150;
var radius= 100;
var startAngle = 2*Math.PI;
var endAngle = 0;
var counterClockwise = false;
ctx.arc(x, y, radius, startAngle, endAngle, counterClockwise);
ctx.fillStyle = "#00aaff";
ctx.fill();
```

Koło + okrąg

Koło z konturem rysujemy, używając poleceń `stroke()` i `fill()`.



Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
ctx.beginPath();
var x = 200;
var y= 150;
var radius= 100;
var startAngle = 2*Math.PI;
var endAngle = 0;
var counterClockwise = false;
ctx.arc(x, y, radius, startAngle, endAngle, counterClockwise);
ctx.lineWidth = 5;
ctx.strokeStyle = "#000000";
ctx.stroke();
ctx.fillStyle = "#00aaff";
ctx.fill();
```

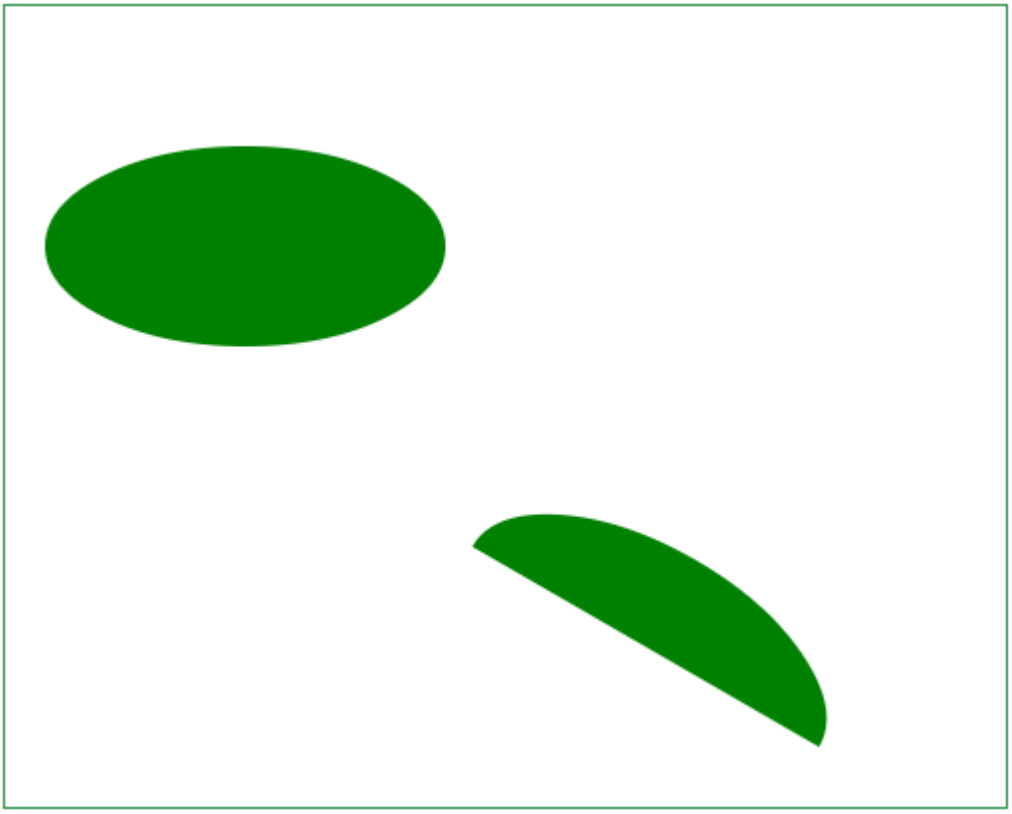
Elipsa

W chwili pisania (grudzień 2014) działa tylko w Operze i Google Chrome

Tworzy elipsę albo wycinek elipsy.

W `ellipse(x,y,radiusX,radiusY, rotation, startAngle, engAngle, anticlockwise)` podajemy:

współrzędne środka elipsy, promień w osi X, promień w osi Y, kąt obrotu w radianach oraz kąty i kierunek rysowania tak jak w funcji arc.



Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
ctx.fillStyle="green";
ctx.ellipse(120, 120, 100, 50, 0, 0, 2*Math.PI, false);
ctx.fill();
ctx.beginPath();
ctx.ellipse(320, 320, 100, 50, Math.PI/4, Math.PI, 2*Math.PI, 0, true);
ctx.fill();
```

Transformacje

Algebra liniowa

Podstawy teoretyczne dotyczące wektorów znajdują się w Dodatku 3, a dotyczące macierzy w Dodatku 4.

Podstawy teoretyczne dotyczące transformacji znajdują się w Dodatku 5.

Macierz przekształceń

Macierz przekształceń w znaczniku <canvas> ma postać:

$$\begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix}$$

gdzie *a,b,c,d,e,f* są liczbami ustalanymi przez programistę. W funkcjach `transform(a,b,c,d,e,f)` i `setTransform(a,b,c,d,e,f)` liczby ułożone są kolejno.

W innych językach kolejność może być inna, np. w języku Java: *a,c,e,b,d,f*.

Zerowanie macierzy

Aktualną macierz <canvas> zastępujemy macierzą jednostkową (domyślną):

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

przy użyciu `setTransfrorm(1,0,0,1,0,0)` . Aktualna macierz jest zastąpiona macierzą zerową.

Translacja

Macierz translacji ma postać:

$$\begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix}$$

a cała operacja wygląda następująco:

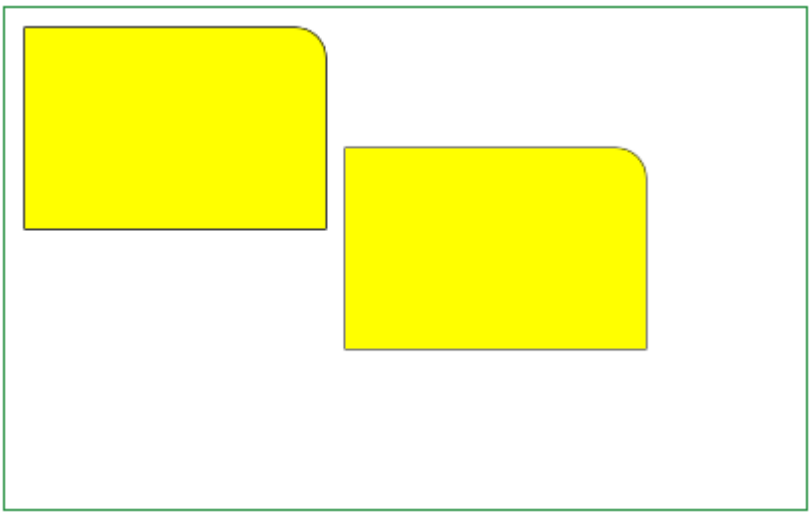
$$\begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x+dx \\ y+dy \\ 1 \end{bmatrix}$$

gdzie *x*, *y* oznaczają aktualne współrzędne punktu, który ulegnie przesunięciu.

Translację, czyli przesunięcie obiektu możemy wykonać trzema metodami.

Sposób 1

Używamy `translate(dx, dy)` . *dx* – oznacza przesunięcie w poziomie, *dy* – określa przesunięcie w pionie. Ta transformacja jest złożeniem aktualnej macierzy przekształceń i żądanej przez programistę translacji.



Listing

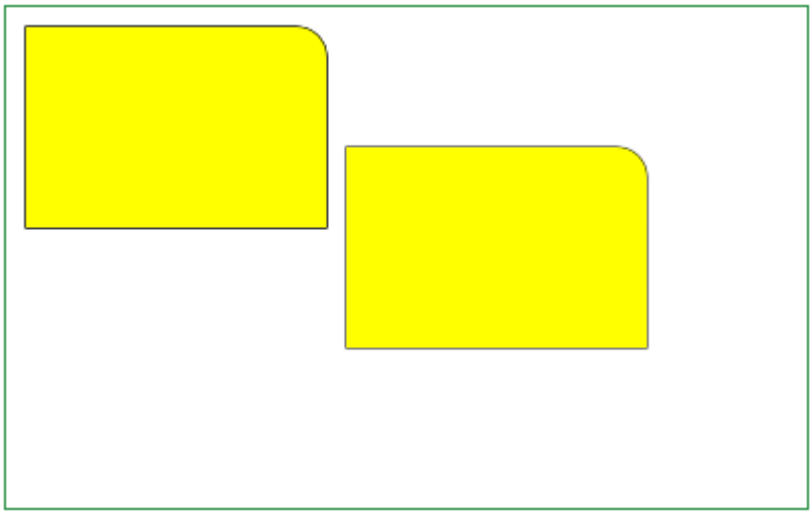
```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var leftX = 10;
var topY = 10;
var width = 150;
var height = 100;
var cornerRadius = 15;
drawRectangle(ctx);
ctx.translate(160, 60);
drawRectangle(ctx);
```

Funkcja drawRectangle()

```
function drawRectangle(context) {
      context.save();
      context.moveTo(leftX, topY);
      context.lineTo(leftX + width - cornerRadius, topY);
      context.arcTo(leftX + width, topY, leftX + width, topY + cornerRadius,
                  cornerRadius);
      context.lineTo(leftX + width, topY + height);
      context.lineTo(leftX, topY + height);
      context.closePath();
      context.restore();
      context.strokeStyle = "black";
      context.stroke();
      context.fillStyle = "yellow";
      context.fill();

};
```

Sposób 2

Ustawiamy translację przy użyciu `setTransform(1,0,0,1,dx,dy)` . Macierz jest zerowana, a następnie ustawiana na podaną transformację.



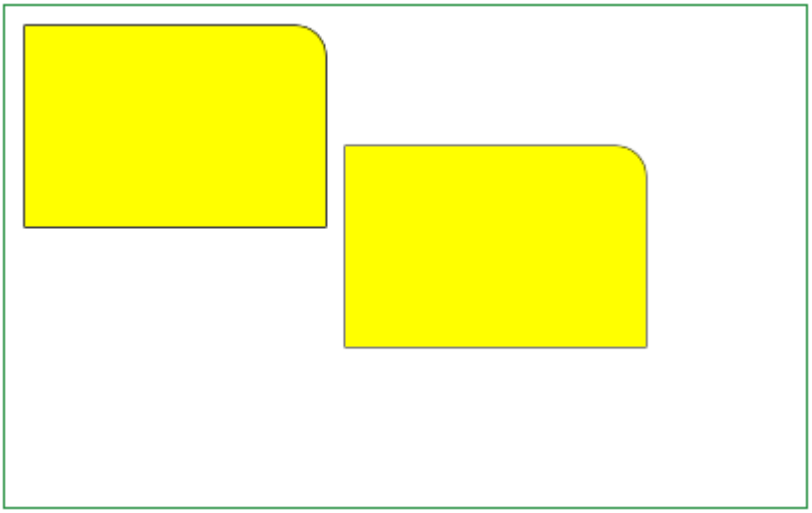
Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var leftX = 10;
var topY = 10;
var width = 150;
var height = 100;
var cornerRadius = 15;
drawRectangle(ctx);
ctx.setTransform(1,0,0,1,160,60);
drawRectangle(ctx);
```

Sposób 3:

Zerujemy aktualną macierz przy użyciu `setTransfrorm(1,0,0,1,0,0)` . Aktualna macierz staje się macierzą zerową.

Ustawiamy translację przy użyciu `transform(1,0,0,1,dx,dy)` . Wynikiem jest pomnożenie macierzy jednostkowej przez macierz translacji, co w rezultacie daje translację, czyli ten sam wynik co w sposobie 2.



Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var leftX = 10;
var topY = 10;
var width = 150;
var height = 100;
var cornerRadius = 15;
drawRectangle(ctx);
ctx.setTransform(1,0,0,1,0,0);
ctx.transform(1,0,0,1,160,60);
drawRectangle(ctx);
```

Skalowanie

Macierz skalowania ma postać:

$$\begin{bmatrix} sx & c & c \\ 0 & sy & c \\ 0 & c & 1 \end{bmatrix}$$

a cała operacja przebiega następująco:

$$\begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} sx \cdot x \\ sy \cdot y \\ 1 \end{bmatrix}$$

gdzie x, y oznaczają aktualne współrzędne punktu, który ulegnie skalowaniu.

Skalowanie możemy wykonać na trzy sposoby.

Sposób 1

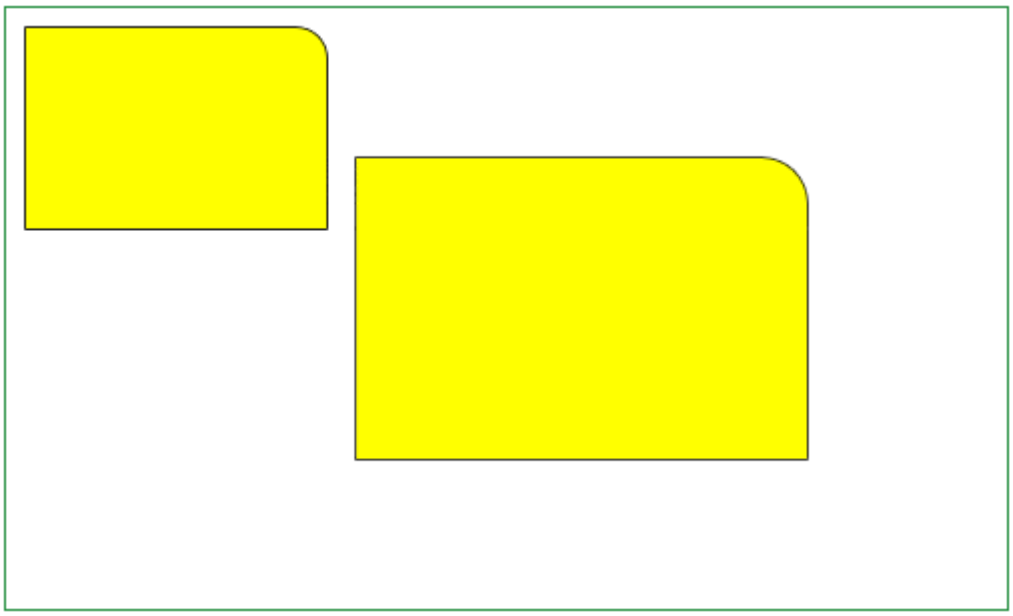
Używamy funkcji `scale(sx,sy)`. *sx* – oznacza współczynnik skalowania w poziomie, *sy* – oznacza współczynnik skalowania w pionie. Ta tranformacja jest złożeniem aktualnej macierzy przekształceń i żądanego przez programistę skalowania.

Żeby zobaczyć skalowaną figurę trzeba dokonać translacji.

Kolejność wykonania translacji i skalowania nie jest obojętna.

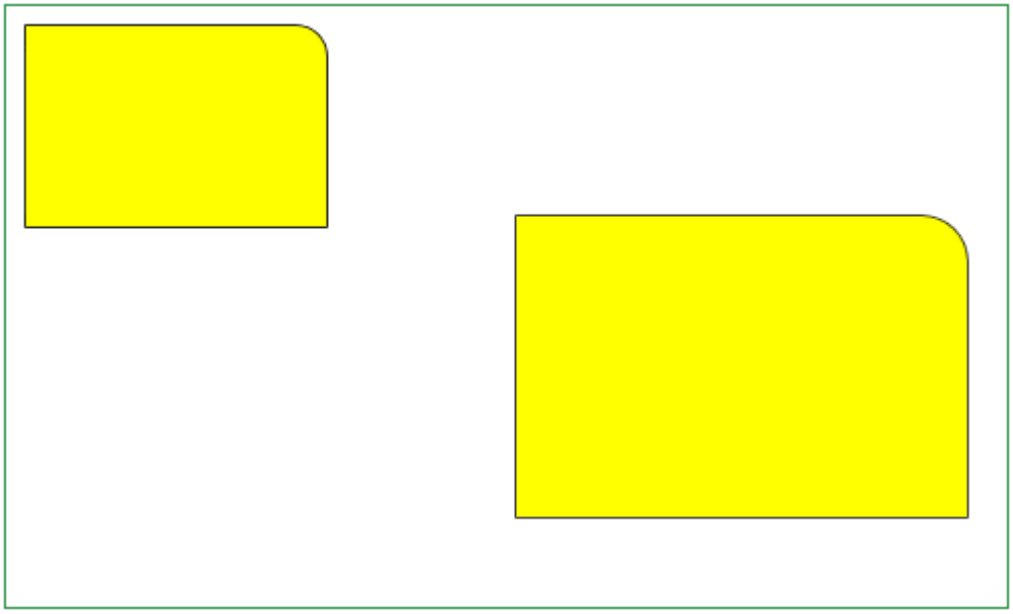
Najpierw translacja, potem skalowanie

```
ctx.translate(160,60);
ctx.scale(1.5, 1.5);
```



Najpierw skalowanie, potem translacja

```
ctx.scale(1.5, 1.5);
ctx.translate(160,60);
```



Listing

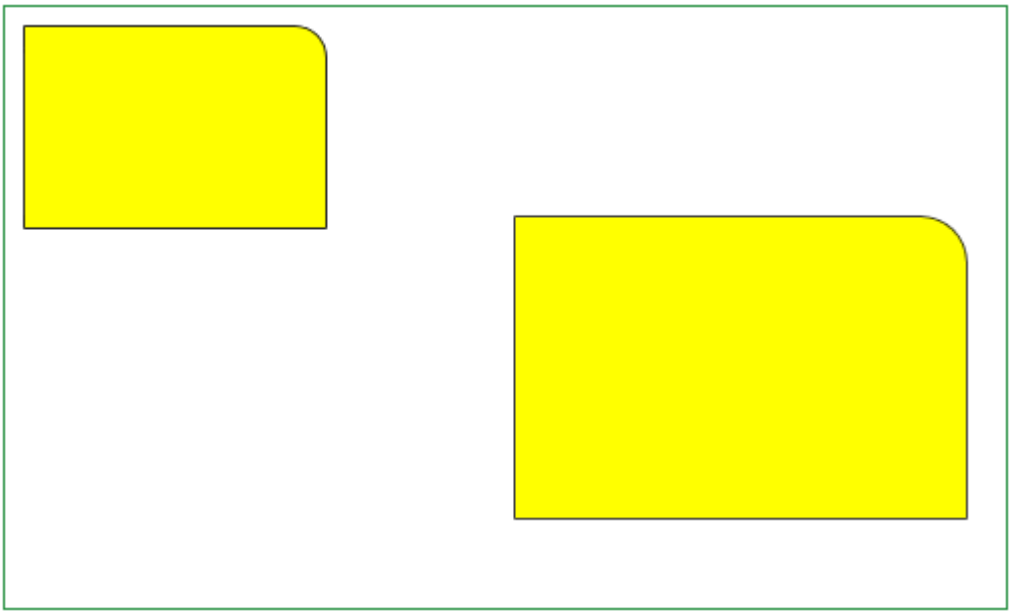
```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var leftX = 10;
var topY = 10;
var width = 150;
var height = 100;
var cornerRadius = 15;
drawRectangle(ctx);
ctx.translate(160,60);
ctx.scale(1.5, 1.5);
drawRectangle(ctx);
var cv1 = document.getElementById("canvas1");
var ctx1 = cv1.getContext("2d");
drawRectangle(ctx1);
ctx1.scale(1.5, 1.5);
ctx1.translate(160,60);
drawRectangle(ctx1);
```

Sposób 2

Ustawiamy skalowanie przy użyciu `setTransform(sx,0,0,sy,0,0)`. Macierz jest zerowana, a następnie ustawiana na podaną transformację.

Żeby zobaczyć skalowaną figurę trzeba dokonać translacji.

Kolejność wykonania translacji i skalowania nie jest obojętna. Operację `setTransform()` musimy wykonać jako pierwszą, gdyż ta funkcja, zerując macierz, anuluje poprzednie operacje.



Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var leftX = 10;
var topY = 10;
var width = 150;
var height = 100;
var cornerRadius = 15;
```

```
drawRectangle(ctx);
ctx.setTransform(1.5,0,0,1.5,0,0);
ctx.translate(160, 60);
drawRectangle(ctx);
```

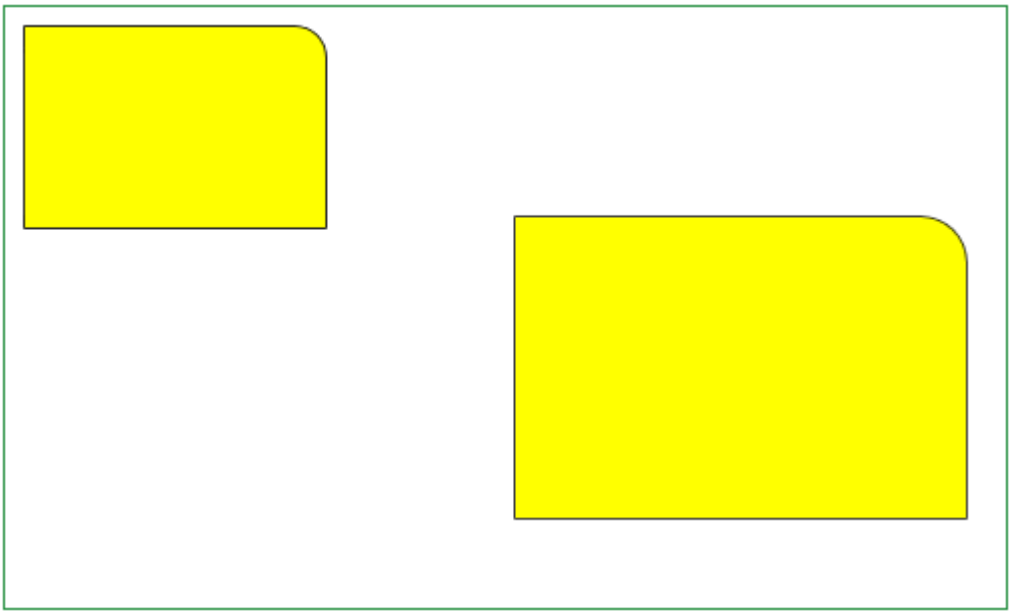
Sposób 3

Zerujemy aktualną macierz przy użyciu `setTransfrorm(1,0,0,1,0,0)`. Aktualna macierz staje się macierzą zerową.

Ustawiamy skalowanie przy użyciu `transform(sx,0,0,sy,0,0)`. Wynikiem jest pomnożenie macierzy jednostkowej przez macierz skalowania, co w rezultacie daje skalowanie, czyli ten sam wynik co w sposobie 2.

Żeby zobaczyć skalowaną figurę trzeba dokonać translacji.

Kolejność wykonania translacji i skalowania nie jest obojętna, a skutki zmiany kolejności są takie jak w sposobie 2.

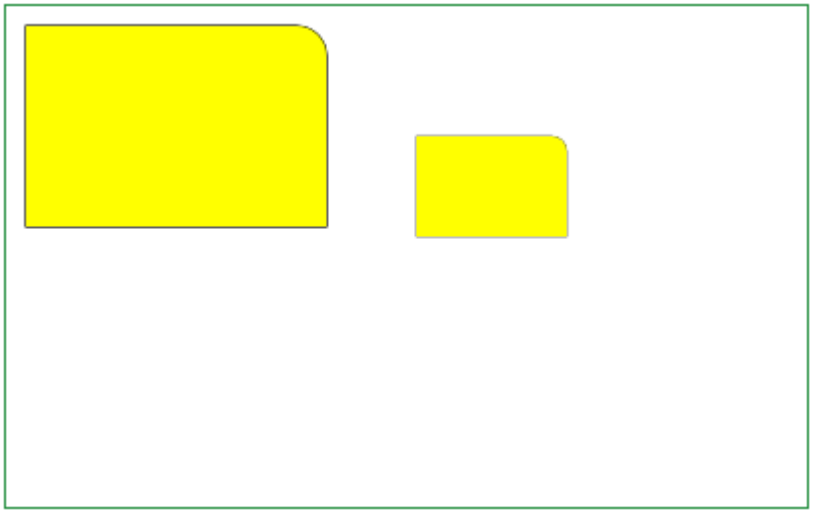


Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var leftX = 10;
var topY = 10;
var width = 150;
var height = 100;
var cornerRadius = 15;
drawRectangle(ctx);
ctx.setTransform(1,0,0,1,0,0);
ctx.transform(1.5, 0, 0, 1.5, 0, 0);
ctx.translate(160, 60);
drawRectangle(ctx);
```

Skalowanie - Zmniejszenie

Zmniejszenie wielkości figury następje, gdy 0<*dx*<1 lub 0<*dy*<1

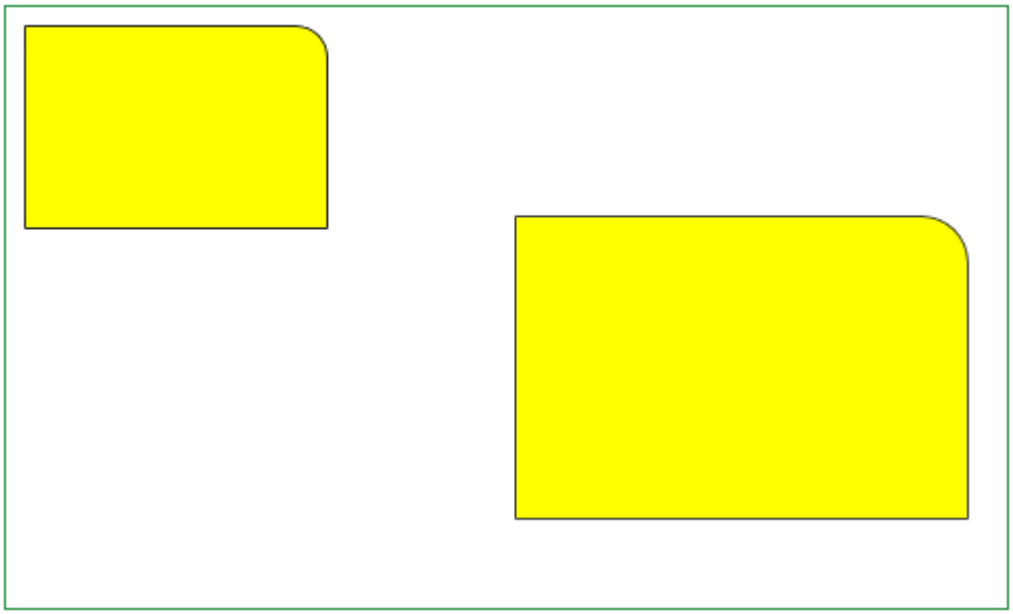


Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var leftX = 10;
var topY = 10;
var width = 150;
var height = 100;
var cornerRadius = 15;
drawRectangle(ctx);
ctx.translate(200, 60);
ctx.scale(0.5,0.5);
drawRectangle(ctx);
```

Skalowanie - Zwiększenie

Zwiększenie wielkości figury następuje, gdy *dx*>1 lub *dy*>1.



Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var leftX = 10;
var topY = 10;
var width = 150;
var height = 100;
var cornerRadius = 15;
drawRectangle(ctx);
ctx.scale(1.5,1.5);
ctx.translate(160, 60);
drawRectangle(ctx);
```

Skalowanie - elipsa

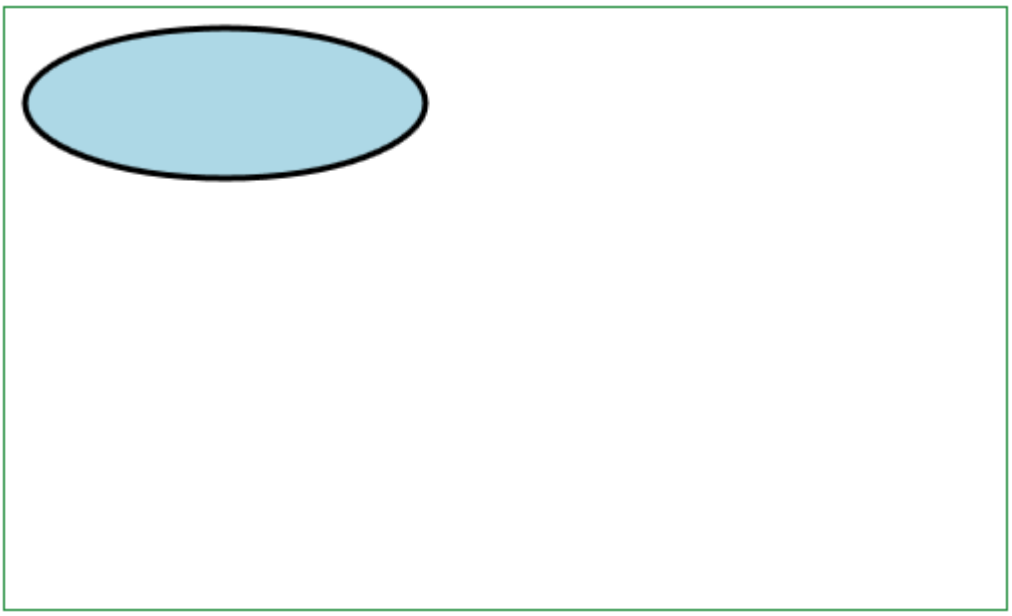
Brak jest funkcji rysującej elipsę. Elipsę możemy uzyskać przez skalowanie koła, ale prawidłowe dobranie położenia elipsy może sprawiać trudność. Dlatego utworzyliśmy funkcję drawEllipse, w której podajemy parametry *x*, *y*, *w*, *h*, czyli położenie lewego górnego rogu prostokąta obejmującego elipsę oraz szerokość i wysokość elipsy.

A oto nasza funkcja:

Listing

```
function drawEllipse(context, x, y, w, h) {
  var s = h / w;
  context.save();
  context.beginPath();
  context.translate(0, y - y * s);
  context.scale(1, s);
  context.arc(x + w / 2, y + w / 2, w / 2, 0, 2 * Math.PI, false);
  context.restore();
};
```

Zmianie uległa długość odcinka *y*. Ponieważ translację wykonaliśmy przed skalowaniem musieliśmy skrócić odcinek *y*. Gdyby translacja była wykonywana po skalowaniu, odcinek *y* musielibyśmy powiększyć.



Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
drawEllipse(ctx, 10, 10, 200, 75);
ctx.fillStyle = 'lightblue';
ctx.fill();
ctx.lineWidth = 3;
ctx.strokeStyle = 'black';
ctx.stroke();
```

Funkcja drawEllipse()

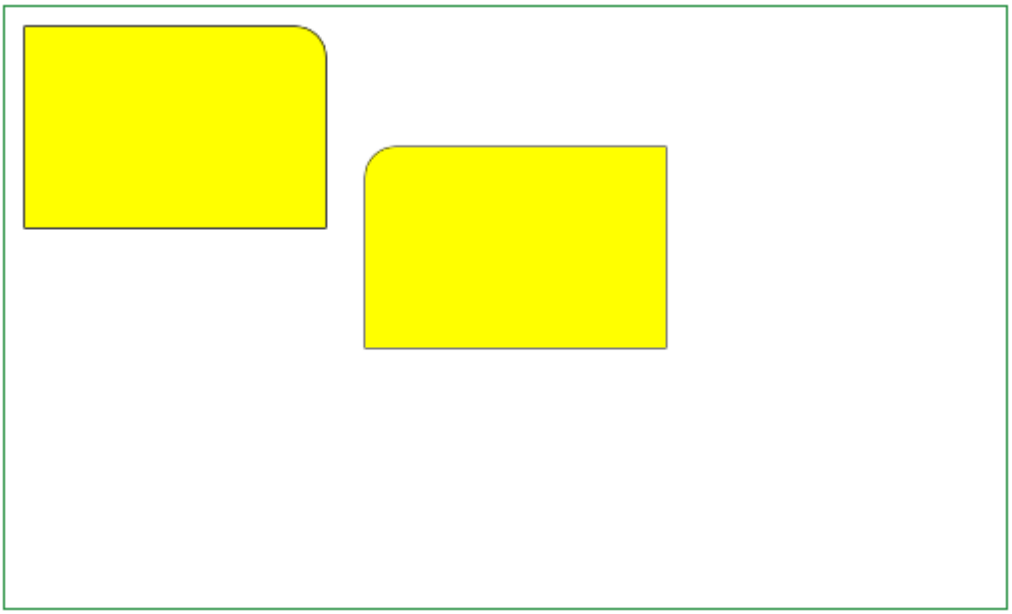
```
function drawEllipse(context, x, y, w, h) {
      var s = h / w;
      context.save();
      context.beginPath();
      context.translate(0, y - y * s);
      context.scale(1, s);
      context.arc(x + w / 2, y + w / 2, w / 2, 0, 2 * Math.PI, false);
      context.restore();

};
```

W dalszym ciągu poznamy inne sposoby utworzenia i narysowania elipsy.

Skalowanie - Lustro w poziomie

Odbicie lustrzane figury tworzymy gdy $dx<0$.

Figura jest odbijana symetrycznie po drugiej stronie osi Y, musimy więc dokonać translacji figury w pole widzenia.



Listing

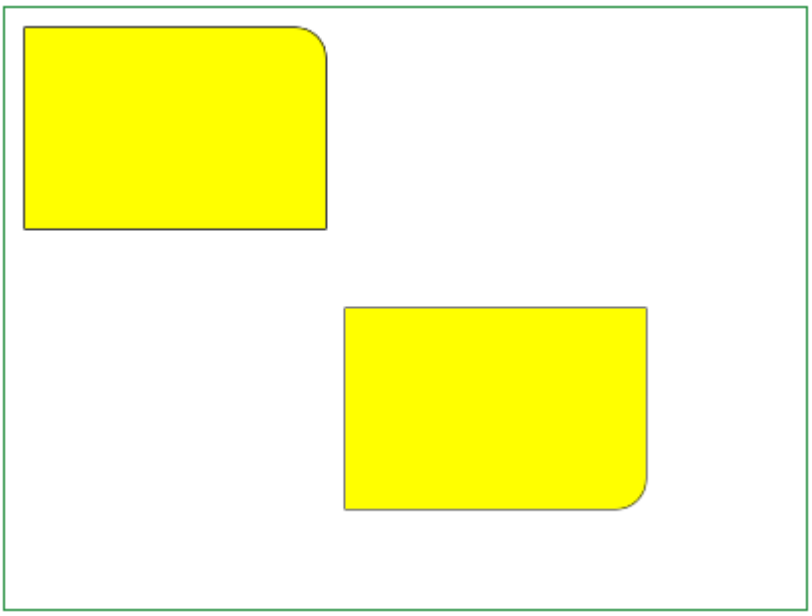
```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
```

```
var leftX = 10;
var topY = 10;
var width = 150;
var height = 100;
var cornerRadius = 15;
drawRectangle(ctx);
ctx.scale(-1,1);
ctx.translate(-340, 60);
drawRectangle(ctx);
```

Skalowanie - Lustro w pionie

Odbicie lustrzane figury tworzymy gdy *dy*<0.

Figura jest odbijana symetrycznie po drugiej stronie osi X, musimy więc dokonać translacji figury w pole widzenia.



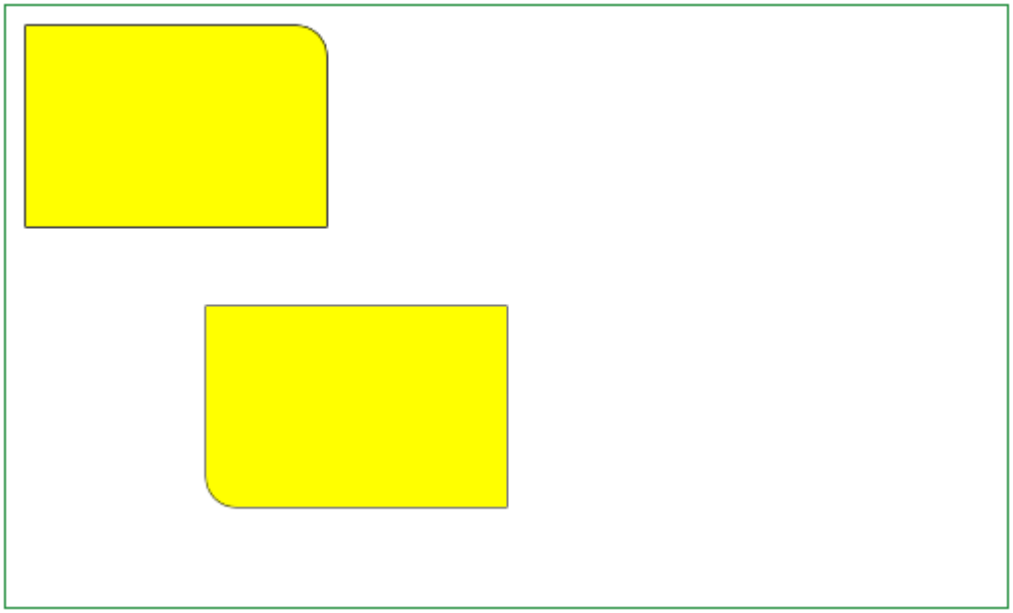
Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var leftX = 10;
var topY = 10;
var width = 150;
var height = 100;
var cornerRadius = 15;
drawRectangle(ctx);
ctx.scale(1,-1);
ctx.translate(160, -260);
drawRectangle(ctx);
```

Skalowanie - Lustro w pionie i poziomie

Odbicie lustrzane figury tworzymy gdy *dy*<0 i *dy*<0.

Figura jest odbijana symetrycznie po drugiej stronie osi X i Y, musimy więc dokonać translacji figury w pole widzenia.



Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var leftX = 10;
var topY = 10;
var width = 150;
var height = 100;
var cornerRadius = 15;
drawRectangle(ctx);
ctx.scale(-1,-1);
ctx.translate(-260, -260);
drawRectangle(ctx);
```

Odbicie względem prostej przechodzącej przez punkt (0,0)

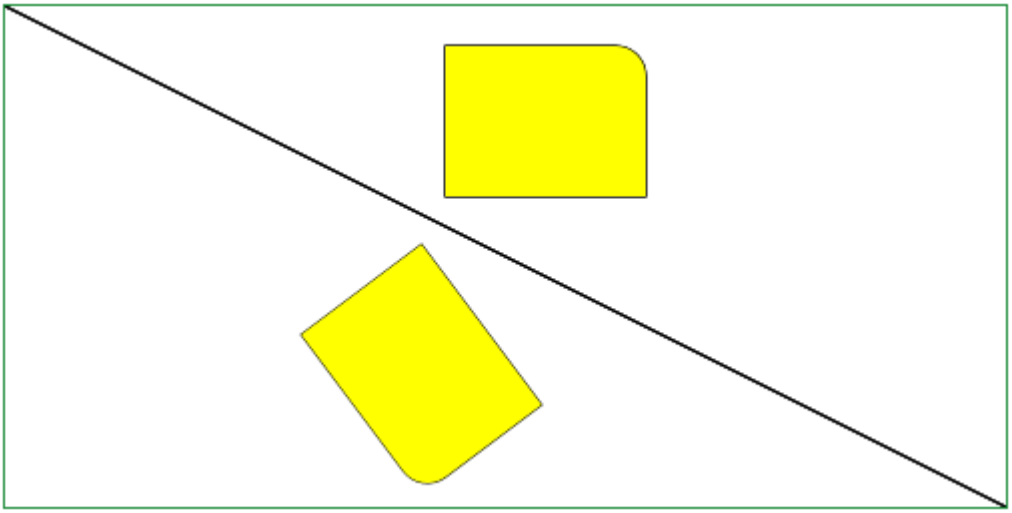
Odbicie przez oś X, oś Y lub obie osie może być wykonane za pomocą skalowania.

Ponieważ prosta przebiega przez brzegi elementu canvas, wiemy, że prosta przecina oś X pod kątem φ, którego tangens = 250/500=0.5, czyli m = 0.5, a b= 0, czyli prosta wyrażona jest równaniem y=mx+b, czyli y=0.5x+0.

Jeżeli tgφ= 0.5 to kąt wynosi 0.4636476090008061 radianów, czyli 26.56505117707799°

Macierz takiego przekształcenia wygląda następująco:

$$\begin{bmatrix} \cos 2\varphi & \sin 2\varphi & 0 \\ \sin 2\varphi & \cos 2\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
ctx.beginPath();
ctx.moveTo(0,0);
ctx.lineTo(500,250);
ctx.strokeStyle = "black";
ctx.stroke();
var leftX = 220;
var topY = 20;
var width = 100;
var height = 75;
var cornerRadius = 15;
drawRectangle(ctx);
var tan=0.5;
var atan=Math.atan(tan);
var cos2=Math.cos(2*atan);
var sin2 = Math.sin(2*atan);
ctx.setTransform(1,0,0,1,0,0);
ctx.transform(cos2,sin2,sin2,-cos2,0,0);
drawRectangle(ctx);
```

Obrót

Punkt (0,0) czyli górny, lewy róg ekranu jest środkiem obrotu. Obrót obiektu następuje o kąt α (podany w radianach). Promieniem obrotu jest odległość od punktu (0,0) do obracanego punktu. Kąty są liczone w kierunku ruchu wskazówek zegara.

Macierz obrotu wygląda następująco:

$$\begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(obrót w kierunku ruchu wskazówek zegara)

a cała operacja wygląda następująco:

$$\begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x\cos\alpha + y\sin\alpha \\ -x\sin\alpha + y\cos\alpha \\ 1 \end{bmatrix}$$

gdzie x, y oznaczają aktualne współrzędne punktu, który ulegnie obrotowi.

Jeżeli zmienimy macierz obrotu na:

$$\begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x\cos\alpha + y\sin\alpha \\ -x\sin\alpha + y\cos\alpha \\ 1 \end{bmatrix}$$

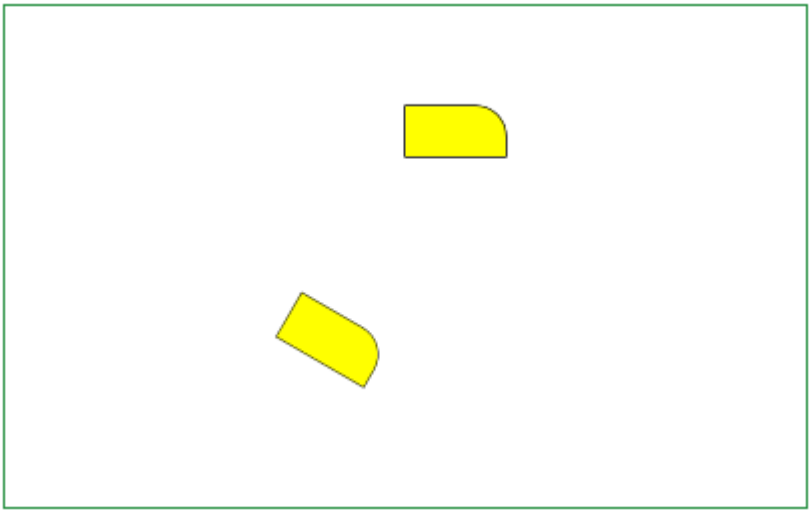
dokonamy obrotu w kierunku przeciwnym do ruchu wskazówek zegara.

Obrót możemy wykonać na trzy sposoby.

Sposób 1

Używamy funkcji `rotate(α)` . α – oznacza kąt obrotu w radianach. Ta tranformacja jest złożeniem aktualnej macierzy przekształceń i żądanego przez programistę obrotu.

Obracamy figurę o Math.PI/6=30° zgodnie z kierunkiem wskazówek zegara.



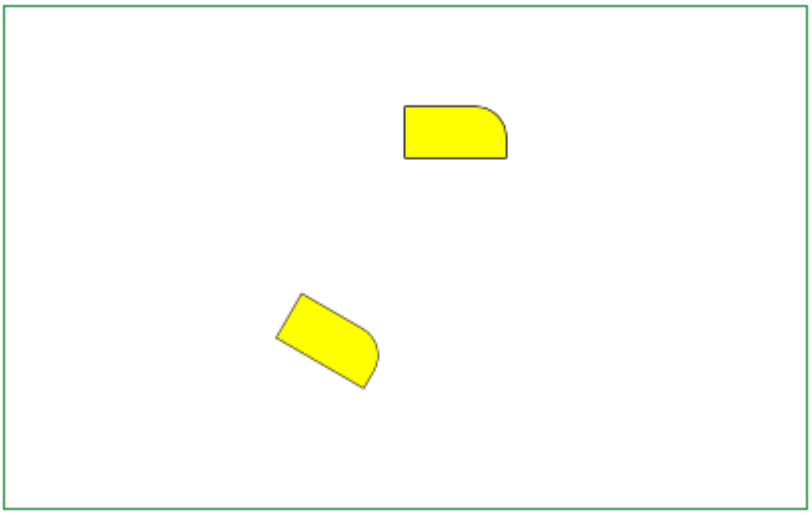
Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var leftX = 200;
var topY = 50;
var width = 50;
var height = 25;
var cornerRadius = 15;
drawRectangle(ctx);
ctx.rotate(Math.PI/6);
drawRectangle(ctx);
```

Sposób 2

Ustawiamy skalowanie przy użyciu setTransform(cosα,-sinα, sinα, cosα,0,0). Macierz jest zerowana, a następnie ustawiana na podaną transformację

Obracamy figurę o Math.PI/6=30° zgodnie z kierunkiem wskazówek zegara.



Listing

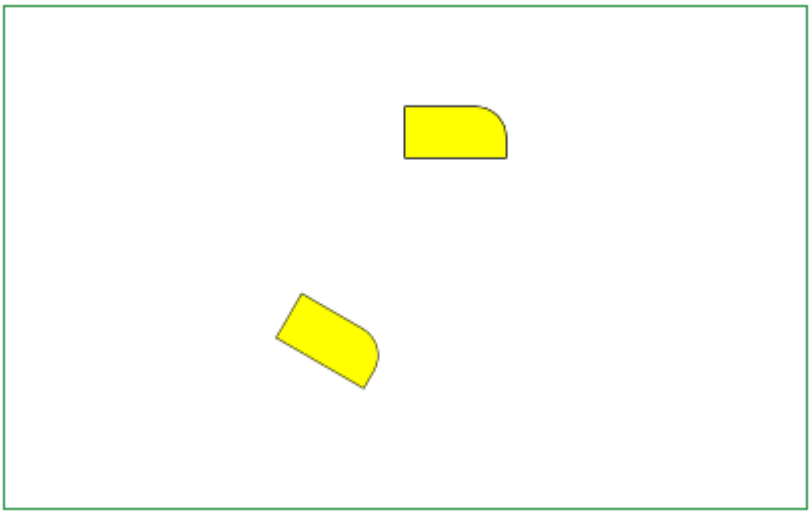
```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var leftX = 200;
var topY = 50;
var width = 50;
var height = 25;
var cornerRadius = 15;
drawRectangle(ctx);
ctx.setTransform(Math.cos(Math.PI/6),Math.sin(Math.PI/6),-
Math.sin(Math.PI/6),Math.cos(Math.PI/6),0,0);
       drawRectangle(ctx);
```

Sposób 3

Zerujemy aktualną macierz przy użyciu setTransfrorm(1,0,0,1,0,0). Aktualna macierz staje się macierzą zerową.

Ustawiamy obrót przy użyciu transform(cosα,-sinα, sinα, cosα,0,0). Wynikiem jest pomnożenie macierzy jednostkowej przez macierz obrotu, co w rezultacie daje obrót, czyli ten sam wynik co w sposobie 2.

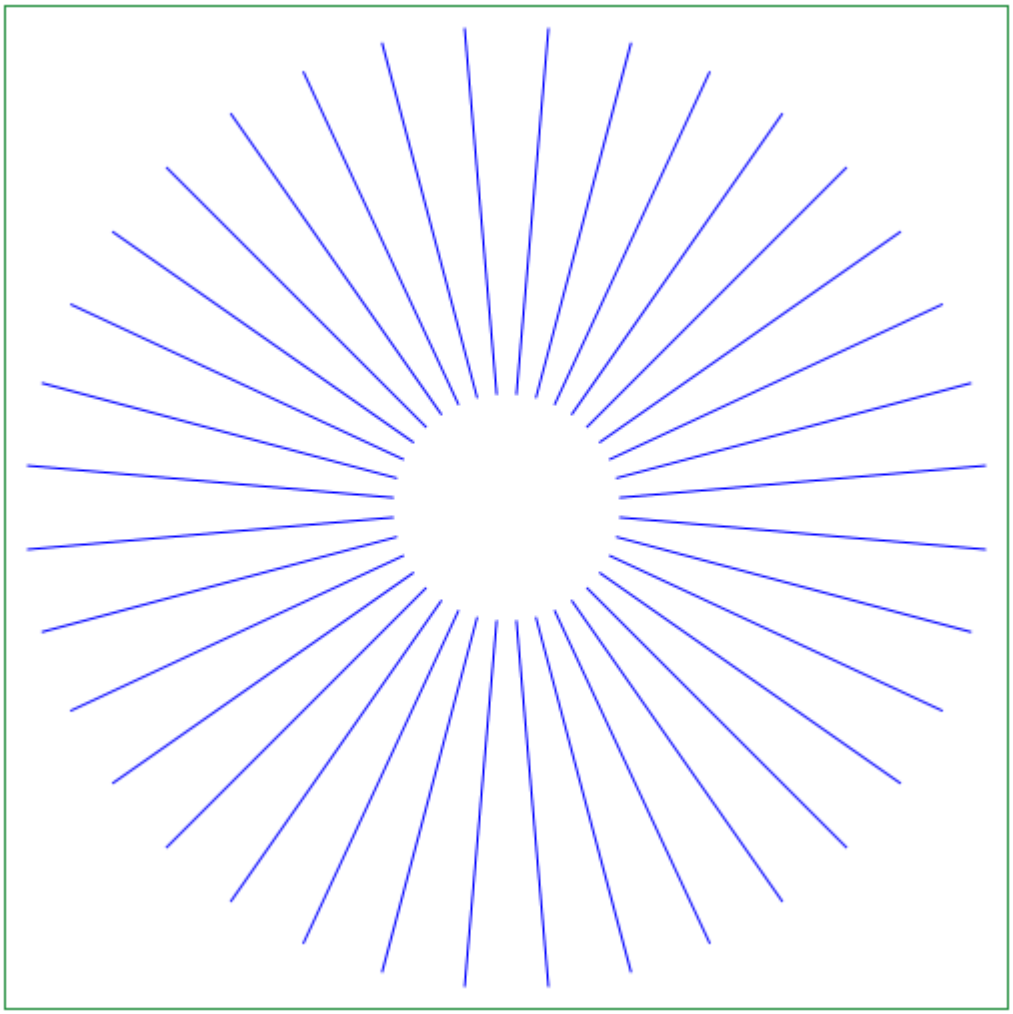
Obracamy figurę o Math.PI/6=30° zgodnie z kierunkiem wskazówek zegara.



Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var leftX = 200;
var topY = 50;
var width = 50;
var height = 25;
var cornerRadius = 15;
drawRectangle(ctx);
var a = Math.PI / 6.0;
var c = Math.cos(a);
var s = Math.sin(a);
ctx.setTransform(1,0,0,1,0,0);
ctx.transform(c, s, -s, c, 0, 0);
drawRectangle(ctx);
```

Obrót iterowany



Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
ctx.translate(250, 250);
var n = 36;
for (var i = 0; i < n; i++) {
      ctx.rotate(2 * Math.PI/n);
      ctx.moveTo(40,40);
      ctx.lineTo(170,170);
}
ctx.strokeStyle = "blue";
ctx.stroke();
```

Przekrzywienie

Macierz przekrzywienia wygląda następująco:

$$\begin{bmatrix} 1 & k_{ux} & 0 \\ k_{uy} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

a całe przekształcenie następująco:

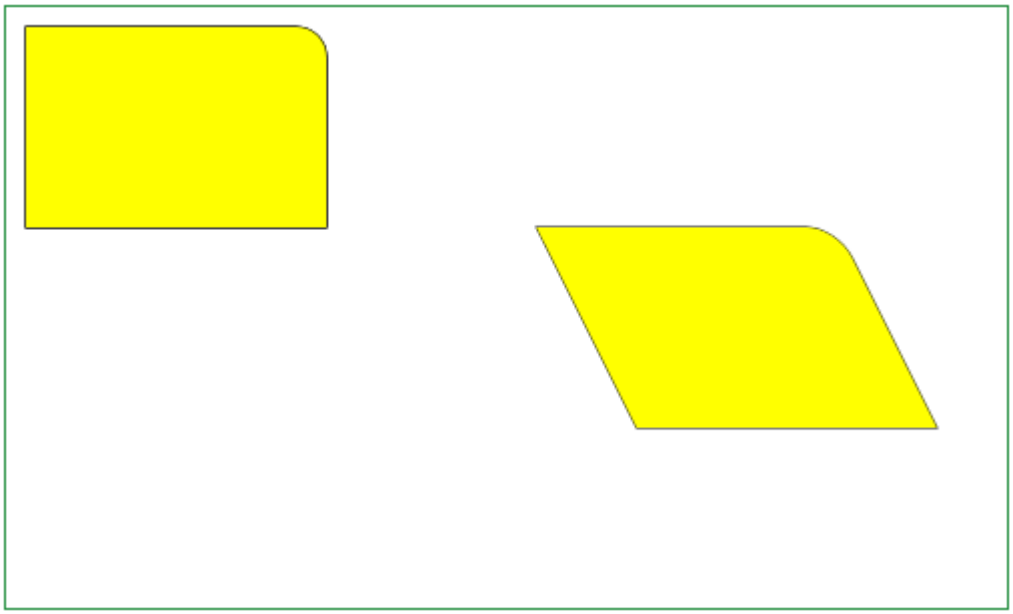
$$\begin{bmatrix} 1 & hx & 0 \\ hy & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + y \cdot hx \\ y + x \cdot hy \\ 1 \end{bmatrix}$$

gdzie *x*, *y* oznaczają aktualne współrzędne punktu, który ulegnie przekrzywieniu. *hx* – to współczynnnik przekoszenia wzdłuż osi X, *h*y – to współczynnik przekoszenia wzdłuż osi Y.

Przekrzywienie możemy wykonać na dwa sposoby.

Sposób 1

Ustawiamy przekrzywienie przy użyciu `setTransform(1,hy, hx, 1,0,0)` . Macierz jest zerowana, a następnie ustawiana na podaną transformację.



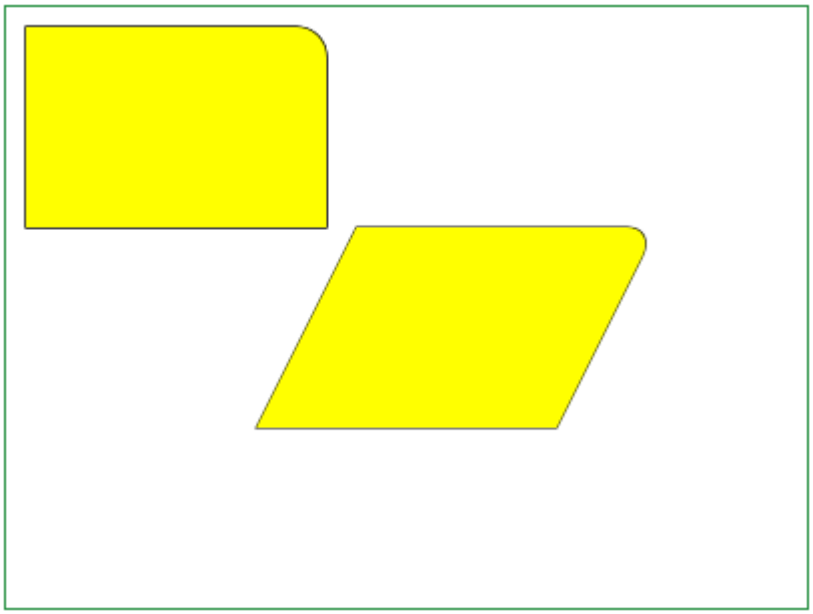
Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var leftX = 10;
var topY = 10;
var width = 150;
var height = 100;
var cornerRadius = 15;
drawRectangle(ctx);
ctx.setTransform(1,0,0.5,1,0,0);
ctx.translate(200,100);
drawRectangle(ctx);
```

Sposób 2

Zerujemy aktualną macierz przy użyciu setTransfrorm(1,0,0,1,0,0). Aktualna macierz staje się macierzą zerową.

Ustawiamy przekrzywienie przy użyciu transform(1,hy,hx,1,0,0). Wynikiem jest pomnożenie macierzy jednostkowej przez macierz przekoszenia, co w rezultacie daje przekrzywienie, czyli ten sam wynik co w sposobie 2.



Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var leftX = 10;
var topY = 10;
var width = 150;
var height = 100;
var cornerRadius = 15;
drawRectangle(ctx);
ctx.setTransform(1,0,0,1,0,0);
ctx.transform(1, 0, -0.5, 1, 0, 0);
ctx.translate(220, 100);
drawRectangle(ctx);
```

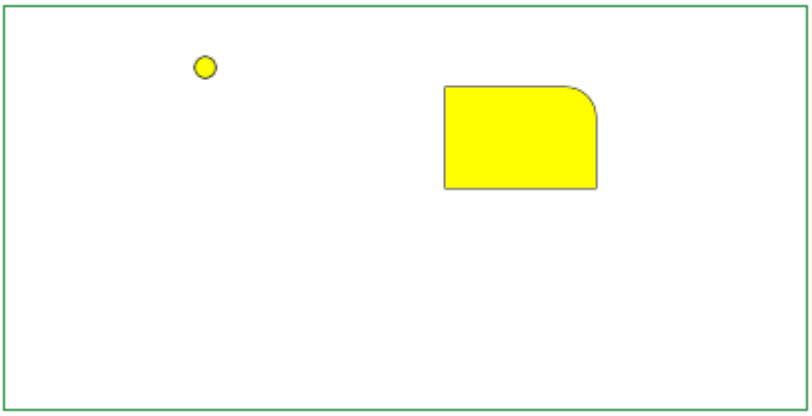
Składanie przekształceń

Do zamiany macierzy używamy setTransform();

Do składania przekształceń używamy `transform(a,b,c,d,e,f)` albo metod specyficznych: `translate()`, `scale()`, `rotate()`.

Obrót wokół dowolnego punktu

Mamy figurę, którą chcemy obrócić wokół punktu (100,30) o 30°

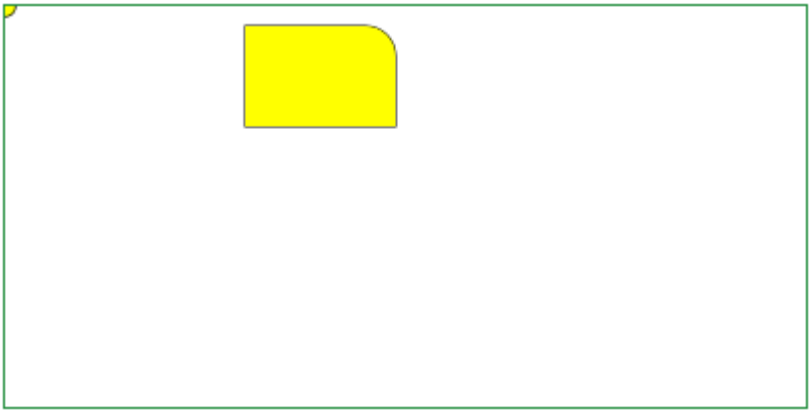


Listing

```
var cv = document.getElementById("canvas1");
var ctx = cv.getContext("2d");
var leftX = 220;
var topY = 40;
var width = 75;
var height = 50;
var cornerRadius = 15;
ctx.beginPath();
ctx.arc(100, 30, 5, 2 * Math.PI, 0, false);
ctx.strokeStyle = "black";
ctx.stroke();
ctx.fillStyle = "green";
ctx.fill();
drawRectangle(ctx);
```

Jeżeli chcemy obrócić figurę względem punktu, który nie jest (0,0) należy:

wykonać translację do punktu (0,0)



Listing

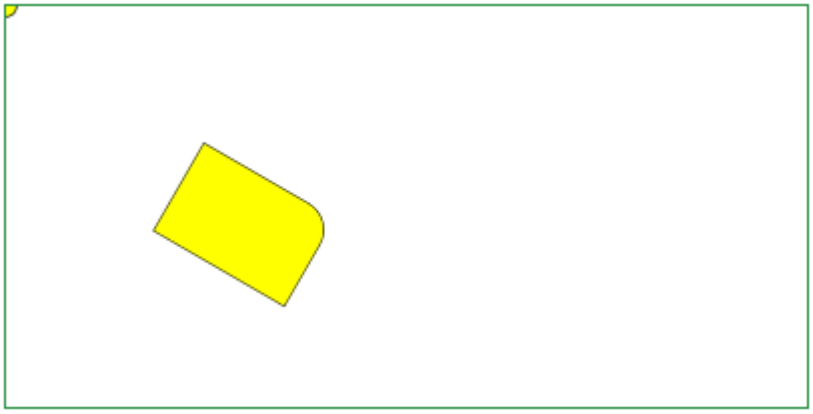
```
var cv = document.getElementById("canvas2");
var ctx = cv.getContext("2d");
var leftX = 220;
var topY = 40;
var width = 75;
var height = 50;
var cornerRadius = 15;
ctx.save();
ctx.beginPath();
ctx.translate(-100, -30);
ctx.arc(100, 30, 5, 2 * Math.PI, 0, false);
ctx.strokeStyle = "black";
```

```
ctx.stroke();
ctx.fillStyle = "green";
ctx.fill();
ctx.restore();
ctx.translate(-100, -30);
drawRectangle(ctx);
```

obrócić figurę o podany kąt

Zwróć uwagę, że w skrypcie transformacje są w odwrotnej kolejności (ze względu na to, że są to operacje na macierzach):

```
ctx.rotate(Math.PI/6);
ctx.translate(-100,-30);
```



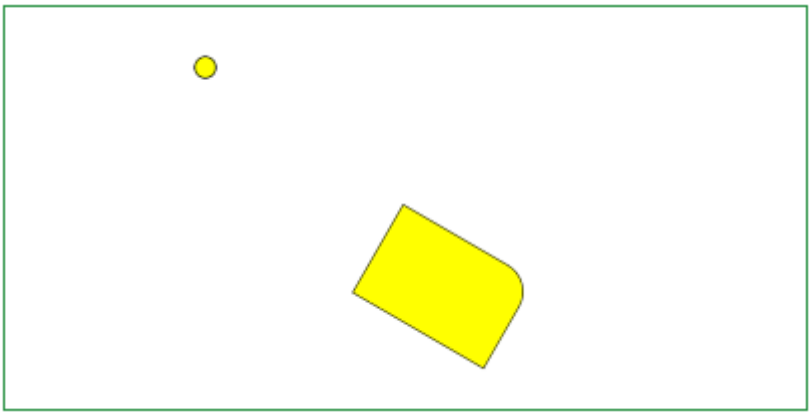
Listing

```
var cv = document.getElementById("canvas3");
var ctx = cv.getContext("2d");
var leftX = 220;
var topY = 40;
var width = 75;
var height = 50;
var cornerRadius = 15;
ctx.save();
ctx.beginPath();
ctx.translate(-100, -30);
ctx.arc(100, 30, 5, 2 * Math.PI, 0, false);
ctx.strokeStyle = "black";
ctx.stroke();
ctx.fillStyle = "green";
ctx.fill();
ctx.restore();
ctx.rotate(Math.PI / 6);
ctx.translate(-100, -30);
drawRectangle(ctx);
```

przesunąć figurę z powrotem (translacja zwrotna)

Zwróć uwagę, że w skrypcie transformacje są w odwrotnej kolejności (ze względu na to, że są to operacje na macierzach):

```
ctx.translate(100,30);
ctx.rotate(Math.PI/6);
ctx.translate(-100,-30);
```

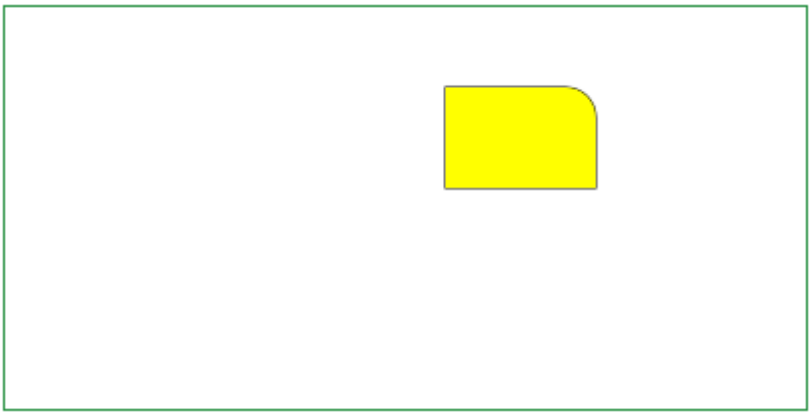


Listing

```
var cv = document.getElementById("canvas4");
var ctx = cv.getContext("2d");
var leftX = 220;
var topY = 40;
var width = 75;
var height = 50;
var cornerRadius = 15;
ctx.save();
ctx.beginPath();
ctx.arc(100, 30, 5, 2 * Math.PI, 0, false);
ctx.strokeStyle = "black";
ctx.stroke();
ctx.fillStyle = "green";
ctx.fill();
ctx.restore();
ctx.translate(100, 30);
ctx.rotate(Math.PI / 6);
ctx.translate(-100, -30);
drawRectangle(ctx);
```

Obrót w miejscu

Mamy figurę, którą chcemy obrócić w miejscu o 30°



Listing

```
var cv = document.getElementById("canvas1");
var ctx = cv.getContext("2d");
var leftX = 220;
var topY = 40;
var width = 75;
```

```
var height = 50;
var cornerRadius = 15;
drawRectangle(ctx);
```

Jeżeli chcemy obrócić figurę względem środka ciężkości, należy:

wykonać translację środka ciężkości do punktu (0,0)

Listing

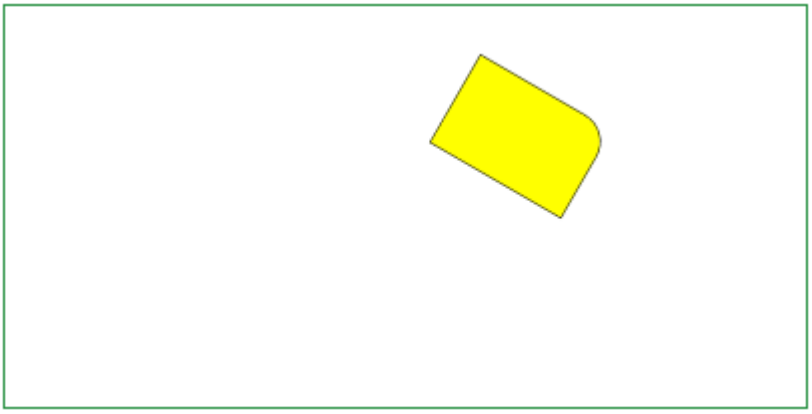
```
var cv = document.getElementById("canvas2");
var ctx = cv.getContext("2d");
var leftX = 220;
var topY = 40;
var width = 75;
var height = 50;
var cornerRadius = 15;
ctx.save();
ctx.translate(-(leftX+width/2), -(topY+height/2));
drawRectangle(ctx);
```

obrócić figurę o podany kąt

Listing

```
var cv = document.getElementById("canvas3");
var ctx = cv.getContext("2d");
var leftX = 220;
var topY = 40;
var width = 75;
var height = 50;
var cornerRadius = 15;
ctx.rotate(Math.PI / 6);
ctx.translate(-(leftX+width/2), -(topY+height/2));
drawRectangle(ctx);
```

przesunąć figurę z powrotem (translacja zwrotna)

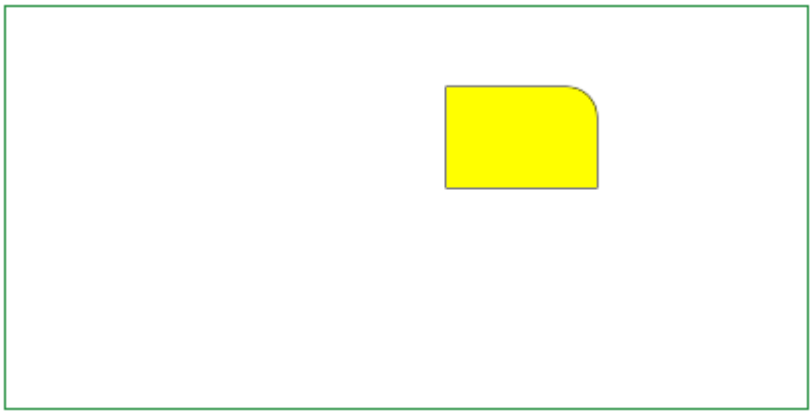


Listing

```
var cv = document.getElementById("canvas4");
var ctx = cv.getContext("2d");
var leftX = 220;
var topY = 40;
var width = 75;
var height = 50;
var cornerRadius = 15;
ctx.translate(leftX+width/2, topY+height/2);
ctx.rotate(Math.PI / 6);
ctx.translate(-(leftX+width/2), -(topY+height/2));
drawRectangle(ctx);
```

Skalowanie w miejscu

Mamy figurę, którą chcemy skalować w miejscu o współczynnik 0.5



Listing

```
var cv = document.getElementById("canvas1");
var ctx = cv.getContext("2d");
var leftX = 220;
var topY = 40;
var width = 75;
var height = 50;
var cornerRadius = 15;
drawRectangle(ctx);
```

Jeżeli chcemy saklować figurę względem środka ciężkości, należy:

wykonać translację środka ciężkości do punktu (0,0)

Listing

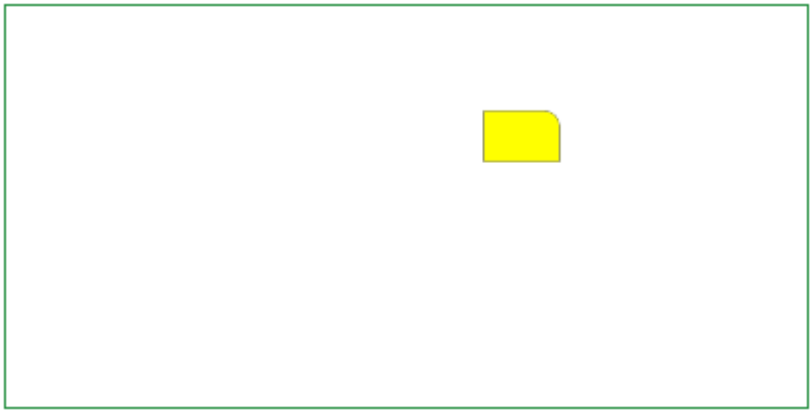
```
var cv = document.getElementById("canvas2");
var ctx = cv.getContext("2d");
var leftX = 220;
var topY = 40;
var width = 75;
var height = 50;
var cornerRadius = 15;
ctx.save();
ctx.translate(-(leftX+width/2), -(topY+height/2));
drawRectangle(ctx);
```

wykonać żądane skalowanie

Listing

```
var cv = document.getElementById("canvas3");
var ctx = cv.getContext("2d");
var leftX = 220;
var topY = 40;
var width = 75;
var height = 50;
var cornerRadius = 15;
ctx.scale(0.5,0.5);
ctx.translate(-(leftX+width/2), -(topY+height/2));
drawRectangle(ctx);
```

przesunąć figurę z powrotem (translacja zwrotna)

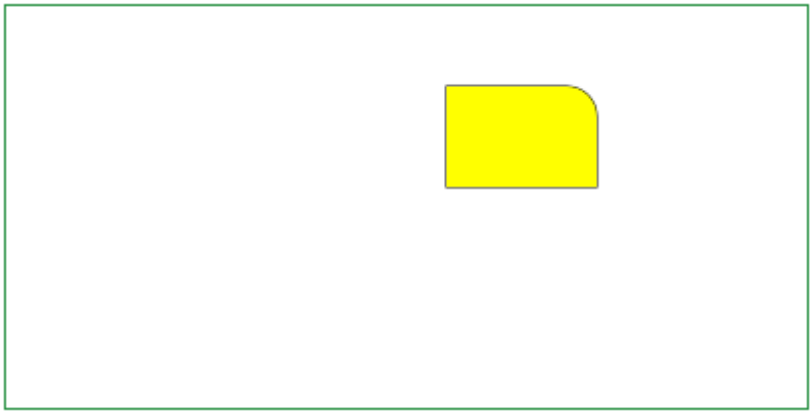


Listing

```
var cv = document.getElementById("canvas4");
var ctx = cv.getContext("2d");
var leftX = 220;
var topY = 40;
var width = 75;
var height = 50;
var cornerRadius = 15;
ctx.translate(leftX+width/2, topY+height/2);
ctx.scale(0.5, 0.5);
ctx.translate(-(leftX+width/2), -(topY+height/2));
drawRectangle(ctx);
```

Przekrzywienie w miejscu

Mamy figurę, którą chcemy przekrzywić w osi X o współczynnik 0.4



Listing

```
var cv = document.getElementById("canvas1");
var ctx = cv.getContext("2d");
var leftX = 220;
var topY = 40;
var width = 75;
var height = 50;
var cornerRadius = 15;
drawRectangle(ctx);
```

Jeżeli chcemy saklować figurę względem środka ciężkości, należy:

wykonać translację środka ciężkości do punktu (0,0)

Listing

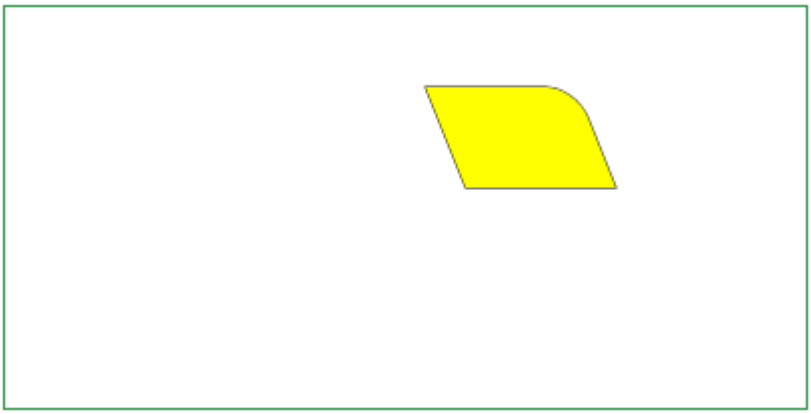
```
var cv = document.getElementById("canvas2");
var ctx = cv.getContext("2d");
var leftX = 220;
var topY = 40;
var width = 75;
var height = 50;
var cornerRadius = 15;
ctx.save();
ctx.translate(-(leftX+width/2), -(topY+height/2));
drawRectangle(ctx);
```

wykonać żądane przekrzywienie

Listing

```
var cv = document.getElementById("canvas3");
var ctx = cv.getContext("2d");
var leftX = 220;
var topY = 40;
var width = 75;
var height = 50;
var cornerRadius = 15;
ctx.transform(1,0,0.4,1,0,0);
ctx.translate(-(leftX+width/2), -(topY+height/2));
drawRectangle(ctx);
```

przesunąć figurę z powrotem (translacja zwrotna)

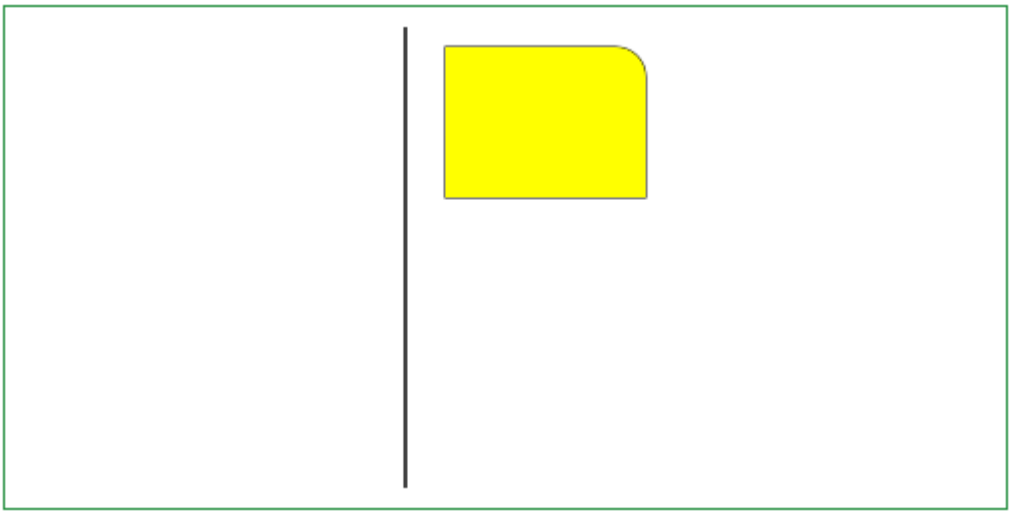


Listing

```
var cv = document.getElementById("canvas4");
var ctx = cv.getContext("2d");
var leftX = 220;
var topY = 40;
var width = 75;
var height = 50;
var cornerRadius = 15;
ctx.translate(leftX+width/2, topY+height/2);
ctx.transform(1,0,0.4,1,0,0);
ctx.translate(-(leftX+width/2), -(topY+height/2));
drawRectangle(ctx);
```

Odbicie względem prostej nie przechodzącej przez punkt (0,0)

Przypadek 1. Prosta jest równoległa do osi Y i nie może być przedstawiona równaniem y=mx+b

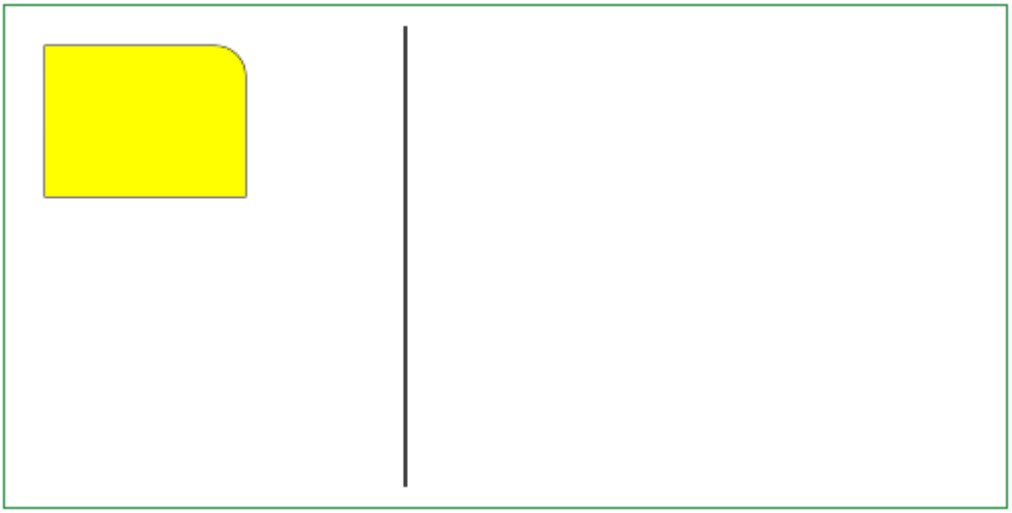


Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
ctx.beginPath();
ctx.moveTo(200,10);
ctx.lineTo(200,240);
```

```
ctx.strokeStyle = "black";
ctx.stroke();
var leftX = 220;
var topY = 20;
var width = 100;
var height = 75;
var cornerRadius = 15;
drawRectangle(ctx);
```

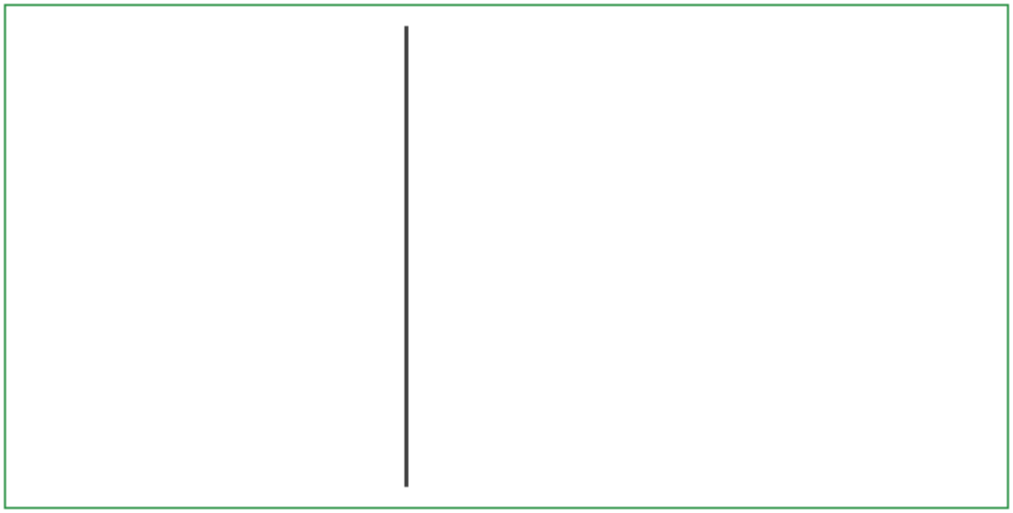
dokonujemy translacji o odległość prostej od osi Y



Listing

```
var cv = document.getElementById("canvas1");
var ctx = cv.getContext("2d");
ctx.save();
ctx.beginPath();
ctx.moveTo(200,10);
ctx.lineTo(200,240);
ctx.strokeStyle = "black";
ctx.stroke();
ctx.restore();
var leftX = 220;
var topY = 20;
var width = 100;
var height = 75;
var cornerRadius = 15;
ctx.translate(-200,0);
drawRectangle(ctx);
```

wykonujemy odbicie figury względem osi Y

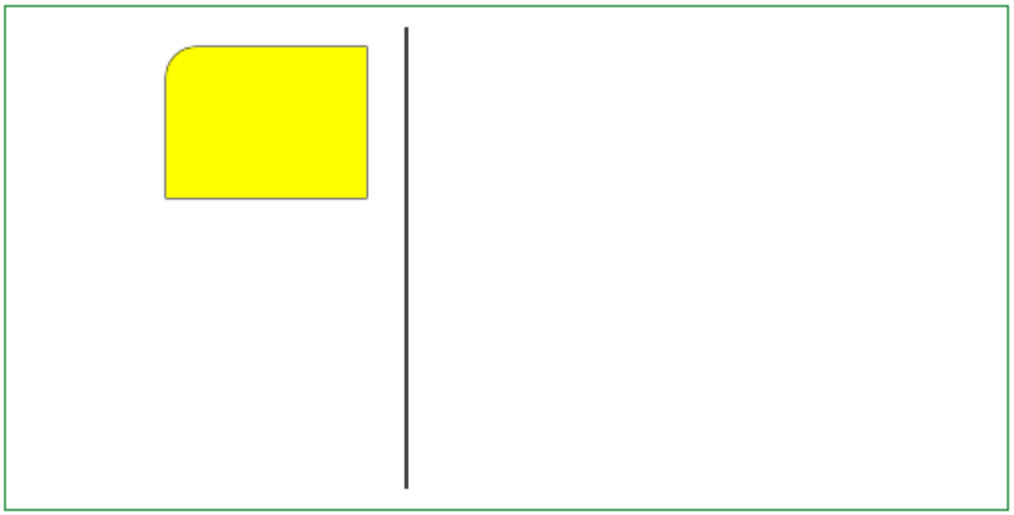


Figury chwilowo nie widzimy.

Listing

```
var cv = document.getElementById("canvas2");
var ctx = cv.getContext("2d");
ctx.save();
ctx.beginPath();
ctx.moveTo(200,10);
ctx.lineTo(200,240);
ctx.strokeStyle = "black";
ctx.stroke();
ctx.restore()
var leftX = 220;
var topY = 20;
var width = 100;
var height = 75;
var cornerRadius = 15;
ctx.scale(-1,1);
ctx.translate(-200,0);
drawRectangle(ctx);
```

wykonujemy translację powrotną



Listing

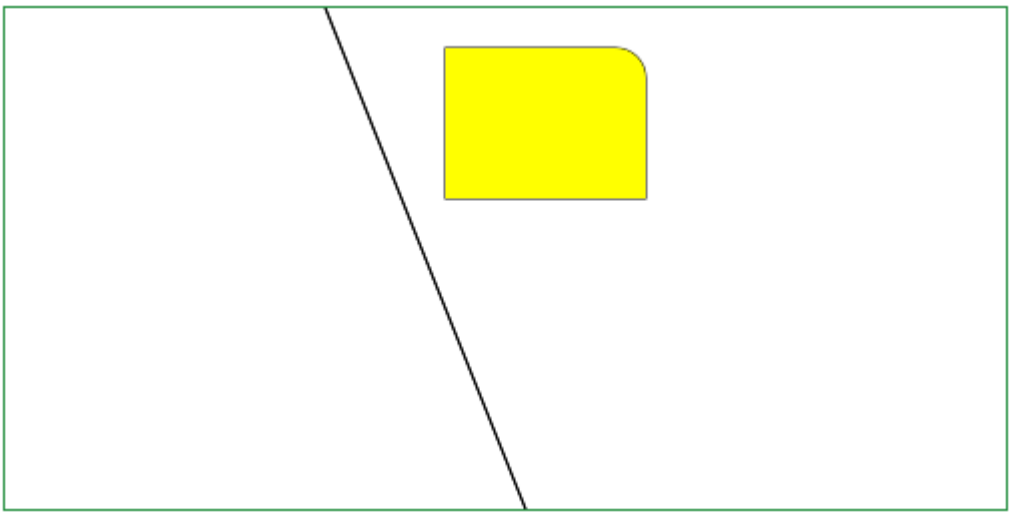
```
var cv = document.getElementById("canvas3");
var ctx = cv.getContext("2d");
ctx.save();
ctx.beginPath();
ctx.moveTo(200,10);
ctx.lineTo(200,240);
ctx.strokeStyle = "black";
ctx.stroke();
ctx.restore();
var leftX = 220;
var topY = 20;
var width = 100;
var height = 75;
var cornerRadius = 15;
ctx.translate(200,0);
ctx.scale(-1,1);
ctx.translate(-200,0);
drawRectangle(ctx);
```

Przypadek 2.Prosta nie jest równoległa do osi Y i może być przedstawiona równaniem y=mx+b

Prosta przechodzi przez dwa punkty (160,0) i (260,250).

tgφ=250/100=2,5=m;

Równanie prostej to y=2.5x-400, czyli *b*=-400

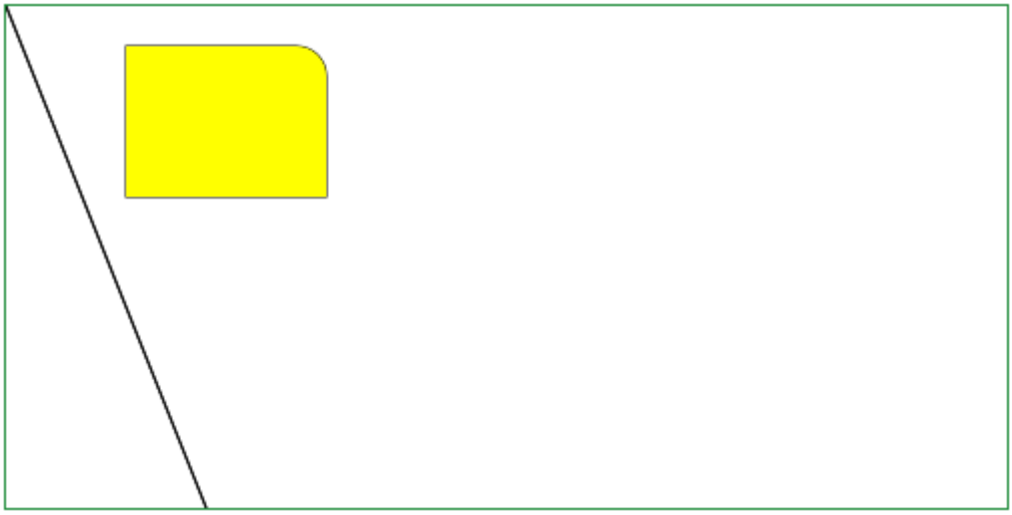


Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
ctx.beginPath();
ctx.moveTo(160,0);
ctx.lineTo(260,250);
ctx.strokeStyle = "black";
ctx.stroke();
var leftX = 220;
var topY = 20;
var width = 100;
var height = 75;
var cornerRadius = 15;
```

```
drawRectangle(ctx);
```

translacja figury o współczynnik *b/m=-400/2.5=-160*



Prosta przechodzi teraz przez punkt (0,0).

Listing

```
var cv = document.getElementById("canvas1");
var ctx = cv.getContext("2d");
ctx.save();
ctx.beginPath();
ctx.translate(-160,0);
ctx.moveTo(160,0);
ctx.lineTo(260,250);
ctx.strokeStyle = "black";
ctx.stroke();
ctx.restore();
var leftX = 220;
var topY = 20;
var width = 100;
var height = 75;
var cornerRadius = 15;
ctx.translate(-160,0);
drawRectangle(ctx);
```

odbicie figury względem przesuniętej linii

Macierz odbicia wygląda tak:

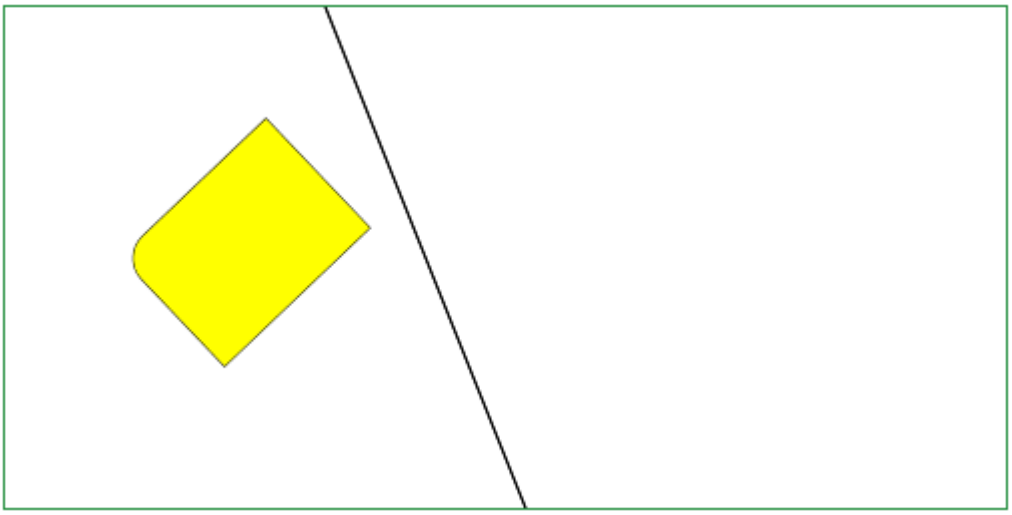
$$\begin{bmatrix} \cos 2\varphi & \sin 2\varphi & 0 \\ \sin 2\varphi & -\cos 2\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

gdzie φ jest kątem pod jakim prosta przecina oś X.

Listing

```
var cv = document.getElementById("canvas2");
var ctx = cv.getContext("2d");
ctx.save();
ctx.beginPath();
ctx.translate(-160,0);
ctx.moveTo(160,0);
ctx.lineTo(260,250);
ctx.strokeStyle = "black";
ctx.stroke();
ctx.restore();
var leftX = 220;
var topY = 20;
var width = 100;
var height = 75;
var cornerRadius = 15;
ctx.transform(Math.cos(2*Math.atan(2.5)),Math.sin(2*Math.atan(2.5)),
      Math.sin(2*Math.atan(2.5)),-Math.cos(2*Math.atan(2.5)),0,0);
ctx.translate(-160,0);
drawRectangle(ctx);
```

translacja powrotna



Listing

```
var cv = document.getElementById("canvas3");
var ctx = cv.getContext("2d");
```

```
ctx.save();
ctx.beginPath();
ctx.moveTo(160,0);
ctx.lineTo(260,250);
ctx.strokeStyle = "black";
ctx.stroke();
ctx.restore();
var leftX = 220;
var topY = 20;
var width = 100;
var height = 75;
var cornerRadius = 15;
ctx.translate(160,0);
ctx.transform(Math.cos(2*Math.atan(2.5)),Math.sin(2*Math.atan(2.5)),
      Math.sin(2*Math.atan(2.5)),-Math.cos(2*Math.atan(2.5)),0,0);
ctx.translate(-160,0);
drawRectangle(ctx);
```

Krzywe Béziera

Jeśli chcesz zapoznać się z krzywymi Béziera zajrzyj do Dodatku 6

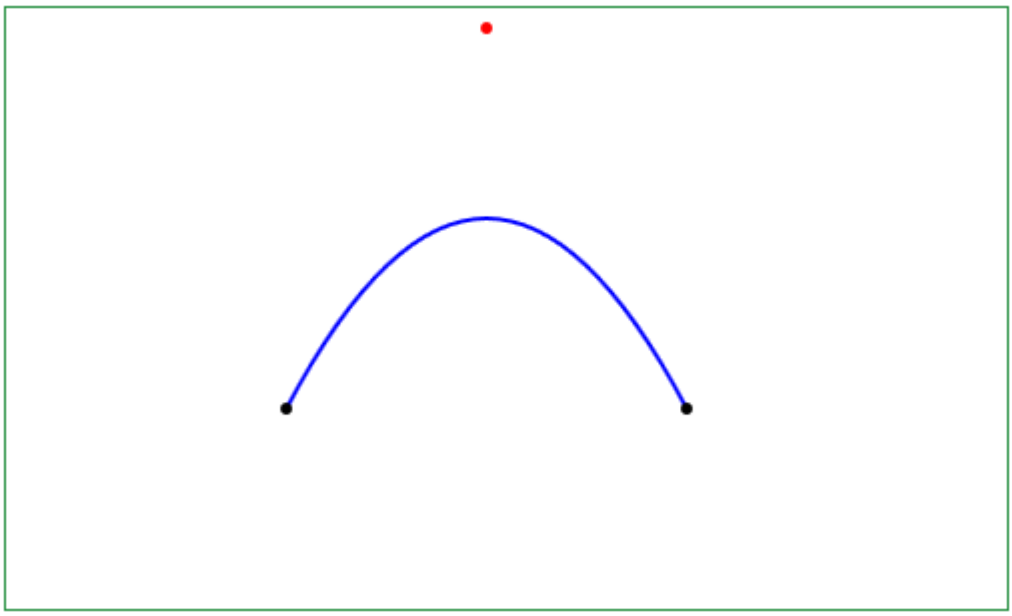
Krzywe Béziera dzielą się w canvas JavaScript na krzywe kwadratowe i krzywe sześcienne.

Kwadratowa krzywa Béziera

Aby narysować kwadratową krzywą Béziera musimy podać współrzędne trzech punktów:

- współrzędne punktu początkowego krzywej
- współrzędne punktu kontrolnego
- współrzędne punktu końcowego krzywej

A oto krzywa rozciągająca się między punktami (140,200), (340, 200) z punktem kontrolnym w pozycji (240,10).



Listing

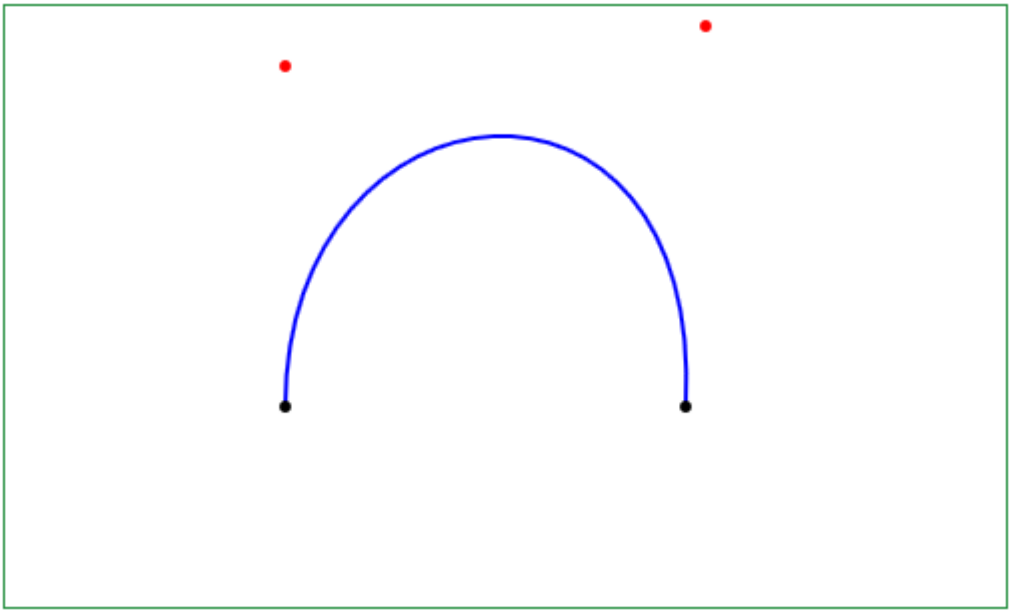
```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
ctx.beginPath();
ctx.moveTo(140, 200);
ctx.quadraticCurveTo(240, 10, 340, 200);
ctx.lineWidth = 2;
ctx.strokeStyle = 'blue';
ctx.stroke();
ctx.beginPath();
fillStyle="blue";
ctx.arc(140, 200, 3, 2*Math.PI, 0, false);
ctx.fill();
ctx.beginPath();
fillStyle="blue";
ctx.arc(340, 200, 3, 2*Math.PI, 0, false);
ctx.fill();
ctx.beginPath();
ctx.fillStyle="red";
ctx.arc(240, 10, 3, 2*Math.PI, 0, false);
ctx.fill();
```

Sześcienna krzywa Béziera

Aby narysować sześcienną krzywą Béziera musimy podać współrzędne czterech punktów:

- współrzędne punktu początkowego krzywej
- współrzędne punktu pierwszego kontrolnego
- współrzędne drugiego punktu kontrolnego
- współrzędne punktu końcowego krzywej

A oto krzywa rozciągająca się między punktami (140,200), (340, 200) z punktami kontrolnym w pozycji (140,30) i (350,10).



Listing

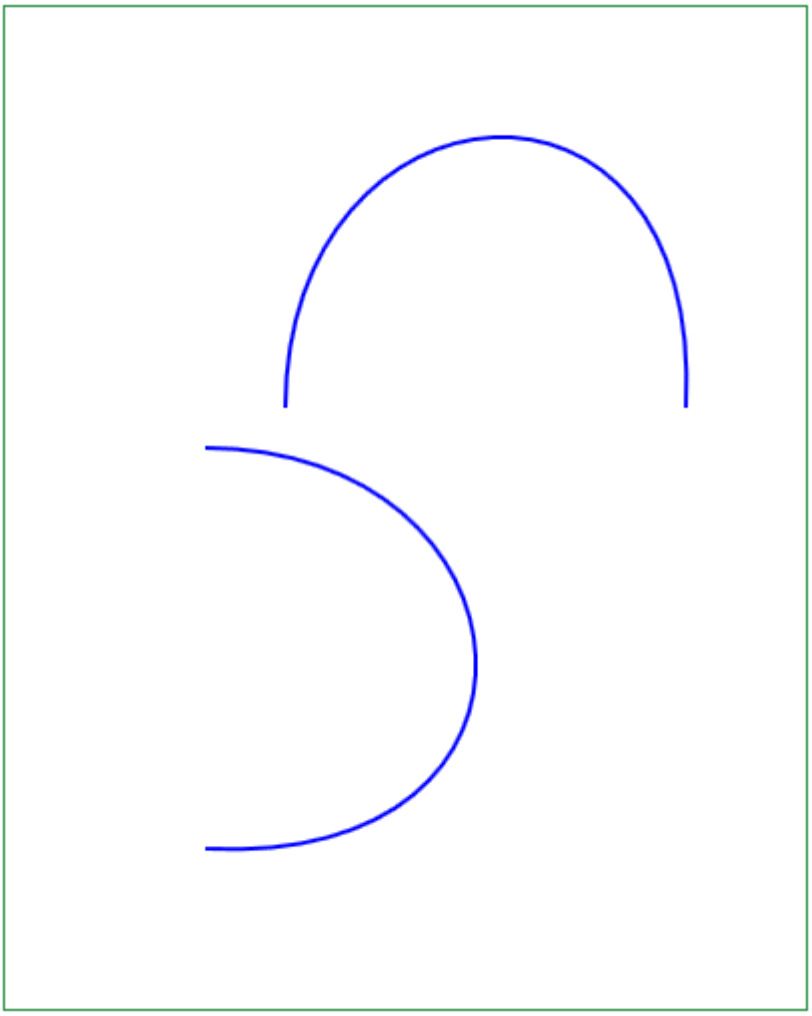
```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
```

```
ctx.beginPath();
ctx.moveTo(140, 200);
ctx.bezierCurveTo(140, 30, 350, 10, 340, 200);
ctx.lineWidth = 2;
ctx.strokeStyle = 'blue';
ctx.stroke();
ctx.beginPath();
fillStyle="blue";
ctx.arc(140, 200, 3, 2*Math.PI, 0, false);
ctx.fill();
ctx.beginPath();
fillStyle="blue";
ctx.arc(340, 200, 3, 2*Math.PI, 0, false);
ctx.fill();
ctx.beginPath();
ctx.fillStyle="red";
ctx.arc(140, 30, 3, 2*Math.PI, 0, false);
ctx.fill();
ctx.beginPath();
ctx.fillStyle="red";
ctx.arc(350, 10, 3, 2*Math.PI, 0, false);
ctx.fill();
```

Transformacja krzywych Béziera

W canvas transformowane są cały kontekst lub ścieżka. Z matematycznego punktu widzenia wystarczy transformacja każdego z trzech lub czterech punktów krzywej i ponowne jej odrysowanie przy uwzględnieniu nowych współrzędnych!

A oto krzywa rozciągająca się między punktami (140,200), (340, 200) z punktami kontrolnym w pozycji (140,30) i (350,10), a następnie transformowana o 90° z zastosowaniem odpowiedniej translacji.



Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
ctx.beginPath();
ctx.moveTo(140, 200);
ctx.bezierCurveTo(140, 30, 350, 10, 340, 200);
ctx.lineWidth = 2;
ctx.strokeStyle = 'blue';
ctx.stroke();
ctx.translate(300, 80);
ctx.rotate(Math.PI / 2);
ctx.beginPath();
ctx.moveTo(140, 200);
ctx.bezierCurveTo(140, 30, 350, 10, 340, 200);
ctx.lineWidth = 2;
ctx.strokeStyle = 'blue';
ctx.stroke();
```

Kolory CSS

Liczby heksadecymalne

(Listing01 w Dodatku 7)

Przyjrzyjmy się tabelce podającej liczby w różnych systemach liczenia:

Liczba dziesiętna	**Liczba binarna**	**Liczba heksadecymalna**
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Weźmy liczbę 1-bajtową (8-bitową), np.

10011011

Każdy bajt możemy zapisać jako parę liczb heksadecymalnych

1001 = 9,

1011 = B

10001011 = 9B

Liczby 8 bitowe pozwalają na zapisanie 256 wartości, czyli wartości od 0 do 255. Jest to bardzo wygodne do zapisania wartości pojedynczego składnika koloru, np. red = 15 = 0F. Dla odróżnienia od innych liczb, liczby heksadecymalne zapisujemy z 0x albo 0X na początku, czyli nasza liczba to 0x0F.

Liczbę 2-bajtową (16-bitową) możemy zapisać w postaci 2 par liczb heksadecymalnych.

Liczbę 3-bajtową (24-bitową) możemy zapisać w postaci 3 par liczb heksadecymalnych. Jest to wygodne do zapisywania koloru w postaci rgb (red, green, blue).

Liczbę 4-bajtową (32-bitową) możemy zapisać w postaci 4 par liczb heksadecymalnych. Jest to wygodne do zapisywania kolorów w systemie rgba (red, green, blue, alfa), czyli z kanałem przezroczystości.

Ponieważ liczba 32-bitowa to liczba typu 'integer' to kolor rgba możemy zapisać również w postaci jednej liczby typu 'integer'.

Przeliczanie liczb dziesiętnych na liczby heksadecymalne:

```
var wzrost = parseInt(175);
var bwzrost = wzrost.toString(16);
```

Przeliczenie z liczby heksadecymalnej na dziesiętną możemy wykonać:

```
var iwzrost = parseInt(bwzrost, 16);
```

W obu przypadkach 16 to podstawa, gdzie można oczywiście użyć innej liczby, np. 2.

Model RGB

System RGB przedstawia kolor w postaci trzech liczb naturalnych lub jednej liczby heksadecymalnej.

System RGB był stworzony pod kątem łatwości obliczania kolorów do wyświetlania na ekranach telewizorów i monitorów.

Jest powszechnie używany na stronach internetowych i w grafice internetowej.

R albo r oznacza 'red' czyli składnik czerwony.

G albo g oznacza 'green', czyli składnik zielony.

B albo b oznacza 'blue' czyli składnik niebieski.

Każdy z powyższych trzech kolorów jest jedną z 256 liczb od 0 do 255.

W systemie RGB można przedstawić 256 x 256 x 256 = 16 777 216 kolorów.

Kolor RGB możemy przedstawić w postaci:

- liczb dziesiętnych `rgb(255, 0, 0)`
- procentów `rgb(100%, 0%, 0%)`;
- liczb zmiennoprzecinkowcych `rgb(1.0, 0, 0)`, każda od 0.00 do 1.0
- liczby heksadecymalnej "color: #FF0000";

Liczba heksadecymalna

6 cyfr

Standardowy zapis trzech kolorów. Kanał alfa α=1.0:

```
>#FF0000
```

8 cyfr

Ostatnie 2 cyfry oznaczają kanał alfa:

#FF0000CC oznacza rgba(255, 0, 0, 0.8)

Trzy cyfry

Zapis #FC0 oznacza, że każdy z kanałów został podzielony na 16 wartości od 0 do F, gdzie 0 oznacza minimum, a F oznacza maksimum.

Czyli mamy 16/16, 12/16, 0/16

4 cyfry

Identycznie jak dla zapisu 3-cyfrowego, czwarta cyfra oznacza kanał alfa:

#FC02 oznacza 16/16, 12/16, 0/16, 2/16

Przeliczanie int na hex

Listing

```
/*
 * funkcja zmieniajaca wartosci RGB na licybe heksdeczmalna. Liczby musza być
 * podane jako inty od 0 do 255 włącznie
 */
function RGBToHex(r, g, b) {
	var r1 = r.toString(16).toUpperCase();
	var g1 = g.toString(16).toUpperCase();
	var b1 = b.toString(16).toUpperCase();
	if (r1.length == 1) {
		r1 = "0" + r1;
	}
	if (g1.length == 1) {
		g1 = "0" + g1;
	}
	if (b1.length == 1) {
		b1 = "0" + b1;
	}
	return "0x" + r1 + g1 + b1;
};
```

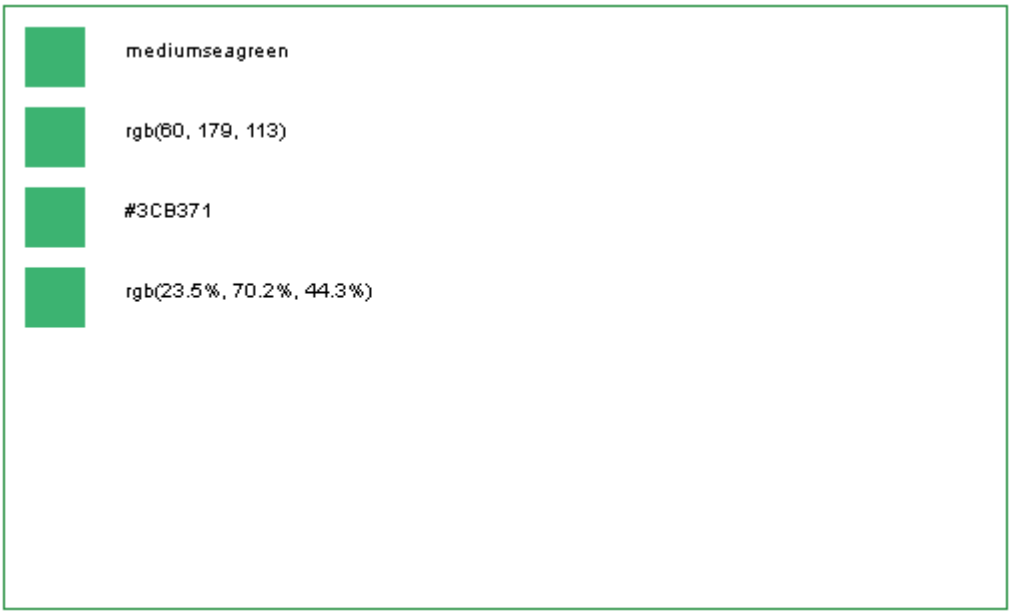
Przeliczanie hex na int

Listing

```
/* funkcja zmieniajaca liczbe heksadecymalna na wartosci RGB
 * Liczba może być podana jako 0x np "0xFFFFFF" lub "#FFFFFF"
 * lub "FFFFFF"
 */
function HexToRGB(hexa) {
	var R = -1;
	var G = -1;
	var B = -1;
	if (((hexa.indexOf("x") == -1) && (hexa.indexOf("#") == -1))) {
		R = parseInt(hexa.substr(0, 2), 16);
		G = parseInt(hexa.substr(2, 2), 16);
		B = parseInt(hexa.substr(4, 2), 16);
	} else {
		if (hexa.indexOf("#") > 0) {
```

```
                    R = parseInt(hexa.substr(1, 2), 16);
                    G = parseInt(hexa.substr(3, 2), 16);
                    B = parseInt(hexa.substr(5, 2), 16);
              } else if (hexa.indexOf("x") > 0) {
                    R = parseInt(hexa.substr(2, 2), 16);
                    G = parseInt(hexa.substr(4, 2), 16);
                    B = parseInt(hexa.substr(6, 2), 16);
              }
       }
       return [ R, G, B ];
};
```

Sposoby zapisania tego samego koloru



Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
ctx.beginPath();
ctx.fillStyle = "mediumseagreen";
ctx.fillRect(10, 10, 30, 30);
ctx.fillStyle = "black";
ctx.fillText("mediumseagreen", 60, 25);
ctx.beginPath();
ctx.fillStyle = "rgb(60, 179, 113)";
ctx.fillRect(10, 50, 30, 30);
ctx.fillStyle = "black";
ctx.fillText("rgb(60, 179, 113)", 60, 65);
ctx.beginPath();
ctx.fillStyle = "#3CB371";
ctx.fillRect(10, 90, 30, 30);
ctx.fillStyle = "black";
ctx.fillText("#3CB371", 60, 105);
ctx.beginPath();
ctx.fillStyle = "rgb(23.5%, 70.2%, 44.3%)";
ctx.fillRect(10, 130, 30, 30);
ctx.fillStyle = "black";
ctx.fillText("rgb(23.5%, 70.2%, 44.3%)", 60, 145);
ctx.beginPath();
```

Palety kolorów

Dostępnych jest wiele palet kolorów RGB:

- Paleta 16 kolorów nazwanych
- Paleta 216 kolorów bezpiecznych dla Internetu
- Rozszerzona paleta kolorów nazwanych
- Kolory z polskimi nazwami

Palety przedstawione są w Dodatku 8.

Model RGBA

A albo a dodaje do RGB czwarty składnik 'alfa' określający transparentność (przezroczystość, stopień krycia) koloru.

W JavaScript A jest wyrażane jako liczba zmiennoprzecinkowa od 0.0 - 1.0.

Przezroczystość definiujemy dla ścieżki kontekstu jako `globalAlpha` .

Jeżeli liczba A = 1.0 to kolor RGBA jest identyczny z kolorem RGB.

Kolory RGBA można przedstawić podobnie jak kolory RGB, ale kolor alfa jest wyrażony w postaci liczby zmiennoprzecinkowej od 0.0 do 1.0:

```
rgba(255, 0, 0, 0.5)
```

Liczba 0 oznacza przezroczystość zupełną (brak krycia), a liczba 1 - pełną nieprzezroczystość (krycie pełne).

Przeliczanie HEX na RGBA

Listing

```
// zwraca tablice R,G,B,A gdzie A jest liczbą float pomiedzy 0.0 - 1.0
var HexToRGBA = function(hexa) {
	var R = -1;
	var G = -1;
	var B = -1;
	var A = -1;
	if (((hexa.indexOf("x") == -1) && (hexa.indexOf("#") == -1))) {
		R = parseInt(hexa.substr(0, 2), 16);
		G = parseInt(hexa.substr(2, 2), 16);
		B = parseInt(hexa.substr(4, 2), 16);
		A = parseInt(hexa.substr(6, 2), 16);
	} else {
		if (hexa.indexOf("#") > 0) {
			R = parseInt(hexa.substr(1, 2), 16);
			G = parseInt(hexa.substr(3, 2), 16);
			B = parseInt(hexa.substr(5, 2), 16);
			A = parseInt(hexa.substr(7, 2), 16);
		} else if (hexa.indexOf("x") > 0) {
			R = parseInt(hexa.substr(2, 2), 16);
			G = parseInt(hexa.substr(4, 2), 16);
			B = parseInt(hexa.substr(6, 2), 16);
			A = parseInt(hexa.substr(8, 2), 16);
		}
	}
	return [ R, G, B, Math.roundToDecimal((A - 0.5) / 255, 4) ];
```

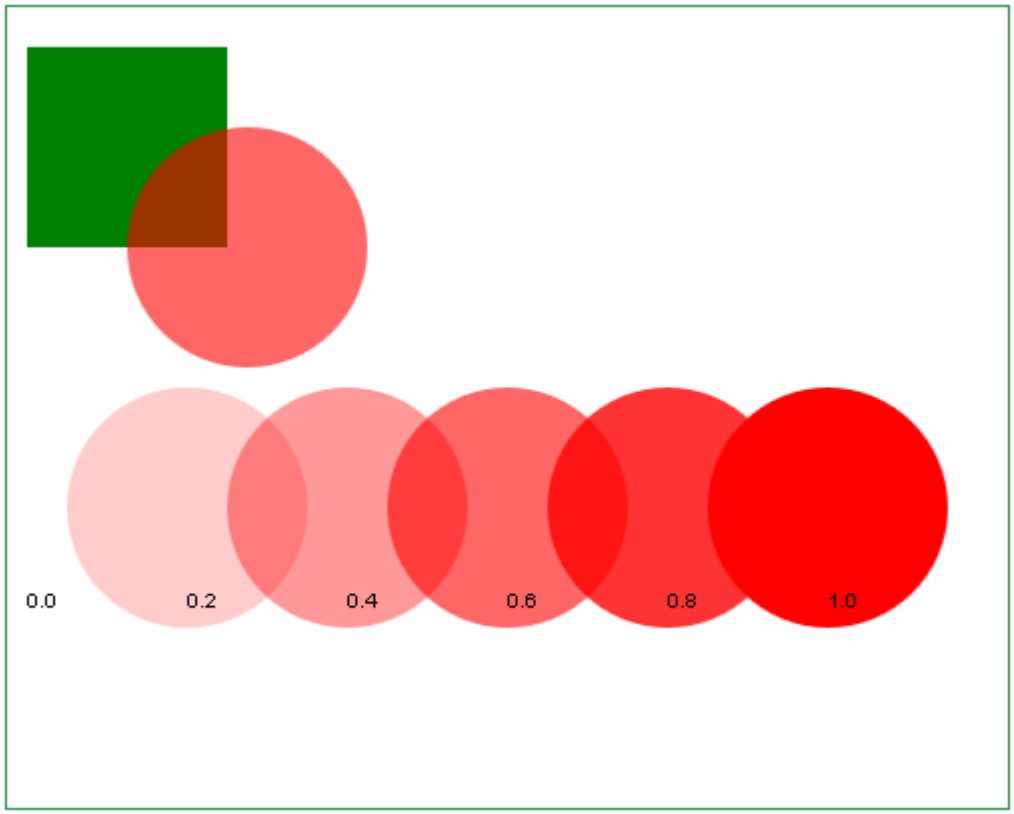
```
};
```

Przeliczanie zRGBA na HEX

Listing

```
// a jest typu float pomiędzy 0.0 - 1.0
var RGBAToHex = function(r, g, b, a) {
	var r1 = parseInt(r).toString(16).toUpperCase();
	var g1 = parseInt(g).toString(16).toUpperCase();
	var b1 = parseInt(b).toString(16).toUpperCase();
	var a1 = parseInt(Math.roundToDecimal(a * 255 + 0.5), 4).toString(16)
			.toUpperCase();
	if (r1.length == 1) {
		r1 = "0" + r1;
	}
	if (g1.length == 1) {
		g1 = "0" + g1;
	}
	if (b1.length == 1) {
		b1 = "0" + b1;
	}
	if (a1.length == 1) {
		a1 = "0" + a1;
	}
	return "0x" + r1 + g1 + b1;
};
```

Różne wartości krycia:



Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
ctx.save();
```

```
ctx.beginPath();
ctx.rect(10, 20, 100, 100);
ctx.fillStyle = "green";
ctx.fill();
ctx.globalAlpha = 0.6;
ctx.beginPath();
ctx.arc(120, 120, 60, 0, 2 * Math.PI, false);
ctx.fillStyle = "red";
ctx.fill();
for (var i = 0; i 7lt; 6; i++) {
      ctx.beginPath();
      ctx.globalAlpha = 0.2 * i;
      ctx.arc(10 + 80 * i, 250, 60, 0, 2 * Math.PI, false);
      ctx.fillStyle = "red";
      ctx.fill();
}
ctx.restore()
ctx.fillStyle = "black";
ctx.fillText("0.0", 10 + 80 * 0, 300);
ctx.fillText("0.2", 10 + 80 * 1, 300);
ctx.fillText("0.4", 10 + 80 * 2, 300);
ctx.fillText("0.6", 10 + 80 * 3, 300);
ctx.fillText("0.8", 10 + 80 * 4, 300);
ctx.fillText("1.0", 10 + 80 * 5, 300);
```

Model HSL i HSLA

HSL to tzw. model kolorów dla artystów. Nie oddaje wszystkich barw widzianych przez ludzkie oko, np. purpury.

H oznacza 'hue' czyli 'odcień', 'barwa' i jest podawany w postaci kąta 0.0 - 360.0^{o}. Barwy zielone mieszczą się w zakresie od 0.0 do 120.0. Barwy niebieskie mieszczą się w przedziale od 120.0 do 240.0, a barwy czerwone od 240.0 do 360.0

S oznacza 'saturation', czyli 'nasycenie' koloru podawane w zakresie 0.0 - 100%

L oznacza 'lightness' czyli 'jasność', podawane w zakresie 0.0 - 100%

HSLA dodaje czwarty parametr - 'krycie' wyrażony w zakresie od 0.0 do 1.0.

W HTML `hue` można podawać w postaci 6 kolorów nazwanych i ich 6 odcieni, ale ten sposób nie działa w `canvas`.

Palety kolorów HSL:

- Paleta nazwanych kolorów HSL
- Paleta kolorów HSL

Palety przedstawione są w Dodatku 8.

Przeliczanie RGB to HSL

Listing

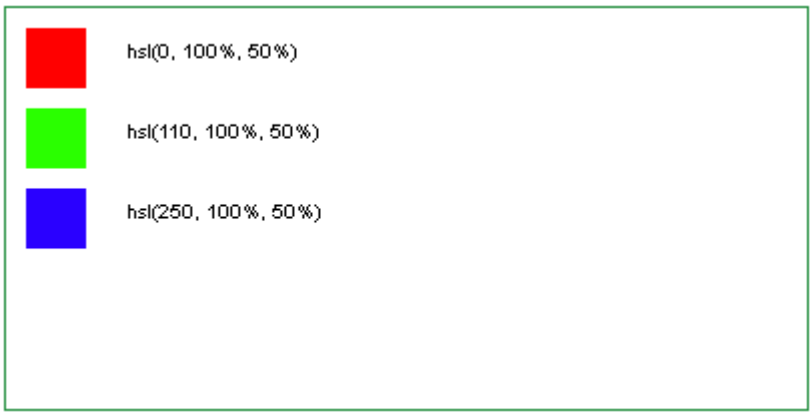
```
//Przyjmuje wartości RGB
//zwraca tablicę HSL (h w stopniach od 0.0 - 360, pozostałe
//w %)
```

```
function RGBToHSL(r, g, b){
    r /= 255, g /= 255, b /= 255;
    var max = Math.max(r, g, b), min = Math.min(r, g, b);
    var sm = max + min;
    var h = sm / 2;
    var s = sm / 2;
    var l = sm / 2;

    if(max == min){
        h = s = 0; // achromatic
    }else{
        var d = max - min;
        s = l > 0.5 ? d / (2 - sm) : d / sm;
        switch(max){
            case r: h = (g - b) / d + (g < b ? 6 : 0); break;
            case g: h = (b - r) / d + 2; break;
            case b: h = (r - g) / d + 4; break;
        }
        h *= 60.0;
    }

    return [roundToDecimal(h,4), roundToDecimal(s,4), roundToDecimal(l,4)];
}
```

Przeliczanie HSL do RGB

Listing

```
//h - barwa w stopniach 0.0 - 360.0
//s - nasycenie w % od 0.0 - 100%
//l - jasność w % od 0.0 - 100%
//zwraca tablice wartości RGB
function HSLToRGB(h, s, l) {
       var r, g, b;
       h /= 360.0;
       if (s == 0) {
              r = g = b = l; // achromatic
       } else {
              var q = l 7lt; 0.5 ? l * (1 + s) : l + s - l * s;
              var p = 2 * l - q;
              r = HueToRGB(p, q, h + 1.0 / 3.0);
              g = HueToRGB(p, q, h);
              b = HueToRGB(p, q, h - 1.0 / 3.0);
       }

       return [ roundToDecimal(r * 255, 0), roundToDecimal(g * 255, 0),
                     roundToDecimal(b * 255, 0) ];
}
//funkcja pomocnicza do obliczania nasycenia
function HueToRGB(a, b, c) {
       if (c < 0)
              c += 1;
       if (c > 1)
              c -= 1;
       if (c < 1.0 / 6.0)
              return a + (b - a) * 6 * c;
       if (c < 1.0 / 2.0)
              return b;
       if (c < 2.0 / 3.0)
              return a + (b - a) * (2. / 3.0 - c) * 6;
       return a;
}
```



Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
ctx.beginPath();
ctx.fillStyle = "hsl(0,100%,50%)";
ctx.fillRect(10, 10, 30, 30);
ctx.fillStyle = "black";
ctx.fillText("hsl(0, 100%, 50%)", 60, 25);
ctx.beginPath();
ctx.fillStyle = "hsl(110,100%,50%)";
ctx.fillRect(10, 50, 30, 30);
ctx.fillStyle = "black";
ctx.fillText("hsl(110, 100%, 50%)", 60, 65);
ctx.beginPath();
ctx.fillStyle = "hsl(250,100%,50%)";
ctx.fillRect(10, 90, 30, 30);
ctx.fillStyle = "black";
ctx.fillText("hsl(250, 100%, 50%)", 60, 105);
```

Model HWB

H oznacza 'hue' czyli 'odcień', 'barwa' i jest podawany w postaci kąta 0.0 - 360.0°

W oznacza 'whiteness' czyli 'białość' i jest podawane w zakresie 0-100%

B oznacza 'blackness' czyli 'czarność' i jest podawane w zakresie 0-100%

Można też podać czwartą wartość oznaczającą stopień krycia od 0.0-1.0

Skryptu obsługującego `hbw()` nie udało się uruchomić w żadnej z przeglądarek ani na `canvas`.

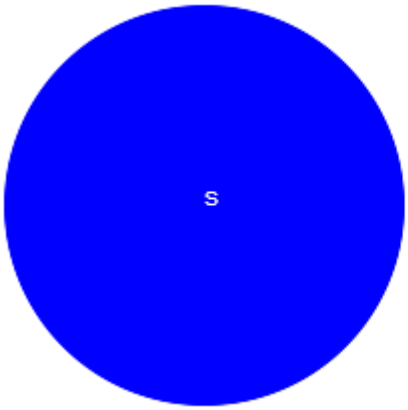
Rachunek zbiorów i składanie (kompozycja) kolorów

Zbiór

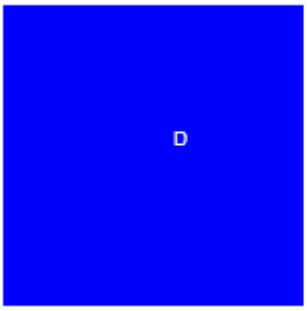
Zbiór jest pojęciem elementarnym i oznacza strukturę matematyczną, na ogół mającą elementy. Zbiór, który nie posiada elementów nazywany jest zbiorem pustym.

W sensie geometrycznym zbiór może być przedstawiony jako figura na płaszczyźnie, składająca się z elementów - punktów.

A oto przykładowy zbiór S (Dodatek 7 Listing02):



oraz przykładowy zbiór D (Dodatek7 Listing03):



Oba zbiory są typu `rgba(0,0,255,1.0)`.

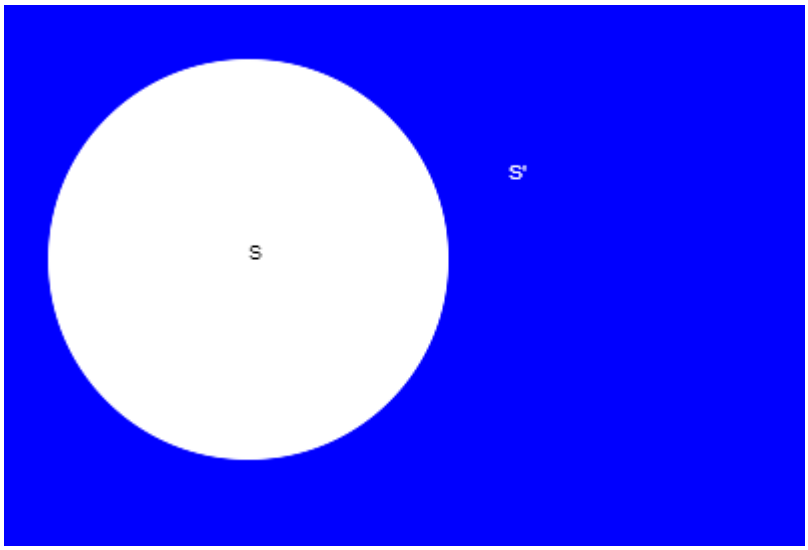
Operacje na zbiorach

Na zbiorach możemy wykonywać operacje logiczne podobne do operacji logicznych stosowanych w logice klasycznej

Dopełnienie zbioru

Jest odpowiednikiem negacji (zaprzeczenia) w logice klasycznej. Dopełnieniem S' zbioru S są wszystkie elementy płaszczyzny, które nie należą do zbioru S.

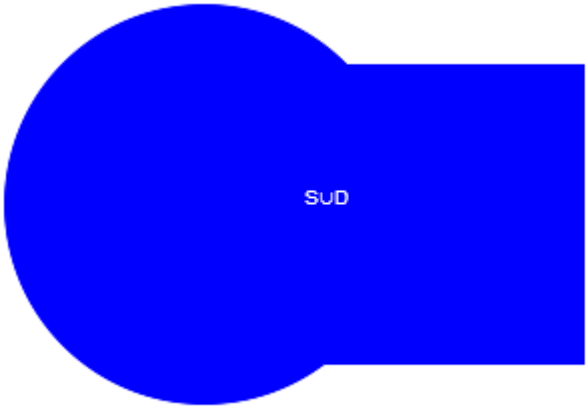
(Dodatek7 Listing04).



Suma zbiorów

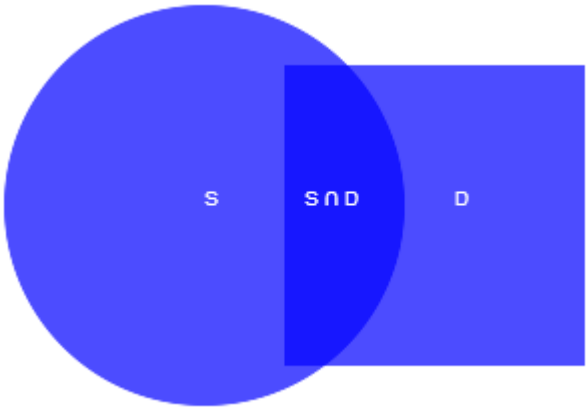
Jest odpowiednikiem alternatywy w logice klasycznej. Sumą zbioru S i D oznaczaną jako $S \cup D$ jest zbiór zawierający wszystkie elementy, które należą do S lub należą do D, czyli również elementy, które należą do obu zbiórów.

(Dodatek 7 Listing05).



Iloczyn zbiorów

Jest odpowiednikiem koniunkcji w logice klasycznej. Iloczynem zbiorów S i D oznaczanym jako $S \cap D$ jest zbiór zawierający wszystkie elementy, które należą do zbioru S i należą do zbioru D.

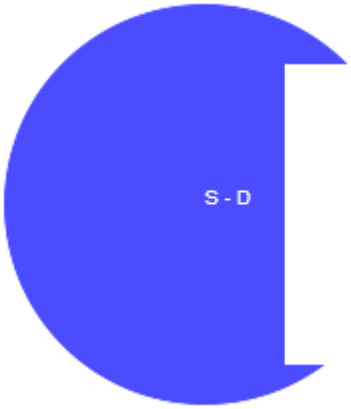


(Dodatek 7 Listing06)

Różnica zbiorów

Nie występuje w logice klasycznej

Różnicą zbiorów S i D oznaczaną jako S-D albo S\D jest zbiór zawierający wszystkie elementy, które należą do S i nie należą do D.

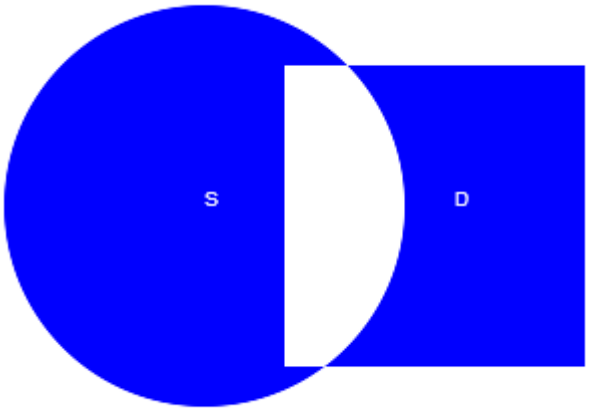


(Dodatek 7 Listing 07)

Różnica symetryczna zbiorów

Jest odpowiednikiem bramki logicznej XOR w logice klasycznej. Różnicą symetryczną elementów S i D oznaczaną jako $S \leftarrow D$, jest zbiór tych wszystkich elementów, które należą do S albo należą do D.

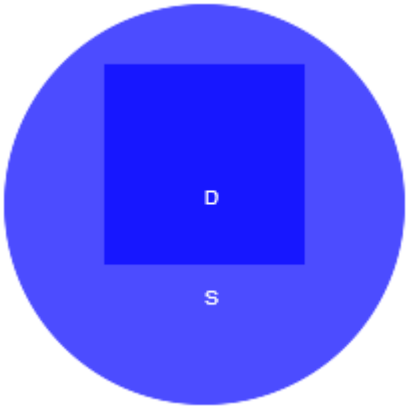
(Dodatek 7 Listing08)



Zawieranie się zbiorów

Jest odpowiednikiem implikacji w logice klasycznej. Zbiór D zawiera się w S, co oznaczmy symbolem $D \subset S$, jeżeli każdy element zbioru D jest elementem zbioru S. Nie mogą istnieć elementy D, które nie są zawarte w zbiorze S.

(Dodatek 7 Listing 09)



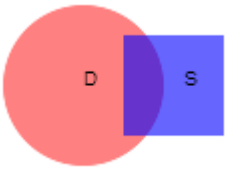
Składanie kolorów w przeglądarkach

Jak już wiemy kolor RGBA może być opisany przez cztery liczby, które oznaczają odpowiednio:

- R - kolor czerwony, 0-255
- G - kolor zielony, 0-255
- B - kolor niebieski, 0-255
- A - przezroczystość, 0-255, oznaczaną jako $\alpha = 0.0 - 1.0$, gdzie 0.0 oznacza zupełną przezroczystość, 1.0 zupełną nieprzezroczystość, a liczby pośrednie - kolory z pewną przezroczystością, gdzie $\alpha = A/255$. Jeżeli A =200, to $\alpha = 200/255 \approx 0.784$

Mamy 2 zbiory (Dodatek 7 Listing10):

- D(destination), rgba(255,0, 0,0.5)
- S (source), rgba(0,0,255,0.6)



Jeżeli przezroczystość S wynosi α_S to tło ma przezroczystość $1-\alpha_S$. Przenikająca część tła to $\alpha_D(1-\alpha_S)$, a blokowana część tła to $\alpha_S(1-\alpha_D)$.

Jeżeli przygotujemy tabelkę dla wszystkich możliwości to otrzymamy:

	S	$S \cap D$	D
tło	$1-\alpha_S$	$(1-\alpha_S)(1-\alpha_D)$	$1-\alpha_D$
tło blokowane	$\alpha_S(1-\alpha_D)$	$\alpha_S\alpha_D$	$\alpha_D(1-\alpha_S)$
tło przechodzące	$\alpha_D(1-\alpha_S)$	$(1-\alpha_S)(1-\alpha_D)$	$\alpha_S(1-\alpha_D)$

Rozważamy tutaj cztery możliwości w sensie teorii zbiorów:

opis	$\cap$	przezroczystość	Bierzemy pod uwagę	Oznaczenie
obszar poza S i obszar poza D, czyli tło T , nie zajęte ani przez S ani przez D	$\bar{S} \cap \bar{D}$	$(1-\alpha_S)(1-\alpha_D)$	0(Tło)	0
obszar S i obszar poza D, czyli obszar S nie nakładający się z D	$S \cap \bar{D}$	$\alpha_S(1-\alpha_D)$	0, S	S
obszar poza S i obszar D, czyli obszar D nie nakładający się z S	$\bar{S} \cap D$	$\alpha_D(1-\alpha_S)$	0, D	D
obszar S i obszar D, czyli obszar nakładania się S i D	$S \cap D$	$\alpha_S\alpha_D$	0, S, D	SD

Jeżeli przemnożymy liczby elementów w kolumnie 'Bierzemy pod uwagę' to otrzymamy: 1 x 2 x 2 x 3 = 12 możliwości, które ujmujemy w tabelce:

Zasada Portera-Duffa	F_S	F_D
clear (w JavaScript nie występuje)	0	0
copy	0	1
destination-atop	$1-\alpha_D$	α_S
destination-in	0	α_S
destination-out	0	$1-\alpha_S$
destination-over	$1-\alpha_D$	1

lighter	1	0
source-atop	α_D	$1-\alpha_S$
source-in	α_D	0
source-out	$1-\alpha_D$	0
source-over	1	$1-\alpha_S$
xor	$1-\alpha_D$	$1-\alpha_S$

F- oznacza frakcję, w jakim kolor S lub D występuje w miejscu złożenia kolorów

Przezroczystość wynikowa α_R będzie wynosiła:

$$\alpha_R = F_S\alpha_S + F_D\alpha_D$$

Kolor wynikowy C_R będzie wynosił:

$$C_R = F_S C_S + F_D C_D$$

gdzie C_S jest składnikiem koloru przemnożonym wstępnie przez α_S (jeżeli np kolor czerwony = 255, a przezroczystość koloru wynosi $\alpha_S = 0.5$ to za C_S podstawiamy 255*0.5 i podobnie dla C_D, gdzie podajemy kolor przemnożony wstępnie przez α_D

Obliczenia

- D(destination), rgba(255,0, 0,0.5)
- S (source), rgba(0,0,255,0.6)

zastosowana była zasada source-over, a zatem

$$\alpha_S = 0.6$$

$$\alpha_D = 0.5$$

Z tabeli:

$$F_S = 1,$$

$$F_D = 1-\alpha_S = 1-0.6 = 0.4$$

$$\alpha_R = F_S\alpha_S + F_D\alpha_D = 1*0.6 + 0.4*0.5 = 0.6 + 0.2 = 0.8$$

$$C_{red} = 1*(0*0.6) + 0.4*(255*0.5) = 0 + 51 = 51$$

$$C_{green} = 1*(0*0.6) + 0.4*(0*0.5) = 0 + 0 = 0$$

$$C_{blue} = 1*(255*0.6) + 0.4*(0*0.5) = 153 + 0 = 153$$

Kolor wynikowy to rgba (51, 0, 153, 0.8).

(Dodatek 7 Listing11)

Do obliczenia koloru można użyć funkcji:

Listing

```
var kolor=function(rgba1, rgba2, rule){
var adst= rgba1.split(",");
var asrc=rgba2.split(",");
var src=parseFloat(asrc[3]);
var dst=parseFloat(adst[3]);
var fs=0;
var fd=0;
switch(rule){
case "copy":
      fs=0;
      fd=1.00;
      break;
case "destination-atop":
      fs=1.0-dst;
      fd=src;
      break;
case "destination-in":
      fs=0;
      fd=src;
      break;
case "destination-out":
      fs=0;
      fd=1.0-src;
      break;
case "destination-over":
      fs=1.0-dst;
      fd=1.0;
      break;
case "lighter":
      fs=1.00;
      fd=0;
      break;
case "source-atop":
      fs=dst;
      fd=1.0-src;
      break;
case "source-in":
      fs=dst;
      fd=0;
      break;
case "source-out":
      fs=1.0-dst;
      fd=0;
```

```
            break;
      case "source-over":
            fs=1.0;
            fd=1.0-src;
            break;
      case "xor":
            fs=1.0-dst;
            fd=1.0-src;
            break;
      }
      var a=fs*src+fd*dst;
      var r=fs*parseInt(asrc[0].substring(5))*src +
fd*parseInt(adst[0].substring(5))*dst;
      var g=fs*parseInt(asrc[1])*src + fd*parseInt(adst[1])*dst;
      var b=fs*parseInt(asrc[2])*src + fd*parseInt(adst[2])*dst;
      return "rgba("+Math.round(r)+","+Math.round(g)+","+Math.round(b)+","+a+")";
      };
```

Plik: rachzb11.html

Listing

```
...
var col1="rgba(255,0,0,0.5)";//destination
var col2="rgba(0,0,255,0.6)";      //source
ctx.fillStyle=kolor(col1, col2, "source-over");
ctx.fillRect(270,75, 50,50);
ctx.fillText(ctx.fillStyle, 200,170);
```

rgba(51, 0, 153, 0.8)

Reguły

copy

$F_S = 0$

$F_D = 1$

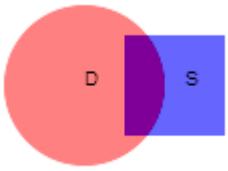
Obszar	0	S	D	SD
Wyświetlany	0	0	D	D

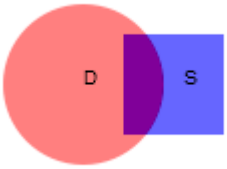
$\alpha_Z = \alpha_D$

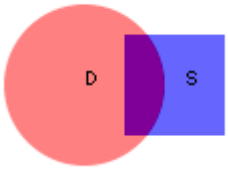
$C_Z = C_D$

(Dodatek 7 Listing13)

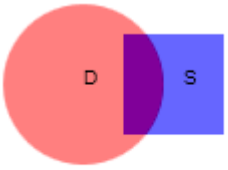
Google Chrome Opera Safari

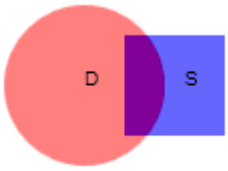






Internet Explorer Firefox





destination-atop

$$F_S = 1 - \alpha_D$$

$$F_D = \alpha_S$$

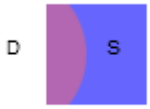
Obszar 0 S D SD
Wyświetlany 0 S 0 D

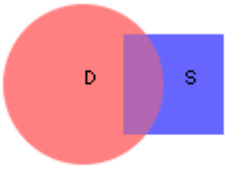
$$\alpha_R = \alpha_S$$

$$C_R = C_S(1 - \alpha_D) + C_D\alpha_S$$

(Dodatek 7 Listing14)

Google Chrome Opera Safari





Internet Explorer Firefox

destination-in

$F_S = 0$

$F_D = \alpha_S$

Obszar 0 S D SD
Wyświetlany 0 0 0 D

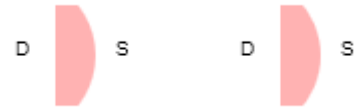
$\alpha_Z = \alpha_S \alpha_D$

$C_Z = C_D \alpha_S$

(Dodatek 7 Listing15)

Google Chrome Opera Safari

Internet Explorer Firefox



destination-out

$F_S = 0$

$F_D = 1 - \alpha_S$

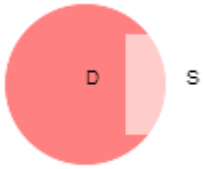
Obszar 0 S D SD
Wyświetlany 0 0 D 0

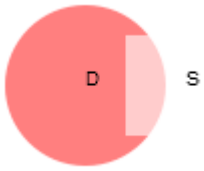
$\alpha_Z = \alpha_D(1 - \alpha_S)$

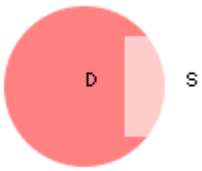
$$C_z = C_D(1-\alpha_S)$$

(Dodatek 7 Listing16)

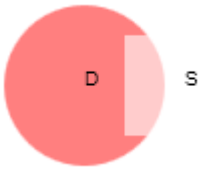
Google Chrome Opera Safari

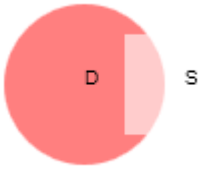






Internet Explorer Firefox





destination-over

$$F_S = 1-\alpha_D$$

$$F_D = 1$$

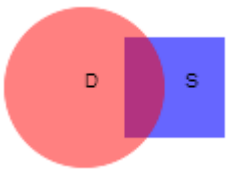
Obszar 0 S D SD
Wyświetlany 0 S D D

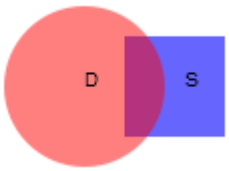
$$\alpha_Z = \alpha_S(1-\alpha_D)+\alpha_D$$

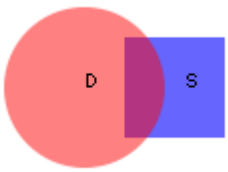
$$C_Z = C_S(1-\alpha_D)+C_D$$

(Dodatek 7 Listing17)

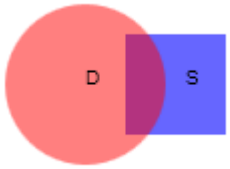
Google Chrome Opera Safari

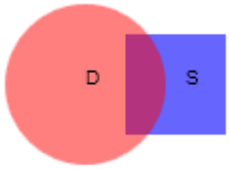






Internet Explorer Firefox





lighter

$$F_S = 1$$

$$F_D = 0$$

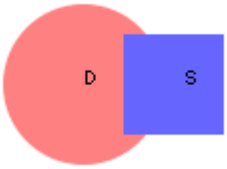
Obszar 0 S D SD
Wyświetlany 0 S 0 S

$$\alpha_R = F_S\alpha_S + F_D\alpha_D = 1*\alpha_S + 0*\alpha_D = \alpha_S$$

$$C_R = F_S C_S + F_D C_D = 1*C_S + 0*C_D = C_S$$

(Dodatek 7 Listing 18)

Google Chrome Opera Safari



Internet Explorer Firefox

source-atop

$$F_S = \alpha_D$$

$$F_D = 1 - \alpha_S$$

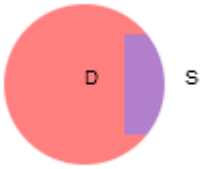
Obszar 0 S D SD
Wyświetlany 0 0 D S

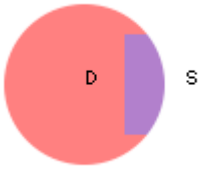
$$\alpha_s = \alpha_D$$

$$C_s = C_s\alpha_s + C_D(1-\alpha_s)$$

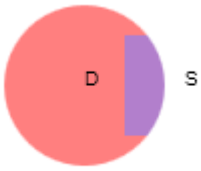
(Dodatek 7 Listing19)

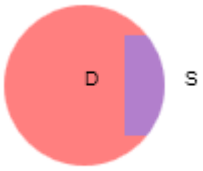
Google Chrome Opera Safari





Internet Explorer Firefox





source-in

$$F_s = \alpha_D$$

$$F_D = 0$$

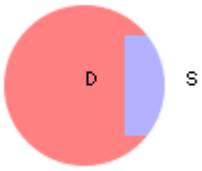
Obszar	0	S	D	SD
Wyświetlany	0	0	0	S

$$\alpha_s = \alpha_s\alpha_L$$

$$C_s = C_s\alpha_s$$

(Dodatek 7 Listing20)

Google Chrome Opera Safari



Internet Explorer Firefox

source-out

$$F_S = 1 - \alpha_D$$

$$F_D = 0$$

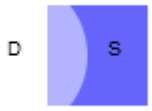
Obszar 0 S D SD
Wyświetlany 0 S 0 0

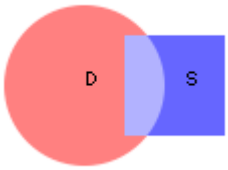
$$\alpha_Z = \alpha_S(1 - \alpha_D)$$

$$C_Z = C_S(1 - \alpha_D)$$

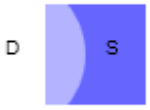
(Dodatek 7 Listing20)

Google Chrome Opera Safari





Internet Explorer Firefox



source-over

$$F_S = 1$$

$$F_D = 1 - \alpha_S$$

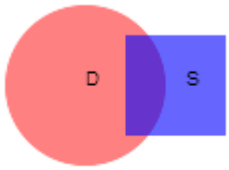
Obszar 0 S D SD
Wyświetlany 0 S D S

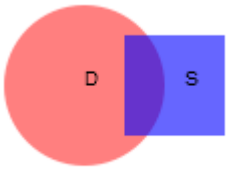
$$\alpha_z = \alpha_S + \alpha_D(1-\alpha_S)$$

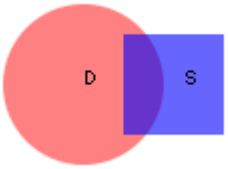
$$C_z = C_S + C_D(1-\alpha_S)$$

(Dodatek 7 Listing22)

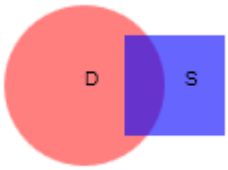
Google Chrome Opera Safari

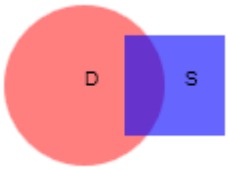






Internet Explorer Firefox





xor

$$F_S = 1-\alpha_D$$

$$F_D = 1-\alpha_S$$

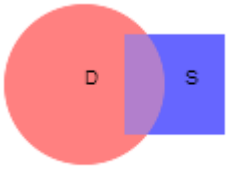
Obszar 0 S D SD
Wyświetlany 0 S D 0

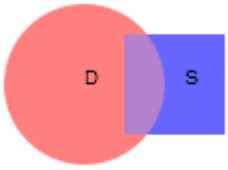
$$\alpha_z = \alpha_S(1-\alpha_D) + \alpha_D(1-\alpha_S)$$

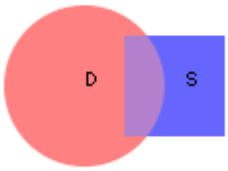
$$C_z = C_S(1-\alpha_D) + C_D(1-\alpha_S)$$

(Dodatek 7 Listing23)

Google Chrome Opera Safari







Internet Explorer Firefox

Jak widzimy implementacja zasad Portera-Duffa w przeglądarkach pozostawia wiele do życzenia.

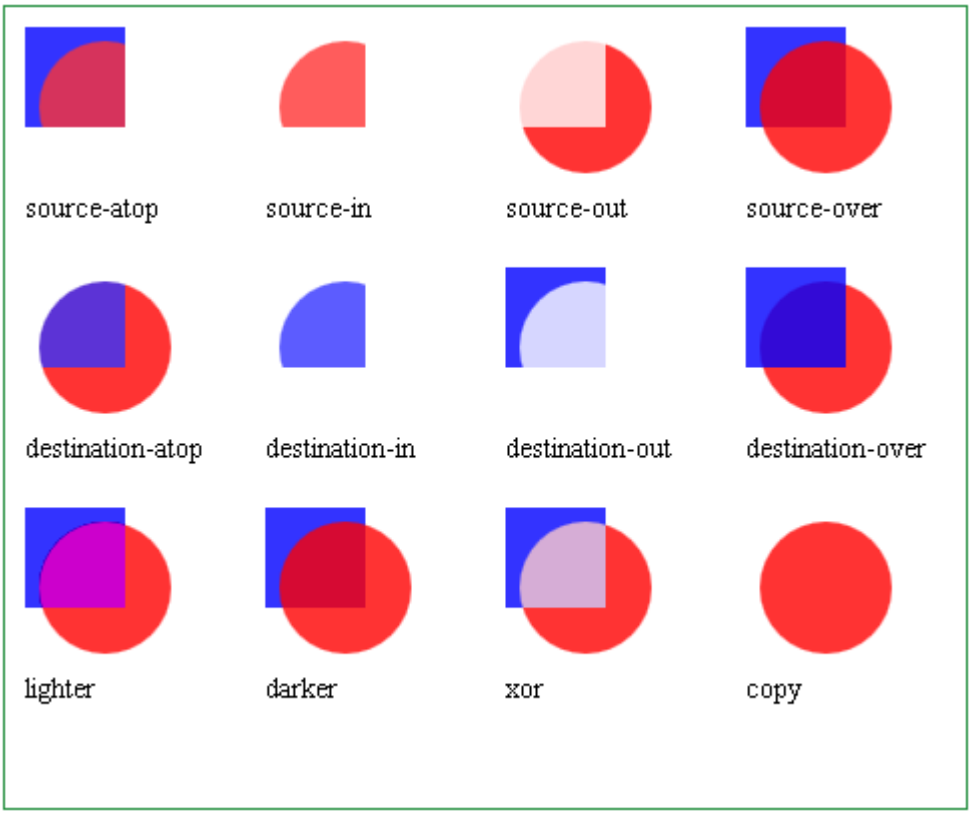
Składanie kolorów - razem

B - niebieski kwadrat - jest obiektem aktualnie istniejącym na canvas i jest 'destination' (miejscem przeznaczenia).

A - czerwone koło - jest obiektem, który jest dodawany i jest 'source' (obiektem źródłowym).

Domyślnym składaniem jest `source-atop` .

Operacja	Opis
copy	Wyświetla tylko A.
destination-atop	Wyświetla B tylko w miejscach, gdzie A i B się nakładają. Wyświetla A w miejscach gdzie A i B się nie nakładają.
destination-in	Wyświetla B tylko w obszarze, w którym A i B nakładają się. A nie jest wyświetlane.
destination-out	Wyświetla B tylko w obszarze, w którym A i B nie nakładają się. A nie jest wyświetlane.
destination-over	Wyswietla całe B, A A tylko w miejscy, gdzie A i B nie nakładają się.
lighter	W obszarach, gdzie A i B nakładają się wyświetla sumę A i B. Tam gdzie się nie nakładają wyświetla A i B normalnie.
darker	W obszarach, gdzie A i B nakładają się wyświetla różnicę A i B. Tam gdzie się nie nakładają wyświetla A i B normalnie.
source-atop	Wyświetla A tam gdzie A i B nakładają się, a B tam gdzie A i B nie nakładają się.
source-in	Wyświetla A tylko w obszarze gdzie A i B nakładają się. B nie jest wyświetlane.
source-out	Wyświetla A tylko w obszarze, gdzie A i B nie nakładają się. B nie jest wyświetlane.
source-over	Wyświetla całe A, a B tylko w miejscach, gdzie A i B nie nakładają się.
xor	Wyświetla A i B w obszarach, w których nie nakładają się. Nic nie wyświetla w miejscach, gdzie A i B nakładają się.



Listing

```
var cv = document.getElementById('canvas');
var ctx = cv.getContext('2d');
var tcv = document.getElementById('tCanvas');
var tctx = tcv.getContext('2d');
var s = 50;
var r = 33;
var so = 40;
var oo = 120;
var arr = [];
arr.push('source-atop');
arr.push('source-in');
arr.push('source-out');
arr.push('source-over');
arr.push('destination-atop');
arr.push('destination-in');
arr.push('destination-out');
arr.push('destination-over');
arr.push('lighter');
arr.push('darker');
arr.push('xor');
arr.push('copy');
ctx.translate(10, 10);
for(var n = 0; n < arr.length; n++) {
  var thiso = arr[n];
  tctx.save();
  tctx.clearRect(0, 0, cv.width, cv.height);
  tctx.beginPath();
  tctx.rect(0, 0, s, s);
  tctx.fillStyle = "rgba(0, 0, 255, 0.8)";
  tctx.fill();
  tctx.globalCompositeOperation = thiso;
  tctx.beginPath();
  tctx.arc(so, so, r, 0, 2 * Math.PI, false);
  tctx.fillStyle = "rgba(255, 0, 0, 0.8)";
```

```
    tctx.fill();
    tctx.restore();
    tctx.font = '14px san-serif';
    tctx.fillStyle = 'black';
    tctx.fillText(thiso, 0, s + 45);
    if(n > 0) {
      if(n % 4 === 0) {
        ctx.translate(oo * -3, oo);
      }
      else {
        ctx.translate(oo, 0);
      }
    }
    ctx.drawImage(tcv, 0, 0);
  }
```

Kolor wynikowy przy składaniu

Pixel B jest elementem obrazka będącego na canvas, a jego kolory to:
Br, Bg, Bb, Ba.

Pixel A jest elementem obrazka dodawanegom do canvas, a jego kolory to:
Ar, Ag, Ab, Aa

Kolor wynikowy pixela po nałożeniu się pixela A na pixel B to:
Wr, Wg, Wb, Wa

Wartości możemy obliczyć według wzorów:

Wa = 1 - (1 - Aa) * (1 - Ba)

Wr = (1 - Aa) * Br + Ar

Wg = (1 - Aa) * Bg + Ag

Wb = (1 - Aa) * Bb + Ab

Przy obliczeniu według wzoru należy wziąć pod uwagę, że wszystkie wartości kolorów są tzw. 'premultiplied alpha', tzn. wartości koloru są już wstępnie pomnożone przez alpha, czyli zamiast r podajemy r*a, itd.

Br=0, Bg=0, Bb=255, Ba=0.8

Ar=255, Ag=0, Ab=0, Aa=0.8

Wa = 1 - (1 - 0.8) * (1 - 0.8)= 1 - 0.2 * 0.2= 1 - 0.04 = 0.96

Wr = (1 - 0.8) * 0*0.8 + 255*0.8 = 255 * 0.8 = 204

Wg = (1 - 0.8) * 0*0.8 + 0*0.8 = 0

Wb = (1 - 0.8) * 255*0.8 + 0*0.8 = 41

rgba(0, 0, 255, 0.8)	rgba(255, 0, 0, 0.8)	rgba(204, 0, 41, 0.96)

Obrazy

Pobieranie obrazów z serwera

Tworzymy obiekt `Image`

Przypisujemy mu `src` wskazując obraz z serwera.

Po załadowaniu obrazka - obrazek wyświetlamy na płótnie.

Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var image = new Image();
      image.onload = function() {
       ctx.drawImage(image, 0,0, 300, 285);
};
image.src = "images/pies.jpg";
```

Pobieranie obrazów ze strony

Mamy obrazek umieszczony w elemencie `img` opatrzony atrybutem `display:none`

Pobieramy obrazek

Wyświetlamy obrazek na płótnie.

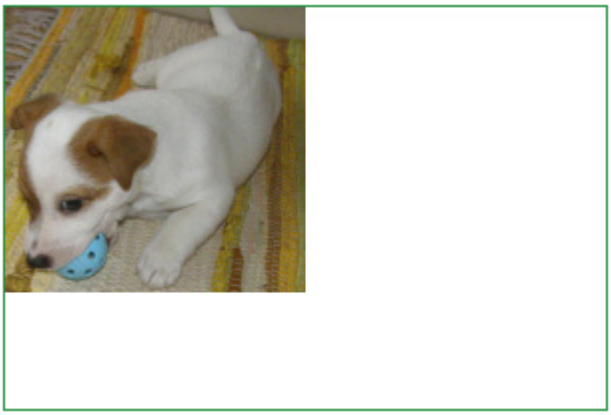
Listing

```
window.onload = function(){
       var cv = document.getElementById("canvas");
       var ctx = cv.getContext("2d");
       var img = document.getElementById("aniolek");
       ctx.drawImage(img, 10,10);
};
```

Skalowanie obrazów

Obraz skalujemy podając wymiary obrazka mniejsze lub większe od rzeczywistych.

Przy powiększanie następuje pikselizacja obrazu.



Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var image = new Image();
      image.onload = function() {
       ctx.drawImage(image, 0,0, 150, 142);
};
image.src = "images/pies.jpg";
```

Wymiarowanie obrazów

Nasz obrazek źródłowy "images/aniolek.jpg" ma wymiary:

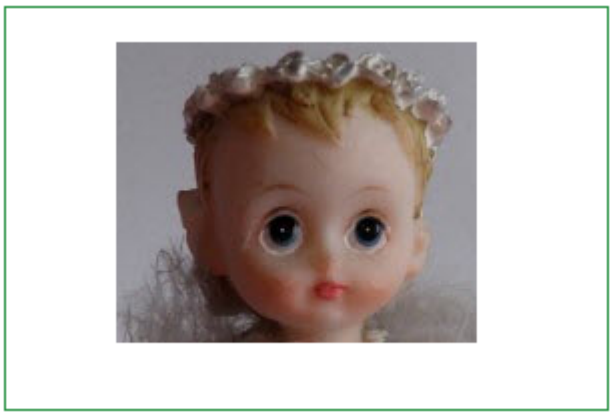
- szerokość (width) = 303px
- wysokość (height) = 407px

Funkcja `drawImage(obrazek, sx, sy, sw, sh, dx, dy, dw, dh)` służy do pobierania fragmentu obrazka źródłowego i wyświetlenia pobranego fragmentu w znaczniku <canvas>.

Parametry z przedrostkiem 's' są danymi pobieranego wycinka. Parametry z przedrostkiem 'd' są danymi obrazka wyświetlanego.

Z obrazka pobieramy wycinek o współrzędnych (sx=55, sy=17, sw=180, sh=150), czyli mamy określone parametry metody `drawImage("images/aniolek.jpg", 55, 17, 180, 150, dx, dy, dw, dh)`.

a następnie wyświetlamy wycinek na <canvas>. Parametry wyświetlenia to: (dx=55, dy=17, dw=180, dh=150), czyli mamy określone parametry fukcjiy
`drawImage("images/aniolek.jpg", sx, sy, sw, sh, 55, 17, 180, 150)`



Listing

```
window.onload = function() {
	var cv = document.getElementById("canvas");
	var ctx = cv.getContext("2d");
	var img = new Image();
	img.onload = function() {
		//ctx.drawImage(img,0,0,303,417,0,0,303,417);//cały
		//ctx.drawImage(img, 0, 0, 303, 208, 0, 0, 303, 208);//gorna polowa
		//ctx.drawImage(img, 0, 208, 303, 208, 0, 208, 303, 208);//dolna
połowa
		ctx.drawImage(img, 55, 17, 180, 150, 55, 17, 180, 150);//wycinek
obrazka
	}
	img.src = "images/aniolek.jpg";
};
```

w ten sposób mamy skompletowaną funkcję
`drawImage("images/aniolek.jpg", 55, 17, 180, 150, 55, 17, 180, 150)`

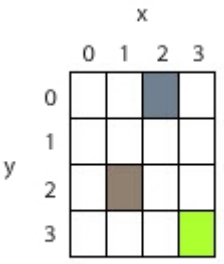
Obrazek został wyświetlony na <canvas> dokładnie w tym samym miejscu co na obrazku wyjściowym. Obrazek pobrany i wyświetlony są tej samej wielkości.

Oczywiście nic nie stoi na przeszkodzie, aby obrazek wyświetlić w innym miejscu lub w innych rozmiarach. Wystarczy zmienić pararametry `dx, dy, dw, dh`.

Operacje na pikselach obrazu

Macierz obrazu

Każdy obraz można przedstawić jako macierz punktów składającą się z rzędów i kolumn.

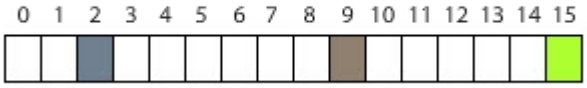


Nasz obrazek składa się z 4 rzędów i 4 kolumn, czyli ma rozmiar 4 x 4 = 16 pikseli.

Podając położenie komórki w macierzy podajemy współrzędne (rząd, kolumna), w których leży dana komórka. Jednak przy pracy z obrazami współrzędne piksela podajemy jako (x,y), gdzie x jest położeniem pixela od lewej strony obrazka w prawo, a y położeniem piksela od góry obrazka w dół, czyli doładnie odwrotnie jak w macierzy matematycznej, czy tablicach dwuwymiarowych.

Nie ma to większego znaczenia ponieważ tutaj dane obrazów zapisujemy w tablicu 1-wymiarowej.

Gdyby informacje o każdym pikselu były zapisane w postaci jednej liczby (tak jest na przykład w języku Java) nasza tablica opisująca obraz miałaby taki wygląd:



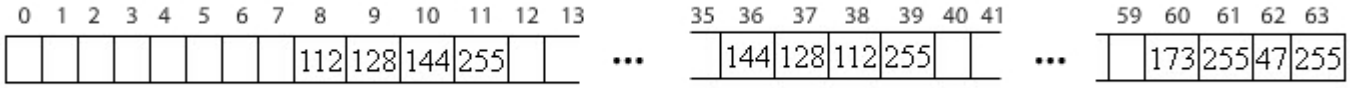
Jednak w JavaScript dane każdego piksela zapisane są w czterech kolejnych komórkach 1-wymiarowej tabeli, które podają w kolejności składowe koloru:

1. r - (red) składnik czerwony
2. g - (green) składnik zielony
3. b - (blue) składnik niebieski
4. a - (alpha) transparentność

Każdy ze składników jest liczbą od 0 do 255

W metodach tworzących kolor składnik alpha podajemy w postaci liczby zmiennoprzecinkowej od 0.0 do 1.0 (0 do 255)

Nasza tablica wygląda zatem tak:



W operacjach na obrazach możemy dokonywać zmiany danych poszczególnych pikseli. Operacje na pikselach i związane z nimi iteracje pokażemy dalej w tekście.

Pobieranie danych obrazu

Mamy obraz ukryty w elemencie `img`.

Pobieramy obrazek.

Wyświetlamy obrazek na płótnie.

Pobieramy dane obrazka przy użyciu funkcji `getImageData()`.

Listing

```
window.onload = function(){
	var cv = document.getElementById("canvas");
	var ctx = cv.getContext("2d");
	//położenie obrazka na canvas
	var imageX = 10;
	var imageY = 10;
	//pobranie obrazka
	var img = document.getElementById("aniolek");
	//wymiary obrazka
	var imageW = img.width;
	var imageH = img.height;
	//odrysowanie obrazka
	ctx.drawImage(img, imageX,imageY);
	//pobranie danych obrazka
	var imageData = ctx.getImageData(imageX, imageY, imageW, imageH);
	var data = imageData.data;
};
```

Iteracja po danych obrazu

Dane są w tablicy 1-wymiarowej. Każde 4 elementy tablicy dotyczą kolorów (r,g,b,a) jednego piksela.

Listing

```
window.onload = function() {
	var cv = document.getElementById("canvas");
	var ctx = cv.getContext("2d");
	//położenie obrazka na canvas
	var imageX = 10;
	var imageY = 10;
	//pobranie obrazka
	var img = document.getElementById("aniolek");
	//wymiary obrazka
	var imageW = img.width;
	var imageH = img.height;
	//odrysowanie obrazka
	ctx.drawImage(img, imageX, imageY);
	//pobranie danych obrazka
	var imageData = ctx.getImageData(imageX, imageY, imageW, imageH);
	var data = imageData.data;
	//iteracja po danych
	var j = data.length;
	for (var i = 0; i < j; i += 4) {
		var red = data[i];
		var green = data[i + 1];
		var blue = data[i + 2];
		var alpha = data[i + 3];
		if (i == 0) {
			//wyświetlanie danych pierwszego piksela
			ctx.fillText("Dane pierwszego piksela: ", 340, 20);
			ctx.fillText("red: " + red, 340, 40);
			ctx.fillText("green: " + green, 340, 60);
```

```
                    ctx.fillText("blue: " + blue, 340, 80);
                    ctx.fillText("alpha: " + alpha, 340, 100);
            }
        }
};
```

Pobieranie danych piksela

Listing

```
window.onload = function() {
        var cv = document.getElementById("canvas");
        var ctx = cv.getContext("2d");
        //położenie obrazka na canvas
        var imageX = 10;
        var imageY = 10;
        //pobranie obrazka
        var img = document.getElementById("aniolek");
        //wymiary obrazka
        var imageW = img.width;
        var imageH = img.height;
        //odrysowanie obrazka
        ctx.drawImage(img, imageX, imageY);
        //pobranie danych obrazka
        var imageData = ctx.getImageData(imageX, imageY, imageW, imageH);
        var data = imageData.data;
        //pobranie danych wskazanego piksela
        //koordynaty x i y są koordynatami w obrębie obrazka i jego
                pobranych
        //danych, a nie względem kontekstu.
```

```
      //piksele są liczone 0,1,2,3,4 ...
      //tablica ma indeksy 0,1,2,3 ...
      var x = 99;
      var y = 99;
      var red = data[((imageW * y) + x) * 4];
    var green = data[((imageW * y) + x) * 4 + 1];
    var blue = data[((imageW * y) + x) * 4 + 2];
    var alpha = data[((imageW * y) + x) * 4 + 3];
  //wyświetlanie danych pierwszego piksela
      ctx.fillText("Dane piksela(99, 99): ", 340, 20);
      ctx.fillText("red: " + red, 340, 40);
      ctx.fillText("green: " + green, 340, 60);
      ctx.fillText("blue: " + blue, 340, 80);
      ctx.fillText("alpha: " + alpha, 340, 100);
};
```

Iteracja po pikselach

Dane piksela (100, 100):

red: 76

green: 42

blue: 33

alpha: 255

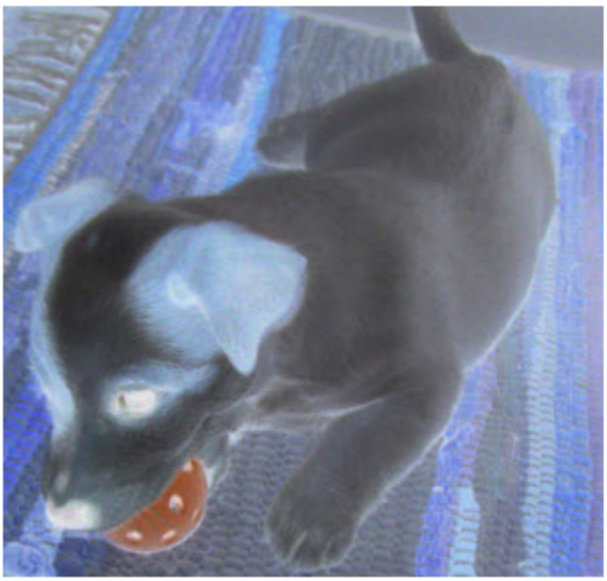
Listing

```
window.onload = function() {
      var cv = document.getElementById("canvas");
      var ctx = cv.getContext("2d");
      //położenie obrazka na canvas
      var imageX = 10;
      var imageY = 10;
      //pobranie obrazka
      var img = document.getElementById("aniolek");
      //wymiary obrazka
      var imageW = img.width;
```

```
        var imageH = img.height;
        //odrysowanie obrazka
        ctx.drawImage(img, imageX, imageY);
        //pobranie danych obrazka
        var imageData = ctx.getImageData(imageX, imageY, imageW, imageH);
        var data = imageData.data;
        //pobranie danych wskazanego piksela
        //koordynaty x i y są koordynatami w obrębie obrazka i jego
              pobranych
        //danych, a nie względem kontekstu.
  // iteracja po pikselach
        for(var y = 0; y < imageH; y++) {
          // loop through each column
          for(var x = 0; x < imageW; x++) {
            var red = data[((imageW * y) + x) * 4];
            var green = data[((imageW * y) + x) * 4 + 1];
            var blue = data[((imageW * y) + x) * 4 + 2];
            var alpha = data[((imageW * y) + x) * 4 + 3];
            if ((x == 99)&&(y==99)) {
                    //wyświetlanie danych piksela
                    ctx.fillText("Dane piksela (100, 100): ", 340, 20);
                    ctx.fillText("red: " + red, 340, 40);
                    ctx.fillText("green: " + green, 340, 60);
                    ctx.fillText("blue: " + blue, 340, 80);
                    ctx.fillText("alpha: " + alpha, 340, 100);
              }
          }
        }
};
```

Filtrowanie obrazów

Negatyw



Listing

```
function negative(img) {
        var cv = document.getElementById("canvas");
        var ctx = cv.getContext("2d");
        //położenie obrazka na canvas
        var imageX = 10;
        var imageY = 10;
```

```
        //pobranie obrazka
        //wymiary obrazka
        var imageW = img.width;
        var imageH = img.height;
        //odrysowanie obrazka
        ctx.drawImage(img, imageX, imageY);
        //pobranie danych obrazka
        var imageData = ctx.getImageData(imageX, imageY, imageW, imageH);
        var data = imageData.data;
        for (var i = 0; i < data.length; i += 4) {
                //data[i] = 255 - data[i];
                //data[i + 1] = 255 - data[i + 1];
                //data[i + 2] = 255 - data[i + 2];
                data[i] = 0xff - data[i] & 0xff;
                data[i + 1] = 0xff - data[i + 1] & 0xff;
                data[i + 2] = 0xff - data[i + 2] & 0xff;
        }
        ctx.putImageData(imageData, imageX, imageY);
};
var image = new Image();
image.src = "images/pies.jpg";
image.onload = function() {
        negative(this);
};
```

Skala szarości

Listing

```
function greyscale(img) {
        var cv = document.getElementById("canvas");
        var ctx = cv.getContext("2d");
        //położenie obrazka na canvas
        var imageX = 10;
        var imageY = 10;
        //pobranie obrazka
        //var img = document.getElementById("aniolek");
        //wymiary obrazka
        var imageW = img.width;
        var imageH = img.height;
        //odrysowanie obrazka
        ctx.drawImage(img, imageX, imageY);
        //pobranie danych obrazka
```

```
        var imageData = ctx.getImageData(imageX, imageY, imageW, imageH);
        var data = imageData.data;
        for (var i = 0; i < data.length; i += 4) {
                var grey = 0.33* data[i] +  0.56*data[i+1] + 0.11*data[i+2];
                data[i] = grey;
                data[i + 1] = grey;
                data[i + 2] = grey;
        }
        ctx.putImageData(imageData, imageX, imageY);
};
var image = new Image();
image.src = "images/pies.jpg";
image.onload = function() {
        greyscale(this);
};
```

Sepia

Listing

```
function sepia(img) {
  var cv = document.getElementById("canvas");
  var ctx = cv.getContext("2d");
  //położenie obrazka na canvas
  var imageX = 10;
  var imageY = 10;
  //pobranie obrazka
  //var img = document.getElementById("aniolek");
  //wymiary obrazka
  var imageW = img.width;
  var imageH = img.height;
  //odrysowanie obrazka
  ctx.drawImage(img, imageX, imageY);
  //pobranie danych obrazka
  var imageData = ctx.getImageData(imageX, imageY, imageW, imageH);
 var data = imageData.data;
 for (var i = 0; i < data.length; i += 4) {
  var grey = 0.33 * data[i] + 0.56 * data[i + 1] + 0.11
         * data[i + 2];
  data[i] = grey + 100;
  data[i + 1] = grey + 50;
  data[i + 2] = grey;
}
```

```
ctx.putImageData(imageData, imageX, imageY);
};
var image = new Image();
image.src = "images/pies.jpg";
image.onload = function() {
	sepia(this);
};
```

Szum

Listing

```
function szum(img) {
	var cv = document.getElementById("canvas");
	var ctx = cv.getContext("2d");
	//położenie obrazka na canvas
	var imageX = 10;
	var imageY = 10;
	//pobranie obrazka
	//var img = document.getElementById("aniolek");
	//wymiary obrazka
	var imageW = img.width;
	var imageH = img.height;
	//odrysowanie obrazka
	ctx.drawImage(img, imageX, imageY);
	//pobranie danych obrazka
	var imageData = ctx.getImageData(imageX, imageY, imageW, imageH);
	var data = imageData.data;
	var factor = 40;
	for (var i = 0; i < data.length; i += 4) {
		var rand = (0.5 - Math.random()) * factor;
		data[i] = data[i] + rand;
		data[i + 1] = data[i+1] + rand;
		data[i + 2] = data[i+2] + rand;
	}
	ctx.putImageData(imageData, imageX, imageY);
};
var image = new Image();
image.src = "images/pies.jpg";
image.onload = function() {
	szum(this);
};
```

Kontrast

Listing

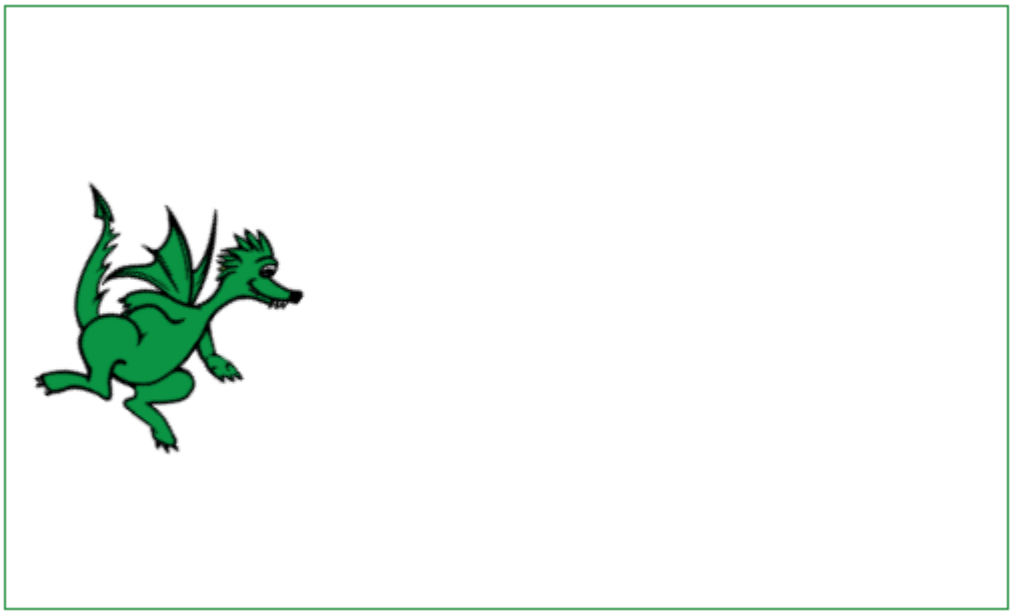
```
function newInt(n, factor){
	var x = n * factor;
	return x > 255 ? 255 : x;
};
function cont(n, factor){
	return (n < 128) ? n/factor : newInt(n, factor);
}
function kontrast(img) {
	var cv = document.getElementById("canvas");
	var ctx = cv.getContext("2d");
	//położenie obrazka na canvas
	var imageX = 10;
	var imageY = 10;
	//pobranie obrazka
	//var img = document.getElementById("aniolek");
	//wymiary obrazka
	var imageW = img.width;
	var imageH = img.height;
	//odrysowanie obrazka
	ctx.drawImage(img, imageX, imageY);
	//pobranie danych obrazka
	var imageData = ctx.getImageData(imageX, imageY, imageW, imageH);
	var data = imageData.data;
	var factor = 1.15;
	for (var i = 0; i < data.length; i += 4) {
		data[i] = cont(data[i], factor);
		data[i + 1] = cont(data[i+1], factor);
		data[i + 2] = cont(data[i+2], factor);
	}
	ctx.putImageData(imageData, imageX, imageY);
};
var image = new Image();
image.src = "images/pies.jpg";
image.onload = function() {
	kontrast(this);
};
```

Operacje na obrazach

Transformacje obrazów

Transformacje figur wykonujemy dokadnie tak samo jak transformacje kształtów.

A oto przykład obrotu.



Listing

```
var cv = document.getElementById("canvas4");
var ctx = cv.getContext("2d");
var image = new Image();
image.src = "images/samsmok.png";
ctx.rotate(Math.PI/5);
image.onload = function() {
      ctx.drawImage(image, 85, 15, 120, 130);
};
```

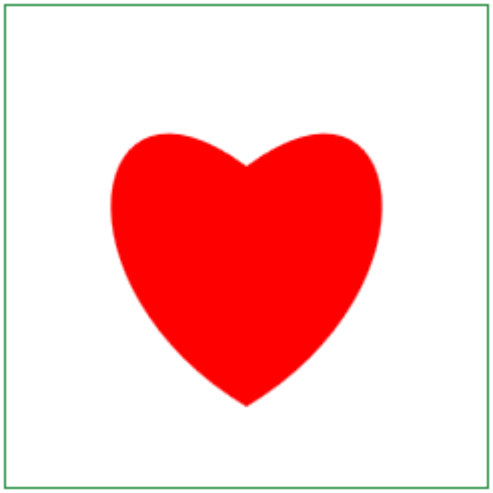
Zapamiętywanie obrazów

Postępowanie:

- Na `canvas` rysujemy obrazek
- Zapisujemy `canvas` przy użyciu funkcji `toDataURL()`

Bity obrazka są zamieniane na ciąg znaków zawierający 64 znaki (kodowanie Base64).

- Ciąg znaków oglądamy przekazując go do elementu <p>



heartData:
AAAA+VemSAAAJvElEQVR4nO3dXcikdR3G8S+2ByKF2ia+oLDFRmGeGAUqghsoxuaB
RgdBJ2snQi8ilRSkWb6guIKiYqKFlXoQRhvGIh6kR5JoYJSgJCqhqbToQpssInh1MHex5ur
MPM/M/H4z8/3AdeI+Ps99///Xn3m530CSJEmSJEmSJEmSJEmSJEmSJEmSJEmSJEmSJEm
SJEmSJEmSJEmSJEmSJEmSJEmSJEmSJEmSJEmSJEmSJEmSJEmSJEmSJEmSJEmSJEmSJEm
SJEmSJEmSJEmSJEmSJEmSJEmSJEmSJEmSJEmSJEmSJEmSJEmSJEmSJEmSJEmSJEmSJEm
SJEmSJEmSJEmSJEnSqgocGdgW+HRgR+CLgV2BCwJnDP+2pXo7uwlsGcbmjGGsdg1jt2
MYy22BI6u3UysmcEzgwsD1gT8GDgYyQfYPP3/z8P8fU70vi3LImN08jMH+Ccfs4PDz16/
bmGnGhleH309YvEnzTOCnQzlX5lV6eHW9cNi3Z2Y8Zr8PfLF6H7UEhrfHlwWem3EJD5
d9gbsDZ1fv90YFzh72Yd8Cxuu5YW58m633yuiz2axfPSbNSxm9bdxePQ7jBLYP2/pS0Vg9
EzijehzUROCUwDWBfxYV8tC8ntErWru3jBl9pLh72MbqcfrnMGenVI+LCgVOCPy1QSEP
lz1p8PY6o2+J9zQYj8PlqcAJ1WOkAhm9Fax6yzxNHg3sKBifc4e/Xb3/4/JMYNuix0eFAl8K
PNagfNPkV4HzFjA25w1/q3p/p8ljgS/Ne2zUQODkLOZb03nk7cDuzOH4aEbHbncPf6N6Pze
SVwMnz3pc1Ejg+MCvG5Rts/lT4NLAsTMYk2OH3/WnBvu12fw6cPwsuqKGMjr8UV2yW
ebJbOKLroyO4z7ZYD9mmetn2Rk1kdFhkOpyzSu7M8UJDhmdsLK7wXbPK+0Ow2kTAicF
HmlQrHlmbyb4IiejL/D2NtjeeeaRwEmL6JYWIHB5g1ItIn8P/CBw9GHG4Ojh3/7eYDsXkcsr
uqYZC5wW+HODQi0y9+eQz8YZfda9v8F2LTJ/DpxW2T3NQFb7s94HZX/gq0MmvZxv1b
K7un/ahMDHA39rUCRTk78FPl7dQ21Q4JIGJTK1uaS6h9qgwG8aFMjU5jfVPdQGBM4M/
KtBgUxt/hU4s7qPmlLgigblMT1yRXUfNaXAgw2KY3rkweo+agoZXaj/coPimB55OV74vzw
COxuUxvTKzupeakKBKxsUxvTKldW91IQCDzUojOmVh6p7qQkFnm5QGNMrT1f3UhMI
bA0caFAY0ysHAlur+6kxAp9tUBbTM5+t7qfGCFzUoCimZy6q7qfGyOj5OdVFMT1zWXU/
NUZGj9yoLorpmWuq+6kxsr4X8Jvxuam6nxojcFuDopieub26nxojoyfoVRfF9MzPqvupMQL
3NiiK6Zn7qvupMeICNu+fe6r7qTEC9zUoiukZX4G7C9zZoCimZ+6s7qfGiIeRzPvH+0R3F7i
qQVFMz1xV3U+NEfhOg6KYnvlOdT81RmBXg6KYntlV3U+NEbiwQVFMz1xY3U+NETi
9QVFMz5xe3U+NEe/IYQ4f78ixLOI9scx74z2xlkVgb4PCmF7ZW91LTShwR4PCmF65o7q
XmlC8rY55b7ydzrIInN+gMKZXzq/upSYUODGwr0FpTI/sC5xY3UtNIfCHBsUxPfKH6j5q
SoFbGhTH9Mgt1X3UlAJfb1Ac0yNfr+6jphTY3qA4pke2V/dRUwpsCTzeoDymNo8HtlT3U
RsQuKlBgUxtvJn7soqXFhovIVxegeMCzzYokanJs4HjqnuoTQj8vEGRTE1+Xt0/bVLg4gZF
MjW5uLp/2qTAhwMHG5TJLDYHAx+u7p9mIPCLBoUyi80vqnunGQl8rUGhzGLztereaUY
y+jb6qQalMovJU/Hb59USuLFBscxicmN13zRjgXMD7zQol5lv3gmcW903zUHg4QYFM/P
Nw9U905wEvtWgYGa++VZ1zzQngU/Ge0avcp4OfLK6Z5qjwLUNimbmk2ur+6U5C3wu8F
KDspnZ5uXA56v7pQUI3N6gcGa2ub26V1qQjA4p+QC01cmBeOhovQTubVA8M5vcW90n
LVhgR4PimdlkR3WfVCDwywblM5vLL6t7pCIZPUPp9QYlNBvL6/GZR+stcFuDIpqN5dbq/
qhY4MzACw3KaKbLC4Ezq/ujBuLZWcsYz7rSSODUeMH/MuWpwKnVvVEj8UqlZYpXH
OndAlsDexqU03xw9gS2VvdFDQUuCLzWoKTm8HktcEF1T9RY4LoGRTWHz3XV/VBzg
U8EHmlQVvPuPBL4RHU/tAQyuo+0T3Pok4PxPs+aRrxmuFO81lfTCXwssK9Bedc9rwY+
Vt0HLaHArsC/G5R4XfNmYFd1D7TE4hMdKuMTFrQ5gW2BvQ3KvG7ZG9hWPf9aAYG

dgRcblHpd8mJgZ/W8a4UELm9Q7HXJ5dXzrRUTOCpwT4Nyr3ruCRxVPd9aQYETMjqsU
V3yVY2HjDRfgS8Hnm9Q9lXL84EvV8+v1kDgGxkdo6wu/arkzcA3qudVayRwdYPir0qurp5
PrZnA0YG7GpR/2XNX4Ojq+dQaCnwq8GCDRbCseTDwqep51BoLnBN4osFiWLY8ETine
v4kAmfHpx1OkwOBs6vnTfqfwMWBfzRYHN3zj8DF1fMlvUfgm4H9DRZJ1+wPfLN6nqT
3Ffhe4K0Gi6Vb3gp8r3p+pLECP2qwYLrlR9XzIk0ksCVwQ4NF0yU3BD5UPS/SxAIfCdza
YPFU59bAR6rnQ5paRq/EDzRYRFV5ILCleh6kDQscG7ipwWJadG4KHFs9/tKmBY7Mej2
D+NrAkdXjLs1U4IdZ7Sc+HAz8sHqcpbkJfDfwRoPFNuu8Efhu9fhKc5fRGVuvNFh0s8or8
QwrrZPAV7IaF0AcCHylejylhQtcFHiswSLcaB4LXFQ9jlKZwFmBPQ0W47TZEzirevykcoH
tgTsbLMpJc2dge/W4SW1kdOrlT9L7bpdvDtvoqZHS4WR0y9oXGyzW/8+L8dav0ngZfUP9e
INF+988Hr9pliYXODnwaIPF+2jg5OrxkJZO4KOBKwOvFSzc14a//dHqcZCW2vCWepGvx
o/6llmaocBnAncsYPHeEfhM9f5KKydwRODbgWfnsHCfHX73EdX7Ka20wHmB381w8e4J
nFu9X9LayOh2PT8OvL2Jhfv28Du87Y1UIbAz8NsNLN7fBnZWb7+09gJbA98PPDfBwn1u
+Nmt1dst6RCBLwTu/4DFe19gR/V2SnofgaMClwb+csjC/cvw346q3j5JE8joVMwHhngqpCR
JkiRJkiRJkiRJkiRJkiRJkiRJkiRJkiRJkiRJkiRJkiRJkiRJkiRJkiRJkiRJkiRJkiRpJf0HVVS
VPtcbDEoAAAAASUVORK5CYII=

Listing

```
function drawHeart(ctx, x, y, w, h){
	var x0 = x + 0.5 * w;
	var y0 = y + 0.3 * h;
	var x1 = x + 0.1 * w;
	var y1 = y //+ 0 * h;
	var x2 = x //+ 0 * w;
	var y2 = y + 0.6 * h;
	var x3 = x + 0.5 * w;
	var y3 = y + 0.9 * h;
	var x4 = x + 1 * w;
	var y4 = y + 0.6 * h;
	var x5 = x + 0.9 * w;
	var y5 = y //+ 0 * h;
	ctx.moveTo(x0, y0);
	ctx.bezierCurveTo(x1, y1, x2, y2, x3, y3);
	ctx.bezierCurveTo(x4, y4, x5, y5, x0, y0);
};
var cv = document.getElementById('canvas');
var ctx = cv.getContext('2d');
ctx.beginPath();
ctx.lineWidth = 5;
ctx.fillStyle = "rgb(255,0,0)";
drawHeart(ctx,20,20,200,200);
ctx.fill();
var dataURL = cv.toDataURL();
var doc = document.getElementById("heartData");
doc.innerHTML="heartData:" + dataURL;
```

Zapamiętywanie i pobieranie danych obrazu toDataUrl()

Postępowanie:

- Tworzymy pusty obrazek
- Elementowi `canvas` nadajemy styl: `display: none`
- Na canvas rysujemy obrazek
- Obrazek z canvas będzie widoczny jako element `img`. Można go kliknąć prawym klawiszem myszy i zapisać jak każdy inny obrazek. Oczywiście można by również kliknąć `canvas`, ale obrazek pochodny niekoniecznie musi być identyczny z `canvas`.

Kliknij prawym klawiszem myszy aby zapamiętać.

Listing

```
function drawHeart(ctx, x, y, w, h){
       var x0 = x + 0.5 * w;
       var y0 = y + 0.3 * h;
       var x1 = x + 0.1 * w;
       var y1 = y;
       var x2 = x;
       var y2 = y + 0.6 * h;
       var x3 = x + 0.5 * w;
       var y3 = y + 0.9 * h;
       var x4 = x + w;
       var y4 = y + 0.6 * h;
       var x5 = x + 0.9 * w;
       var y5 = y;
       ctx.moveTo(x0, y0);
       ctx.bezierCurveTo(x1, y1, x2, y2, x3, y3);
       ctx.bezierCurveTo(x4, y4, x5, y5, x0, y0);
};
var cv = document.getElementById('canvas');
     var ctx = cv.getContext('2d');
     ctx.beginPath();
  ctx.lineWidth = 5;
  ctx.fillStyle = "rgb(255,0,0)";
  drawHeart(ctx,20,20,200,200);
  ctx.fill();
  var dataURL = cv.toDataURL();
  document.getElementById('obrazek').src = dataURL;
```

Uwagi do toDataURL()

`dataURL` możemy użyć w następujących konfiguracjach:

1. `cv.toDataURL()`
2. `cv.toDataURL("image/png")`
3. `cv.toDataURL("image/jpeg", 1.0)`

Dwa pierwsze zapisy są identyczne ponieważ domyślnie obraz zapisywany jest jako 'png'. Tło obrazka jest czarne: rgb(0,0,0), ale ponieważ kanał alfa jest ustawiany na 0, czyli obrazek jest przezroczysty: rgba(0,0,0,0).

Przez obrazek będzie jest widoczne np. białe tło.

Jeżeli użyjemy trzeciego zapisu otrzymamy tło czarne rgb(0,0,0).

Jest to wynikiem faktu, że tablica danych jest pusta, czyli rgb(0,0,0).

W takiej sytuacji alfa jest przyjmowane domyślnie jako 1.0, a więc otrzymujemy rgb(0,0,0,1), czyli kolor czarny.

Jeśli nie chcesz koloru czarnego możesz wypełnić canvas żądanym kolorem, a następnie rysować na canvas.

Użycie toBlob()

W momencie pisania (grudzień 2014) działa tylko w Firefox-ie.

Rysujemy obrazek na canvas.

Używamy `toBlob()` z jednym argumentem - funkcją zwrotną.

Funcja przekształca canvas w obrazek PNG (w postaci obiektu binarnego blob).

blob jest przypisywany do elementu 'obrazek' jako źródło.

Nazwa Blob to nazwa interfejsu używanego głównie w JDBC. Klasy implementujące tworzą Binary Large OBject.

W JavaScript jest w zasadzie bezużyteczny, ale jeśli przekażemy obiekt na serwer, to możemy obiekt umieścić w bazie danycha i/lub prekształcić w obrazek, przy użyciu np. metod języka Java.

Kliknij prawym klawiszem myszy aby zapamiętać

Listing

```
function drawHeart(ctx, x, y, w, h){
       var x0 = x + 0.5 * w;
       var y0 = y + 0.3 * h;
       var x1 = x + 0.1 * w;
       var y1 = y;
       var x2 = x;
       var y2 = y + 0.6 * h;
       var x3 = x + 0.5 * w;
       var y3 = y + 0.9 * h;
       var x4 = x + w;
       var y4 = y + 0.6 * h;
       var x5 = x + 0.9 * w;
       var y5 = y;
       ctx.moveTo(x0, y0);
       ctx.bezierCurveTo(x1, y1, x2, y2, x3, y3);
       ctx.bezierCurveTo(x4, y4, x5, y5, x0, y0);
};

var cv = document.getElementById('canvas');
     var ctx = cv.getContext('2d');
```

```
    ctx.beginPath();
  ctx.lineWidth = 5;
  ctx.fillStyle = "rgb(255,0,0)";
  drawHeart(ctx,20,20,200,200);
  ctx.fill();
  cv.toBlob(function(blob){
        var obrazek = document.getElementById('obrazek');
        var url=URL.createObjectURL(blob);
        obrazek.src=url;
  });
```

Użycie toBlob() (2)

Działa tylko w Firefox-ie

Rysujemy obrazek na canvas.

Używamy `toBlob()` z trzema argumentami - funkcją zwrotną, trybem obrazka i dokładnością obrazka.

Funcja przekształca canvas w obrazek JPEG (w postaci obiektu binarnego blob) z pełną dokładnością.

blob jest przypisywany do elementu 'obrazek' jako źródło.

Kliknij prawym klawiszem myszy aby zapamiętać

Listing

```
function drawHeart(ctx, x, y, w, h){
       var x0 = x + 0.5 * w;
       var y0 = y + 0.3 * h;
       var x1 = x + 0.1 * w;
       var y1 = y;
       var x2 = x;
       var y2 = y + 0.6 * h;
       var x3 = x + 0.5 * w;
       var y3 = y + 0.9 * h;
       var x4 = x + w;
       var y4 = y + 0.6 * h;
       var x5 = x + 0.9 * w;
       var y5 = y;
       ctx.moveTo(x0, y0);
```

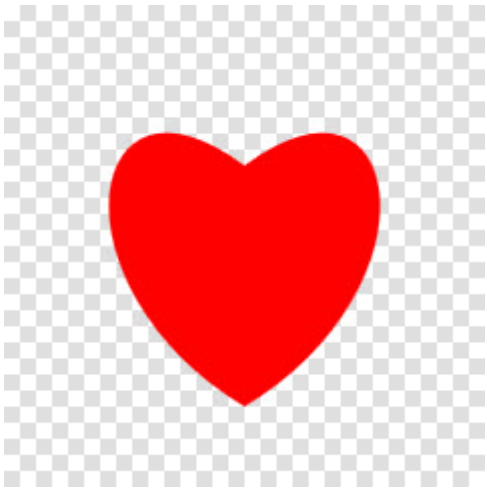
```
      ctx.bezierCurveTo(x1, y1, x2, y2, x3, y3);
      ctx.bezierCurveTo(x4, y4, x5, y5, x0, y0);
};

var cv = document.getElementById('canvas');
     var ctx = cv.getContext('2d');
     ctx.beginPath();
  ctx.lineWidth = 5;
  ctx.fillStyle = "rgb(255,0,0)";
  drawHeart(ctx,20,20,200,200);
  ctx.fill();
  cv.toBlob(function(blob){
        var obrazek = document.getElementById('obrazek');
        var url=URL.createObjectURL(blob);
        obrazek.src=url;
        //URL.revokeObjectURL(url);
  }, "image/jpeg", 1.0);
```

Uwagi do toBlob()

`toBlob` możemy użyć w następujących konfiguracjach:

1. `cv.toBlob(callback)`
2. `cv.toDataURL(callback, "image/png")`
3. `cv.toDataURL(callback, "image/jpeg", 1.0)`

Dwa pierwsze zapisy są identyczne ponieważ domyślnie obraz zapisywany jest jako 'png'. Tło obrazka jest czarne: rgb(0,0,0), ale ponieważ kanał alfa jest ustawiany na 0, czyli obrazek jest przezroczysty: rgba(0,0,0,0).



Przez obrazek będzie jest widoczne np. białe tło.

Jeżeli użyjemy trzeciego zapisu otrzymamy tło czarne rgb(0,0,0).

Jest to wynikiem faktu, że tablica danych jest pusta, czyli rgb(0,0,0).

W takiej sytuacji alfa jest przyjmowane domyślnie jako 1.0, a więc otrzymujemy rgb(0,0,0,1), czyli kolor czarny.

Jeśli nie chcesz koloru czarnego możesz wypełnić canvas żądanym kolorem, a następnie rysować na canvas.

Na końcu funkcji zwrotnej możemy użyć `URL.revokeObjectURL(url);`, ale wtedy nie

będzie można skobiować obrazka z elementu .

Desenie, gradienty i cienie

Desenie

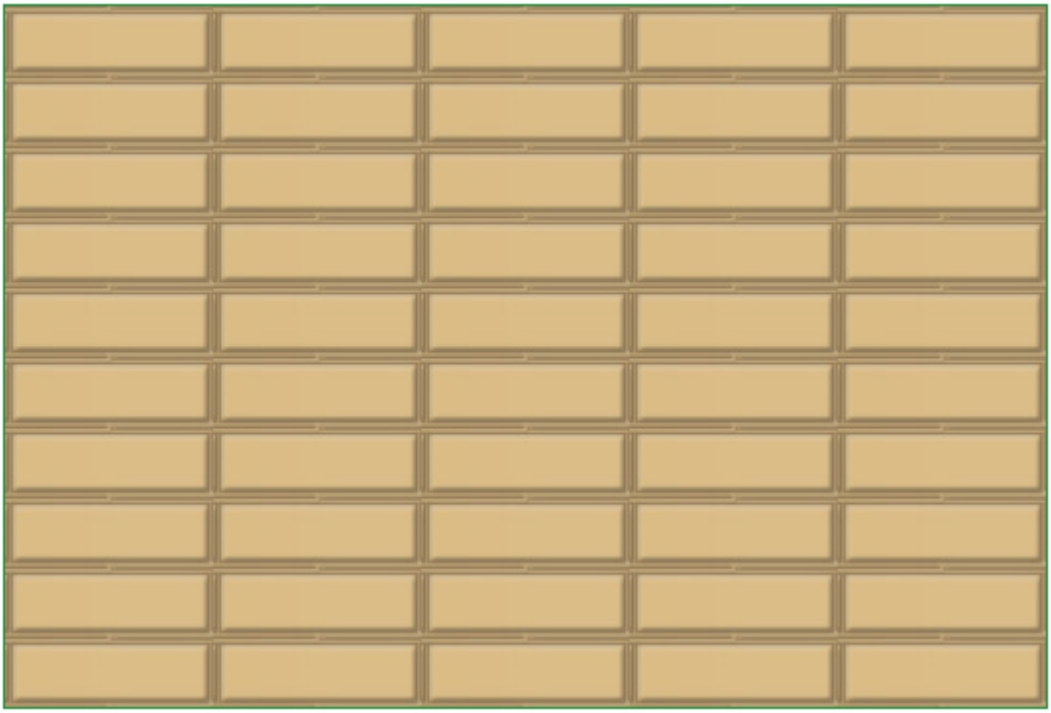
Deseń tworzymy poleceniem `createPattern(obrazek, powtórzenie)` , w którym podajemy obrazek, który będzie deseniem (teksturą), oraz sposób powtórzenia deseniu.

Sposób powtórzenia ma 4 możliwości:

- 'repeat' - powtórzenie obrazka w pionie i w poziomie
- 'no-repeat - wyświetla pojedynczy obrazek
- 'repeat-x' - powtarza obrazek w rzędzie
- 'repeat-y' - powtarza obrazek w kolumnie

Nie wszystkie przeglądarki obsługują prawidłowo tę funkcję.

'repeat'

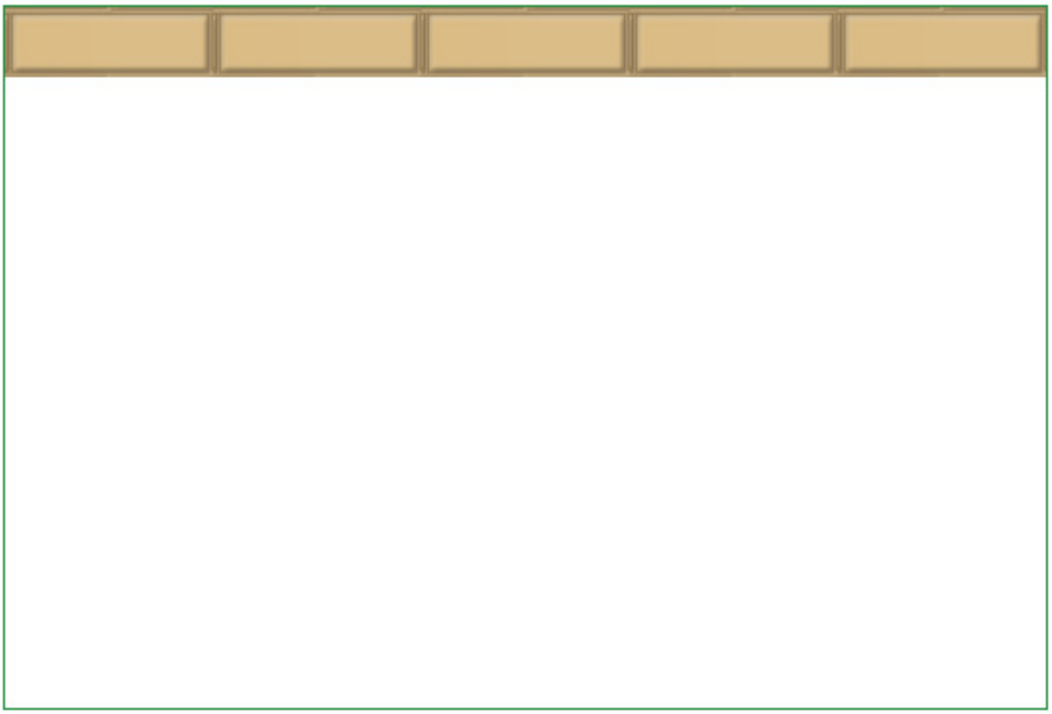


Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var image = new Image();
   image.onload = function() {
        var pattern = ctx.createPattern(image, 'repeat');
         ctx.rect(0, 0, cv.width, cv.height);
         ctx.fillStyle = pattern;
         ctx.fill();
   };
image.src = "images/cegla.png";
```

'repeat-x'

W chwili pisania (grudzień 2014) nie działa w przeglądarce Firefox.

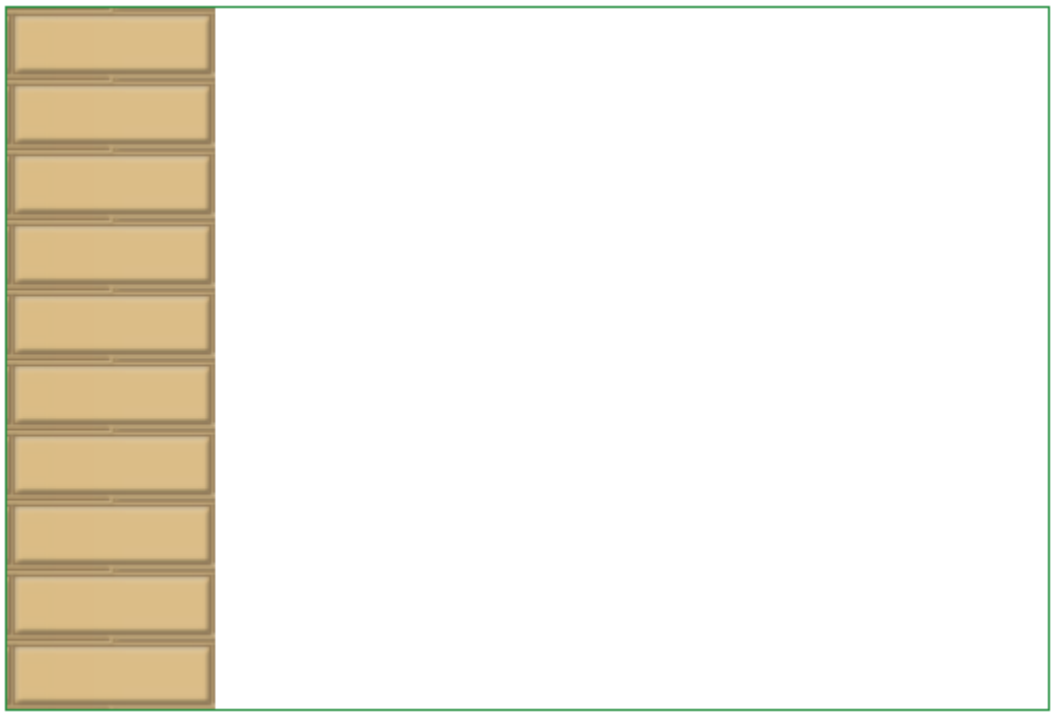


Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var image = new Image();
   image.onload = function() {
        var pattern = ctx.createPattern(image, 'repeat-x');
         ctx.rect(0, 0, cv.width, cv.height);
         ctx.fillStyle = pattern;
         ctx.fill();
   };
image.src = "images/cegla.png";
```

'repeat-y'

W chwili pisania (grudzień 2014) nie działa w przeglądarce Firefox.



Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var image = new Image();
   image.onload = function() {
        var pattern = ctx.createPattern(image, 'repeat-y');
         ctx.rect(0, 0, cv.width, cv.height);
         ctx.fillStyle = pattern;
         ctx.fill();
   };
image.src = "images/cegla.png";
```

'no-repeat'

W chwili pisania (grudzień 2014) nie działa w przeglądarce Firefox.

Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var image = new Image();
image.onload = function() {
        var pattern = ctx.createPattern(image, 'no-repeat');
     ctx.rect(0, 0, cv.width, cv.height);
     ctx.fillStyle = pattern;
     ctx.fill();
};
image.src = "images/cegla.png";
```

Gradient liniowy

Gradient liniowy tworzymy przy użyciu polecenia `createLinearGradient(x0,y0,x1,y1)` .

Punkty wyznaczają linię wzdłuż której układa się gradient

Gradient musi mieć co najmniej dwa kolory, ale może mieć ich znacznie więcej.

Miejsca gdzie wchodzą kolory oznaczamy używając polecenia `addColorStop(x, kolor)` .

x - to liczba zmiennoprzecinkowa oznaczająca względną odległość od początku linii wyznaczającej gradient.

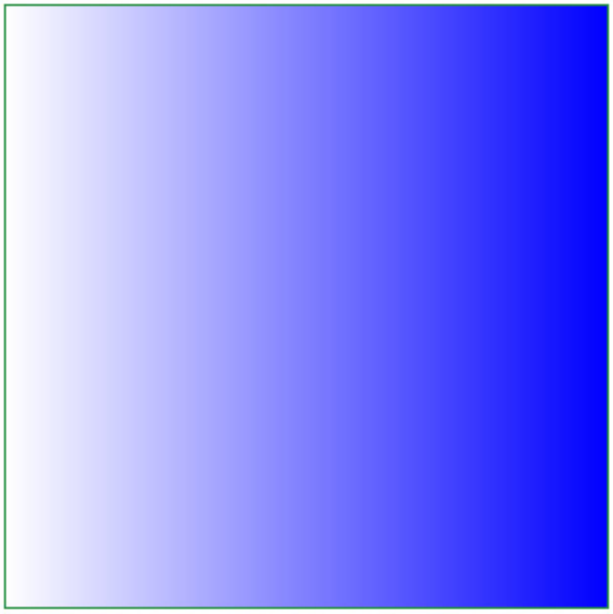
Liczba 0 oznacza początek linii, liczba 1 - koniec linii, liczba 0.5 środek linii, etc.

Rozciągość gradientu w kierunku prostopadłym zależy jedynie od kształtu figury wypełnionej gradientem.

Gradient liniowy poziomy

Linią gradientu jest linia prosta będąca górną krawędzią wypełnianego kwadratu.

Gradient tworzą dwa kolory: biały na początku linii i niebieski na końcu linii gradientu.



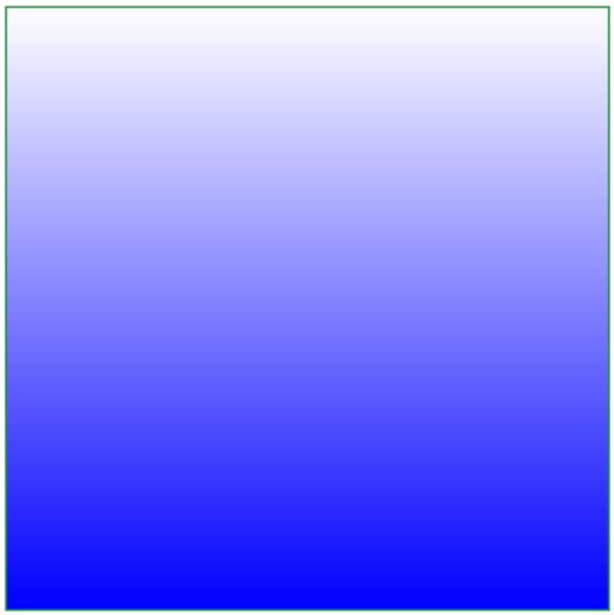
Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var w = cv.width;
var h = cv.height;
var grd = ctx.createLinearGradient(0, 0, w,0);
grd.addColorStop(0, "white");
grd.addColorStop(1, "blue");
ctx.fillStyle=grd;
ctx.rect(0,0, w, h);
ctx.fill();
```

Gradient liniowy pionowy

.Linia gradientu przebiega wzdłuż lewego brzegu wypełnianego prostokąta.

Gradient tworzą dwa kolory: biały na początku linii gradientu i niebieski na końcu linii gradientu.



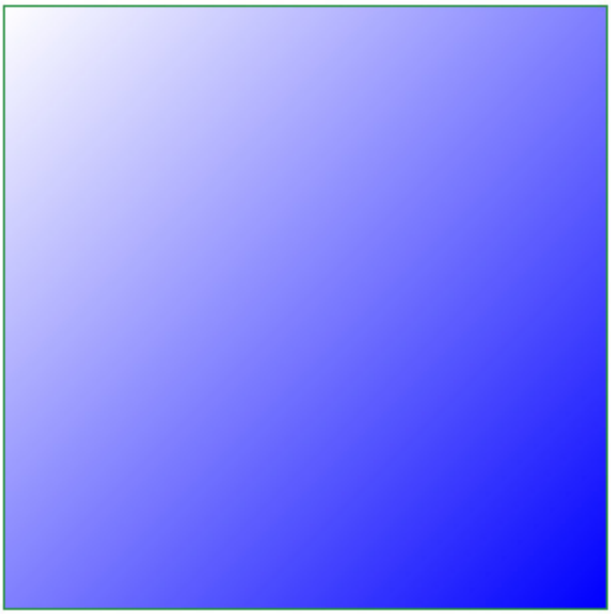
Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var w = cv.width;
var h = cv.height;
var grd = ctx.createLinearGradient(0, 0, 0,h);
grd.addColorStop(0, "white");
grd.addColorStop(1, "blue");
ctx.fillStyle=grd;
ctx.rect(0,0, w, h);
ctx.fill();
```

Gradient liniowy kątowy

Linia gradientu przebiega wzdłuż przekątnej wypełnianego kwadratu.

Gradient składa się z dwóch kolorów, biały na początku linii i niebieski na końcu linii.



Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var w = cv.width;
var h = cv.height;
var grd = ctx.createLinearGradient(0, 0, w,h);
grd.addColorStop(0, "white");
grd.addColorStop(1, "blue");
ctx.fillStyle=grd;
ctx.rect(0,0, w, h);
ctx.fill();
```

Gradient liniowy wielokolorowy

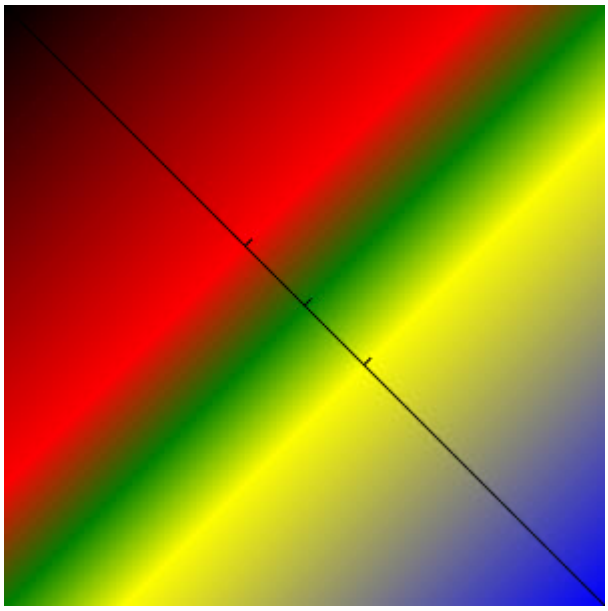
Linia gradientu przebiega wzdłuż przekątnej wypełnianego kwadratu.

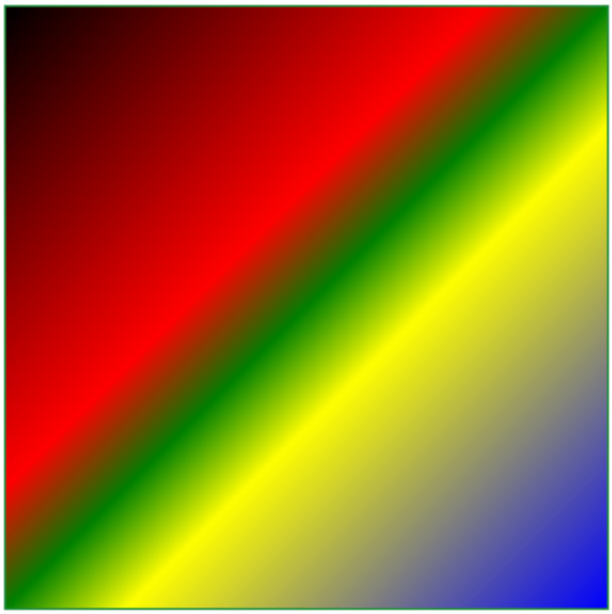
Gradient tworzy 5 kolorów:

- czarny - na początku linii
- niebieski - na końcu linii

oraz kolory pośrednie umieszczone pomiędzy nimi:

- czerwony - w odległości 0.4 * długość linii
- zielony - w połowie długości linii
- żółty - w odległości 0.6 * długość linii





Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var w = cv.width;
var h = cv.height;
var grd = ctx.createLinearGradient(0, 0, w,h);
grd.addColorStop(0, "black");
grd.addColorStop(0.4, "red");
grd.addColorStop(0.5, "green");
grd.addColorStop(0.6, "yellow");
grd.addColorStop(1, "blue");
ctx.fillStyle=grd;
ctx.rect(0,0, w, h);
ctx.fill();
```

Gradient radialny (kołowy)

Gradient radialny tworzymy przy użyciu polecenia
`createRadialGradient(x1,y1,r1,x2,y2,r2)` .

x_1,y_1 - to współrzędne środka koła o promieniu r_1

x_2,y_2 - to współrzędne środka koła o promieniu r_2

Gradient jest rozpięty między tymi dwoma kołami.

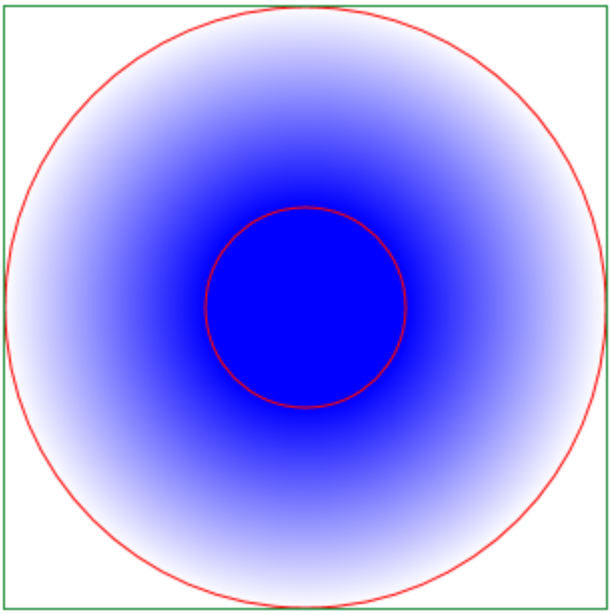
Podobnie jak w gradiencie liniowym polecenia `addColorStop(liczba, kolor)` wyznaczają położenie użytych kolorów.

Gradient kołowy (1)

Gradient wyznaczony jest przez dwa współśrodkowe koła o różnych promieniach.

Gradient ma dwa kolory, z których pierwszy, niebieski, tworzy mniejsze koło, a kolor drugi, biały tworzy większe koło.

Czerowne koła - pomocnicze pokazują okręgi kół między którymi rozpięty jest gradient.



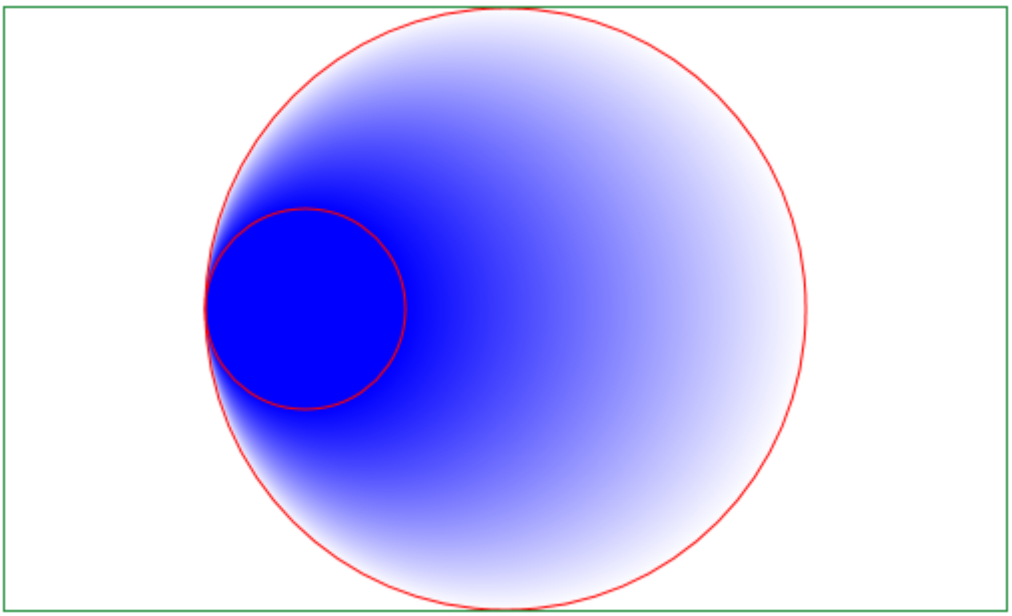
Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var w = cv.width;
var h = cv.height;
var grd = ctx.createRadialGradient(150, 150, 50, 150, 150, 150);
grd.addColorStop(0, "blue");
grd.addColorStop(1, "white");
ctx.fillStyle = grd;
ctx.rect(0, 0, w, h);
ctx.fill();
```

Gradient kołowy (2)

Gradient utworzony przez mniejsze koło niebieskie i większe koło białe.

Czerwone koła - pomocnicze pokazują okręgi kół pomiędzy którymi rozpięty jest gradient.



Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var w = cv.width;
var h = cv.height;
var grd = ctx.createRadialGradient(150, 150, 50, 250, 150, 150);
grd.addColorStop(0, "blue");
grd.addColorStop(1, "white");
ctx.fillStyle=grd;
ctx.rect(0,0, w, h);
ctx.fill();
```

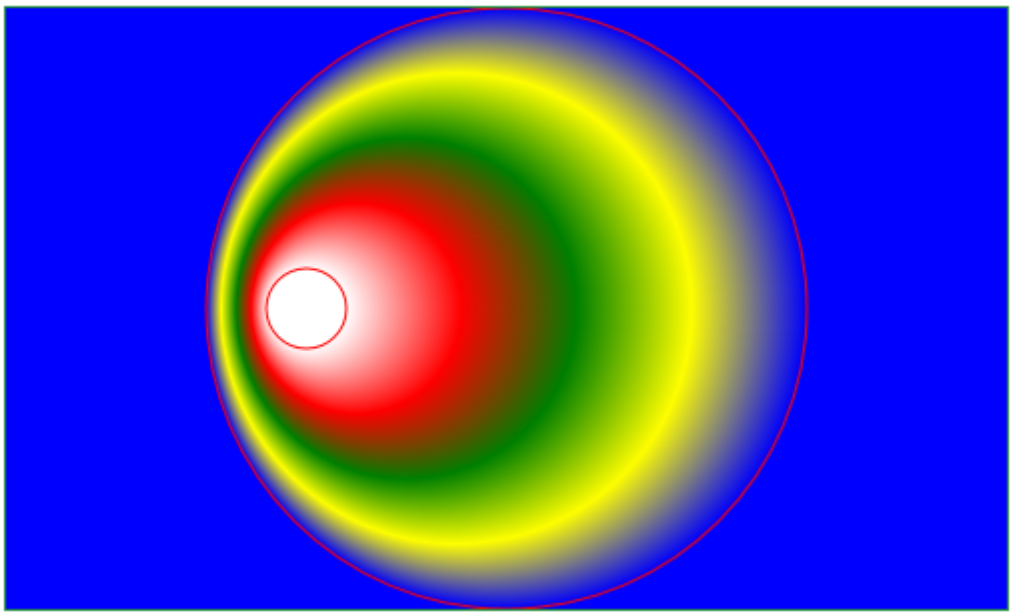
Gradient kołowy (3)

Gradient rozpięty jest między małym białym kołem, a dużym kołem niebieskim.

Między nimi dodano trzy równo rozmieszczone kolory: czerwony (0.25), zielony (0.5), żółty(0.75).

Czerwone koła - pomocnicze - pokazują okręgi kół pomiędzy którymi rozpięty jest gradient.

Ponieważ kolor niebieski nie ma ogranicznika, jest rozciągany na całą płaszczyznę.



Listing

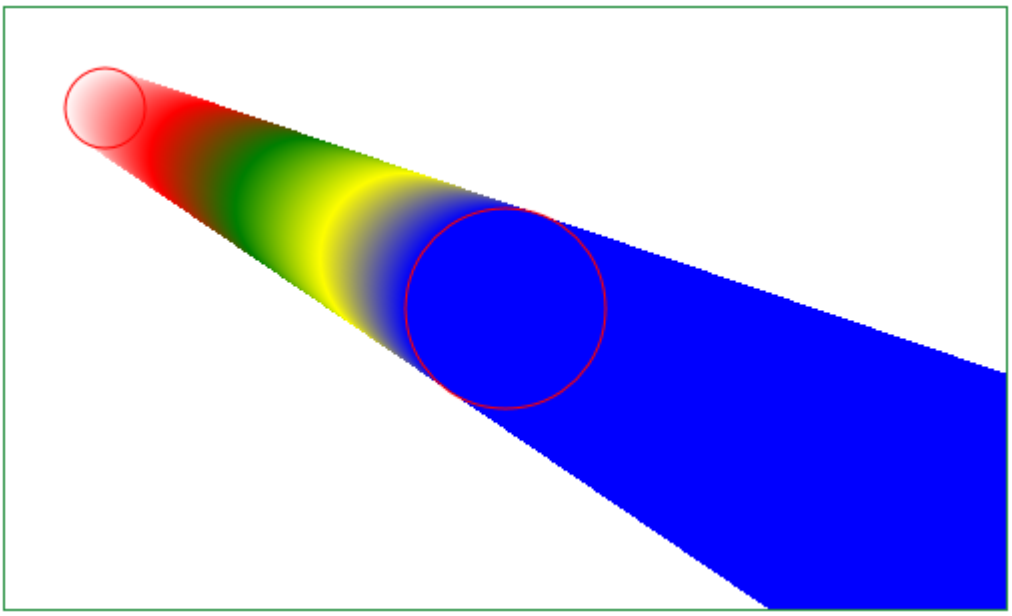
```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var w = cv.width;
var h = cv.height;
var grd = ctx.createRadialGradient(150, 150, 20, 250, 150, 150);
grd.addColorStop(0, "white");
grd.addColorStop(0.25, "red");
grd.addColorStop(0.5, "green");
grd.addColorStop(0.75, "yellow");
grd.addColorStop(1, "blue");
ctx.fillStyle = grd;
ctx.rect(0, 0, w, h);
ctx.fill();
```

Gradient kołowy (4)

Prostokąt pokrywający się wymiarami z canvas wypełniamy gradientem kołowym rozpościerającym się między małym białym kołem, a dużym kołem niebieskim.

Na lini łączącej środki kół umieszczono trzy kolory: czerwony, zielony i żółty.

Ponieważ kolor niebieski nie ma ogranicznika, rozciąga się do brzegów płótna, ale tylko w obrębie powierzchni wyznczonej przez proste - styczne do obu kół.



Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var w = cv.width;
var h = cv.height;
var grd = ctx.createRadialGradient(50, 50, 20, 250, 150, 50);
grd.addColorStop(0, "white");
grd.addColorStop(0.25, "red");
grd.addColorStop(0.5, "green");
grd.addColorStop(0.75, "yellow");
grd.addColorStop(1, "blue");
ctx.fillStyle = grd;
ctx.rect(0, 0, w, h);
ctx.fill();
```

Cienie

Cienie ustala się wykorzystując cztery właściwości:

- shadowColor - kolor cienia
- shadowBlur - stopień rozmycia cienia (liczby całkowite dodatnie lub ujemne)
- shadowOffsetX - przesunięcie cienia na osi X w stosunku do figury - liczby całkowite dodatnie lub ujemne
- shadowOffsetY - przesunięcie cienia na osi Y w stsunku do figury - liczby całkowite dodatnie lub ujemne

Są to właściwości kontekstu, a więc następna rysowana figura będzie miała cień z ostatnio podanymi właściwościami.

Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
ctx.rect(20, 20, 200, 100);
ctx.fillStyle = "#8EF6FF";
ctx.shadowColor = "#bbbbbb";
ctx.shadowBlur = 20;
ctx.shadowOffsetX = 15;
ctx.shadowOffsetY = 15;
ctx.fill();
ctx.shadowBlur = 5;
ctx.shadowOffsetX = 5;
ctx.shadowOffsetY = 5;
ctx.fillStyle = "black";
ctx.font = "1.5em sans-serif";
ctx.fillText("Ten napis ma delikatny cień ... ", 10, 225);
ctx.beginPath();
ctx.moveTo(140, 300);
ctx.quadraticCurveTo(440, 210, 340, 400);
ctx.lineWidth = 2;
ctx.strokeStyle = 'blue';
ctx.shadowColor = "red";
ctx.shadowBlur = 15;
ctx.shadowOffsetX = 5;
ctx.shadowOffsetY = 5;
ctx.stroke();
```

Przycinanie

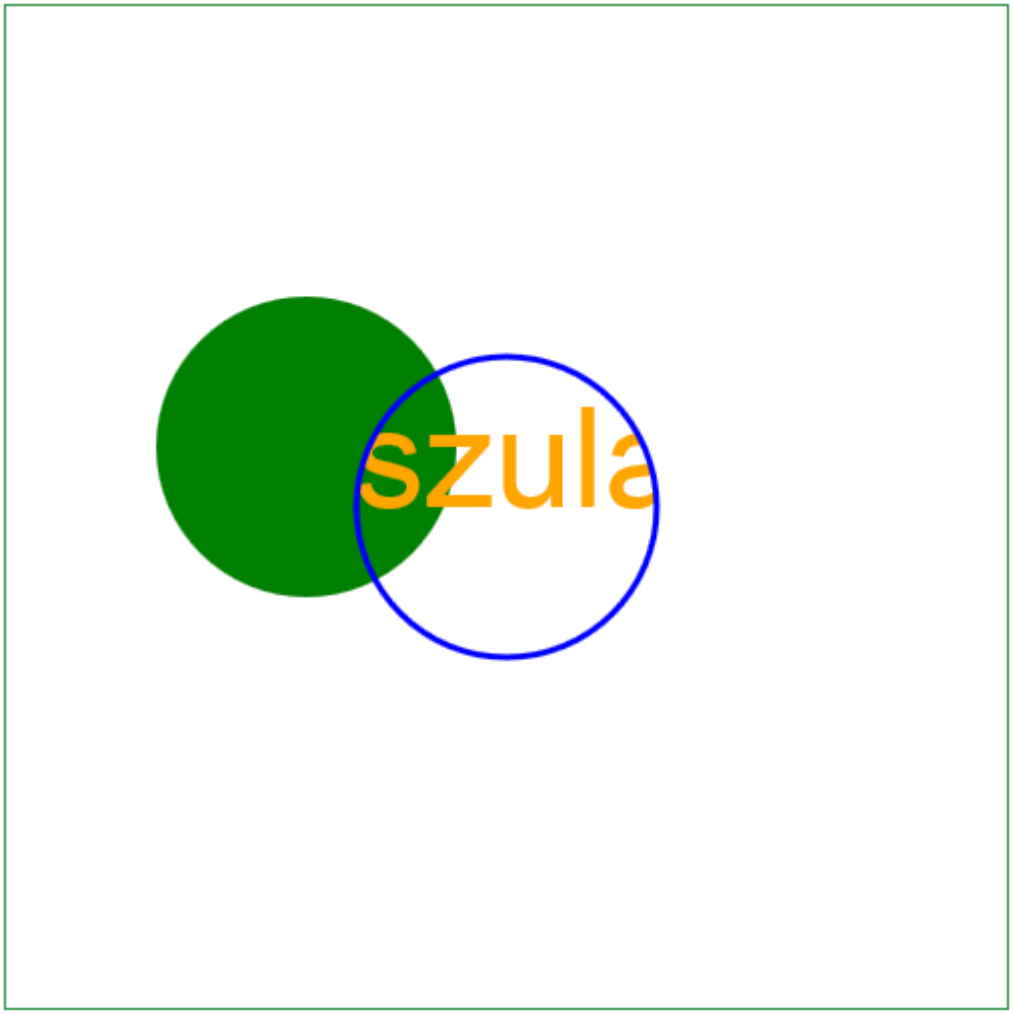
Przycinanie jest operacją określającą, co na obrazie będzie widoczne, a co nie będzie widoczne.

Wszystko co narysowane zostanie przed operacją maskowania - będzie widoczne 'pod spodem' i nie będzie maskowane (zielone koło).

Operacją rysowania na ostatniej ścieżce otwartej przed wydaniem polecenia `clip()` służy do ustalenia obszaru przycięcia (obszaru maski). Obszar widoczny (otwór maski) to obszar będący przecięciem (iloczynem logicznym) obszaru canvas i obszaru wyznaczanego przez ostatnią ścieżkę (w przykładzie jest to obszar koła). Reszta canvas jest niewidoczna.

Wszystko co narysujemy po wydaniu polecenia `clip()` , a przed `restore()` będzie rysowane (pomarańczowy napis) na canvas pod maską (widoczne lub nie - część niewidoczna nie jest tak naprawdę rysowana).

Jeśli chcemy rysować 'na wierzchu' (niebieski okrąg) musimy odtworzyć stan canvas przed wydaniem polecenia `save()`.



Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var x = 250;
var y = 250;
var radius = 75;
ctx.fillStyle="green";
ctx.arc(x-100, y-30, radius, 0, 2 * Math.PI, false);
ctx.fill();
ctx.save();
ctx.beginPath();
ctx.arc(x, y, radius, 0, 2 * Math.PI, false);
ctx.clip();
ctx.beginPath();
ctx.fillStyle = "orange";
ctx.font = "70px sans-serif";
ctx.fillText("Urszula", 100, 250);
ctx.restore();
ctx.beginPath();
ctx.arc(x, y, radius, 0, 2 * Math.PI, false);
ctx.lineWidth = 3;
ctx.strokeStyle = "blue";
ctx.stroke();
```

Praca z tekstem

Ustawienia czcionki

Dane czcionki podajemy w następującej kolejności:

"styl wariant grubość wielkość nazwa"

Każdą z wartości domyślnych określonych jako 'normal' można pominąć, np.
'normal normal normal 16px courier' możemy podać jako:
font='16px courier'

Rodzaj czcionki

Domyślną czcionką w znaczniku <canvas> jest czcionka 'sans-serif' o wielkości '10px'.

Jeśli rodzina nie jest podana - napis wyświetlony jest czcionką 'sans-serif'.

Przy określaniu czcionki można użyć nazwy konkretnej czcionki, np.

- times
- lucida console
- cambria
- courier new
- etc.

Możemy też podać nazwę rodziny czcionek i pozostawić przeglądarce wybór odpowiedniej konkretnej czcionki, np.

- 'serif' - czcionka szeryfowa

- 'sans-serif' - czcionka bezszeryfowa
- 'cursive' - kursywa
- 'fantasy' - czcionka fantazyjna
- 'monospace' - czcionka równoodstępowa

Przy określaniu możemy też podać nazwę konkretnej czcionki lub czcionek oraz na końcu, po przecinku, nazwę rodziny czcionek. Wówczas przeglądarka wyświetla napisy konkretną czcionką, ale jeśli jej nie znajdzie wyświeli dostepną czcionkę z podanej rodziny, np.

- 'times, serif'
- 'nieznana, monospace'
- 'nieznana nieznana, fantasy'

Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
ctx.font = "10px sans-serif";
ctx.fillText('10px san-serif', 20, 20);
ctx.font = "20px times";
ctx.fillText('20px times', 20, 40);
ctx.font = "20px lucida console";
ctx.fillText('20px lucida console', 20, 60);
ctx.font = "20px cambria";
ctx.fillText('20px cambria', 20, 80);
ctx.font = "20px courier new";
ctx.fillText('20px courier new', 20, 100);
ctx.font = "20px serif";
ctx.fillText('20px serif', 20, 140);
ctx.font = "20px sans-serif";
ctx.fillText('20px sans-serif', 20, 160);
ctx.font = "20px cursive";
```

```
ctx.fillText('20px cursive', 20, 180);
ctx.font = "20px fantasy";
ctx.fillText('20px fantasy', 20, 200);
ctx.font = "20px monospace";
ctx.fillText('20px monospace', 20, 220);
ctx.font = "20px times, serif";
ctx.fillText('20px times, serif', 20, 260);
ctx.font = "20px nieznana, monospace";
ctx.fillText('20px nieznana, monospace', 20, 280);
ctx.font = "20px nieznana nieznana, fantasy";
ctx.fillText('20px nieznana nieznana, fantasy', 20, 300);
```

Wielkość czcionki

Wielkość czcionki podajemy przed nazwą czcionki lub rodziny czcionek: '10px san-serif'. Jednostka powinna być podana po cyfrze bez spacji.

Jeśli wielkość czcionki nie jest podana to wyświetlana jest czcionka wielkości domyślnej '10px'

Wielkość czcionki możemy określić w jednostkach:

- px - bezwzględna wartość w pikselach wyświetlacza, np '12px'. Wielkość czcionki zmieni się jeśli użytkownik zmieni rozdzielczość ekranu, gdyż zmieni się liczba pikseli na cal.
- pt - bezwzględna wartośc w punktach (drukarskich), np '1pt'. '1pt' ma 1/72 cala, czyli 0,35 mm. Wielkość czcionki nie zmienia się przy zmianie wielkości monitora.
- em - względna wartość w stosunku do wielkości czcionki domyślnej w przeglądarce, np. '0.5em'. '1em' to wielkość aktualnie używanej czcionki.
- % - względna wartość w stosunku do wielkości czcionki domyślnej w przeglądarce.

Na stronach internetowych najczęściej wielkość czcionki podaje się w %, a wielkości marginesów i odstępów w 'em'-ach.

Jeśli czcionka domyślna przeglądarki ustawiona jest np. na 16px to ustawienie wielkości czcionki na '70%' spowoduje wyświetlenie napisu o wielkości 11 pikseli. Ustawienie na 0.9em w stosunku do 11 pikseli wyświetli napis o wielkości 10px.

Jeżeli użytkownik zmieni w przeglądarce wielkość fontu na stronie, a wielkości są ustawione na 'px' i 'pt' - to zmiany te nie odniosą skutku. Dlatego lepiej jest określać wielkość w '%' i 'em'-ach - po zmianie wielkości fontu w przeglądarce - zmieniają się wielkości napisów na monitorze.

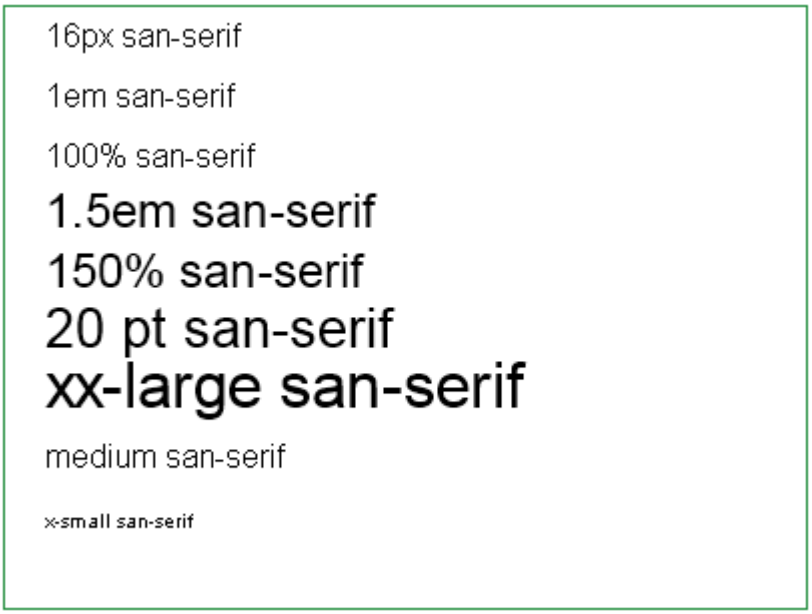
Jeśli jednak wymiary czcionki są ważne (np. mierzone) to lepiej używać jednostek bezwzględnych.

Wielkość czcionki można również określić używając określeń:

- 'xx-small'
- 'x-small'
- 'small'
- 'medium'
- 'large'

- 'x-large'
- 'xx-large'
- 'smaller' - określa wielkość czcionki względem fontu rodzica.
- 'larger' - określa wielkość czcionki względem fontu rodzica.

Powyższe, pierwsze siedem określeń określa czcionkę od najmniejszej do największej. Wszystkie te określenia są względne. Określają wielkość w stosunku do domyślnej wielkości czcionki ustawionej w przeglądarce.



Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
ctx.font = "16px sans-serif";
ctx.fillText('16px san-serif', 20, 20);
ctx.font = "1em sans-serif";
ctx.fillText('1em san-serif', 20, 50);
ctx.font = "100% sans-serif";
ctx.fillText('100% san-serif', 20, 80);
ctx.font = "1.5em sans-serif";
ctx.fillText('1.5em san-serif', 20, 110);
ctx.font = "150% sans-serif";
ctx.fillText('150% san-serif', 20, 140);
ctx.font = "20pt sans-serif";
ctx.fillText('20 pt san-serif', 20, 170);
ctx.font = "xx-large sans-serif";
ctx.fillText('xx-large san-serif', 20, 200);
ctx.font = "medium sans-serif";
ctx.fillText('medium san-serif', 20, 230);
ctx.font = "x-small sans-serif";
ctx.fillText('x-small san-serif', 20, 260);
```

Grubość czcionki

Grubość czcionki jest podawana przed wielkością czcionki i przed nazwą czcionki lub rodziny:
np. 'bold 18px lucida'

Jeżeli grubość czcionki nie jest określona tekst pisany jest czcionką o normalnej grubości.

Grubość czcionki może być podana jako jedna z wartości:
100, 200, 300, 400, 500, 600, 700, 800, 900
albo jako:

- normal (400) - wartość domyślna
- bold (700) - czcionka wytłuszczona
- bolder - czcionka o 100 bardziej wytłuszczona niż czcionka podstawowa
- lighter - czcionka o 100 mniej wytłuszczona niż czcionka podstawowa

16px san-serif

normal 16px san-serif

bold 16px san-serif

900 16px san-serif

bolder 16px san-serif

Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
ctx.font = "16px sans-serif";
ctx.fillText('16px san-serif', 20, 20);
ctx.font = "normal 16px sans-serif";
ctx.fillText('normal 16px san-serif', 20, 50);
ctx.font = "bold 16px sans-serif";
ctx.fillText('bold 16px san-serif', 20, 80);
ctx.font = "900 16px sans-serif";
ctx.fillText('900 16px san-serif', 20, 110);
ctx.font = "bolder 16px sans-serif";
ctx.fillText('bolder 16px san-serif', 20, 140);
```

Wariant czcionki

Wariant jest podawany przed grubością, wielkością i nazwą czcionki, np. 'normal bold 16px sans-serif'.

Można podać następujące wartości:

- 'normal' - czcionka normalna
- 'small-caps' - kapitaliki (drukowane litery)

Jeśli wariant nie jest podany domyślną wartością jest 'normal'.

normal normal 16px san-serif

SMALL-CAPS NORMAL 16PX SAN-SERIF

SMALL-CAPS BOLD 16PX SAN-SERIF

Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
ctx.font = "normal normal 16px sans-serif";
ctx.fillText('normal normal 16px san-serif', 20, 20);
ctx.font = "small-caps normal 16px sans-serif";
ctx.fillText('small-caps normal 16px san-serif', 20, 50);
ctx.font = "small-caps bold 16px sans-serif";
ctx.fillText('small-caps bold 16px san-serif', 20, 80);
```

Styl czcionki

Styl jest podawany przed wariantem, grubością, wielkością i nazwą czcionki, np. 'normal normal bold 16px sans-serif'.

Można podać następujące wartości:

- 'normal' - czcionka normalna
- 'italic' - kursywa
- 'oblique' - pochylony

Jeśli styl nie jest podany domyślną wartością jest 'normal'.

normal normal normal 16px san-serif

italic normal normal 16px san-serif

oblique normal normal 16px san-serif

Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
ctx.font = "normal normal normal 16px sans-serif";
ctx.fillText('normal normal normal 16px san-serif', 20, 20);
ctx.font = "italic normal normal 16px sans-serif";
ctx.fillText('italic normal normal 16px san-serif', 20, 50);
ctx.font = "oblique normal normal 16px sans-serif";
```

```
ctx.fillText('oblique normal normal 16px san-serif', 20, 80);
```

Kolor tekstu

Kolor tekstu ustawiamy właściwości `fillStyle` i `strokeStyle`.

Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
ctx.font = 'bold 90px Cambria';
ctx.fillStyle = 'red';
ctx.fillText('Smok', 10, 150);
```

Kontur i wypełnienie tekstu

Kontur i wypełnienie ustawiamy ustawiając właściwości `strokeStyle` i `fillStyle`.

Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
ctx.font = 'bold 60px Cambria';
ctx.fillStyle = 'red';
ctx.fillText('Smok', 10, 70);
ctx.strokeStyle = 'black';
ctx.strokeText('Smok', 10, 150);
ctx.fillText('Smok', 10, 220);
ctx.strokeText('Smok', 10, 220);
```

Deseń tekstu

Aby stworzyć deseń wczytujemy obrazek, tworzymy deseń używając polecenia createPattern(obrazek, sposób powielenia).

Następnie właściwości fillStyle przypisujemy utworzony deseń.

Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var image = new Image();
   image.onload = function() {
        var pattern = ctx.createPattern(image, 'repeat');
         ctx.fillStyle = pattern;
         ctx.font="192px san-serif";
         ctx.fillText("Smok", 50, 150);
         ctx.strokeText("Smok", 50, 150);
   };
   image.src = "images/cegla.png";
```

Tekst z gradientem

Tworzymy gradient, właściwość `fillStyle` ustawiamy na gradient.

Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var w = cv.width;
var h = cv.height;
var grd = ctx.createLinearGradient(0, 0, w,0);
grd.addColorStop(0, "white");
grd.addColorStop(1, "blue");
ctx.fillStyle=grd;
ctx.font="128px sans-serif";
ctx.fillText("Smok", 50, 200);
```

Wyrównywanie tekstu

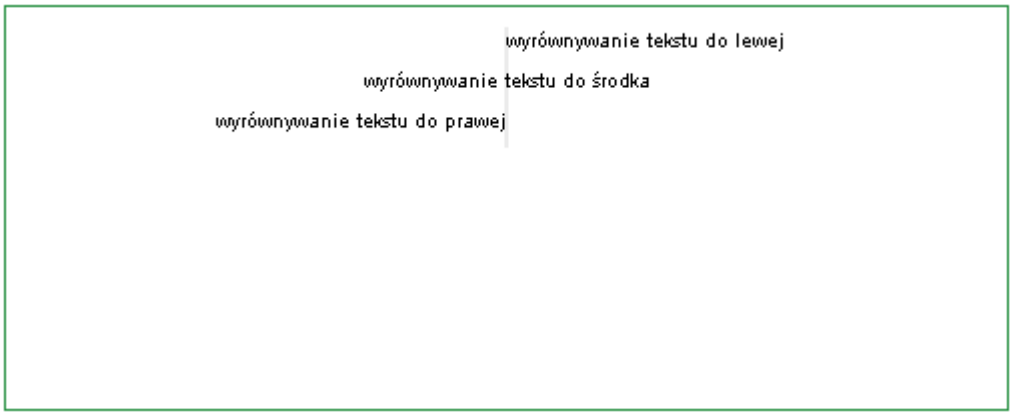
Wyrównywanie tekstu jest określane jako:

- start (domyślne)
- center
- end
- right
- left

Jeśli atrybut "dir" znacznika <canvas> jest równy "ltr", czyli teksty są pisane od lewej do prawej strony wówczas określenie 'start' = "left", a 'end' = 'right'.

Jeśli atrybut "dir" znacznika <canvas> jest równy "rtl", czyli teksty są pisane od prawej do lewej strony wówczas określenie 'start' = "right", a 'end' = 'left'.

Tekst jest wyrównywany w stosunku do miejsca (x,y) określającego położenie napisu względem <canvas>.



Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
ctx.beginPath();
ctx.strokeStyle = "#afafaf";
ctx.lineWidth = 0.5;
ctx.moveTo(250, 10);
ctx.lineTo(250, 70);
ctx.stroke();
ctx.beginPath();
ctx.fillStyle = 'black';
ctx.textAlign = "start";
ctx.fillText("wyrównywanie tekstu do lewej", 250, 20);
ctx.textAlign = "center";
ctx.fillText("wyrównywanie tekstu do środka", 250, 40);
ctx.textAlign = "end";
ctx.fillText("wyrównywanie tekstu do prawej", 250, 60);
```

Metryka tekstu

Na razie możemy mierzyć tylko szerokość tekstu.

Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var tekst = 'Smok';
ctx.font = '50px arial';
ctx.fillStyle = 'blue';
ctx.fillText(tekst, 10, 50);
var metryka = ctx.measureText(tekst);
```

```
var width = metryka.width;
ctx.font = '20px arial';
ctx.fillStyle = "black";
ctx.fillText("szerokość napisu "+ width + " px", 10, 90);
```

Linia tekstu

Linię tekstu 'textBaseline' można ustawić dla języków:

- 'hanging' - dla sanskrytu
- 'ideografic' - dla języków takich jak chiński czy japoński
- 'alphabetic' - dla języków z alfabetem łacińskim - linia przebiegająca pod małymi literami, np. o, m, r.

W obrębie języka można ustawić:

- 'top' - linia nad tekstem dotykająca do najwyższego elementu tekstu.
- 'middle' - linia przebiegająca przez środek tekstu
- 'bottom' - linia pod tekstem dotykająca do najniższego elementu tekstu

Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var tekst = 'Frog';
ctx.font = '50px arial';
ctx.fillStyle = 'blue';
ctx.fillText(tekst, 10, 50);
var metryka = ctx.measureText(tekst);
var width = metryka.width;
ctx.font = '20px arial';
ctx.fillStyle = "black";
ctx.textBaseline = "alphabetic";
ctx.font = "50px arial";
ctx.fillStyle = "blue";
ctx.textBaseline = "bottom";
ctx.fillText(tekst, 136, 50);
ctx.textBaseline = "middle";
ctx.fillText(tekst, 262, 50);
ctx.textBaseline = "top";
ctx.fillText(tekst, 388, 50);
ctx.lineWidth=1;
ctx.beginPath();
ctx.moveTo(10, 51);
ctx.lineTo(530, 51);
```

```
ctx.stroke();
ctx.fillStyle = "#afafaf";
ctx.font = "12px Arial";
ctx.fillText("alphabetic", 40, 60);
ctx.fillText("bottom", 176, 60);
ctx.fillText("midle", 302, 70);
ctx.fillText("top", 438, 100);
```

Zawijanie tekstu

Listing

```
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
var maxWidth = 400;
var lineHeight = 25;
var x = (cv.width - maxWidth) / 2;
var y = 60;
var text = "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin
euismod lacinia elit, ut luctus quam elementum sit amet. Pellentesque pulvinar
tempus est vel viverra. Vestibulum vulputate lobortis neque molestie iaculis.
Nullam convallis nibh sit amet pellentesque bibendum. Ut posuere laoreet metus, nec
finibus nisl posuere sed. Donec elit felis, viverra vitae posuere ut, blandit quis
ipsum. Pellentesque laoreet, lorem et fringilla volutpat, ante dui sodales erat,
non porttitor lacus lectus sit amet est. Phasellus imperdiet ante nec enim laoreet
tempor. Suspendisse accumsan dolor eu neque bibendum auctor. Pellentesque felis
risus, tincidunt a pharetra ac, placerat ac dui. Donec egestas, ligula eget
bibendum vestibulum, orci sem iaculis mauris, at blandit odio magna sed risus.
Maecenas vitae neque interdum, faucibus risus iaculis, malesuada justo. Duis felis
```

```
justo, elementum vel nisl et, volutpat mollis quam. Fusce efficitur semper enim vel
fermentum. ";
	ctx.font = '12pt';
	ctx.fillStyle = '#333';
	var words = text.split(' ');
	var line = '';
	for (var n = 0; n < words.length; n++) {
		var testLine = line + words[n] + ' ';
		var metrics = ctx.measureText(testLine);
		var testWidth = metrics.width;
		if (testWidth > maxWidth && n > 0) {
			ctx.fillText(line, x, y);
			line = words[n] + ' ';
			y += lineHeight;
		} else {
			line = testLine;
		}
	}
	ctx.fillText(line, x, y);
```

Transformacja tekstu

Tworzymy smoczą pieczęć :-)

Listing

```
	var cv = document.getElementById("canvas");
	var ctx = cv.getContext("2d");
	var image = new Image();

	image.src = "images/samsmok.png";
	ctx.fillStyle="#238E3E";
	ctx.font="bold 21px 'Courier New'";
	ctx.translate(150,150);
	ctx.rotate(-Math.PI/22.0);
	ctx.translate(-150,-150);
	image.onload = function() {
		ctx.drawImage(image, 85, 85, 120, 130);
	};
	var x=140;
	var y=250;
	var s= "Rok Smoka 23 stycznia 2012 - 9 lutego 2013";
```

```
for(var i=0; i<s.length;i++){
        ctx.fillText(s.substring(i,i+1),x,y);
        ctx.translate(150,150);
        ctx.rotate(-Math.PI/22.0);
        ctx.translate(-150,-150);
}
```

Encje

Istnienie rozlicznych stron kodowych powoduje problemy z wyświetlaniem znaków innych niż znaki języka angielskiego (ASCII). Podobny problem powstaje, gdy na stronie chcemy umieścić znaki, które używane są w standardzie HTML, CSS czy JavaScript.

Wtedy należy użyć encji, czyli znaków zastępujących przewidzianych dla HTML lub znaków zastępujących rozpoznawanych jako Unicode (UTF-8).

Encje są przydatne również, gdy chcemy umieścić w tekście znaki niedostępne z klawiatury.

Wybrane encje HTML oraz sekwencje Java/JavaScript i CSS dla polskich liter i innych znaków narodowych, interpunkcyjnych, matematycznych i specjalnych

Tab. Polskie litery

Litera	(X)HTML			Sekwencja Java/JavaScript	Sekwencja CSS
	Encja dziesiętna	Encja hex	Encja symboliczna		
ą	ą	ą	—	\u0105	\000105
ć	ć	ć	—	\u0107	\000107
ę	ę	ę	—	\u0119	\000119
ł	ł	ł	—	\u0142	\000142
ń	ń	ń	—	\u0144	\000144
ó	ó	ó	ó	\u00F3 \xF3	\0000F3
ś	ś	ś	—	\u015B	\00015B
ź	ź	ź	—	\u017A	\00017A
ż	ż	ż	—	\u017C	\00017C
Ą	Ą	Ą	—	\u0104	\000104
Ć	Ć	Ć	—	\u0106	\000106
Ę	Ę	Ę	—	\u0118	\000118
Ł	Ł	Ł	—	\u0141	\000141
Ń	Ń	Ń	—	\u0143	\000143
Ó	ò	ò	Ó	\u00F2 \xF2	\0000F2
Ś	Ś	Ś	—	\u015A	\00015A
Ź	Ź	Ź	—	\u0179	\000179
Ż	Ż	Ż	—	\u017B	\00017B

Tab. Niektóre inne litery

Litera	(X)HTML	Sekwencja	Sekwencja

	Encja dziesiętna	Encja hex	Encja symboliczna	Java/JavaScript	CSS
à	à	à	à	\u00E0 \xE0	\0000E0
ê	ê	ê	ê	\u00EA \xEA	\0000EA
À	À	À	À	\u00C0 \xC0	\0000C0
Ê	Ê	Ê	Ê	\u00CA \xCA	\0000CA
é	é	é	é	\u00E9 \xE9	\0000E9
É	É	É	É	\u00C9 \xC9	\0000C9
ọ	ọ	ọ	—	\u1ECD	\001ECD
ä	ä	ä	ä	\u00E4 \xE4	\0000E4
ö	ö	ö	ö	\u00F6 \xF6	\0000F6
ü	ü	ü	ü	\u00FC \xFC	\0000FC
ß	ß	ß	ß	\u00DF \xDF	\0000DF
Ä	Ä	Ä	Ä	\u00C4 \xC4	\0000C4
Ö	Ö	Ö	Ö	\u00D6 \xD6	\0000D6
Ü	Ü	Ü	Ü	\u00DC \xDC	\0000DC

Tab. Znaki diakrytyczne, których można użyć z literami (stawiane po literze)

Znak	(X)HTML Encja dziesiętna	Encja hex	Encja symboliczna	Sekwencja Java/JavaScript	Sekwencja CSS
́	́	́	—	\u0301	\000301
̨	̨	̨	—	\u0328	\000328
̇	̇	̇	—	\u0307	\000307
̈	̈	̈	—	\u0308	\000308
̀	̀	̀	—	\u0300	\000300
̧	̧	̧	—	\u0327	\000327
̂	̂	̂	—	\u0302	\000302
̃	̃	̃	—	\u0303	\000303
̊	̊	̊	—	\u030A	\00030A
̋	̋	̋	—	\u030B	\00030B
̌	̌	̌	—	\u030C	\00030C
̄	̄	İ	—	\u0304	\000304
̆	̆	̆	—	\u0306	\000306
̦	̦	̦	—	\u0326	000326
̣	̣	̣	—	\u0323	\000323

Tab. Spacje i znaki interpunkcyjne

Znak	(X)HTML Encja dziesiętna	Encja hex	Encja symboliczna	Sekwencja Java/JavaScript	Sekwencja CSS
				\u00A \xA0	\0000A0
				\u2002	\00202
				\u2003	\002003
			—	\u2004	\002004
			—	\u2005	\002005
			—	\u2006	\002006
				\u2009	\002009
			—	\u200A	\00200A
			—	\u200B	\00200B
–	–	–	–	\u2013	\002013
—	—	—	—	\u2014	\002014
				\u00AD \xAD	\0000AD
-	‐	‐	—	\u2010	\002010
-	‑	‑	—	\u2011	\002011
„	„	„	„	\u201E	\00201E
”	”	”	”	\u201D	\00201D
’	’	’	’	\u2019	\002019
«	«	«	«	\u00AB \xAB	\0000AB
»	»	»	»	\u00BB \xBB	\0000BB
"	"	"	"	\u0022 \x22	\000022
'	'	'	(')	\u0027 \x27	\000027
…	…	…	…	\u2026	\002026
–	‒	‒	—	\u2012	\002012

Tab. Symbole matematyczne i techniczne

Sym-bol	(X)HTML Encja dziesiętna	Encja hex	Encja symboliczna	Sekwencja Java/JavaScript	Sekwencja CSS
×	×	×	×	\u00D7 \xD7	\0000D7
÷	÷	÷	÷	\u00F7 \xF7	\0000F7
⁄	⁄	⁄	⁄	\u2044	\002044
∕	∕	∕	—	\u2215	\002215
−	−	−	−	\u2212	\002212
<	<	<	<	\u003C \x3C	\00003C
>	>	>	>	\u003E \x3E	\00003E
≪	≪	≪	—	\u226A	\00226A
≫	≫	≫	—	\u226B	\00226B

≮	≮	≮	—	\u226E	\00226E
≯	≯	≯	—	\u226F	\00226F
≈	≈	≈	≈	\u2248	\002248
≤	≤	≤	≤	\u2264	\002264
≥	≥	≥	≥	\u2265	\002265
≠	≠	≠	≠	\u2260	\002260
√	√	√	√	\u221A	\00221A
∛	∛	∛	–	\u221B	\00221B
‾	‾	‾	—	\u203E	\00203E
⌀	⌀	⌀	—	\u2300	\002300
∆	∆	∆	—	\u2206	\002206
∞	∞	∞	∞	\u221E	\00221E
Ω	Ω	Ω	—	\u2126	\002126
µ	µ	µ	µ	\u00B5 \xB5	\0000B5
‰	‰	‰	‰	\u2030	\002030
°	°	°	°	\u00B0 \xB0	\0000B0
′	′	′	′	\u2032	\002032
″	″	″	″	\u2033	\002033
¬	¬	¬	¬	\u00AC \xAC	\0000AC
∧	∧	∧	∧	\u2227	\002227
∨	∨	∨	∨	\u2228	\002228
⊻	⊻	⊻	—	\u22BB	\0022BB
∈	∈	∈	∈	\u2208	\002208
∉	∉	∉	∉	\u2209	\002209
∋	∋	∋	∋	\u220B	\00220B
∪	∪	∪	∪	\u222A	\00222A
∩	∩	∩	∩	\u2229	\002229
⊂	⊂	⊂	⊂	\u2282	\002282
⊄	⊄	⊄	⊄	\u2284	\002284
⊃	⊃	⊃	⊄	\u2283	\002283
∀	∀	∀	∀	\u2200	\002200
∃	∃	∃	∃	\u2203	\002203
∅	∅	∅	∅	\u2205	\002205
π	π	π	π	\u03C0	\0003C0
∟	∟	∟	—	\u221F	\00221F
∠	∠	∠	∠	\u2220	\002220
∡	∡	∡	—	\u2221	\002221
α	α	α	α	\u03B1	\0003B1
β	β	β	β	\u03B2	\0003B2

γ	γ	γ	γ	\u03B3	\0003B3
∥	∥	∥	—	\u2225	\002225
∦	∦	∦	—	\u2226	\002226

Tab. Ułamki

Znak	(X)HTML Encja dziesiętna	(X)HTML Encja hex	(X)HTML Encja symboliczna	Sekwencja Java/JavaScript	Sekwencja CSS
⅛	⅛	⅛	—	\u215B	\00215B
⅙	⅙	⅙	—	\u2159	\002159
⅕	⅕	⅕	—	\u2155	\002155
¼	¼	¼	¼	\u00BC \xBC	\0000BC
⅓	⅓	⅓	—	\u2153	\002153
⅜	⅜	⅜	—	\u215C	\00215C
⅖	⅖	⅖	—	\u2156	\002156
½	½	½	½	\u00BD \xBD	\0000BD
⅗	⅗	⅗	—	\u2157	\002157
⅝	⅝	⅝	—	\u215D	\00215D
⅔	⅔	⅔	—	\u2154	\002154
¾	¾	¾	¾	\u00BE \xBE	\0000BE
⅘	⅘	⅘	—	\u2158	\002158
⅚	⅚	⅚	—	\u215A	\00215A
⅞	⅞	⅞	—	\u215E	\00215E
⅟	⅟	⅟	—	\u215F	\00215F

Cyfry i ine znaki w górnym i dolnym indeksie

Znak	(X)HTML Encja dziesiętna	(X)HTML Encja hex	(X)HTML Encja symboliczna	Sekwencja Java/JavaScript	Sekwencja CSS
⁰	⁰	⁰	—	\u2070	\002070
¹	¹	¹	¹	\u00B9 \xB9	\0000B9
²	²	²	²	\u00B2 \xB2	\0000B2
³	³	³	³	\u00B3 \xB3	\0000B3
⁴	⁴	⁴	—	\u2074	\002074
⁵	⁵	⁵	—	\u2075	\002075
⁶	⁶	⁶	—	\u2076	\002076
⁷	⁷	⁷	—	\u2077	\002077

⁸	⁸	⁸	—	\u2078	\002078
⁹	⁹	⁹	—	\u2079	\002079
ⁿ	ⁿ	ⁿ	—	\u207F	\00207F
⁺	⁺	⁺	—	\u207A	\00207A
⁻	⁻	⁻	—	\u207B	\00207B
⁽	⁽	⁽	—	\u207D	\00207D
⁾	⁾	⁾	—	\u207E	\00207E
₀	₀	₀	—	\u2080	\002080
₁	₁	₁	—	\u2081	\002081
₂	₂	₂	—	\u2082	\002082
₃	₃	₃	—	\u2083	\002083
₄	₄	₄	—	\u2084	\002084
₅	₅	₅	—	\u2085	\002085
₆	₆	₆	—	\u2086	\002086
₇	₇	₇	—	\u2087	\002087
₈	₈	₈	—	\u2088	\002088
₉	₉	₉	—	\u2089	\002089

Tab. Strzałki

Znak	(X)HTML Encja dziesiętna	Encja hex	Encja symboliczna	Sekwencja Java/JavaScript	Sekwencja CSS
←	←	←	←	\u2190	\002190
↑	↑	↑	↑	\u2191	\002191
→	→	→	→	\u2192	\002192
↓	↓	↓	↓	\u2193	\002193
↔	↔	↔	↔	\u2194	\002194
↕	↕	↕	—	\u2195	\002195
↖	↖	↖	—	\u2196	\002196
↗	↗	↗	—	\u2197	\002197
↘	↘	↘	—	\u2198	\002198
↙	↙	↙	—	\u2199	\002199
⇆	⇆	⇆	—	\u21C6	\0021C6
⇐	⇐	⇐	⇐	\u21D0	\0021D0
⇑	⇑	⇑	⇑	\u21D1	\0021D1
⇒	⇒	⇒	⇒	\u21D2	\0021D2
⇓	⇓	⇓	⇓	\u21D3	\0021D3

⇔	⇔	⇔ ⇔	\u21D4	\0021D4
⇕	⇕	⇕ —	\u21D5	\0021D5
⇖	⇖	⇖ —	\u21D6	\0021D6
⇗	⇗	⇗ —	\u21D7	\0021D7
⇘	⇘	⇘ —	\u21D8	\0021D8
⇙	⇙	⇙ —	\u21D9	\0021D9
↵	↵	↵ ↵	\u21B5	\0021B5

Tab. Figury szachowe i kolory karciane

Znak	Encja dzie-siętna	(X)HTML Encja hex	Encja symboliczna	Sekwencja Java/JavaScript	Sekwencja CSS	Uwagi
♔	♔	♔	—	\u2654	\002654	Biały król
♕	♕	♕	—	\u2655	\002655	Biały hetman
♖	♖	♖	—	\u2656	\002656	Biała wieża
♗	♗	♗	—	\u2657	\002657	Biały goniec
♘	♘	♘	—	\u2658	\002658	Biały skoczek
♙	♙	♙	—	\u2659	\002659	Biały pionek
♚	♚	♚	—	\u265A	\00265A	Czarny król
♛	♛	♛	—	\u265B	\00265B	Czarny hetman
♜	♜	♜	—	\u265C	\00265C	Czarna wieża
♝	♝	♝	—	\u265D	\00265D	Czarny goniec
♞	♞	♞	—	\u265E	\00265E	Czarny skoczek
♟	♟	♟	—	\u265F	\00265F	Czarny pionek
♠	♠	♠	♠	\u2660	\002660	Piki (symbol wypełniony)
♥	♥	♥	♥	\u2665	\002665	Kiery (symbol wypełniony). Kolor uzyskany dzięki stylowi
♦	♦	♦	♦	\u2666	\002666	Kara (symbol wypełniony). Kolor uzyskany dzięki stylowi
♣	♣	♣	♣	\u2663	\002663	Trefle (symbol wypełniony)
♤	♤	♤	—	\u2664	\002664	Piki (kontur)
♡	♡	♡	—	\u2661	\002661	Kiery (kontur).

						Kolor uzyskany dzięki stylowi
♢	♢	♢	—	\u2662	\002662	Kara (kontur). Kolor uzyskany dzięki stylowi
♧	♧	♨	—	\u2668	\002668	Trefle (kontur)

Tab. Inne znaki specjalne

Znak	(X)HTML Encja dziesiętna	(X)HTML Encja hex	(X)HTML Encja symboliczna	Sekwencja Java/JavaScript	Sekwencja CSS
&	&	&	&	\u0026 \x26	\000026
§	§	§	§	\u00A7 \xA7	\0000A7
•	•	•	•	\u2022	\002022
‹	‹	‹	‹	\u2039	\002039
›	›	›	’	\u203A	\00203A
†	†	†	†	\u2020	\002020
‡	‡	‡	‡	\u2021	\002021
™	™	™	™	\u2122	\002122
®	®	®	®	\u00AE \xAE	\0000AE
©	©	©	©	\u00A9 \xA9	\0000A9
€	€	€	€	\u20AC	\0020AC
¢	¢	¢	¢	\u00A2 \xA2	\0000A2
£	£	£	£	\u00A3 \xA3	\0000A3
¥	¥	¥	¥	\u00A5 \xA5	\0000A5

Pomocnicze kody Java

Tworzenie tabel kolorów HSL

```
package canvas1;
import java.io.*;

public class HSL{
	public static final float[] hues = {0, 7.5f, 15, 22.5f, 30, 37.5f, 45,
			52.5f, 60, 75, 90, 105, 120, 150, 180, 210, 240, 255, 270, 285,
		300, 315, 330, 345};
	public static final String[] sats = {"0%", "10%", "20%", "30%", "40%",
			"50%", "60%", "70%", "80%", "90%", "100%"};

	public static void main(String[] args) {
		int len = hues.length;
		int len1 = sats.length;
		PrintWriter pw = null;
		try{
```

```
                pw = new PrintWriter("assets/hslcolors.html");
            } catch (FileNotFoundException e){
                e.printStackTrace();
            }
            pw.println("<!DOCTYPE html>");
            pw.println("<html lang=\"pl\">");
            pw.println("<head>");
            pw.println("<meta charset=\"utf-8\">");
            pw.println("</head>");
            pw.println("<body>");
            for(int i = 0; i < len; i++){
                pw.print("<table>");
                pw.println("<caption>Hue " + hues[i] + "&deg;</caption>");
                pw.println("<tr>");
                pw.println("<th scope=\"col\"> </th>");
                pw.println("<th colspan=\"12\" scope=\"col\">Lightness</th>");
                pw.println("</tr>");
                pw.println("<tr>");
                pw.println("<th rowspan=\"12\" scope=\"row\">Saturation</th>");
                pw.println("<td> </td>");
                pw.println("<td>0%</td>");
                pw.println("<td>10%</td>");
                pw.println("<td>20%</td>");
                pw.println("<td>30%</td>");
                pw.println("<td>40%</td>");
                pw.println("<td>50%</td>");
                pw.println("<td>60%</td>");
                pw.println("<td>70%</td>");
                pw.println("<td>80%</td>");
                pw.println("<td>90%</td>");
                pw.println("<td>100%</td>");
                pw.println("</tr>");
                for(int j = 0; j < len1; j++){
                    pw.println("<tr>");
                    pw.println("<td>" + sats[j] + "</td>");
                    for(int k = 0; k < len1; k++){
                        pw.println("<td style=\"background-color: hsl(" +
hues[i]
                                + "," + sats[j] + "," + sats[k]
                                        + ");\"> </td>");
                    }
                    pw.println("</tr>");
                }
                //-
                pw.println("</table>");
                pw.println("<p> </p>");
            }
            pw.println("</body>");
            pw.println("</html>");
            pw.flush();
            pw.close();
        }
}
```

Tworzenie tabel kolorów HWB

```
package canvas1;
import java.io.*;

public class HWB{
```

```
	public static final int[] hues = {0, 15, 30, 45, 60, 75, 90, 105, 120, 135,
		150, 165, 180, 195, 210, 225, 240, 255, 270, 285, 300, 315,
330,
		345};
	public static final String[] sats = {"0%", "10%", "20%", "30%", "40%",
		"50%", "60%", "70%", "80%", "90%", "100%"};

	public static void main(String[] args) {
		int len = hues.length;
		int len1 = sats.length;
		PrintWriter pw = null;
		try{
			pw = new PrintWriter("assets/hwbcolors.html");
		} catch (FileNotFoundException e){
			e.printStackTrace();
		}
		pw.println("<!DOCTYPE html>");
		pw.println("<html lang=\"pl\">");
		pw.println("<head>");
		pw.println("<meta charset=\"utf-8\">");
		pw.println("</head>");
		pw.println("<body>");
		for(int i = 0; i < len; i++){
			pw.print("<table>");
			pw.println("<caption>Hue " + hues[i] + "&deg;</caption>");
			pw.println("<tr>");
			pw.println("<th scope=\"col\"> </th>");
			pw.println("<th colspan=\"12\" scope=\"col\">Whiteness</th>");
			pw.println("</tr>");
			pw.println("<tr>");
			pw.println("<th rowspan=\"12\" scope=\"row\">Blackness</th>");
			pw.println("<td> </td>");
			pw.println("<td>0%</td>");
			pw.println("<td>10%</td>");
			pw.println("<td>20%</td>");
			pw.println("<td>30%</td>");
			pw.println("<td>40%</td>");
			pw.println("<td>50%</td>");
			pw.println("<td>60%</td>");
			pw.println("<td>70%</td>");
			pw.println("<td>80%</td>");
			pw.println("<td>90%</td>");
			pw.println("<td>100%</td>");
			pw.println("</tr>");
			for(int j = 0; j < len1; j++){
				pw.println("<tr>");
				pw.println("<td>" + sats[j] + "</td>");
				for(int k = 0; k < len1; k++){
					pw.println("<td style=\"background-color: hwb(" +
hues[i]
						+ "," + sats[j] + "," + sats[k]
						+ ");\"> </td>");
				}
				pw.println("</tr>");
			}
			//-
			pw.println("</table>");
			pw.println("<p> </p>");
		}
```

```
            pw.println("</body>");
            pw.println("</html>");
            pw.flush();
            pw.close();
        }
}
```

Tworzenie tabeli kolorów Web216

```
package canvas1;
import java.io.*;

public class Web216{
        public static final String[] web216 = {"00", "33", "66", "99", "CC", "FF"};

        public static void main(String[] args) {
                int len = web216.length;
                PrintWriter pw = null;
                try{
                        pw = new PrintWriter("assets/web216.html");
                } catch (FileNotFoundException e){
                        e.printStackTrace();
                }
                pw.println("<!DOCTYPE html>");
                pw.println("<html lang=\"pl\">");
                pw.println("<head>");
                pw.println("<meta charset=\"utf-8\">");
                pw.println("</head>");
                pw.println("<body>");
                pw.print("<table>");
                pw.print("<caption>Tab. Paleta Web Safe Colors</caption>");
                pw.println("<tr>");
                pw.println("<th scope=\"col\">Kolor</th>");
                pw.println("<th scope=\"col\">HEX</th>");
                pw.println("<th scope=\"col\">R</th>");
                pw.println("<th scope=\"col\">G</th>");
                pw.println("<th scope=\"col\">B</th>");
                pw.println("</tr>");
                for(int i = 0; i < len; i++){
                        for(int j = 0; j < len; j++){
                                for(int k = 0; k < len; k++){
                                        String hex =
"#".concat(web216[i]).concat(web216[j])
                                                .concat(web216[k]);
                                        String r =
String.valueOf(Integer.parseInt(web216[i], 16));
                                        String g =
String.valueOf(Integer.parseInt(web216[j], 16));
                                        String b =
String.valueOf(Integer.parseInt(web216[k], 16));
                                        pw.println("<tr>");
                                        pw.println("<td style= \"background-color: " + hex
                                                + "\"> </td>");
                                        pw.println("<td>" + hex + "</td>");
                                        pw.println("<td>" + r + "</td>");
                                        pw.println("<td>" + g + "</td>");
                                        pw.println("<td>" + b + "</td>");
                                        pw.println("</tr>");
```

```
                        }
                }
            }
            pw.println("</table>");
            pw.println("</body>");
            pw.println("</html>");
            pw.flush();
            pw.close();
        }
}
```

Klasa pomocnicza Util

```
package canvas1;
public class Util{
        private Util(){}

        /**
         * Zamienia polskie znaki w podanym łańcuchu znakowym
         * na encje nazwane HTML
         * @param str - łańcuch znaków
         * @return - zmieniony łańcuch znaków
         */
        public static String textToHTML2(String str) {
                String[] zn = {"ą", "ć", "ę", "ł", "ń", "ó", "ś", "ź", "ż", "Ą", "Ć",
                                "Ę", "Ł", "Ń", "Ó", "Ś", "Ź", "Ż"};
                String[] zs = {"&#261;", "&#263;", "&#281;", "&#322;", "&#324;",
                                "&#243;", "&#347;", "&#378;", "&#380;", "&#260;",
"&#262;",
                                "&#280;", "&#321;", "&#323;", "&#211;", "&#346;",
"&#377;",
                                "&#379;"};
                for(int i = 0; i < zn.length; i++){
                        if(str.contains(zn[i])){
                                str = str.replace(zn[i], zs[i]);
                        }
                }
                return str;
        }

        /**
         * Zamienia polskie znaki oraz niektóre inne znaki w podanym łańcuchu
         * znakowym na encje nazwane HTML
         * @param str - łańcuch znaków
         * @return - zmieniony łańcuch znaków
         */
        public static String textToHTMLFull(String str) {
                String[] zn = {"ą", "ć", "ę", "ł", "ń", "ó", "ś", "ź", "ż", "Ą", "Ć",
                                "Ę", "Ł", "Ń", "Ó", "Ś", "Ź", "Ż", "<", ">", "&", "'",
"\""};
                String[] zs = {"&#261;", "&#263;", "&#281;", "&#322;", "&#324;",
                                "&#243;", "&#347;", "&#378;", "&#380;", "&#260;",
"&#262;",
                                "&#280;", "&#321;", "&#323;", "&#211;", "&#346;",
"&#377;",
                                "&#379;", "&lt;", "&gt;", "&amp;", "&#039;", "&#034;"};
                for(int i = 0; i < zn.length; i++){
                        if(str.contains(zn[i])){
                                str = str.replace(zn[i], zs[i]);
                        }
```

```
		}
		return str;
	}

	/**
	 * Zamienia polskie znaki w podanym łańcuchu znakowym
	 * na sekwencje Unicode UTF-8
	 * @param str - łańcuch znaków
	 * @return - zmieniony łańcuch znaków
	 */
	public static String textToUTF(String str) {
		String[] zn = {"ą", "ć", "ę", "ł", "ń", "ó", "ś", "ź", "ż", "Ą", "Ć",
				"Ę", "Ł", "Ń", "Ó", "Ś", "Ź", "Ż"};
		String[] zs = {"\u0105", "\u0107", "\u0119", "\u0142", "\u0144",
				"\u00F3", "\u015B", "\u017A", "\u017C", "\u0104",
"\u0106",
				"\u0118", "\u0141", "\u0143", "\u00D3", "\u015A",
"\u0179",
				"\u017B"};
		for(int i = 0; i < zn.length; i++){
			if(str.contains(zn[i])){
				str = str.replace(zn[i], zs[i]);
			}
		}
		return str;
	}

	/**
	 * Zamienia polskie znaki oraz niektóre inne znaki w podanym łańcuchu
	 * znakowym na sekwencje Unicode UTF-8
	 * @param str - łańcuch znaków
	 * @return - zmieniony łańcuch znaków
	 */
	public static String textToUTFFull(String str) {
		String[] zn = {"ą", "ć", "ę", "ł", "ń", "ó", "ś", "ź", "ż", "Ą", "Ć",
				"Ę", "Ł", "Ń", "Ó", "Ś", "Ź", "Ż", "<", ">", "&", "'",
"\""};
		String[] zs = {"\u0105", "\u0107", "\u0119", "\u0142", "\u0144",
				"\u00F3", "\u015B", "\u017A", "\u017C", "\u0104",
"\u0106",
				"\u0118", "\u0141", "\u0143", "\u00D3", "\u015A",
"\u0179",
				"\u017B", "\u003C", "\u003E", "\u0024", "\u0027",
"\\u0022"};
		for(int i = 0; i < zn.length; i++){
			if(str.contains(zn[i])){
				str = str.replace(zn[i], zs[i]);
			}
		}
		return str;
	}
}
```

Proste narzędzie do testowania kodu canvas

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<style>
```

```
textarea {
        height: 580px;
        width: 480px;
        font-family: Consolas, Courier, monotype;
        float: left;
        margin: 10px;
}

iframe {
        height: 580px;
        width: 480px;
        float:left;
        margin: 10px;
}
pre{
        font-family: Consolas, Courier, monotype;
        font-size:0.75em;
        margin-left:1em;
        padding-left:1.07em;
}
.listing{
        font-family: Arial, serif;
        font-size:1.05em;
        font-weight:bold;
        color:#888888;
}
</style>
<script>
        var $ = function(id) {
                return document.getElementById(id);
        };
        var uruchom_click = function() {
                var code = $("code").value;
                var code1 = $("icode");
                code1.srcdoc = code;
        };
        window.onload = function() {
                $("uruchom").onclick = uruchom_click;
                var code = $("code").value;
                var code1 = $("icode");
                code1.srcdoc = code;
        };
</script>
</head>
<body>
        <section>
                <h4>Wprowadź kod i kliknij przycisk 'Uruchom':</h4>
                <textarea id="code" name="code">
&lt;!DOCTYPE html&gt;
&lt;html&gt;
&lt;body&gt;
&lt;canvas id="canvas" width="450" height="550"
                style="border: 1px solid #238E3E;"&gt;Zawartość możesz
zobaczyć w
przeglądarce obsługującej element &lt;canvas&gt;
z kontekstem &#034;2d&#034;&lt;/canvas&gt;
&lt;script type="text/javascript"&gt;
var cv = document.getElementById("canvas");
var ctx = cv.getContext("2d");
ctx.beginPath();
ctx.rect(50, 50, 200, 100);
```

```
ctx.fillStyle = "#0000ff";
ctx.fill();


&lt;/script&gt;
&lt;/body&gt;
&lt;/html&gt;
                </textarea>
                <iframe name="icode" id="icode" sandbox="allow-
scripts"></iframe>
                <br style="clear:both"/>
                <input type="button" id="uruchom" name="uruchom"
value="Uruchom">
        </section>
</body>
</html>
```

Dodatek 1. Przypomnienie z matematyki

Potęgowanie i pierwiastkowanie

Potęgowanie

a, b - liczba rzeczywista
n - liczba naturalna

$a^n = a * a * a * a$ (n-mnożeń)

$a^0 = 1$ (a $\neq$ 0)

$a^{-n} = \frac{1}{a^n}$ (a $\neq$ 0)

$(a^n)^m = a^{n*m}$

$a^n a^m = a^{n+m}$

$\frac{a^n}{a^m} = a^{m-n}$ (a $\neq$ 0)

$0^n = 0$ (n $\neq$ 0)

$(-a)^n = a^n$ (jeżeli n jest nieparzyste)

$(-a)^n = -a^n$ (jeżeli n jest parzyste)

$\left(\frac{a}{b}\right)^n = \frac{a^n}{b^n}$ (b $\neq$ 0)

$(ab)^n = a^n b^n$

Pierwiastkowanie

$\sqrt[n]{0} = 0$

$\sqrt[n]{1} = 1$

$\sqrt[n]{a} = a^{\frac{1}{n}}$ (a $\geq$ 0)

$$\frac{1}{\sqrt[n]{a}} = a^{-\left(\frac{1}{n}\right)} \quad (a > 0)$$

$$a^{\frac{n}{m}} = \sqrt[m]{a^n} = \left(\sqrt[m]{a}\right)^n$$
$$\sqrt[n]{ab} = \sqrt[n]{a} * \sqrt[n]{b}$$

$$\sqrt[m]{\sqrt[n]{a}} = \sqrt[n*m]{a}$$

$$\sqrt[n]{\frac{a}{b}} = \frac{\sqrt[n]{a}}{\sqrt[n]{b}}$$

Ponieważ w JavaScript jest jedynie funkcja do obliczania pierwiastka 2-stopnia, pierwiastki n-tego stopnia możesz obliczyć dzięki poniższemu wzorowi:
$$\sqrt[n]{a} = a^{\frac{1}{n}} \quad (a \geq 0)$$

Aby obliczyć $\sqrt[5]{a}$ obliczasz `Math.pow(a, 1/5)`.

Logarytmy

Logarytm $\log_a b$

b - liczba logarytmowana
a - podstawa logarytmu

$$\log_a b = c \leftrightarrow a^c = b$$
Czytamy: logarytm b przy podstawie a równa się c wtedy i tylko wtedy, gdy a do potęgi c równa się b.

$\log_2 8 = 3$ gdyż $2^3 = 8$
Czytamy: logarytm z 8 przy podstawie 2 równa się 3, gdyż 2 do 3-ciej równa się 8

$\log_2 16 = 4$, $16 = 2^4$

$\log_a 1 = 0$ (a>0, a $\neq$ 1)
$\log_a a = 1$ (a>0, a $\neq$ 1)
$\log_a (bd) = \log_a b + \log_a d$
$\log_a \frac{b}{d} = \log_a b - \log_a d$
$\log_a b^r = r \log_a b$ (r jest liczbą rzeczwistą)

Logarytm naturalny $\ln a$

Logarytm o podstawie e, gdzie e to stała Eulera =2.71828182845...

$\log_e x = \ln x$
$\ln x = \log_e x = 2{,}3026 \lg x$

Logarytm dziesiętny $\log_{10} a$ albo $\lg a$

$\log_{10} x = \lg x$
$\lg 1 = \lg 10^0 = 0$
$\lg 10 = \lg 10^1 = 1$
$\lg 100 = \lg 10^2 = 2$
$\lg b = \lg e * \ln b$
$\lg e \approx 0{,}434294$

Przeliczanie logarytmów

Jeżeli a>0 i a≠1 oraz b>0 i b≠1 a x >0, to

$$\log_a x = \frac{\log_b x}{\log_b a}$$

$$\log_a x = \frac{\ln x}{\ln a}$$

Przeliczanie logarytmów z dziesiętnych na naturalne i odwrotnie:

$$\lg x = \frac{1}{\ln 10} \ln x$$
$$\ln x = \ln 10 * \lg x$$

Logarytmy w JavaScript

`Math.LOG10E` = $\log_{10} e = \lg e$ 0.43429448190325181667
`Math.LN10` = $\log_e 10 = \ln 10$ = 2.30258509299404590 11
`Math.LN2` = $\log_e 2 = \ln 2$ =0.69314718055994528623
`Math.E` = 2.71828182845
`Math.log(x)` oblicza logarytm naturalny z x = $\log_e x = \ln x$

Ponieważ te oznakowania są mylące, proponuję używanie następujących funkcji:

ln

```
//logarytm naturalny z b
var ln = function(b){
      return Math.log(b);
};
```

lg

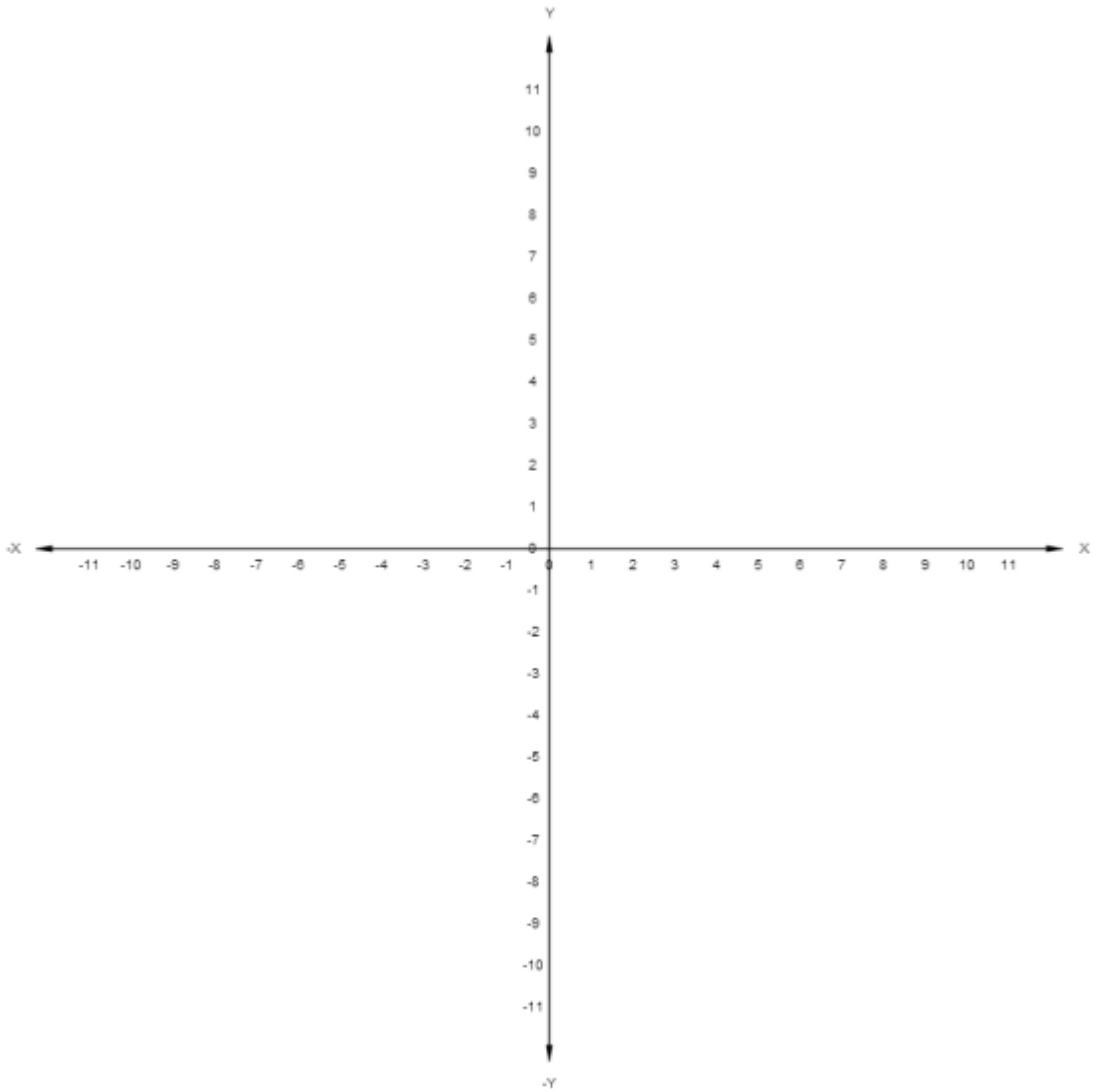
```
//logarytm dziesiętny z b
var lg = function(b){
      return Math.LOG10E*Math.log(b);
};
```

log

```
//logarytm o dowolnej podstawie także o podstawie 10 lub e
//logarytm z b przy podstawie a
var log=function(b, a){
      return Math.log(b)/Math.log(a);
};
```

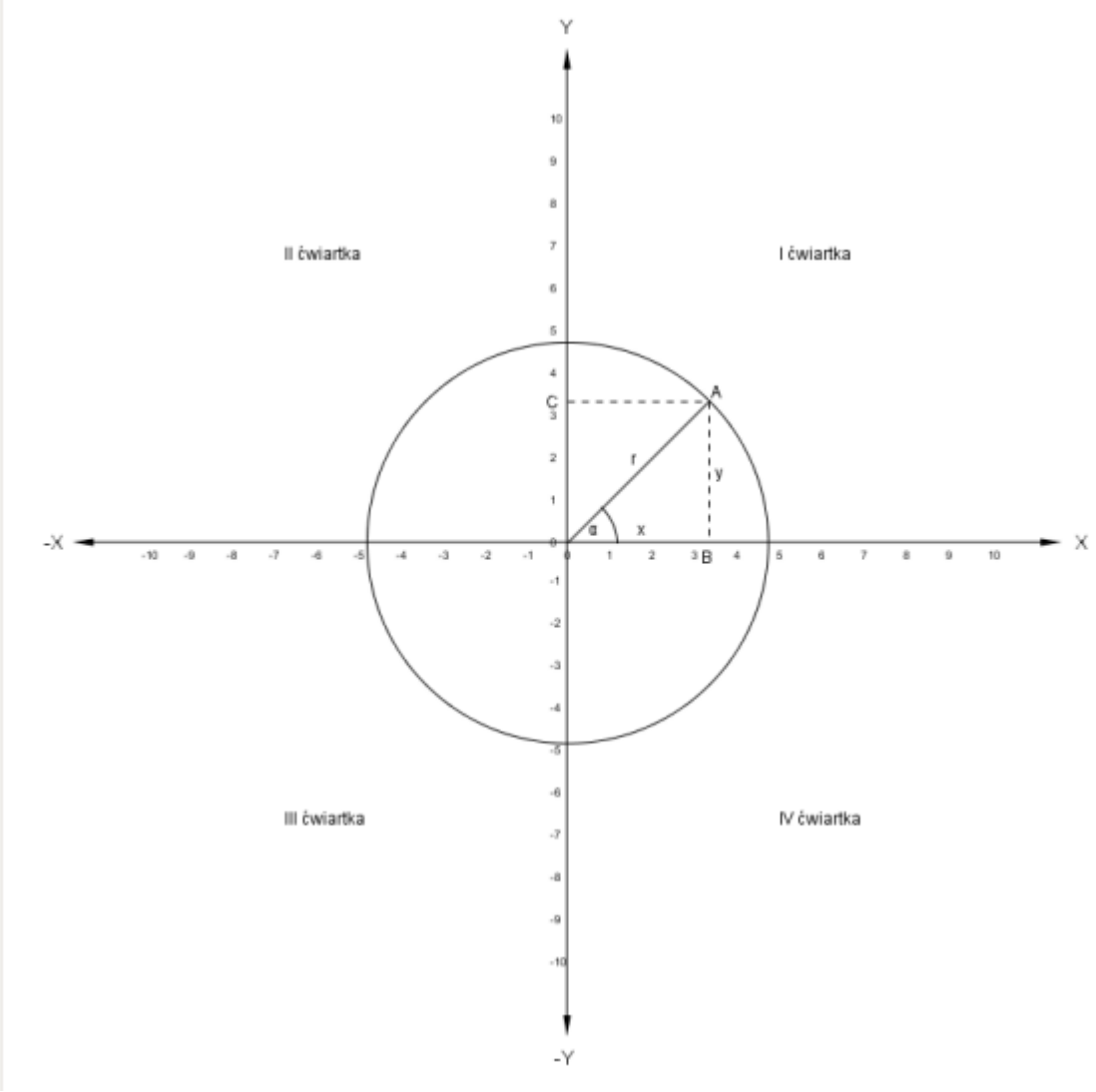
Trygonometria

Oto układ współrzędnych nazywany układem współrzędnych kartezjańskich:



Rys 1. (Dodatek 7 Listing24)

Układ dzieli się na 4 ćwiartki.
Na rysunku rysujemy punkt A, a następnie łączymy ten punkt ze środkiem linii współrzędnych. Punkt A rzutujemy na oś X i oś Y., oznaczając te punkty odpowiednio B i C. Odcinek 0B =AC nazwiemy x, odcinek 0C = AB y, a odcinek 0A - r. Kąt A0B nazwiemy α (alfa).
Na koniec wykreślmy koło o promieniu r, którego środek leży w środku układu współrzędnych. Otrzymujemy obrazek:



Rys. 2.

Jak widzimy odcinki 0A, AB i 0B tworzą trójkąt prostokątny Odcinki 0B (x) i BA (y) nazywamy przyprostokątnymi, odcinek 0A (r) przeciwprostokątną. Z twierdzenia Pitagorasa wiemy, że:

$x^2+y^2 = r^2$, czyli

$$r^2 = \sqrt{x^2 + y^2}$$

Miary kąta

Kąty używane w funkcjach trygonometrycznych mierzy sie w stopniach (°) lub radianach. Koło dzieli się na 360 stopni, a zatem 1° to 1/360 koła. Ćwiartka zajmuje 90°. Kąt 90° nazywany jest kątem prostym.
1° dzieli się na 60′ (minut kątowych)
1′ dzieli się na 60″ (sekund kątowych)
W żeglarstwie używa się rumbów. Rumb = 1/32*360° = 11°15′
W niektórych krajach używa się gradów = 1/100*90° = 0.9° = 54′

Kąty (w językach programowania) na ogół mierzy się w kierunku przeciwnym niż ruch wskazówek zegara, zaczynając od godz. 3.00 (oś X układu współrzędnych). W żeglarstwie (róża wiatrów) kąty mierzy się w kierunku zgodnym ze wskazówkami zegara, zaczynając od godz. 12.00.
W językach programowania znacznie częściej używa się radianów.
Obwód koła obliczamy według wzoru: $2\pi r$, gdzie:
π - to słynna liczba PI
r - to promień koła
Jeśli ze wzoru usuniemy r to otrzymamy
$2\pi = 360°$
$3/2\pi = 270°$
$\pi = 180°$
$\pi/2 = 90°$
$\pi/4 = 45°$
$\pi/6 = 30°$
Czytamy to następująco kąt 2 pi radianów = 360°, etc.
A zatem 1 radian = 180°/ π = 57°17′44″,80625
Radiany są szczególnie wygodne przy transformacjach. Wykonując obrót - π możesz potraktować jak stałą. Zmieniać będą się tylko cyferki przy π lub za π. Pokażemy to w innym miejscu tej książki.

Przeliczanie stopni na radiany i radianów na stopnie

Przeliczanie stopni na radiany i odwrotnie możesz wykonać używając poniższych funkcji:

```
var degToRad = function(deg) {
      return Math.PI * deg / 180.0;
};

var radToDeg = function(rad) {
      return rad * 180.0 / Math.PI;
};
```

Jeśli wolisz stopnie od radianów możesz używać następujących funkcji:

```
var sinDeg = function(angleDeg) {
      return Math.sin(angleDeg * Math.PI / 180);

};

var cosDeg = function(angleDeg) {
      return Math.cos(angleDeg * Math.PI / 180);
```

```
};

var tanDeg = function(angleDeg) {
      return Math.tan(angleDeg * Math.PI / 180);
};
```

Funkcje trygonometryczne kąta pełnego

sinus

$\sin\alpha = y/r$ dla $r \neq 0$

cosinus

$\cos\alpha = x/r$ dla $r \neq 0$

tangens

$tg\alpha = y/x$ dla $x \neq 0$

cotangens

$ctg\alpha = r/y$ dla $y \neq 0$
Rzadko używana funkcja. Jeśli jest potrzebna na ogół oblicza się $tg\alpha$, a następnie oblicza $1.0/tg\alpha$.

secans

$sec\alpha = r/x$ dla $x \neq 0$
Rzadko używana funkcja. Nie będzie dalej omawiana.

cosecans

$cosec\alpha = r/y$ dla $y \neq 0$
Rzadko używana funkcja. Nie będzie dalej omawiana.

Wzory podstawowe

$tg\alpha = \sin\alpha/\cos\alpha = 1/ctg\alpha$

$ctg\alpha = \cos\alpha/\sin\alpha = 1/tg\alpha$

$sec\alpha = 1/\cos\alpha = cosec\alpha/ctg\alpha$

$cosec\alpha = 1/\sin\alpha = sec\alpha/tg\alpha$

$\sin^2\alpha + \cos^2\alpha = 1;$

Suma kątów w trójkącie = 180°

Znaki funkcji

Ponieważ w ćwiartkach II, III i IV wartości współrzędnych mogą przyjmować wartości ujemne, w związku z tym również wartości funkcji mogą przyjmować wartości ujemne.

Ćwiartka	sin	cos	tg	ctg	sec	cos
I	+	+	+	+	+	+
II	+	-	-	-	-	+
III	-	-	+	+	-	-
IV	-	+	-	-	+	-

Zapamiętanie znaków pierwszych 4 funkcji ułatwia wierszyk:

"W I wszystkie są dodatnie, w II tylko sinus, w III tangens i cotangens, a w IV cosinus".

Wzory redukcyjne

kąt φ		sinφ	cosφ	tgφ	ctgφ
°	rad				
-α	-α	-sinα	cosα	-tgα	-ctgα
90° + α	π/2 + α	cosα	-sinα	-ctgα	-tgα
90° - α	π/2 - α	cosα	sinα	ctgα	tgα
180° + α	π + α	-sinα	-cosα	tgα	ctgα
180° - α	π - α	sinα	-cosα	-tgα	-ctgα
270° + α	3/2π + α	-cosα	sinα	-ctgα	-tgα
270° - α	3/2π - α	-cosα	-sinα	ctgα	tgα
360° + α	2π + α	sinα	cosα	tgα	ctgα
360° - α	2π - α	-sinα	cosα	-tgα	-ctgα

Przykłady:
cos(180° + α) = -cosα
tg(270° + α) = -ctgα
cos(π/2 + α) = -sinα

Jeżeli chciałbyś intensywniej zajmować się trygonometrią powinieneś zakupić tablice matematyczne np. wydawane przez wyd. Adamantan, gdzie znajdziesz mnóstwo wzorów pozwalających na uproszczenie obliczeń.

Funkcje cyklometryczne

Funkcje cyklometryczne to funkcje odwrotne do funkcji trygonometrycznych. W funkcjach trygonometrycznych podajesz kąt α i obliczasz np., sinα, cosα, tanα, ctgα. W funkcjach cyklometrycznych podajesz sinα, cosα, tanα, ctgα i obliczasz kąt, którego wartość funkcji podałeś.
Można je określić jednoznacznie tylko w tych przedziałach, w których funkcje te są różnowartościowe.
Funkcje te oznacza się przedrostkiem 'arc' (od łac. *arcus*)

arcsinx

Funkcją odwrotną do funkcji y = sinx, jest funkcja y = arcsinx (czytaj: arcus sinus x), gdzie: y jest miarą łukową kąta z przedziału: <-½ π , ½ π >., którego sin = x.
Funkcja jest określona i ciągła dla x ε <-1, 1>
Symbol ε - oznacza przynależność do zbioru (czytaj: ε leży w przedziale ...).
W JavaScript jest oznaczane jako `asin()`.
Wartość kąta podawana jest w radianach.
Jako argument funkcji możesz podać (zob. rys 2) albo wartość sinα albo stosunek y/r.

arccosx

Funkcją odwrotną do funkcji y = cosx, jest funkcja y = arccosx, gdzie:
y jest miarą łukową kąta z przedziału: <0, π >., którego cos = x.
Funkcja jest określona i ciągła dla x ε <-1, 1>
W JavaScript jest oznaczne jako `acos()`.
Wartość kąta podawana jest w radianach.
Jako argument funkci możesz podać (zob. rys. 2) albo wartość cosα albo x/r

arctgx

Funkcją odwrotną do funkcji y = tgx, jest funkcja y = arctgx, gdzie:
y jest miarą łukową kąta z przedziału: <-½ π , ½ π >., którego tg = x.
Funkcja jest określona i ciągła dla x ε R (zbiór liczb rzeczywistych) i rosnąca.
Ta funkcja jest najczęściej używana. W JavaScript i innych językach programowania na ogół występuje w dwóch wersjach (przyjrzyj się rys 2):

- `atan(x)`, gdzie podajesz x= tgα i otrzymujesz kąt α. Możesz też podać y/x
- `atan(y, x)` gdzie podajesz długości boków (y, x) trójkąta i otrzymujesz kąt α

Wartość kąta podawana jest w radianach.

arcctg

Funkcją odwrotną do funkcji y = ctgx, jest funkcja y = arcctgx, gdzie:
y jest miarą łukową kąta z przedziału: <0, π >., którego ctg = x.
Funkcja jest określona i ciągła dla x ε R.
W JavaScript nie występuje.

Gdybyś chciał wykonywać obliczenia w stopniach możesz użyć funkcji:

```
var atanDeg = function(ratio) {
      return Math.atan(ratio) * 180 / Math.PI;

};

var atan2Deg = function(yy, xx) {
      return Math.atan2(yy, xx) * 180 / Math.PI;
};

var asinDeg = function(ratio) {
      return Math.asin(ratio) * 180 / Math.PI;
};

var acosDeg = function(ratio) {
      return Math.acos(ratio) * 180 / Math.PI;
};
```

Funkcje hiperboliczne

Istnieje 6 funkcji hiperbolicznych, z których w praktyce programowania używa się jedynie tangensa hiperbolicznego tgh.

tgh

$$tgh = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

gdzie *e* jest liczbą Eulera (podstawa logarytmów naturalnych) = 2.718282... = `Math.E()`

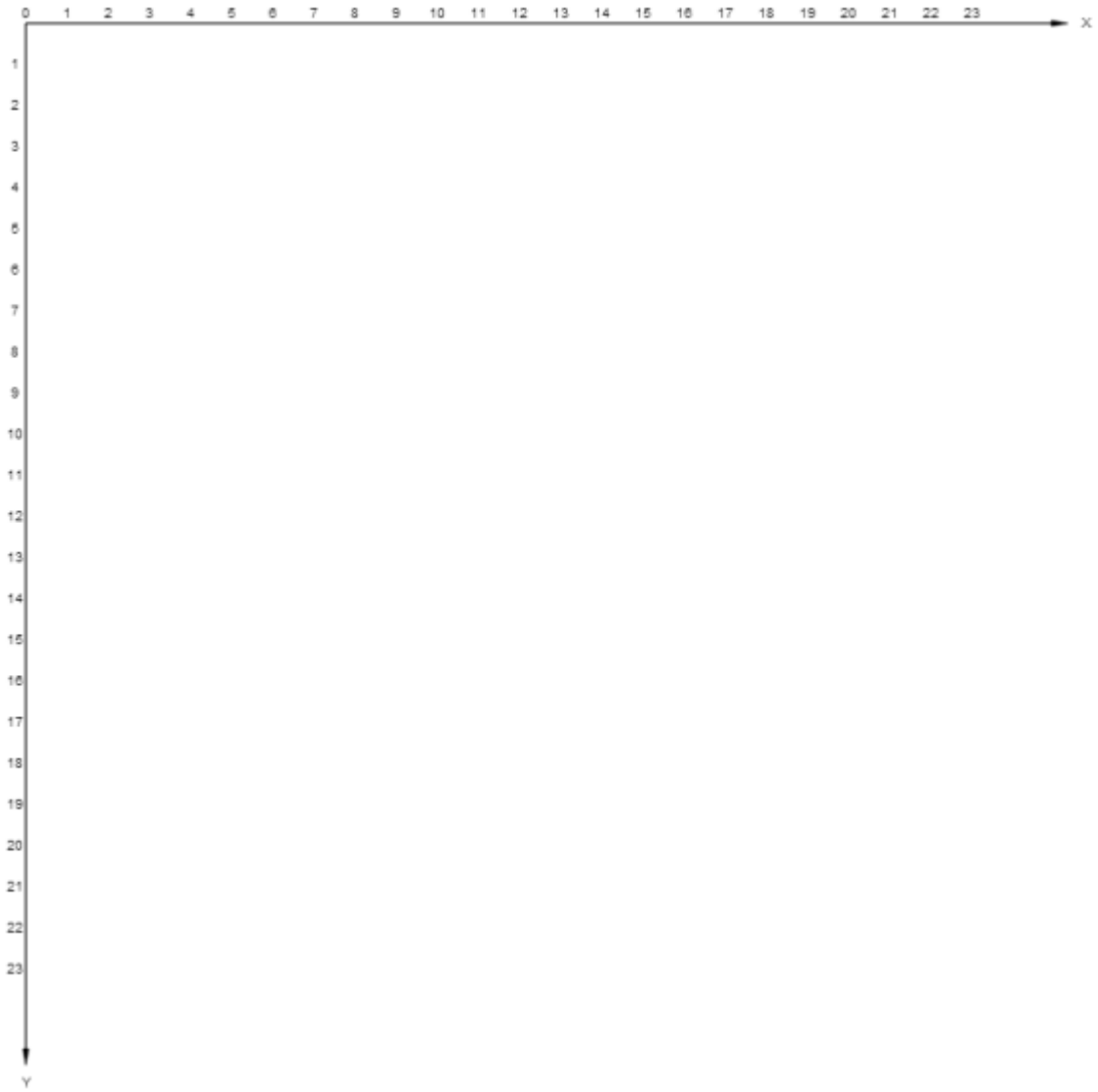
Wartość `tanh` możemy obliczyć używając funkcji:

```
var tanh = function(u) {
      var a = Math.exp(u);
      var b = Math.exp(-u);
      var c = (a - b) / (a + b);
      return c;
};
```

Wyliczona wartość funkcji zawiera się w przedziale <-1, 1>. Używana jest w obliczeniach przeprowadzanych w sieciach neuronowych.

Układ współrzędnych w JavaScript

Ponieważ ekran komputera byłoby bardzo niewygodnie dzielić na 4 ćwiartki, cały ekran



komputera znajduje się w pierwszej ćwiartce. Środek układu współrzędnych przypada w lewym górnym rogu ekranu. Liczby na osi X zwiększają się w prawo, a liczby na osi Y

zwiększają się w dół. To zapewnia wygodną obsługę i dobrą orientację w położeniach figur (Dodatek 7 Listing25).

Algorytm

```
var cv = document.getElementById('canvas');
var ctx = cv.getContext('2d');
var w = cv.width;
var h = cv.height;
var x0 = w / 2.0;
var y0 = h / 2.0;
var distx = x0 / 13;
var disty = y0 / 13;
var col = "black";
var col1 = "red";
var lw = 0.5;
var bw = 12;
var bh = 10;
var cl = true;
var dist = 30;
var m = 30;

function drawAxes(axes) {
	ctx.save();
	ctx.beginPath();
	ctx.fillStyle = "black";
	ctx.textBaseline = "middle";
	ctx.textAlign = "center";
	if (axes == "complex" || axes == "cartesian") {
		ctx.fillText("X", w - distx / 2, y0);
		ctx.fillText("-X", 0 + distx / 2, y0);
		ctx.fillText("Y", x0, 0 + disty / 2, y0);
		ctx.fillText("-Y", x0, h - disty / 2);
		for ( var i = 0; i < 12; i++) {
			ctx.fillText(i.toString(), x0 + i * m, y0 + 12);
		}
		for ( var k = 11; k > 0; k--) {
			ctx.fillText((-k).toString(), 99 + (10 - k) * m, y0 + 12);
		}
		if (axes == "complex") {
			ctx.fillStyle = "red";
			col1 = "red";
		}
		if (axes == "cartesian") {
			ctx.fillStyle = "black";
			col1 = "black";
		}
		for ( var j = 0; j < 12; j++) {
			ctx.fillText(j.toString(), x0 - 12, y0 - j * m);
		}
		for ( var n = 11; n > 0; n--) {
			ctx.fillText((-n).toString(), x0 - 12, y0 + n * m);
		}

		drawArrow(x0, y0, x0 - distx, lw, 0, bw, bh, cl, col);
		drawArrow(x0, y0, x0 - distx, lw, 180, bw, bh, cl, col);
		drawArrow(x0, y0, y0 - disty, lw, 90, bw, bh, cl, col1);
		drawArrow(x0, y0, y0 - disty, lw, 270, bw, bh, cl, col1);
	}
```

```
        if(axes=="js"){

                ctx.fillText("X", w - 12 , 15);
                ctx.fillText("Y", 15,  h - 12);
                for(var i = 0; i < 24; i++){
                        ctx.fillText(i.toString(), 15 + i * m , 8);
                }

                for(var j = 23; j > 0; j--){
                        ctx.fillText(j.toString(), 7, 15+ j * m);
                }
                drawArrow(15, 15, w - 40, lw, 0, bw,bh, cl, col);
                drawArrow(15,15, w - 40,lw, 270,bw,bh,cl,col);

        }
        ctx.restore();

};
```

Inne

2α oznacza podwojony kąt, np. sin2α jest sinusem podwojonego kąta α
$\sin^2\alpha$ oznacza $(\sin\alpha)^2$
Jeśli będziesz potrzebował innych wzorów powinieneś zaopatrzyć się w Tablice matematyczne.

Dodatek 2. Równania prostej

Linię prostą można wyrazić w postaci różnych równań. Nas interesują głównie dwie postaci:

- postać ogólna
- postać kierunkowa

Postać ogólna

Jest to równanie typu:

$Ax + By + C = 0,$

dla $A \neq 0$ lub $B \neq 0$.

Wyobraźmy sobie punkt A o współrzędnych Ax, Ay:

$A = (A_x, A_y) = (4, 3)$

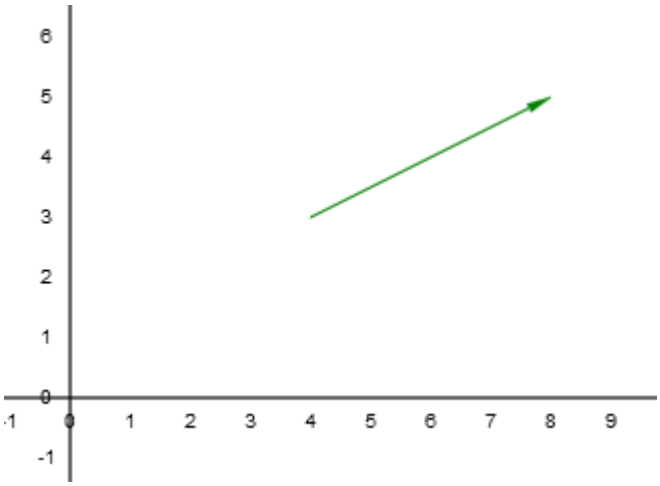
oraz punkt B o współrzędnych Bx, By:

$B = (B_x, B_y) = (8, 5).$

Można by je wyrazić w postaci wektora swobodnego (nie zaczepionego w punkie (0,0)), rozciągającego się od punktu A do punktu B:

$$V = \begin{bmatrix} B_x - A_x \\ B_y - A_y \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}$$

(Dodatek 7 Listing26)



Warto zauważyć, że gdyby ten wektor przesunąć tak, aby jego punkt zaczepienia przypadał w punkcie (0,0) to byłby to zwykły wektor V=[4,2].

Jeżeli przyjmiemy jakiś punkt, np, $(x_0, y_0) = (2,2)$ to

nasze równanie prostej przyjmie postać:

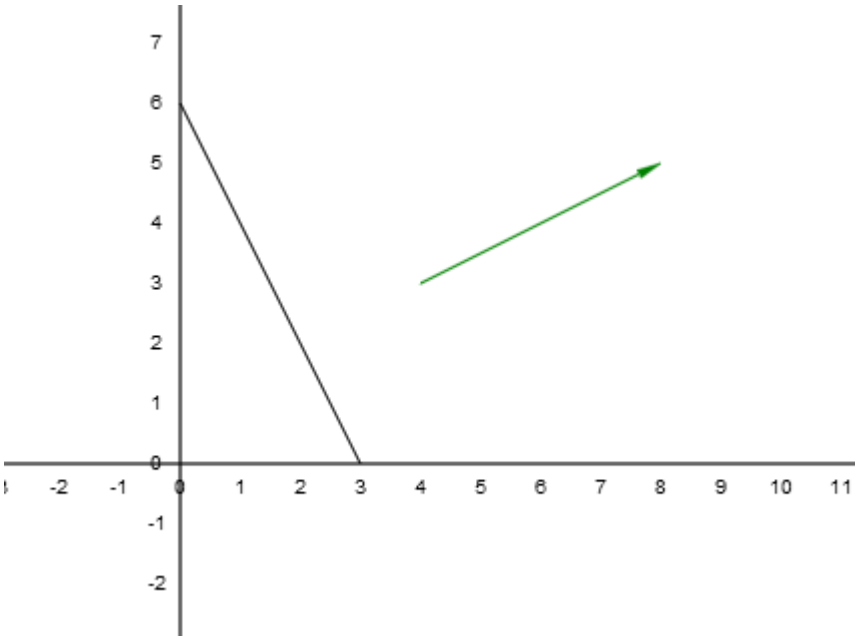
A = 4,
B = 2,
$C = -(Ax_0+By_0) = -(4*2+2*2) = -12$

czyli 4x + 2y - 12 =0,
czyli 2x + y - 6 = 0;(zwróć uwagę, że $\frac{A}{B} = \frac{2}{1} = 2$)
y = - 2x + 6,

Jeżeli x = 0, y = 6,
Jeżeli x = 1, y = 4,
Jeżeli x = 2, y = 2; (to nasz punkt (x_0, y_0)),
Jeżeli x = 3, y = 0, etc.

Jeżeli narysujemy prostą 2x + y - 6 = 0, to będzie ona przechodziła przez punkt (x_0, y_0) i będzie prostopadła do wektora swobodnego.

(Dodatek 7 Listing27).



Postać kierunkowa

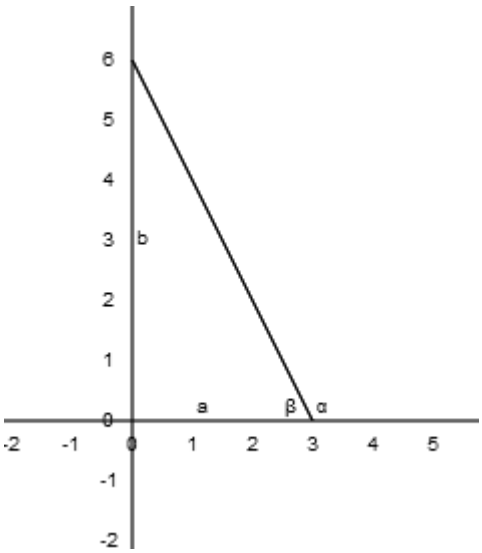
(Dodatek 7 Listing28)

Jest to równanie:

y = mx + b,

gdzie m = - $\frac{b}{a}$ = tgα (współczynnik kierunkowy), dla a ≠ 0.

Przyjrzyjmy się jeszcze raz linii prostej z poprzedniego rysunku.



w poprzednim równaniu doszliśmy do stanu:

y = -2x + 6,

$m = tg\alpha = tg(180°-\beta) = -tg\beta = -\frac{b}{a} = -\frac{6}{3} = -2$

b = 6

y = mx +b

Zwróć uwagę, że:

- Współczynnik b z równania kierunkowego oznacza miejsce przecięcia osi Y przez prostą (gdy x =0). Jeśli b = 0, prosta przechodzi przez środek układu współrzędnych.
- dwie proste, które mają identyczny współczynnik kierunkowy m są do siebie równoległe
- dwie proste, których m1 * m2 = -1 są do siebie prostopadłe
- Prosta nie może być opisana równaniem kierunkowym, gdy linia jest równoległa do osi Y, gdyż a = 0 i nie można wyliczyć współczynnika m.

Współczynnik kierunkowy prostej obliczamy przy użyciu metody:

```
Line.prototype.slope = function() {
	var xx1 = this.start.x - this.end.x;
	var yy1 = this.start.y - this.end.y;
	if (xx1 != 0.0) {
		return yy1 / xx1;
	} else {
		throw new Error("Nie moge dzielic przez 0");
	}
};
```

Współczynnik kierunkowy prostej przechodzącej przez dwa punkty obliczymy przy użyciu funkcji:

```
var slope = function(point1, point2) {
	var xx2 = 0.0;
	var yy2 = 0.0;
	if (point1 instanceof Vector2f && point2 instanceof Vector2f) {
		xx2 = point1.x - point2.x;
		yy2 = point1.y - point2.y;
	}
	if (xx2 != 0.0) {
		return yy2 / xx2;
	} else {
		throw new Error("linia nie może być równoległa do osi Y!");

	}
};
```

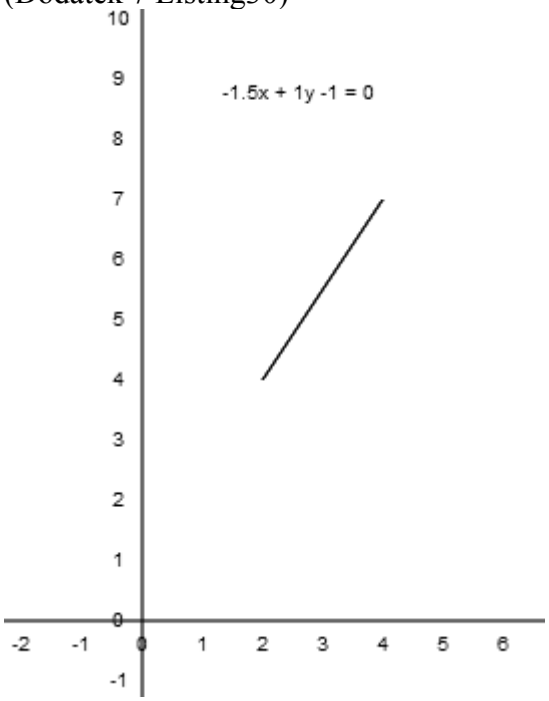
(Dodatek 7 Listing29)

Praca z obiektem Line

Współrzędne równania ogólnego możemy odczytać z obiektu Line i zapisać w postaci obiektu `Vector3f`:

```
var findEquation = function(line) {
	var m = line.slope();
	var a = -m;
	var b = 1;
	var c = m * line.start.x - line.start.y;
	return (new Vector3f(a, b, c));
};
```

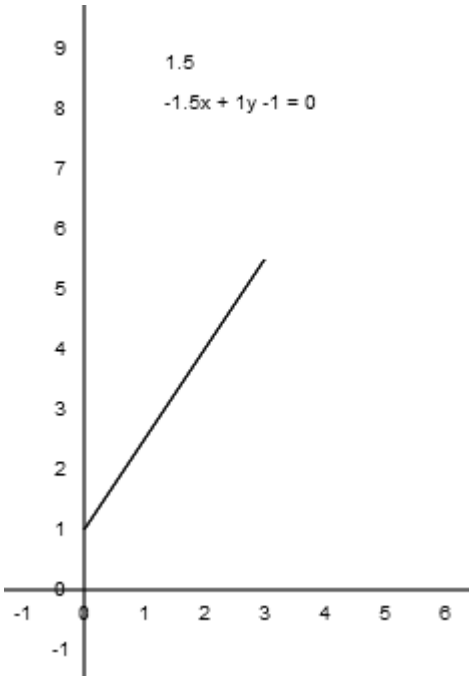
(Dodatek 7 Listing30)



Mając współczynniki równania ogólnego prostej możemy stworzyć obiekt `Line`:

(Dodatek 7 Listing30)

```
var findLine = function(a, b, c) {
	var x1 = 0;
	var x2 = 3;
	var y1 = -c / b - a / b * x1;
	var y2 = -c / b - a / b * x2;
	return new Line(new Vector2f(x1, y1), new Vector2f(x2, y2));
};
```



Jak widzimy jest to ta sama linia, tylko rozpięta między innymi punktami.

Równoległość prostych

Proste są równoległe, jeśli ich współczynniki kierunkowe są równe:

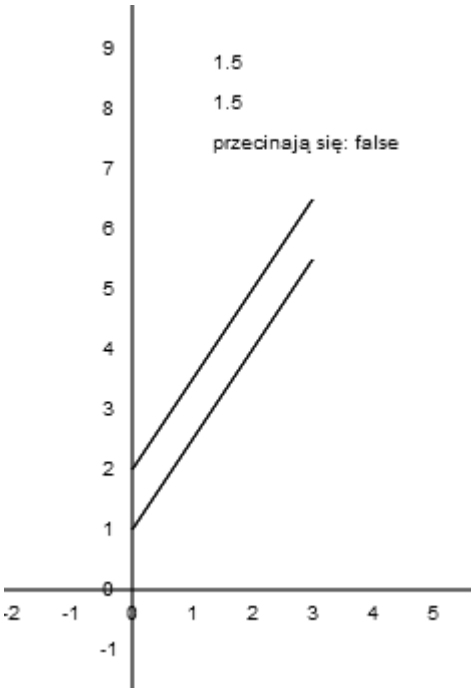
```
var areIntersect = function(line1, line2) {
      var m1 = line1.slope();
      var m2 = line2.slope();
      return (m1 != m2);
};
```

W przypadku równania ogólnego warunkiem równoległości prostych jest:

$A_1B_2 - A_2B_1 = 0$,

gdzie A i B to współczynniki równania ogólnego prostej.

(Dodatek 7 Listing32)



Odległość prostych równoległych

Odległość prostych równoległych wyrażonych równaniem ogólnym:

$$d=\frac{|C_2-C_1|}{\sqrt{A^2+B^2}},$$

gdzie A i B oznaczają współczynniki jednej z prostych.

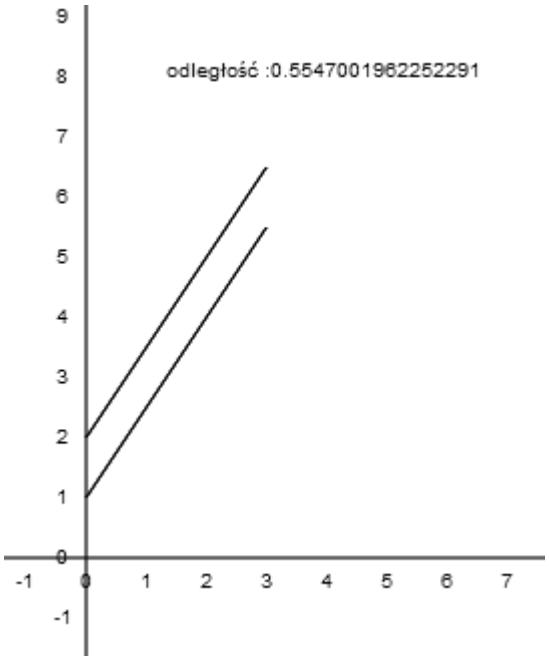
Odległość prostych równoległych wyrażonych równaniem kierunkowym:

$$d=\frac{|b_2-b_1|}{\sqrt{1+m^2}}$$

gdzie m - to współczynnik jednej z prostych.

```
var distance2 = function(line1, line2){
      var v1 =findEquation(line1);
      var v2 = findEquation(line2);
      var dist = Math.abs(v2.c-v1.c)/Math.sqrt(v1.a*v1.a+v1.b*v1.b);
      return dist;
};
```

(Dodatek 7 Listing33)



Prostopadłość prostych

Proste wyrażone równaniem kierunkowym są prostopadłe, jeśli:

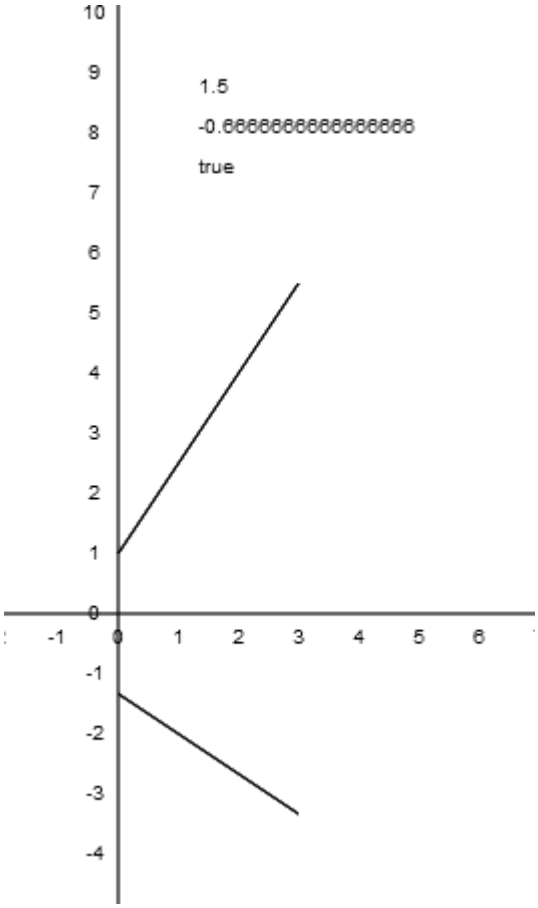
$m_1 * m_2 = -1$

Dla prostych wyrażonych równaniem ogólnym proste są prostopadłe jeśli:

$A_1A_2 + B_1B_2 = 0$

```
var areNormal = function(line1, line2) {
      var m1 = line1.slope();
      var m2 = line2.slope();
      return m1 * m2 == -1;
};
```

(Dodatek 7 Listing34)



Jak widzimy proste są prostopadłe, gdyż 1.5 * -0.6666 = -1.

Kąt między prostymi

Dla prostych, które nie są prostopadłe kąt między prostymi można obliczyć:

W przypadku równania ogólnego:

$$\operatorname{tg}\varphi = \frac{A_1B_2 - A_2B_1}{A_1A_2 + B_1B_2}$$

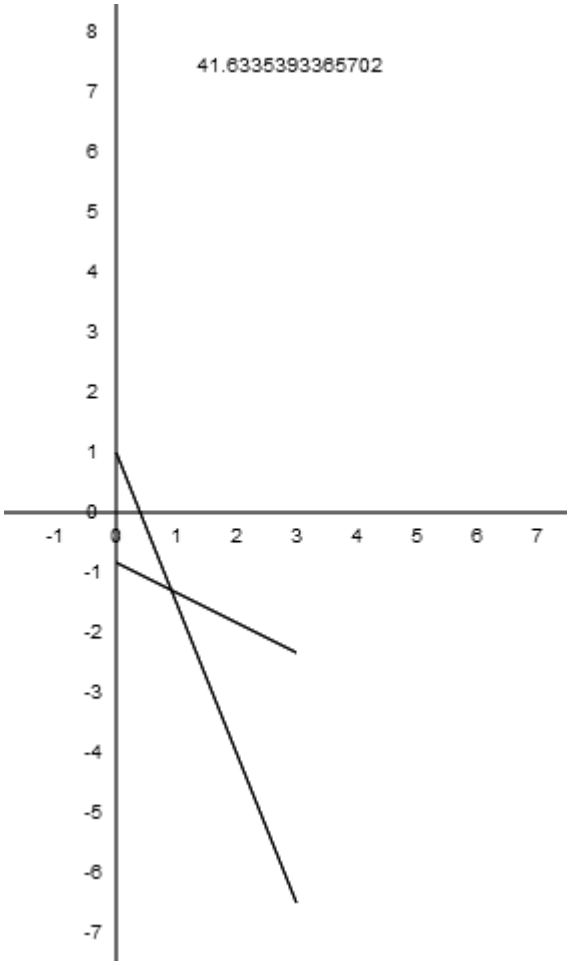
W przypadku równania kierunkowego:

$$\operatorname{tg}\varphi = \frac{m_2 - m_1}{1 + m_1m_2}$$

```
var angleBetween = function(line1, line2) {
      var m1 = line1.slope();
      var m2 = line2.slope();
      var angle = (m2 - m1) / (1 + m1 * m2);
      return atanDeg(angle);
};

var atan2Deg = function(yy, xx) {
```

```
        return Math.atan2(yy, xx) * 180.0 / Math.PI;
};
```

(Dodatek 7 Listing35)



Kąt między prostymi wynosi 41.6°.

Punkt przecięcia prostych

Punkt przecięcia P(x,y) dwóch prostych wyrażonych równaniami ogólnymi:

$$x = \frac{B_1C_2 - B_2C_1}{A_1B_2 - A_2B_1}$$

$$y = \frac{C_1A_2 - C_2A_1}{A_1B_2 - A_2B_1}$$

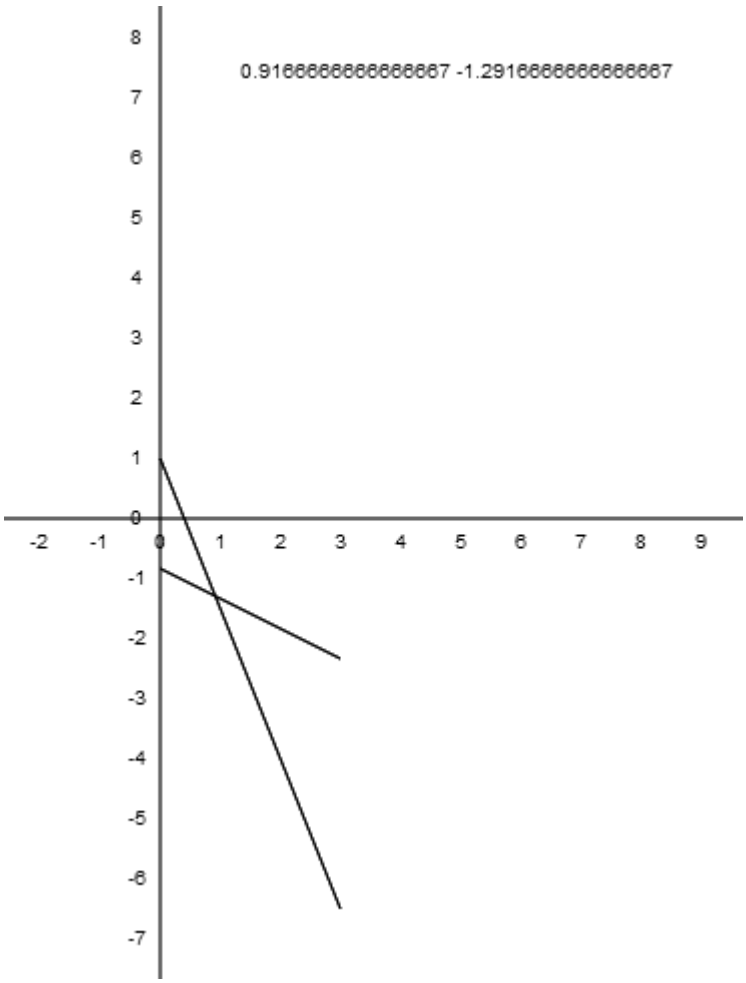
Punkt przecięcia P(x,y) dwóch prostych wyrażonych równaniami kierunkowymi:

$$x = \frac{b_2 - b_1}{m_2 - m_1}$$

$$y = \frac{m_2 b_1 - m_1 b_2}{m_2 - m_1}$$

```
var interceptPointL = function(line1, line2) {
	var xx = 0;
	var yy = 0;
	if (areIntersect(line1, line2)) {
		var v1 = findEquation(line1);
		var v2 = findEquation(line2);
		xx = (v2.b * v1.c - v1.b * v2.c) / (v2.a * v1.b - v1.a * v2.b);
		yy = (-v2.c - v2.a * xx) / v2.b;
	} else {
		return null;
	}
	return (new Vector2f(xx, yy));
};
```

(Dodatek 7 Listing36)




```
var interceptPointE = function(v1, v2) {
	var xx = 0;
	var yy = 0;
	var line1 = findLine(v1.a, v1.b, v1.c);
	var line2 = findLine(v2.a, v2.b, v2.c);
```

```
        if (areIntersect(line1, line2)) {
                xx = (v2.b * v1.c - v1.b * v2.c) / (v2.a * v1.b - v1.a * v2.b);
                yy = (-v2.c - v2.a * xx) / v2.b;
        } else {
                return null;
        }
        return (new Vector2f(xx, yy));
};
```

(Dodatek 7 Listing37)

Odległość punktu od prostej

Odległość punktu $P(x_0, y_0)$ od prostej wyrażonej równaniem ogólnym:

$$d = \frac{|Ax_0 + By_0 + C|}{\sqrt{A^2 + B^2}}$$

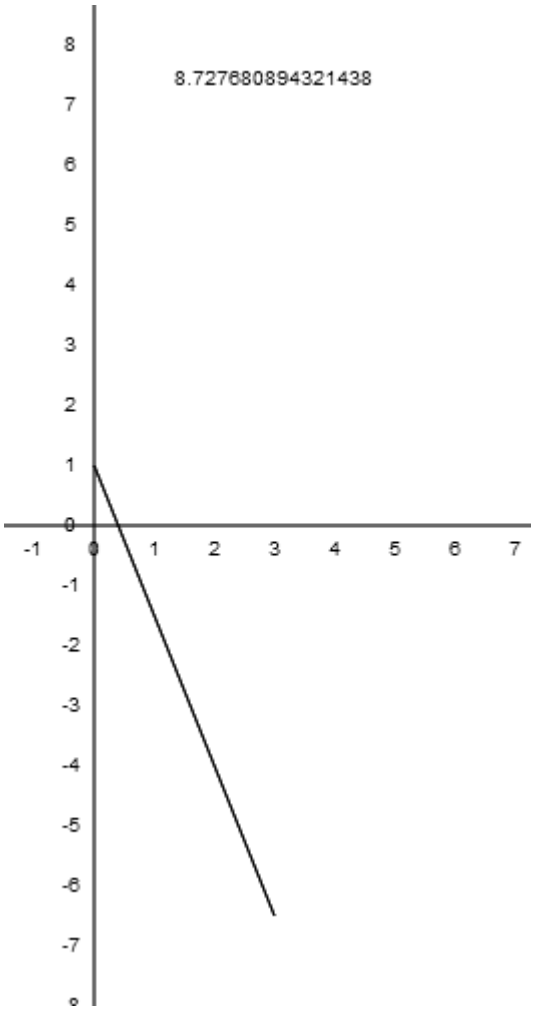
Odległość punktu $P(x_0, y_0)$ od prostej wyrażonej równaniem kierunkowym:

$$d = \frac{|mx_0 - y_0 + b}{\sqrt{1 + m^2}}$$

```
var distance3 = function(line, point) {
        var v = null;
        var d = 0.0;
        if ((line instanceof Line) && (point instanceof Vector2f)) {
                v = findEquation(line);
                d = (v.a * point.x + v.b * point.y + v.c)
                                / (Math.sqrt(v.a * v.a + v.b * v.b));
        }
        return d;
};
```

(Dodatek 7 Listing38)

Jak widzimy punkt P(7,7) jest odległy od linii prostej o 8,73.



Prosta prostopadła do danej prostej przechodząca przez punkt

Gdy znamy odległość punktu od prostej, w przypadku badania kolizji obiektów może być konieczne wyznaczenie prostej prostopadłej do danej prostej i przechodzącej przez punkt, którego odległość właśnie obliczyliśmy:

Jeżeli nasza prosta jest wyrażona równaniem kierunkowym:

$y = m_0x + b$

a nasz punkt P ma współrzędne $P(x_0, y_0)$,

to równanie prostej prostopadłej i przechodzącej przez ten punkt ma postać:

$y_1 = m_1x + y_0 - m_1x_0$

gdzie $m_1 = -\frac{1}{m_0}$

Jeżeli nasza prosta jest wyrażona równaniem ogólnym:

$Ax + By + C = 0$

a nasz punkt P ma współrzędne $P(x_0, y_0)$,

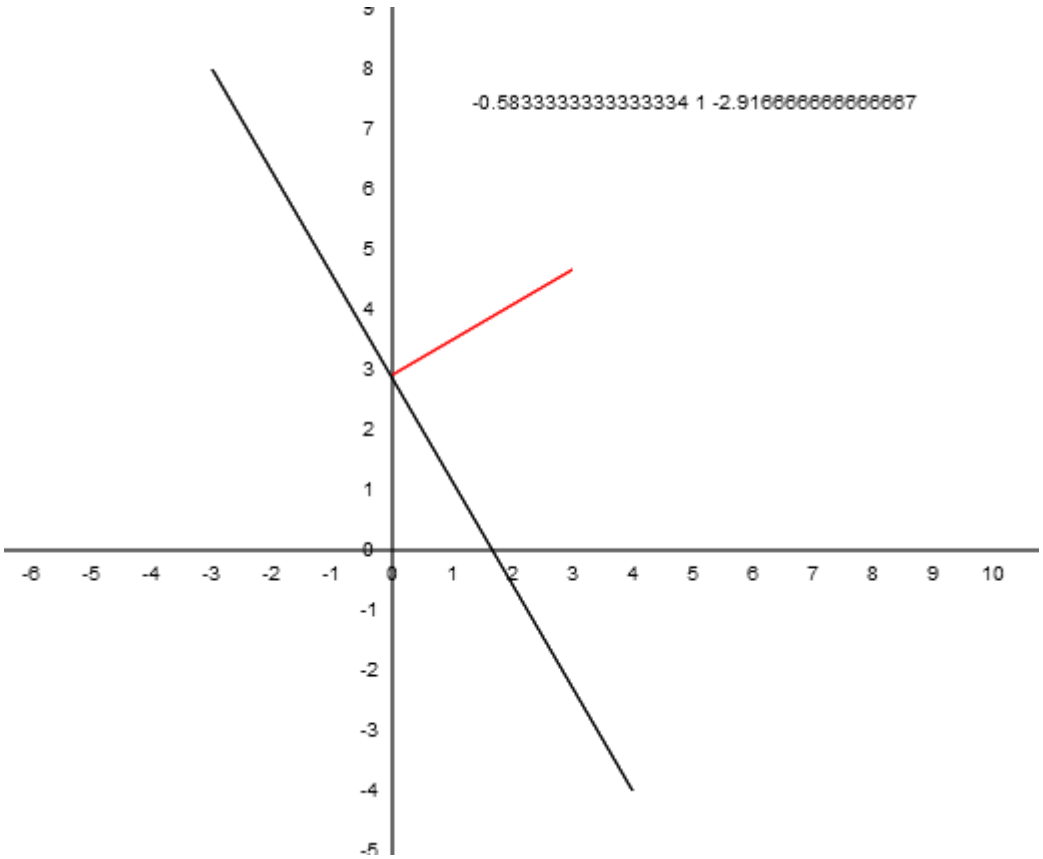
to równanie prostej prostopadłej do tej prostej i przechodzącej przez ten punkt ma postać:

$Bx - Ay + (Ay_0 - Bx_0) = 0;$

```
var lineThruPoint = function(line, point) {
      var line1 = null;
      if ((line instanceof Line) && (point instanceof Vector2f)) {
            var v3f = findEquation(line);
            line1 = findLine(v3f.b, -v3f.a, v3f.a * point.y - v3f.b * point.x);
      }
      return line1;
};
```

Teraz możemy obliczyć współrzędne punktu przecięcia (np. miejsce kolizji) według wzorów podanych wcześniej.

(Dodatek 7 Listing39)



Prosta prostopadła do danej i przechodząca przez punkt P(7, 7) jest wyrażona równaniem:

$-0.583x + y - 2.917 = 0;$

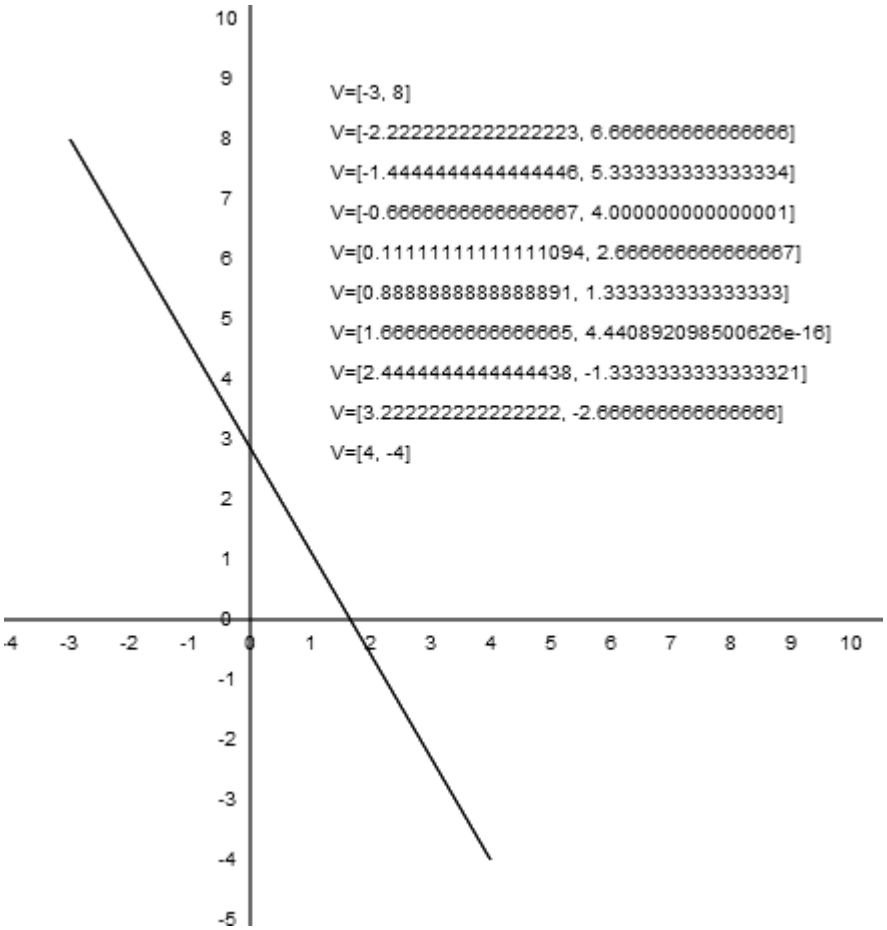
Wyznaczanie punktów na prostej

Czasami chcemy wyznaczyć daną liczbę punktów na prostej, zakładając, że punkty są równomiernie rozmieszczone:

```
var pointsOnLine = function(nPoints, point1, point2) {
	var tabl = new Array(nPoints);
	var delta = 1.0 / (nPoints - 1.0);
	var point = null;
	var ax = point1.x;
	var ay = point1.y;
	var bx = point2.x;
	var by = point2.y;
	for(var i = 0; i < nPoints; i++){
		var t = i * delta;
		var tt = 1.0 - t;
		var x = t * bx + tt * ax;
		var y = t * by + tt * ay;
		point = new Vector2f(x, y);
		tabl[i] = point;
	}
	return tabl;
};

var pointsOnLine = function(nPoints, line) {
	var tabl = new Array(nPoints);
	var delta = 1.0 / (nPoints - 1.0);
	var point = null;
	var ax = line.start.x;
	var ay = line.start.y;
	var bx = line.end.x;
	var by = line.end.y;
	for ( var i = 0; i < nPoints; i++) {
		var t = i * delta;
		var tt = 1.0 - t;
		var x = t * bx + tt * ax;
		var y = t * by + tt * ay;
		point = new Vector2f(x, y);
		tabl[i] = point;
	}
	return tabl;
};
```

(Dodatek 7 Listing40)

Wyznaczamy współrzędne 10 punktów leżących na linii:



Niezbędne klasy i metody

Aby móc korzystać z klasy `Line` potrzebne są klasy `Vector2f` i `Vector3f`. Sa to wektory swobodne, czyli wektory nie zaczepione w punkcie 0,0, ale wektory rozpięte pomiędzy dwoma punktami na płaszczyźnie. W przypadku wektorów typu `Vector3f` ich trzy współrzędne będą nam służyły do grupowania współrzędnych z równania linii.

```
var Vector2f = function(x, y) {
      switch (arguments.length) {
      case 0:
            this.x = 0.0;
            this.y = 0.0;
            break;
      case 2:
            this.x = arguments[0];
            this.y = arguments[1];
            break;
      }
};

var Vector3f = function() {
      switch (arguments.length) {
      case 0:
            this.a = 0.0;
            this.b = 0.0;
            this.c = 0.0;
```

```
            break;
        case 3:
            this.a = arguments[0];
            this.b = arguments[1];
            this.c = arguments[2];
            break;
        }
};
```

Metoda `distance()` pozwala na obliczenie odległości między dwoma wektorami lub dwoma punktami lub wektorem i punktem.
Aby narysować wektor swobodny użyjemy funkcji `drawVectorf()`:

```
function drawVectorf(x1, y1, x2, y2, axes, fillStyle) {
        ctx.save();
        ctx.beginPath();
        ctx.fillStyle = fillStyle;
        var radius = 0;
        var angle = 0;
        var vect = new Vector2f(x2 - x1, y2 - y1);
        radius = Math.sqrt(Math.pow(vect.y, 2) + Math.pow(vect.x, 2));
        angle = atan2Deg(vect.y, vect.x);

        if (axes == "complex" || axes == "cartesian") {
            drawArrow(x0 + x1 * 30, y0 - y1 * 30, radius * 30, lw, angle, bw, bh,
                        cl, fillStyle);

        }
        if (axes == "js") {
            drawArrow(x1 + 15, y1 + 15, radius * 30, lw, -angle, bw, bh, cl,
                        fillStyle);

        }
        if (axes == "norm") {
            drawArrow(x1, y1, arrowLength(x1, y1, x2, y2), lw, -arrowAngle(x1,
y1,
                        x2, y2), bw, bh, cl, fillStyle);
        }
        ctx.restore();
};
```

Zamiast tej funkcji w przypadku rysowania w elemencie <canvas> moglibyśmy użyć bezpośrednio metody `drawArrow()`.

Tworzymy klasę `Line`:

```
var Line = function() {
        switch (arguments.length) {
        case 2:
            if ((arguments[0] instanceof Vector2f)
                        && (arguments[1] instanceof Vector2f)) {
                this.start = arguments[0];
                this.end = arguments[1];
                this.length = this.start.distance(this.end);
                this.cosinus = (this.end.x - this.start.x) / this.length;
                this.sinus = (this.end.y - this.start.y) / this.length;
            }
```

```
            break;
        case 4:
            this.start = new Vector2f(arguments[0], arguments[1]);
            this.end = new Vector2f(arguments[2], arguments[3]);
            this.length = this.start.distance(this.end);
            this.cosinus = (arguments[0] - arguments[2]) / this.length;
            this.sinus = (arguments[1] - arguments[3]) / this.length;
            break;
        }
};
```

Linię będziemy rysowali przy użyciu metody `drawLine()`.

```
function drawLine() {
        ctx.save();
        ctx.beginPath();
        var x1 = 0.0;
        var y1 = 0.0;
        var x2 = 0.0;
        var y2 = 0.0;
        var axes = arguments[0];
        ctx.fillStyle = arguments[1];
        switch (arguments.length) {
        case 3:
            if (arguments[2] instanceof Line) {
                x1 = arguments[2].start.x;
                y1 = arguments[2].start.y;
                x2 = arguments[2].end.x;
                y2 = arguments[2].end.y;
            }
            break;
        case 4:
            if (arguments[2] instanceof Vector2f
                        && arguments[3] instanceof Vector2f) {
                x1 = arguments[2].x;
                y1 = arguments[2].y;
                x2 = arguments[3].x;
                y2 = arguments[3].y;
            }
            break;
        case 6:
            x1 = arguments[2];
            y1 = arguments[3];
            x2 = arguments[4];
            y2 = arguments[5];
            break;
        }
        if (axes == "complex" || axes == "cartesian") {
            ctx.moveTo(x0 + x1 * 30, y0 - y1 * 30);
            ctx.lineTo(x0 + x2 * 30, y0 - y2 * 30);

        }
        if (axes == "js") {
            ctx.moveTo(15 + x1 * 30, 15 + y1 * 30);
            ctx.lineTo(15 + x2 * 30, 15 + y2 * 30);
            ctx.stroke();

        }
        if (axes = "norm") {
```

```
            ctx.moveTo(x1, y1);
            ctx.lineTo(x2, y2);
            ctx.stroke();
      }
      ctx.restore();
};
```

Odległość między punktami obliczamy przy użyciu metody:

```
var distance = function() {
      var distance = 0.0;
      switch (arguments.length) {
      case 2:
            if (argument[0] instanceof Vector2f && arguments[1] instanceof
Vector2f) {
                  var x = arguments[0].x - arguments[1].x;
                  var y = arguments[0].y - arguments[1].y;
                  distance = Math.sqrt(x * x + y * y);
            }
            break;

      case 4:
            var x1 = arguments[0] - arguments[2];
            var y1 = arguments[1] - arguments[3];
            distance = Math.sqrt(x1 * x1 + y1 * y1);
            break;
      }
      return distance;

};
```

Dodatek 3. Wektory

Definicje podane w tym dziale niekoniecznie są ścisłymi definicjami matematycznymi. Są dobrane tak, aby pomóc zrozumieć w czym tkwi sedno rzeczy.

Skalary

Takie wielkości jak wiek, waga, wzrost są wielkościami skalarnymi (skalarami). Skalar jest jednowymiarowy, czyli wyrażany jedną liczbą. W algebrze liniowej może być uważany za wektor bez kierunku.
x= 7.5

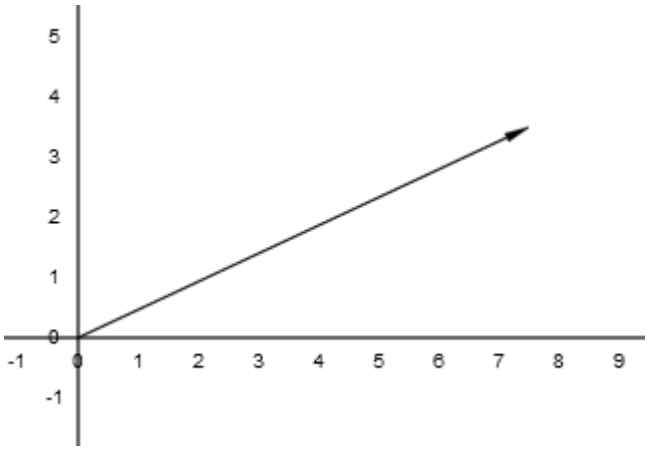
Wektory dwuwymiarowe 2d

Wektor jest wielkością posiadającą 2 lub więcej wymiarów. Jeżeli chcemy ustalić położenie punktu na mapie, czy planie miasta musimy użyć współrzędnych. Jeżeli chcielibyśmy z danego punktu przejść do innego również musimy podać dwie wielkości - odległość i kierunek: np. 962 m na północny wschód.
Wektor dwuwymiarowy, czyli 2d, posiada dwie współrzędne x i y i w tekście zapisywany jest:

V = [7.5, 3.5]

Graficzną reprezentacją wektora jest strzałka zaczepiona jednym końcem na przecięciu osi współrzędnych (0, 0) i skierowana do punktu, którego współrzędne podaje wektor (7.5, 3.5).

(Dodatek 7 Listing41)



Klasa opisująca wektor może wyglądać np. tak:

```
var Vector2d = function(x, y) {
	this.x = x;
	this.y = y;
};
```

Do wyświetlania wektora użyjemy metody:

(Dodatek 7 Listing42)

```
Vector2d.prototype.toString = function() {
	return "V=[" + this.x + ", " + this.y + "]";
};
```

Do porównywania, czy dwa wektory są równe użyjemy metody:

(Dodatek 7 Listing43)

```
Vector2d.prototype.equals = function(vect) {
	if (vect instanceof Vector2d) {
		return this.x == vect.x && this.y == vect.y;
	}
	return false;
};
```

Współrzędne kartezjańskie a biegunowe

Zamiast podawać położenie końca wektora możemy podać jego długość i kierunek (zob. rys 2 i opisy w dodatku poświęconym trygonometrii), czyli promień r okręgu, którego środek znajduje się w środku układu współrzędnych (0, 0) oraz kąt r w stosunku do osi X. r i α są to tzw. współrzędne biegunowe.
Do przeliczania współrzędnych z jednego systemu na drugi służy klasa `Polar` oraz dwie funkcje:

(Dodatek 7 Listing44)

```
var Polar = function(radius, angle) {
	this.radius = radius;
	this.angle = angle;
};
```

(Dodatek 7 Listing45)

```
Polar.prototype.toString = function() {
	return ("[r: " + this.radius + ",  \u03C6" + ": " + this.angle + "
		\u00BA]");
};
```

(Dodatek 7 Listing46)

```
Polar.prototype.equals = function(polar) {
	if (polar instanceof Polar) {
		return this.radius == polar.radius && this.angle == polar.angle;
	}
	return false;
};
```

(Dodatek 7 Listing47 i Listing48)

```
var cartToPolar = function(args) {
	var polar = null;
	switch (arguments.length) {
	case 1:
		if (arguments[0] instanceof Vector2d) {
			var radius = Math.sqrt(Math.pow(arguments[0].y, 2)
				+ Math.pow(arguments[0].x, 2));
			var angle = atan2Deg(arguments[0].y, arguments[0].x);
			polar = new Polar(radius, angle);
		}
		break;
	case 2:
		if ((typeof arguments[0] == "number")
				&& (typeof arguments[1] == "number")) {
			var radius1 = Math.sqrt(Math.pow(arguments[1], 2)
				+ Math.pow(arguments[0], 2));
			var angle1 = atan2Deg(arguments[1], arguments[0]);
			polar = new Polar(radius1, angle1);
		}
		break;
	}
	return polar;
};
```

(Dodatek 7 Listing49 i Listing50)

```
var polarToCart = function(args) {
	var vector = null;
	switch (arguments.length) {
	case 1:
		if (arguments[0] instanceof Polar) {
			var xx = arguments[0].radius * cosDeg(arguments[0].angle);
			var yy = arguments[0].radius * sinDeg(arguments[0].angle);
```

```
                vector = new Vector2d(xx, yy);
            }
            break;
        case 2:
            if ((typeof arguments[0] == "number")
                        && (typeof arguments[1] == "number")) {
                var xx1 = arguments[0] * cosDeg(arguments[1]);
                var yy1 = arguments[0] * sinDeg(arguments[1]);
                vector = new Vector2d(xx1, yy1);
            }
            break;
        }
        return vector;
};
```

Długość wektora

Długość wektora obliczamy przy użyciu twierdzenia Pitagorasa o bokach trójkąta prostokątnego:

$c^2 = a^2 + b^2$

Jeśli chcemy znaleźć długość naszego wektora używamy metody ((Dodatek 7 Listing51):

```
Vector2d.prototype.getLength = function() {
        var xx = Math.pow(this.x, 2);
        var yy = Math.pow(this.y, 2);
        return Math.sqrt(xx + yy);
};
```

Jeżeli chcemy ustawić długość wektora używamy metody ((Dodatek 7 Listing52):

```
Vector2d.prototype.setLength = function(length){
        var r = this.getLength();
        if(r){
            this.scale(length/r);
        }else{
            this.x = length;
        }
};
```

Zmienna `length` w wektorze jest równa zmiennej `radius` w 'polarze'.

Kąt wektora

(Dodatek 7 Listing53)

Jeśli chcemy znaleźć kąt wektora - używamy metody:

```
Vector2d.prototype.getAngle = function() {
        var angle = atan2Deg(this.y, this.x);
        return angle;
};
```

Jeśli chcemy ustawić kąt wektora używamy metody:

```
Vector2d.prototype.setAngle = function(angle){
	var r = this.getLength();
	this.x = r*cosDeg(angle);
	this.y = r*sinDeg(angle);
};
```

Zmienna `angle` w wektorze jest równa zmiennej `angle` w 'polarze'.

Dodawanie wektorów

Aby dodać dwa wektory dodajemy współrzędną x jednego wektora, do wpółrzędnej x drugiego wektora, a współrzędna y pierwszego do współrzędnej y drugiego wektora.
W procesie dodawania albo zmieniamy jeden z wektorów i nadal mamy dwa wektory albo tworzymy trzeci wektor będący sumą dodawanych wektorów:

(Dodatek 7 Listing54)

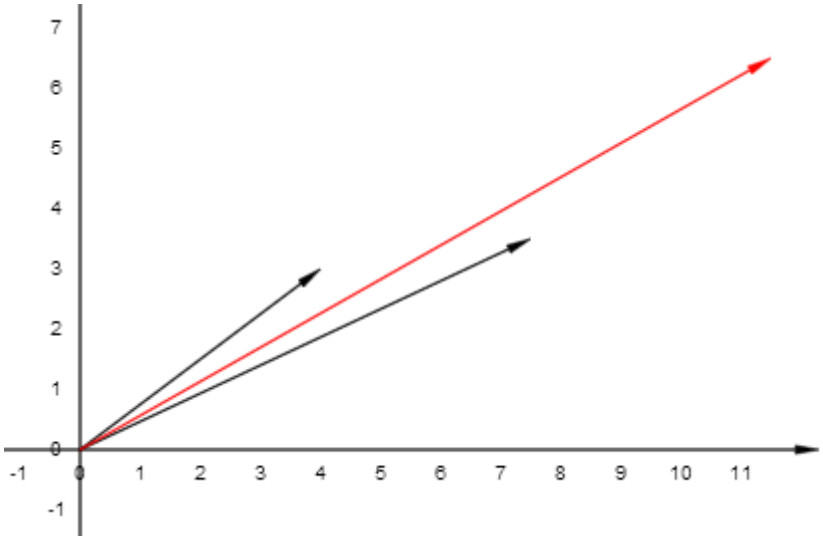
```
Vector2d.prototype.add= function(vector){
	if (vector instanceof Vector2d) {
		this.x += vector.x;
		this.y += vector.y;
	}
};
```

(Dodatek 7 Listing55)

```
Vector2d.prototype.addN = function(vector) {
	if (vector instanceof Vector2d) {
		var xx = vector.x + this.x;
		var yy = vector.y + this.y;
		return (new Vector2d(xx, yy));
	}
	return null;
};
```

Dodajemy wektor **V** = [7.5, 3.5] do wektora **V** = [4,3]. W wyniku otrzymujemy wektor **V** = [11.5, 6.5]
Gdybyśmy drugi wektor przesunęli tak, żeby jego punktem zaczepienia był koniec wektora pierwszego, to końce wektorów przesuniętego i wynikowego zetknęły by się ze sobą.

Graficznie wygląda to tak:



Odejmowanie wektorów

(Dodatek 7 Listing56)

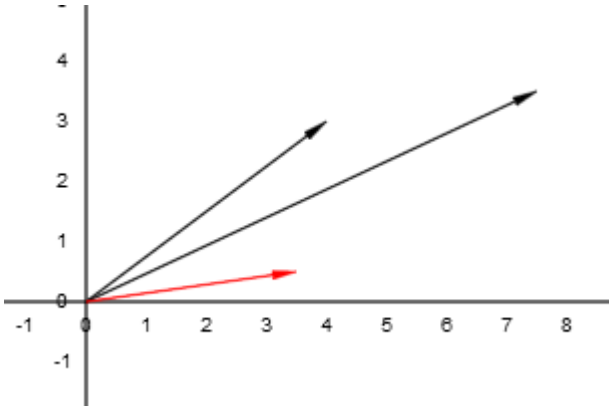
Przy odejmowaniu odejmujemy odpowiadające sobie wartości drugiego wektora od wartości pierwszego wektora:

```
Vector2d.prototype.substract = function(vector) {
	if (vector instanceof Vector2d) {
		this.x -= vector.x;
		this.y -= vector.y;
	}
};
```

(Dodatek 7 Listing57)

```
Vector2d.prototype.substractN = function(vector) {
	if (vector instanceof Vector2d) {
		var xx = this.x - vector.x;
		var yy = this.y - vector.y;
		return (new Vector2d(xx, yy));
	}
	return null;
};
```

Odejmujemy wektor **V** = [4,3] od wektora **V** = [7.5, 3.5] . W wyniku otrzymujemy wektor **V** = [3.5, 0.5]
Gdybyśmy wektor odejmowany przesunęli tak, żeby jego wierzchołek zetknął się z wierzchołkiem drugiego wektora to punkt zaczepienia odejmowanego wektora zetknąłby się z wierzchołkiem wektora wynikowego:



Skalowanie wektora

Skalowanie oznacza pomnożenie x i y wektora przez skalar. W przypadku zmiennych biegunowych oznacza pomnożenie r przez skalar.

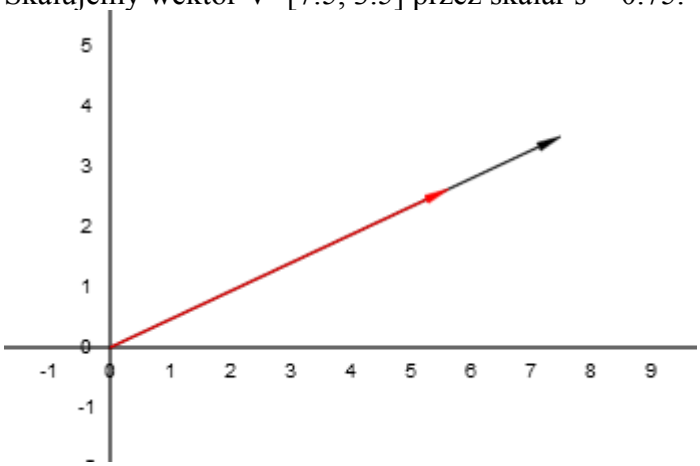
(Dodatek 7 Listing58)

```
Vector2d.prototype.scale = function(num) {
	this.x *= num;
	this.y *= num;
};
```

(Dodatek 7 Listing59)

```
Vector2d.prototype.scaleN = function(num) {
	var xx = this.x * num;
	var yy = this.y * num;
	return (new Vector2d(xx, yy));
};
```

Jeżeli skalar jest liczbą całkowitą wektor ulega przedłużeniu, jeżeli jest ułamkiem - ulega skróceniu.
Skalujemy wektor V=[7.5, 3.5] przez skalar s = 0.75. Wektor wynikowy V=[5.625, 2.625].



Normalizacja wektora

Normalizacja jest sprowadzeniem x i y do takich wartości, aby długość wektora była równa 1.

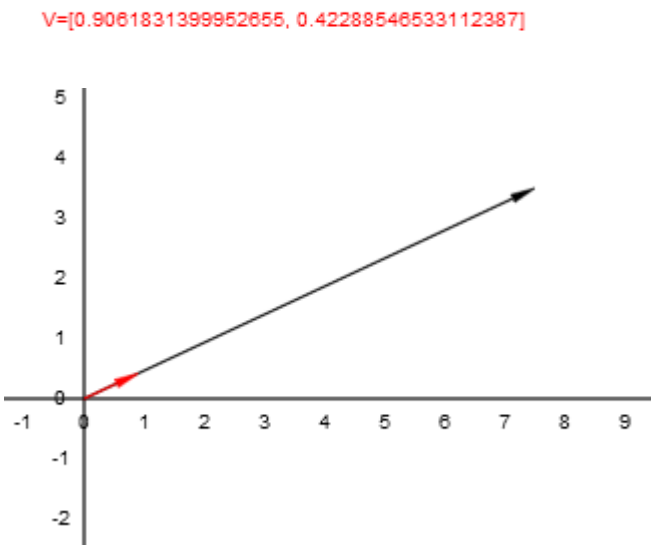
(Dodatek 7 Listing60)

```
Vector2d.prototype.normalize = function() {
	var len = this.getLength();
	this.x /= len;
	this.y /= len;
};
```

(Dodatek 7 Listing61)

```
Vector2d.prototype.normalizeN = function() {
	var len = this.getLength();
	return new Vector2d(this.x /= len, this.y /= len);
};
```

Normalizujemy wektor =[7.5, 3.5].
Jego współrzędne są teraz następujące:



Iloczyn skalarny wektorów

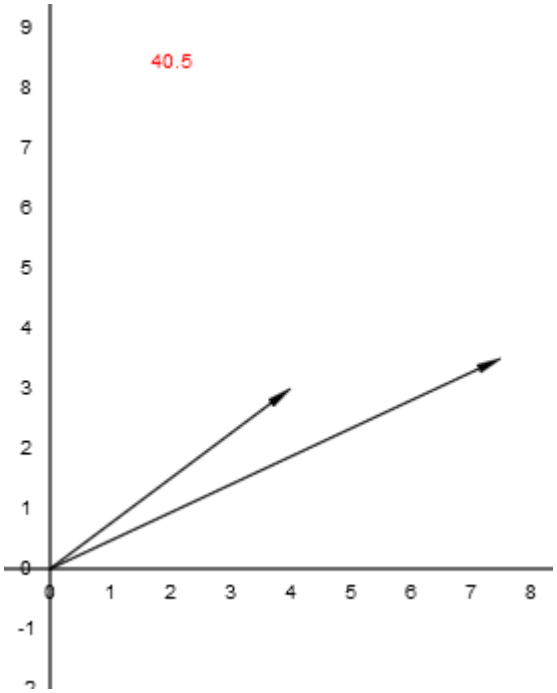
(Dodatek 7 Listing62)

Iloczyn skalarny dwóch wektorów otrzymujemy przez pomnożenie x jednego wektora przez x drugiego wektora oraz pomnożenie y pierwszego wektora przez y drugiego wektora i zwrócenie sumy tych dwóch wartości. Zwrócona liczba jest wartością skalarną.

```
Vector2d.prototype.dot = function(vector) {
	var x1 = 0;
	var x2 = 0;
	if (vector instanceof Vector2d) {
		x1 = this.x * vector.x;
		x2 = this.y * vector.y;
	}
	return (x1 + x2);
};
```

Jeżeli zwrócona wartość = 0; to wektory są do siebie prostopadłe. Jeżeli nie jest zerem, to jeśli liczba jest ujemna - kąt między wektorami jest większy niż 90°. Jeśli liczba jest dodatnia - kąt między wektorami jest mniejszy od 90°.

Obliczamy iloczyn skalarny dwóch wektorów: V_1=[7.5, 3.5] i V_2=[4,3] Iloczyn skalarny wynosi: 40.5. Kąt między wektorami jest mniejszy od 90°.



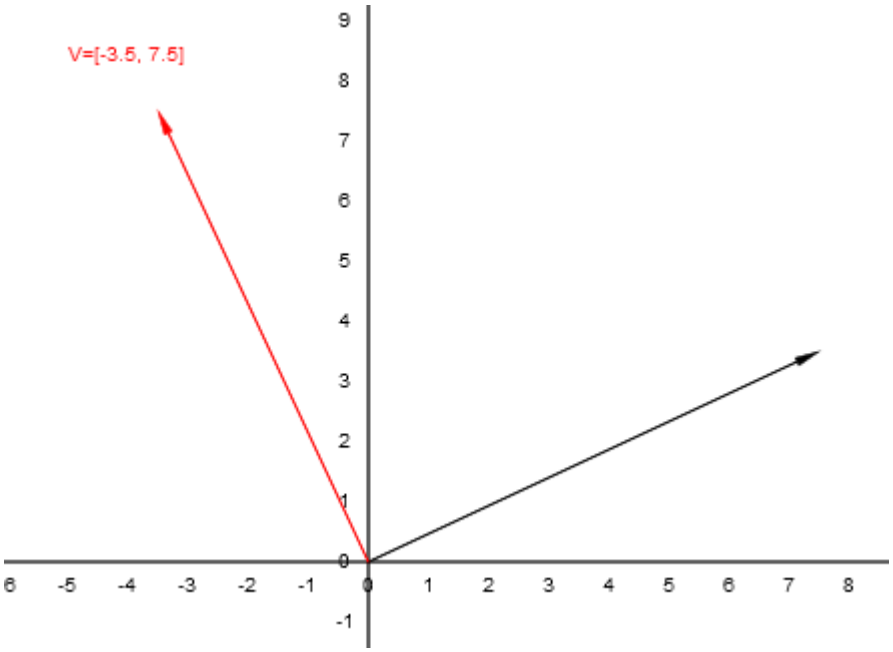
Normalna wektora

Normalna wektora to wektor prostopadły do danego wektora. Normalną wektora obliczamy przy użyciu poniższej metody:

(Dodatek 7 Listing63)

```
Vector2d.prototype.normal = function(direction) {
	var x1 = 0;
	var y1 = 0;
	switch (direction) {
	case "left":
		x1 = -this.y;
		y1 = this.x;
		break;
	case "right":
		x1 = this.y;
		y1 = -this.x;
		break;
	}
	return (new Vector2d(x1, y1));
};
```

Szukamy normalnej do wektora V=[7.5,3.5]; Normalna to: V=[-3.5, 7.5].



Jeśli chcemy sprawdzić czy podany wektor jest normalną wektora to używamy metody:

(Dodatek 7 Listing64)

```
Vector2d.prototype.isNormalTo = function(vector) {
	if (vector instanceof Vector2d) {
		return (this.dot(vect) == 0);
	}
	return false;
};
```

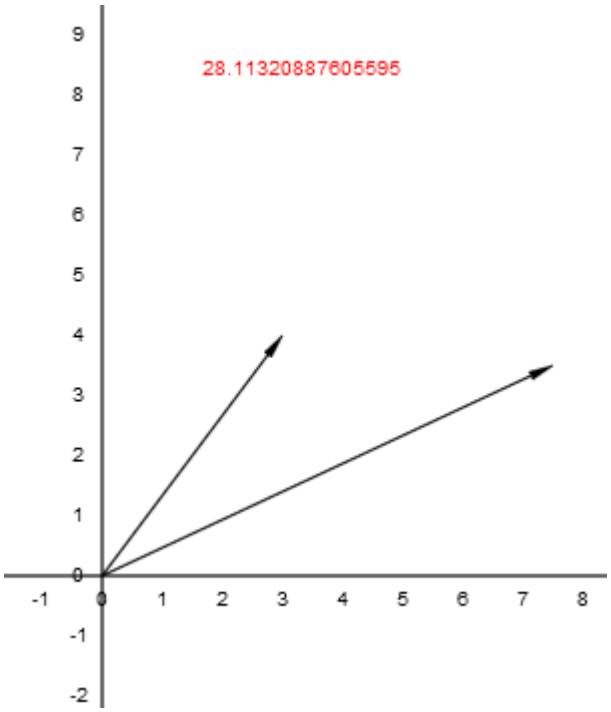
Kąt między wektorami

Jeśli badany wektor nie jest normalną naszego wektora, to kąt między wektorami możemy obliczyć używając metody:

(Dodatek 7 Listing65)

```
Vector2d.prototype.angleBetween = function(vector) {
	if (vector instanceof Vector2d) {
		var d = this.dot(vector);
		var c = d / (this.length * vector.length);
		return acosDeg(c);
	}
};
```

Badamy kąt pomiędzy vektorem V=[7.5, 3.5], a V=[3,4].. Kąt wynosi:



Iloczyn wektorowy

```
Vector2d.prototype.cross = function(vect) {
	if (vect instanceof Vector2d) {
		var z = this.x * vect.y - this.y * vect.x;
		return (new Vector2d(0, z));
	}
	return null;
};
```

O jego sensie geometrycznym powiemy przy omawianiu wyznacznika macierzy w dodatku poświęconym macierzom.

Funkcje rysujące obiekty Vector2d i Polar

```
function drawVector(vector, axes, fillStyle) {
	ctx.save();
	ctx.beginPath();
	ctx.fillStyle = fillStyle;
	var radius = 0;
	var angle = 0;
	if (vector instanceof Vector2d) {
		radius = Math.sqrt(Math.pow(vector.y, 2) + Math.pow(vector.x, 2));
		angle = atan2Deg(vector.y, vector.x);
	}
	if (axes == "complex" || axes == "cartesian") {
		drawArrow(x0, y0, radius * 30, lw, angle, bw, bh, cl, fillStyle);

	}
	if (axes == "js") {
		drawArrow(15, 15, radius * 30, lw, -angle, bw, bh, cl, fillStyle);

	}
```

```
        if (axes == "norm") {
                drawArrow(0, 0, radius, lw, angle, bw, bh, cl, fillStyle);
        }
        ctx.restore();

};
function drawPolar(polar, axes, fillStyle) {
        ctx.save();
        ctx.beginPath();
        ctx.fillStyle = fillStyle;
        if (polar instanceof Polar) {

                if (axes == "complex" || axes == "cartesian") {
                        drawArrow(x0, y0, polar.radius * 30, lw, polar.angle, bw, bh,
cl,
                                        fillStyle);

                }
                if (axes == "js") {
                        drawArrow(15, 15, polar.radius * 30, lw, -polar.angle, bw, bh,
cl,
                                        fillStyle);

                }
                if (axes = "norm") {
                        drawArrow(0, 0, polar.radius, lw, -polar.angle, bw, bh, cl,
                                        fillStyle);
                }
        }
        ctx.restore();

};
```

Wektory 3d

Do osi X i Y możemy dodać oś Z i utworzyć system trójwymiarowy. Do szerokości i wysokości dołączamy głębokość. Wektor ma 3 współrzędne x, y, z i w tekście jest zapisywany jako:

V = [7, 3, 4]

Graficzną reprezentacją wektora jest strzałka zaczepiona jednym końcem na przecięciu osi współrzędnych (0,0,0) i skierowana do punktu, którego współrzędne podaje wektor.

Klasa opisująca wektor trójwymiarowy:

```
var Vector3d = function(x, y, z) {
        this.x = x;
        this.y = y;
        this.z = z;
};
```

Do wyświetlenia wektora użyjemy metody:

```
Vector3d.prototype.toString = function() {
        return "V=[" + this.x + ", " + this.y + ", " + this.z + "]";
```

```
};
```

Do porównania wektorów użyjemy metody:

```
Vector3d.prototype.equals = function(vect) {
	if (vect instanceof Vector3d) {
		return this.x == vect.x && this.y == vect.y && this.z == vect.z;
	}
	return false;
};
```

Długość wektora

```
Vector3d.prototype.getLength = function() {
	var xx = Math.pow(this.x, 2);
	var yy = Math.pow(this.y, 2);
	var zz = Math.pow(this.z, 2);
	return Math.sqrt(xx + yy + zz);

};

Vector2d.prototype.setLength = function(length) {
	var r = this.getLength();
	if (r) {
		this.scale(length / r);
	} else {
		this.x = length;
	}

};
```

Dodawanie wektorów

```
Vector3d.prototype.add = function(vect) {
	if (vect instanceof Vector3d) {
		this.x += vect.x;
		this.y += vect.y;
		this.z += vect.z;
	}

};

Vector3d.prototype.addN = function(vect) {
	if (vect instanceof Vector3d) {
	}
	var xx = vect.x + this.x;
	var yy = vect.y + this.y;
	var zz = vect.z + this.z;
	return (new Vector3d(xx, yy, zz));

};
```

Odejmowanie wektorów

```
Vector3d.prototype.substract = function(vect) {
	if (vect instanceof Vector3d) {
```

```
            this.x -= vect.x;
            this.y -= vect.y;
            this.z -= vect.z;
        }

};

Vector3d.prototype.substractN = function(vect) {
        if (vect instanceof "Vector3d") {
            var xx = vect.x - this.x;
            var yy = vect.y - this.y;
            var zz = vect.z - this.z;
            return new Vector3d(xx, yy, zz);
        }
        return null;

};
```

Skalowanie wektora

```
Vector3d.prototype.scale = function(num) {
        this.x *= num;
        this.y *= num;
        this.z *= num;

};

Vector3d.prototype.scaleN = function(num) {
        var xx = this.x * num;
        var yy = this.y * num;
        var zz = this.z * num;
        return (new Vector3D(xx, yy, zz));

};
```

Normalizacja wektora

```
Vector3d.prototype.normalize = function() {
        var len = this.length;
        return (new Vector3d(this.x / len, this.y / len, this.z / len));

};
```

Iloczyn skalarny wektorów

```
Vector3d.prototype.dot = function(vect) {
        if (vect instanceof Vector3d) {
            var x1 = this.x * vect.x;
            var x2 = this.y * vect.y;
            var x3 = this.z * vect.z;
            return (x1 + x2 + x3);
        }

};
```

Kąt między wektorami

```
Vector3d.prototype.angleBetween = function(vect) {
	if (vect instanceof Vector3d) {
		var d = this.dot(vect);
		var c = d / (this.getLength() * vect.getLength());
		return acosDeg(c);
	}

};
```

Iloczyn wektorowy

```
Vector3d.prototype.cross = function(vect) {
	if (vect instanceof Vector3d) {
		var cx = this.y * vect.z - this.z * vect.y;
		var cy = this.z * vect.x - this.x * vect.z;
		var cz = this.x * vect.y - this.y * vect.x;
		return (new Vector3d(cx, cy, cz));
	}
	return null;
};
```

Wektor zwracany przez tę metodę zwraca wektor prostopadły (normalny) do mnożonych wektorów. Jeżeli w powyższej metodzie zamienilibyśmy wektory miejscami to wektor wynikowy byłby inny. Kolejność wykonywania operacji jest tutaj ważna.

Normalna powierzchni

Dwa dowolne wektory trójwymiarowe wyznaczają powierzchnię. Normalna powierzchni to wektor prostopadły do powierzchni i mający długość 1:

```
Vector3d.prototype.surface = function(vect) {
	if (vect instanceof Vector3d) {
		var temp1 = this.cross(vect);
		var temp2 = temp1.normalize();
		return temp2;
	}
	return null;
};
```

Jest używana przy np. wyznaczaniu ścieżki ruchu po kolizji dwóch obiektów.

Dodatek 4. Macierze

Macierz to jedno lub dwuwymiarowa tablica liczb. W JavaScript może być pokazana jako tablica lub tablica dwuwymiarowa (tablica tablic).
W rozwiązywaniu równań i w geometrii najczęściej używa się macierzy kwadratowych, w których liczba rzędów i kolumn jest taka sama. Jeżeli chcemy pracować z przekształceniami musimy, ze względów matematycznych, używać macierzy o
rząd wielkości większej niż liczba wymiarów. Ponieważ pracujemy w dwóch wymiarach musimy zastosować macierze 3x3.

Ze względów efektywnościowych mądrzej i lepiej było by używać odpowiednio długiej tablicy 1-wymiarowej. Tutaj stosuję jednak klasyczną tablicę tablic. Jeśli chcesz drogi Czytelniku możesz to zmienić i stworzyć macierz opartą o 1-wymiarową tablicę, co znacznie przyśpieszy Twoje obliczenia.

Macierze obsługuje klasa Matrix:

```
var Matrix = function() {
	switch (arguments.length) {
	case 0:
		this.m = 3;
		this.n = 3;
		this.array = new Array(this.m);
		for ( var i = 0; i < this.m; i++) {
			this.array[i] = new Array(this.n);
		}
		this.setToValue(0);
		break;
	case 1:
		if (arguments[0] instanceof Array) {
			this.array = arguments[0];
			this.m = this.array.length;
			this.n = this.array[0].length;
		} else if (arguments[0] instanceof Matrix) {
			this.array = arguments[0].array;
			this.m = arguments[0].m;
			this.n = arguments[0].n;
		}

		break;
	case 2:
		if (arguments[0] instanceof Array) {
			this.array = oneToTwo(arguments[0], arguments[1]);
			this.m=this.array.length;
			this.n=this.array[0].length;
		}
		if (typeof arguments[0] == "number") {
			this.m = arguments[0];
			this.n = arguments[1];
			this.array = new Array(this.m);
			for ( var i = 0; i < this.m; i++) {
				this.array[i] = new Array(this.n);
			}
			this.setToValue(0);
		}
		break;
	}

};
```

Tworzenie macierzy przzu użyciu różnych konstruktorów opisano w: (Dodatek 7 Listing 66, 67, 68, 69, 70)
A oto macierz kwadratowa 3x3, która jednocześnie jest macierzą zerową ponieważ jej wszyskie elementy są równe 0:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Do wyzerowania macierzy użyjemy metody:

```
Matrix.prototype.setToValue = function(s) {
	for ( var i = 0; i < this.m; i++) {
		for ( var j = 0; j < this.n; j++) {
			this.array[i][j] = s;
		}
	}
};
```

Macierz, której wszystkie elementy leżące na tzw. głównej przekątnej są równe 1, a pozostałe są równe 0, nazywana jest macierzą jednostkową:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Macierz przestawiamy na jednostkową używając metody:

```
Matrix.prototype.setToIdentity = function() {
	if (this.isSquared()) {
		for ( var i = 0; i < this.m; i++) {
			for ( var j = 0; j < this.n; j++) {
				this.array[i][j] = (i == j ? 1.0 : 0.0);
			}
		}
	}
};
```

Przykład użycia metody znajduje się w (Dodatek 7 Listing 71)

Macierz 1x3, która posiada tylko jeden wiersz jest nazywana macierzą wierszową albo wektorem wierszowym:

$$\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$$

Macierz 3x1, która ma tylko jedną kolumnę nazywana jest macierzą kolumnową albo wektorem kolumnowym:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Macierzy o postaci:

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix}$$

odpowiada tablica `array[3][3]`.

Indeksy przy a_{00} oznaczają kolejno nr rzędu i nr kolumny macierzy w której znajduje się liczba. Element a_{00} w tablicy byłby oznaczony `array[0][0]`.

Równość macierzy

Macierze są równe wtedy i tylko wtedy gdy mają:

- takie same wymiary
- każdy element pierwszej macierzy i odpowiadający mu element drugiej macierzy są równe

Równość macierzy sprawdzamy przy użyciu metody:

```
Matrix.prototype.equals = function(matrix) {
	if(!(matrix instanceof Matrix)){
		return false;
	}
	if ((this.m != matrix.m) || (this.n != matrix.n)) {
		return false;
	}

	for ( var i = 0; i < this.m; i++) {
		for ( var j = 0; j < this.n; j++) {
			if (this.array[i][j] != matrix.array[i][j]) {
				return false;
			}
		}
	}
	return true;
};
```

Przykład sprawdzania równości macierzy znajduje się w (Dodatek 7 Listing 72)

Dodawanie macierzy

Aby dodać dwie macierze dodajemy do siebie ich odpowiadające sobie elementy i tworzymy trzecią macierz. Obie macierze muszą mieć ten sam rząd.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} + \begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 10 & 10 & 10 \\ 10 & 10 & 10 \\ 10 & 10 & 10 \end{bmatrix}$$

Macierze dodajemy przy użyciu metod:

```
Matrix.prototype.add = function(matrix) {
```

```
        this.checkSize(matrix);
        for ( var i = 0; i < this.m; i++) {
                for ( var j = 0; j < this.n; j++) {
                        this.array[i][j] += matrix.array[i][j];
                }
        }
};

Matrix.prototype.add2 = function(matrix) {
        this.checkSize(matrix);
        var temp = new Matrix(this.m, this.n);
        var temp1 = temp.getArray();
        for ( var i = 0; i < this.m; i++) {
                for ( var j = 0; j < this.n; j++) {
                        temp1[i][j] = this.array[i][j] + matrix.array[i][j];
                }
        }
        return temp;
};
```

Przykłady dodawania macierzy ze zwracaniem i bez zwracania są przedstawione w (Dodatek 7 Listing 73, 74)

Odejmowanie macierzy

Aby odjąć dwie macierze odejmujemy od siebie odpowiadające sobie elementy i tworzmy trzecią macierz. Obie macierze muszą mieć ten sam rząd:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} - \begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix} = \begin{bmatrix} -8 & -6 & -4 \\ -2 & 0 & 2 \\ 4 & 6 & 8 \end{bmatrix}$$

```
Matrix.prototype.substract = function(matrix) {
        checkSize(matrix);
        for ( var i = 0; i < this.m; i++) {
                for ( var j = 0; j < this.n; j++) {
                        this.array[i][j] = this.array[i][j] - matrix.array[i][j];
                }
        }
};

Matrix.prototype.substract2 = function(matrix) {
        checkSize(matrix);
        var temp = new Matrix(this.m, this.n);
        var temp1 = temp.getArray();
        for ( var i = 0; i < this.m; i++) {
                for ( var j = 0; j < this.n; j++) {
                        temp1[i][j] = this.array[i][j] - matrix.array[i][j];
                }
        }
        return temp;
};
```

Przykłady odejmowania ze zwracaniem i bez przedstawiono w (Dodatek 7 Listing 75, 76).

Mnożenie skalarne

Aby pomnożyć macierz przez skalar mnożymy każdy element macierzy przez skalar:

$$3 * \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} = \begin{bmatrix} 3 & 6 & 9 \\ 12 & 15 & 18 \\ 21 & 24 & 27 \end{bmatrix}$$

```
MMatrix.prototype.multiplys = function(s) {
	for ( var i = 0; i < this.m; i++) {
		for ( var j = 0; j < this.n; j++) {
			this.array[i][j] = s * this.array[i][j];
		}
	}
};

Matrix.prototype.multiplys2 = function(s) {
	var temp = new Matrix(this.m, this.n);
	var temp1 = temp.getArray();
	for ( var i = 0; i < this.m; i++) {
		for ( var j = 0; j < this.n; j++) {
			temp1[i][j] = s * this.array[i][j];
		}
	}
	return temp;
};
```

Przykłady mnożenia skalarnego ze zwracaniem i bez zwracania znajdują się w plikach (Dodatek 7 Listing 77, 78)

Mnożenie macierzy

Macierz możemy pomnożyć przez inną macierz, w tym macierz kolumnową albo wierszową. Jeżeli mnożymy macierz A przez macierz B to liczba kolumn A musi być równa liczbie wierszy B. Macierz wynikowa będzie miała tyle wierszy ile ma macierz A i tyle kolumn ile ich miała macierz B. Mnożenie macierzy na ogół nie jest przemienne. Mnożąc A x B otrzymamy inny wynik niż mnożąc B x A. Mnożenie w jednej kolejności może być wykonalne, a w drugą stronę nie. Jeśli jest wykonalne zmianie może ulec wymiar wyniku. Mnożenie macierzy jest przemienne jedynie dla niektórych macierzy kwadratowych. Pomnóżmy przez siebie dwie macierze:

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} \text{x} \begin{bmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \\ b_{20} & b_{21} & b_{22} \end{bmatrix} =$$

$$\begin{bmatrix} a_{00} * b_{00} + a_{01} * b_{10} + a_{02} * b_{20} & a_{00} * b_{01} + a_{01} * b_{11} + a_{02} * b_{21} & a_{00} * b_{02} + a_{01} * b_{12} + a_{02} * b_{22} \\ a_{10} * b_{00} + a_{11} * b_{10} + a_{12} * b_{20} & a_{10} * b_{01} + a_{11} * b_{11} + a_{12} * b_{21} & a_{10} * b_{02} + a_{11} * b_{12} + a_{12} * b_{22} \\ a_{20} * b_{00} + a_{12} * b_{10} + a_{22} * b_{20} & a_{20} * b_{01} + a_{21} * b_{11} + a_{22} * b_{21} & a_{20} * b_{02} + a_{21} * b_{12} + a_{22} * b_{22} \end{bmatrix}$$

Jeżeli przypomnimy sobie mnożenie skalarne wektorów i potraktujemy pierwszą tablicę jako trzy wektory wierszowe, a drugą tablicę jako trzy wektory kolumnowe, to:

- pierwszy wiersz macierzy wynikowej to trzy iloczyny skalarne pierwszego wektora a przez kolejne wektory b
- drugi wiersz macierzy wynikowej to trzy iloczyny skalarne drugiego wektora a przez kolejne wektory b
- trzeci wiersz macierzy wynikowej to trzy iloczyny skalarne trzeciego wektora a przez kolejne wektory b

Zauważmy, że mnożymy A x B, ale zaczynamy od prawej strony, czyli od B.
Jeżeli mnożone macierze mają różne wielkości musimy najpierw sprawdzić dopuszczalność mnożenia:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \text{ x } \begin{bmatrix} 7 \\ 8 \\ 9 \end{bmatrix} =$$

2 x 3 3 x 1

Ponieważ liczba kolumn w pierwszej macierzy = 3 i liczba wierszy w drugiej macierzy = 3 mnożenie jest wykonalne. Wynik będzie postaci 2 x 1.

$$= \begin{bmatrix} 7+16+27 \\ 28+40+54 \end{bmatrix} = \begin{bmatrix} 50 \\ 122 \end{bmatrix}$$

Dokonajmy odwrotnego mnożenia:

$$\begin{bmatrix} 7 \\ 8 \\ 9 \end{bmatrix} \text{x} \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} =$$

3 x 1 2 x 3

Ponieważ liczba kolumn w pierwszej macierzy = 1, a liczba wierszy w drugiej macierzy = 2 mnożenia nie można wykonać.

Pomnóżmy:

$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \text{ x } \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix} =$$

1 x 3 3 x 1 Mnożenie jest wykonalne. Wynik jest postaci 1 x 1, czyli jest to skalar.

$$= \begin{bmatrix} 4+10+18 \end{bmatrix} = \begin{bmatrix} 32 \end{bmatrix} = 32.$$

Pomnóżmy:

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \text{ x } \begin{bmatrix} 4 & 5 & 6 \end{bmatrix} =$$

3 x 1 1 x 3 Mnożenie jest wykonalne. Wynik jest postaci 3 x 3.

$$= \begin{bmatrix} 4 & 8 & 12 \\ 5 & 10 & 15 \\ 6 & 12 & 18 \end{bmatrix}$$

```
Matrix.prototype.multiply = function(matrix) {
	if (matrix.m != this.n) {
		throw new Error("Nieprawidłowe wymiary macierzy.");
	}
	var temp1 = new Array(this.m);
	for ( var p = 0; p < this.m; p++) {
		temp1[p] = new Array(matrix.n);
	}
	var bc = new Array(this.n);
	for ( var j = 0; j < matrix.n; j++) {
		for ( var k = 0; k < this.n; k++) {
			bc[k] = matrix.array[k][j];
		}
		for ( var i = 0; i < this.m; i++) {
			var ai = this.array[i];
			var s = 0.0;
			for ( var r = 0; r < this.n; r++) {
				s += ai[r] * bc[r];
			}
			temp1[i][j] = s;
		}
	}
	this.array = temp1;
	this.n = matrix.n;
};

Matrix.prototype.multiply2 = function(matrix) {
	if (matrix.m != this.n) {
		throw new Error("Nieprawidłowe wymiary macierzy");
	}
	var temp = new Matrix(this.m, matrix.n);
	var temp1 = new Array(this.m);
	for ( var p = 0; p < this.m; p++) {
		temp1[p] = new Array(matrix.n);
	}
	var bc = new Array(this.n);
	for ( var j = 0; j < matrix.n; j++) {
		for ( var k = 0; k < this.n; k++) {
			bc[k] = matrix.array[k][j];
		}
		for ( var i = 0; i < this.m; i++) {
			var ai =this.array[i];
			var s = 0.0;
			for ( var r = 0; r <this.n; r++) {
				s += ai[r] * bc[r];
			}
```

```
                    temp1[i][j] = s;
                }
        }
        temp.array=temp1;
        return temp;

};
```

Przykłady mnożenia macierzy są w (Dodatek 7 Listing 79, 80)

Jeśli mnożymy macierz A przez macierz jednostkową I, to jest to równoznaczne z mnożeniem przez 1 w zwykłej algebrze, czyli macierz A nie ulega zmianie. Mnożenie takie jest przemienne, tzn. może być wykonywane w dowolnej kolejności:

AI = IA = A

Wyznacznik macierzy

Oznaczany jest jako detA albo |A|.

Wyznacznik macierzy 1x1 jest obliczany następująco:

$\det[a] = a$

Wyznacznik macierzy 2x2 jest obliczany następująco:

$\det\begin{bmatrix} a & b \\ c & d \end{bmatrix} = ad - bc$

Wyznacznik macierzy 3x3 może być wyznaczany wieloma metodami, z których najprostszą jest metoda Sarrusa:

$\det\begin{bmatrix} a & b & e \\ c & d & f \\ g & h & i \end{bmatrix} = (adi + bfg + ech) - (edg + afh + bci)$

Np. $\det\begin{bmatrix} 1 & 2 & 3 \\ -2 & 0 & 1 \\ 5 & 1 & 3 \end{bmatrix} = (0 + 10 - 6) - (0 + 1 - 12) = 4 - (-11) = 4 + 11 = 15;$

Jeśli będziemy mieli do czynienia z macierzą transformacji:

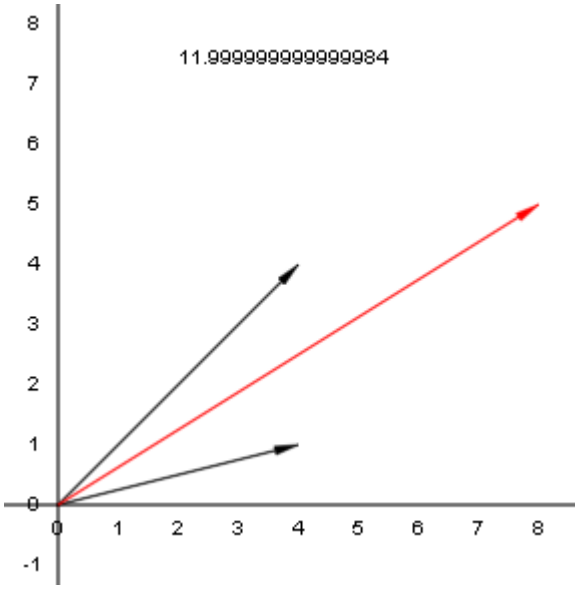
$\det\begin{bmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & 1 \end{bmatrix} = (ad1 + 000 + 0c0) - (0d0 + a00h + bc1) = ad - bc$

a w innym przypadku:

$$\det\begin{bmatrix} a & b & e \\ c & d & f \\ 0 & 0 & 1 \end{bmatrix} = (ad1 + bf0 + ec0) - (ed0 + af0 + bc1) = ad1 - bc1 = ad - bc$$

Geometryczna interpretacja wyznacznika

(Dodatek 7 Listing 83)



Załóżmy, że mamy dwa wektory V_1=[4,4] i V_2= [4,1]. Jeżeli narysujemy je na płaszczyźnie i narysujemy wektor będący sumą tych wektorów to otrzymamy równoległobok, którego dłuższą przekątną jest wektor oznaczony na czerwono będący suma tych dwóch wektorów. Pole równoległoboku to:

$S = a*b*\sin\alpha$

gdzie:
S - pole równoległoboku
a - jest długością pierwszego wektora (I bok równoległoboku),
b - jest długością drugiego wektora (II bok równoległoboku),
α jest kątem między wektorami

W naszym przykładzie S = 12;

$$\det\begin{bmatrix} 4 & 4 & 0 \\ 4 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = 4*1 - 4*4 = -12,\ |\det| = 12$$

Jak widzimy wyznacznik macierzy jest równy polu równoległoboku rozpiętego na płaszczyźnie na tych wektorach.

(Dodatek 7 Listing 85)

Jeżeli obliczymy wektor 2d, który jest iloczynem wektorowym obu podanych wektorów to jego długość będzie równa 12.
Jeżeli obliczymy wektor 3d, który jest iloczynem wektorowym obu podanych wektorów z częścią z = 0, to długość tego wektora będzie wynosiła 12.

Jeżeli macierz będzie miała trzy 'pełne' wektory to wyznacznik macierzy 3 x 3 będzie objętością równoległościanu rozpiętego w przestrzeni na tych 3 wektorach.

Jeżeli weźmiemy trzy wektory V_1=[a,b], V_2=[c,d], V_3 =[g,h] i narysujemy je na płaszczyźnie to pole S trójkąta wyznaczonego przez końce tych wektorów wynosi:

$$S = \frac{1}{2}\det\begin{bmatrix} a & b & 1 \\ c & d & 1 \\ g & h & 1 \end{bmatrix} = (ad1 + b1g + 1ch) - (1dg + a1h + bc1) =$$

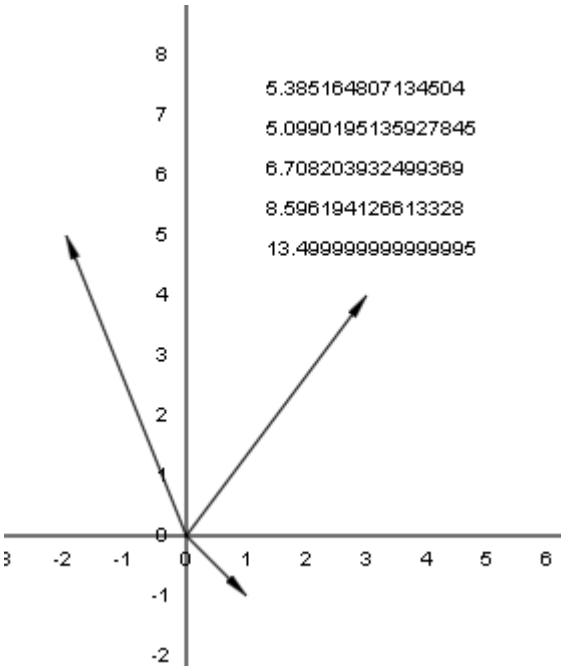
$$= (ad + bg + ch) - (dg + ah + bc)$$

Jeżeli nasz trójkąt to: V_1=[1,-1], V_2=[3,4], V_3=[-2,5]

$$\det\begin{bmatrix} 1 & -1 & 1 \\ 3 & 4 & 1 \\ -2 & 5 & 1 \end{bmatrix} = (4 + 2 + 15)-(-8+5-3) = 21 -(-6)=21+6=27;$$

pole $S=\frac{1}{2}*27 = 13.5$;

(Dodatek 7 Listing 84)



Z geometrii wynika, że boki trójkąta wynoszą:

x = Math.sqrt($(a-c)^2+(b-d)^2$) = Math.sqrt($(1-3)^2+(-1-4)^2$) = Math.sqrt(4+25) = 5.385,

y = Math.sqrt($(c-g)^2+(d-h)^2$) = Math.sqrt($(3+2)^2+(4-5)^2$) = Math.sqrt(25+1)=5.099,
z = Math.sqrt($(g-a)^2+(h-b)^2$) = Math.sqrt($(-2-1)^2+(5+1)^2$)= Math.sqrt(9+36)=6.708,

Połowa obwodu p = (x + y + z)/2 = 5.385+5.099+6.708 = 17.192/2 = 8.596

Pole S trójkąta z wzoru Herona

$$S=\sqrt{p(p-x)(p-y)(p-z)} = \sqrt{8.596(8.596-5.385)(8.596-5.099)(8.596-6.708)} =$$
$$= \sqrt{8.596*3.211*3.497*1.888} = \sqrt{182.236} = 13.499.$$

Możemy to też obliczyć używając iloczynu skalarnego:
Obliczamy iloczyny skalarne V_1xV_2, V_2xV_3, V_3xV_1. Obliczamy połowy długości tych wektorów. Dodajemy długości dwóch pierwszych i odejmujemy trzecią. Wynik jest dokładnie 13.5 (Dodatek 7 Listing 71).

Do obliczania wyznacznika możemy użyć metody:

```
Matrix.prototype.detSarrus = function() {
	var a = this.array[0][0];
	var b = this.array[0][1];
	var e = this.array[0][2];
	var c = this.array[1][0];
	var d = this.array[1][1];
	var f = this.array[1][2];
	var g = this.array[2][0];
	var h = this.array[2][1];
	var i = this.array[2][2];
	return ((a*d*i + b*f*g + e*c*h) - (e*d*g + a*f*h + b*c*i));
};

var detSarrus = function(matrix) {
	var sarrus = 0.0;
	switch (matrix.m) {
	case 1:
		sarrus = matrix.array[0][0];
		break;
	case 2:
		sarrus = matrix.array[0][0] * matrix.array[1][1] - matrix.array[0][1]
				* matrix.array[1][0];
		break;
	case 3:
		var a = matrix.array[0][0];
		var b = matrix.array[0][1];
		var e = matrix.array[0][2];
		var c = matrix.array[1][0];
		var d = matrix.array[1][1];
		var f = matrix.array[1][2];
		var g = matrix.array[2][0];
		var h = matrix.array[2][1];
		var i = matrix.array[2][2];
		sarrus = ((a * d * i + b * f * g + e * c * h) - (e * d * g + a * f *
h + b
				* c * i));
		break;

	}
```

```
        return sarrus;

};
```

Właściwości wyznacznika

Właściwości wyznacznika macierzy kwadratowej:

1. Jeżeli kolumna macierzy lub wiersz macierzy zawierają same 0, to wyznacznik macierzy wynosi 0
2. Jeżeli zamienimy miejscami dwa wiersze (albo dwie kolumny) to wyznacznik zmieni swój znak (+ na - albo - na +)
3. Jeżeli dwie kolumny (albo dwa wiersze) macierzy są jednakowe to wyznacznik tej macierzy wynosi 0
4. Jeżeli wszystkie elementy pewnego wiersza (albo kolumny) zawierają wspólny czynnik, to czynnik ten można wyłączyć przed wyznacznik
5. Jeżeli każda z komórek macierzy zawiera wspólny czynnik to czynnik ten po podniesieniu do potegi równej ilości rzędów (kolumn) można wyłączyć przed wyznacznik
6. Jeżeli do elementów dowolnej kolumny (albo wiersza) dodamy odpowiadające im elementy innej kolumny (wiersza) pomnożone przez dowolną liczbę to wartość wyznacznika pozostanie nie zmieniona.
7. Wyznaczniki macierzy i jej macierzy transponowanej są jednakowe
8. Jeżeli obie macierze są tego samego stopnia to wyznacznik iloczynu macierzy równa się iloczynowi wyznaczników: det(AB) = det(A)*det(B)

Ad. 4

$$\det\begin{bmatrix} 1 & 2 & 3 \\ 4 & 4 & 5 \\ 6 & 8 & 7 \end{bmatrix} = 2\det\begin{bmatrix} 1 & 1 & 3 \\ 4 & 2 & 5 \\ 6 & 4 & 7 \end{bmatrix}$$

Ad. 5

$$\det\begin{bmatrix} 2 & 4 & 6 \\ 8 & 10 & 12 \\ 14 & 16 & 18 \end{bmatrix} = 2^3\det\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Ad. 6

$$\det\begin{bmatrix} 2 & 4 & 6 \\ 8 & 10 & 12 \\ 14 & 16 & 18 \end{bmatrix} = \det\begin{bmatrix} 2 & 4+2*2 & 6 \\ 8 & 10+2*8 & 12 \\ 14 & 16+2*14 & 18 \end{bmatrix}$$

Transpozycja macierzy

Transpozycja macierzy to zamiana wierszy z kolumnami (pierwszy wiersz z pierwszą kolumną, drugi wiersz z drugą kolumną, etc). Wartości leżące na przekątnej głównej nie zmieniają się.
Mamy tablicę:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Po transpozycji otrzymujemy:

$$\begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

Jeśli transponujemy macierz wierszową:

$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

to otrzymamy macierz kolumnową:

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

i odwrotnie.

```
Matrix.prototype.transpose = function() {
	var temp1 = new Array(this.n);
	for ( var k = 0; k < this.n; k++) {
		temp1[k] = new Array(this.m);
	}
	for ( var i = 0; i < this.m; i++) {
		for ( var j = 0; j < this.n; j++) {
			temp1[j][i] = this.array[i][j];
		}
	}
	this.setArray(temp1);
};

Matrix.prototype.transpose2 = function() {
	var temp = new Matrix(this.n, this.m);
	var temp1 = temp.getArray();
	for ( var i = 0; i < this.m; i++) {
		for ( var j = 0; j < this.n; j++) {
			temp1[j][i] = this.array[i][j];
		}
	}
	return temp;
};
```

W działaniach na macierzach macierz transponowaną oznacza się jako A^T.

Przykłady transponowania macierzy zawarto w plikach (Dodatek 7 Listing 81, 82)

Jak widzimy w przypadku macierzy, które nie są kwadratowe, przy transpozycji zamianie ulegają również wymiary macierzy.

Dzielenie macierzy

Dzielenia macierzy nie definiuje się. W przypadku konieczności dzielenia zastępuje się je mnożeniem przez odwrotność. Np. dzielenie przez 2 jest równoznaczne z mnożeniem przez $\frac{1}{2}$, gdyż $2*\frac{1}{2}=1$.

$\frac{1}{2}$ może być zapisana jako 2^{-1},
czyli

$2*2^{-1}=1$;

Macierz odwrotna

W przypadku macierzy A przy dzieleniu potrzebujemy macierzy odwrotnej A^{-1}, czyli takiej, że po jej pomnożeniu przez A otrzymamy macierz jednostkową I:

$A\ A^{-1} = A^{-1}*A = I$

Macierz mająca macierz odwrotną nazywana jest macierzą odwracalną. Jeżeli wyznacznik macierzy jest równy 0, to macierz nie ma odwrotności (jest nieodwracalna) i nazywana jest macierzą osobliwą. Jeżeli macierz jest odwracalna to wyznacznik jest większy od 0 i wtedy jest nazywana macierzą nieosobliwą.

Macierz odwrotna do macierzy A:

$$A^{-1} = \frac{1}{\det A} * D^T$$

gdzie D jest transponowaną macierzą dopełnień algebraicznych odpowiadających elementów macierzy A

Podmacierz

Macierz powstała przez usunięcie jednej lub więcej kolumn (lub wierszy) z danej macierzy. Indeksy podmacierzy zaczynaja się od 1.

Macierz A= $\begin{bmatrix} 2 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$

Podmacierz B_{33} = $\begin{bmatrix} 2 & 2 \\ 4 & 5 \end{bmatrix}$

Podmacierz możemy znaleźć używając metody:

```
var submatrix = function(matrix, row, col){
      var i1 = row-1;
      var i2 = col-1;
      var ar = matrix.cloneArray();
      ar = shortenArray(ar,i1);
      for(var i =0;i< ar.length;i++){
            ar[i]=shortenArray(ar[i],i2);
      }
      return new Matrix(ar);
};
```

(Dodatek 7 Listing 87)

Minor

Wyznacznik podmacierzy danej macierzy. Minor M_{ij} to wyznacznik macierzy, z której usunięto i-ty wiersz i j-tą kolumnę. Indeksy minorów i podmacierzy zaczynają się od 1 (ze względu na wykładniki potęg).

Podmacierz B_{33} = $\begin{bmatrix} 2 & 2 \\ 4 & 5 \end{bmatrix}$

Minor M_{33} tej podmacierzy to 2*5 - 2*4 = 2

```
var minor = function(matrix, row, col) {
      var sub = submatrix(matrix, row, col);
      return detSarrus(sub);
};
```

(Dodatek 7 Listing 88)

Dopełnienie algebraiczne

Jest to iloczyn:

$(-1)^{i+j}$ i minora M_{ij}

A= $\begin{bmatrix} 2 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$

Podmacierz $A_{22} = \begin{bmatrix} 2 & 3 \\ 7 & 9 \end{bmatrix}$

Minor $M_{22} = \det A_{22} = \det \begin{bmatrix} 2 & 3 \\ 7 & 9 \end{bmatrix} = 2*9 - 3*7 = -3$

Dopełnienie algebraiczne elementu $A_{22} = (-1)^{2+2} * -3 = 1*-3 = -3$.

W macierzy dopełnień zaznaczamy:

$$\begin{bmatrix} & & \\ & -3 & \\ & & \end{bmatrix}$$

```
var algComplement = function(matrix, row, col){
      return ac = minor(matrix,row,col)* Math.pow(-1, row+col);
};
```

(Dodatek 7 Listing 89)

Jeżeli wykonamy wszystkie obliczenia uzyskamy macierz dopełnień:

$$\begin{bmatrix} -3 & 6 & -3 \\ 6 & -3 & -2 \\ -3 & 0 & 2 \end{bmatrix}$$

(Dodatek 7 Listing 90)

Transpozycja macierzy dopełnień

(Dodatek 7 Listing 91)

Otrzymujemy macierz transponowaną:

$$\begin{bmatrix} -3 & 6 & -3 \\ 6 & -3 & 0 \\ -3 & -2 & 2 \end{bmatrix}$$

Wyznacznik macierzy wyjściowej

Wyznacznik macierzy wyjściowej wynosi -3 (Dodatek 7 Listing 91)

Macierz odwrotna

Mnożymy macierz transponowaną przez skalar (odwrotność wyznacznika macierzy wyjściowej)

$$A^{-1} = \frac{1}{\det A} * D^T = -\frac{1}{3}\begin{bmatrix} -3 & 6 & -3 \\ 6 & -3 & 0 \\ -3 & -2 & 2 \end{bmatrix} = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 1 & 0 \\ 1 & \frac{2}{3} & -\frac{2}{3} \end{bmatrix}$$

(Dodatek 7 Listing 92)

Sprawdzenie

$$A = \begin{bmatrix} 2 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

$$A^{-1} = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 1 & 0 \\ 1 & \frac{2}{3} & -\frac{2}{3} \end{bmatrix}$$

$$AA^{-1} = \begin{bmatrix} 2 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}\begin{bmatrix} 1 & -2 & 1 \\ -2 & 1 & 0 \\ 1 & \frac{2}{3} & -\frac{2}{3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(Dodatek 7 Listing 93)

.

Ponieważ wynik mnożenia jest macierzą jednostkową, tzn. że macierz A^{-1} jest macierzą odwrotną do macierzy A.

Do odwrócenia macierzy możemy użyć metody:

```
Matrix.prototype.reverte = function() {
	var matrixac = new Matrix(3, 3);
	for ( var i = 0; i < 3; i++) {
		for ( var j = 0; j < 3; j++) {
			ac = algComplement(this, i + 1, j + 1);
			matrixac.setMN(i, j, ac);
		}
	}
	var matrix1 = matrixac.transpose2();
	var det = this.detSarrus();
	var matrixt = matrix1.multiplys2(1.0 / det);
	return matrixt;
};
```

Rozwiązanie prostego równania

Mamy układ 2 równań z dwiema niewiadomymi:

3x + 2y = 17

5x - 3y = 3

Możemy to równanie zapisać w postaci macierzowej (można by to przedstawić za pomocą macierzy 2x2, ale z pewnych względów, o których powiem później, przedstawiam zagadnienie w postaci macierzy 3x3):

$$\begin{bmatrix} 3 & 2 & 0 \\ 5 & -3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 17 \\ 3 \\ 1 \end{bmatrix}$$

Żeby uzyskać wynik musimy podzielić obie strony równania przez macierz stojącą po lewej stronie. Ponieważ dzielenie nie jest określone pomnożymy obie strony przez odwrotność tej macierzy:

$$\begin{bmatrix} 3 & 2 & 0 \\ 5 & -3 & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} = \frac{1}{\det} \begin{bmatrix} -3 & -2 & 0 \\ -5 & 3 & 0 \\ 0 & 0 & -19 \end{bmatrix}$$

Ponieważ det = -19 $\neq 0$ równanie ma rozwiązanie.
Po pomnożeniu mamy:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = -\frac{1}{19} \begin{bmatrix} -3 & -2 & 0 \\ -5 & 3 & 0 \\ 0 & 0 & -19 \end{bmatrix} \begin{bmatrix} 17 \\ 3 \\ 1 \end{bmatrix} = -19 \begin{bmatrix} -3*17-6 \\ -5*17+9 \\ -19 \end{bmatrix} = \begin{bmatrix} -57 \\ -76 \\ -19 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \\ 1 \end{bmatrix}$$

Nasze równanie ma rozwiązanie: x = 3, y = 4.

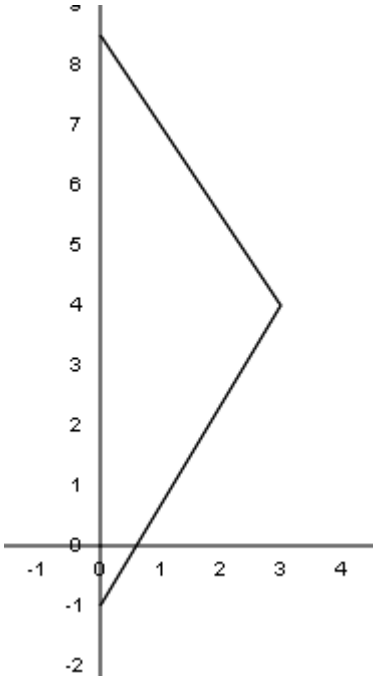
Wszystkie obliczenia w (Dodatek 7 Listing 94)

Możemy to też zapisać:

$$\begin{bmatrix} 17 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 & 2 & 0 \\ 5 & -3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 3x+2y \\ 5x-3y \\ 1 \end{bmatrix}$$

Geometrycznie rzecz biorąc punkt P(3,4) jest punktem przecięcia się obu linii:

(Dodatek 7 Listing 95)



Nasz punkt P(x, y) = P(3, 4) został transformowany na punkt P(17, 3). Znając wynik transformacji P(17, 3) możemy, jak widzimy powyżej, znaleźć punkt transformowany. Znając punkt transformowany możemy znaleźć punkt wyjściowy transformacji.

Inne metody klasy Matrix:

```
Matrix.prototype.toString = function() {
	var st = "";
	for ( var i = 0; i < this.m; i++) {
		for ( var j = 0; j < this.n; j++) {
			st = st + this.array[i][j] + "\n";
		}
	}
	return st;
};

Matrix.prototype.getArray = function() {
	return this.array;
};

Matrix.prototype.setArray = function(array) {
	if (array instanceof Array) {
		this.array = array;
		this.m = this.array.length;
		this.n = this.array[0].length;
	}
};

Matrix.prototype.cloneArray = function() {
	var ar = new Array(this.m);
	for ( var k = 0; k < this.m; k++) {
		ar[k] = new Array(this.n);
	}
	for ( var i = 0; i < this.m; i++) {
		for ( var j = 0; j < this.n; j++) {
```

```
                    ar[i][j] = this.array[i][j];
                }
        }
        return ar;
};

Matrix.prototype.getM = function() {
        return this.m;
};

Matrix.prototype.getN = function() {
        return this.n;
};

Matrix.prototype.getMN = function(m, n) {
        return this.array[m][n];
};

Matrix.prototype.setMN = function(m, n, s) {
        this.array[m][n] = s;
};

Matrix.prototype.isSquared = function() {
        if (this.m == this.n) {
                return true;
        }
        return false;
};

Matrix.prototype.checkSize = function(matrix) {
        if (matrix instanceof Matrix) {
                if (matrix.m != this.m || matrix.n != this.n) {
                        throw new Error("Nieprawidłowe wymiary.");
                }
        }

};

var shortenArray = function(array, index) {
        for ( var i = index + 1; i < array.length; i++) {
                array[i - 1] = array[i];
        }
        array.pop();
        return array;
};

var cloneArray = function(array) {
        var b = new Array();
        return b.concat(array);
};
```

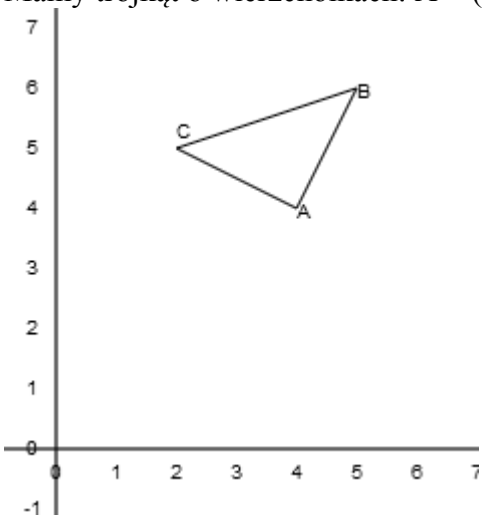
Więcej metod i informacji o zastosowaniach macierzy znajdziesz w rozdziale poświęconym transformacjom.

Dodatek 5. Przekształcenia afiniczne

Przekształcenia to po prostu sposoby, na które możemy postąpić z figurą geometryczną. W trakcie tych przekształceń figury zachowują się jak ciała sztywne. Zachowana jest równoległość linii. Przekształcenia, które zachowują również odległości i kąty nazywane są przekształceniami izometrycznymi. Przekształceniami, które możemy zastosować są:

- przesunięcie (translacja)
- skalowanie
- obrót (rotacja)
- odbicie (refleksja)
- przekrzywienie (pochylenie)

Mamy trójkąt o wierzchołkach: A = (4,4), B=(5,6) i C=(2,5):



Translacja

(Dodatek 7 Listing96)

Jest przekształceniem izometrycznym. Translacja przesuwa współrzędne punktu x o podaną wartość dx wzdłuż osi X, a współrzędne punktu y o podaną wartość dy wzdłuż osi Y.
Z punktu widzenia rachunku macierzy translację możemy wyrazić jako sumę dwóch wektorów kolumnowych z których jeden oznacza współrzędne punku, a drugi wartość przesunięcia. Wektor wyjściowy zawiera nowe wpółrzędne punktu:

$$\begin{bmatrix} 4 \\ 4 \\ 0 \end{bmatrix} + \begin{bmatrix} 3 \\ 4 \\ 0 \end{bmatrix} = \begin{bmatrix} 7 \\ 8 \\ 0 \end{bmatrix}$$

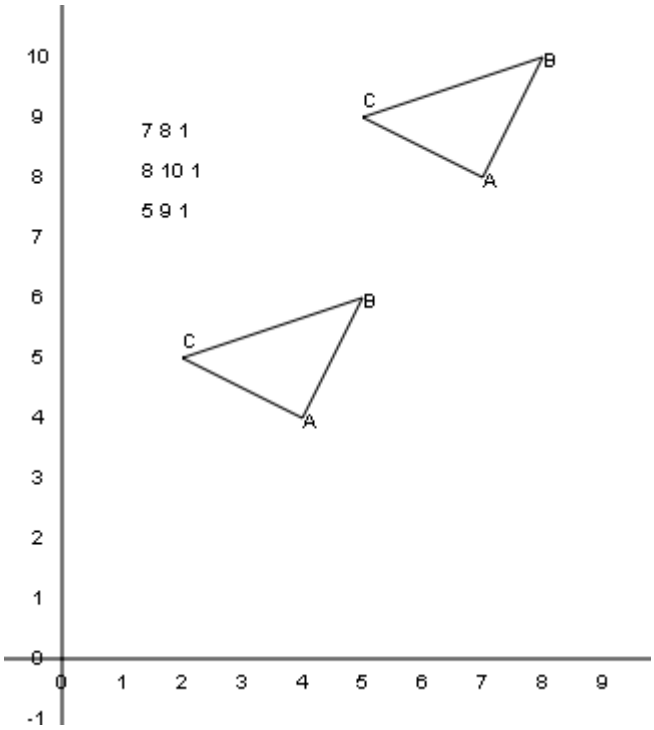
Jeżeli nasz punkt miał współrzędne (4,4) i został na osi X przesunięty o 3 i na osi Y o 4, to obecnie znajduje się w punkcie (7,8).
Ponieważ jednak wszystkie inne przekształcenia wyrażamy w postaci mnożenia przez wektor to wygodniej jest podawać translację jako mnożenie wektora przez macierz przekształcenia. Jest to macierz jednostkowa z dodanymi wartościami parametrów translacji:

$$\begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \text{x} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1*x+0*y+dx*1 \\ 0*x+1*y+dy*1 \\ 0*x+0*y+1*1 \end{bmatrix} = \begin{bmatrix} x+dx \\ y+dy \\ 1 \end{bmatrix}$$

gdzie x i y to współrzędne punktu, a dx i dy to wartości przesunięcia odpowiednio na osi X i osi Y, czyli:

$$\begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{bmatrix} \text{x} \begin{bmatrix} 4 \\ 4 \\ 1 \end{bmatrix} = \begin{bmatrix} 1*4+0*4+3*1 \\ 0*4+1*4+4*1 \\ 0*4+0*4+1*1 \end{bmatrix} = \begin{bmatrix} 4+3 \\ 4+4 \\ 1 \end{bmatrix} = \begin{bmatrix} 7 \\ 8 \\ 1 \end{bmatrix}$$

Jeżeli wykonamy translację dla wszystkich punktów - pozostawiam to Czytelnikowi - możemy narysować nowy trójkąt:



Możemy skorzystać z metody `setToTranslate()`, aby ustawić odpowiednie współrzędne w macierzy przekształceń:

```
Matrix.prototype.setToTranslate = function(dx, dy){
        this.array[0][2]=dx;
        this.array[1][2]=dy;
};

var setToTranslate = function(matrix, dx, dy) {
        matrix.array[0][2] = dx;
        matrix.array[1][2] = dy;
        return matrix;
};
```

Pamiętajmy o tym, że przekształcany punkt, jest parametrem metody `multiply` lub `multiply2` wykonującej mnożenie wektora przez macierz przekształceń.

Skalowanie

Zmienia wymiary obiektu o współczynnik sx na osi X i wspólczynnik sy na osi Y. Jeżeli współczynnik jest większy od zera obiekt jest powiększany, jeżeli mniejszy od zera - obiekt jest pomniejszany. Skalowanie nie jest przekształceniem izometrycznym ponieważ zmienia odległości i kąty.

Macierz przekształceń dla skalowania to:

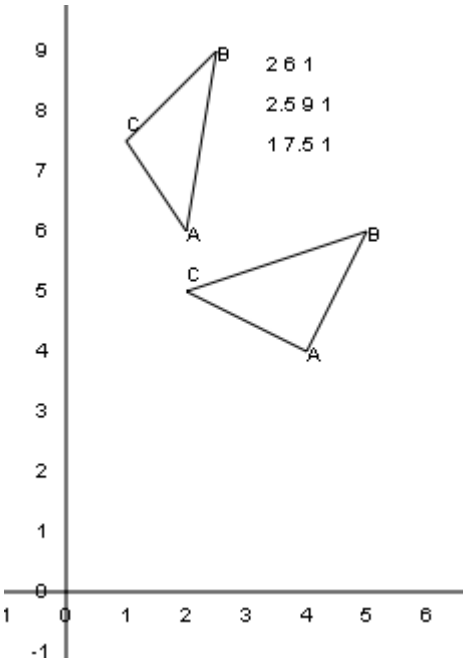
$$\begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{x} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} sx*x+0*y+0*1 \\ 0*x+sy*y+0*1 \\ 0*x+0*y+1*1 \end{bmatrix} = \begin{bmatrix} sx*x \\ sy*y \\ 1 \end{bmatrix}$$

Jeżeli punkt A = (4*4) naszego trójkąta poddamy skalowaniu sx=0.5, sy=1.5, to otrzymamy:

$$\begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 1.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{x} \begin{bmatrix} 4 \\ 4 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.5*4+0*4+0*1 \\ 0*4+1.5*4+0*1 \\ 0*4+0*4+1*1 \end{bmatrix} = \begin{bmatrix} 0.5*4 \\ 1.5*4 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 6 \\ 1 \end{bmatrix}$$

Jeżeli obliczymy pozostałe wierzchołki - pozostawiam to Czytelnikowi - to możemy narysować nasz nowy trójkąt:

(Dodatek 7 Listing97)



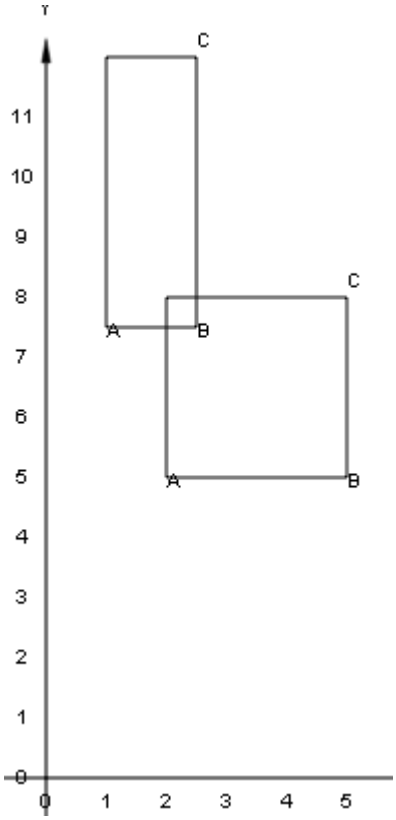
Położenie punktu A =(2,6), etc.

Przy skalowaniu możemy skorzystać z metody `setToScale()`:

```
Matrix.prototype.setToScale = function(sx, sy) {
      this.array[0][0] = sx;
      this.array[1][1] = sy;
};

var setToScale = function(matrix, sx, sy) {
      matrix.array[0][0] = sx;
      matrix.array[1][1] = sy;
      return matrix;
};
```

Skalowanie jest lepiej widoczne na takich figurach jak np. kwadrat, gdzie po prostu otrzymujemu prostokąt, jeżeli sx ≠ sy , a jeżeli sx = sy otrzymujemy zmniejszony lub powiększony kwadrat.

(Dodatek 7 Listing98)



Jeżeli chcemy zmienić rozmiar obiektu tylko wzdłuż osi Y stosujemy skalowanie (1, sy), jeżeli tylko względem osi X stosujemy skalowanie (sx, 1).

Więcej o skalowaniu znajduje się w podrozdziale poświęconym odbiciu (refleksji).

Obrót

Obrót względem punktu (0,0)

Rotacja to obrót obiektu o dany kąt. Obrót odbywa się względem punktu (0,0) osi współrzędnych. Aby jej dokonać potrzebujemy punktu i kąta obrotu α. Obrót jest przekształceniem izometrycznym, gdyż nie zmienia kątów figury, a tylko jej położenie. Macierz obrotu wygląda następująco:

$$\begin{bmatrix} \cos a & \sin a & 0 \\ -\sin a & \cos a & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Załóżmy, że chcemy obrócić nasz trójkąt o wierzchołkach A = (4,4), B=(5,6) i C=(2,5): o kąt 30° .

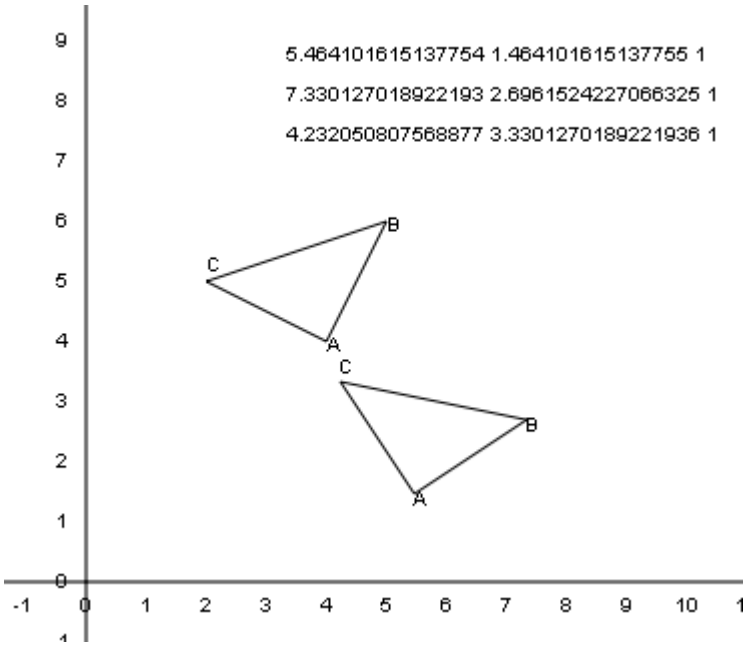
$\cos 30° = \frac{\sqrt{3}}{2} = 0.866$

$\sin 30° = 1/2 = 0.5$

$$\begin{bmatrix} 0.866 & 0.5 & 0 \\ -0.5 & 0.866 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{x} \begin{bmatrix} 4 \\ 4 \\ 1 \end{bmatrix} = \begin{bmatrix} 4*0.866+4*0.5+4*0 \\ -4*0.5+4*0.866+0*1 \\ 4*0+4*0+1*1 \end{bmatrix} = \begin{bmatrix} 5.464 \\ 1.464 \\ 1 \end{bmatrix}$$

Obliczyliśmy położenie wierzchołka A. Jeżeli policzymy współrzędne pozostałych wierzchołków - pozostawiam to Czytelnikowi - to możemy wykreślić nowy trójkąt:

(Dodatek 7 Listing99)



Kąty są liczone w kierunku ruchu wskazówek zegara.
Czasami w literaturze i Internecie spotyka się macierz obrotu 2D wyrażoną jako:

$$\begin{bmatrix} \cos a & -\sin a & 0 \\ \sin a & \cos a & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Różnica polega na tym, że nowe współrzędne punktu byłyby:

$$\begin{bmatrix} 1.464 \\ 5.464 \\ 1 \end{bmatrix}$$

czyli dokonaliśmy obrotu o kąt 30° w kierunku przeciwnym do kierunku ruchu wskazówek zegara, czyli przeciwnym kierunku niż w naszym przykładzie.

Możemy użyć metody:

```
Matrix.prototype.setToRotation = function(angleDeg) {
	var angleRad = degToRad(angleDeg);
	var t1 = Math.cos(angleRad);
	var t2 = Math.sin(angleRad);
	this.array[0][0] = t1;
	this.array[0][1] = t2;
	this.array[1][0] = -t2;
	this.array[1][1] = t1;
};

var setToRotation = function(matrix, angleDeg) {
	var angleRad = degToRad(angleDeg);
	var t1 = Math.cos(angleRad);
	var t2 = Math.sin(angleRad);
	matrix.array[0][0] = t1;
	matrix.array[0][1] = t2;
	matrix.array[1][0] = -t2;
	matrix.array[1][1] = t1;
	return matrix;
};
```

Gdybyś chciał, aby rotacja dokonywana była w kierunku przeciwnym niż ruch wskazówek zegara wystarczy w powyższej metodzie zmienić znaki na:

```
matrix.array[0][1] = -t2;
matrix.array[1][0] = t2;
```

Macierz obrotu o 90° :

$$\begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Odbicie

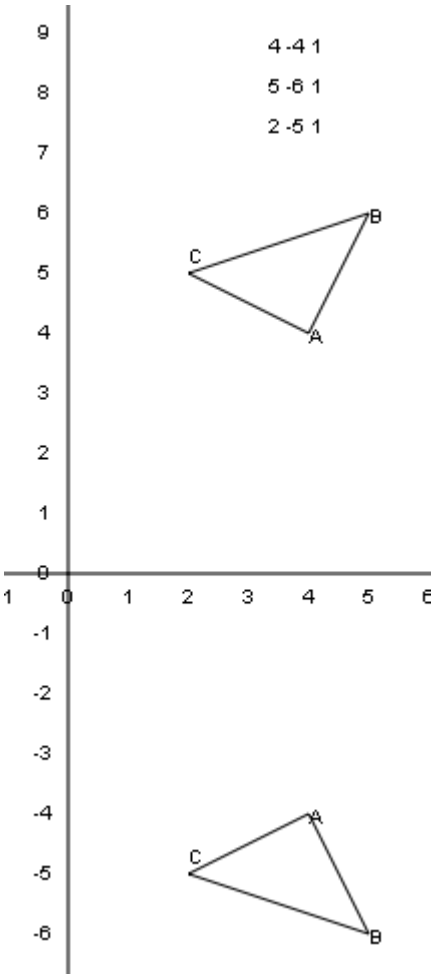
Refleksja jest przekształceniem izometrycznym. Tworzy lustrzane, symetryczne odbicie obiektu względem danej prostej. Jeżeli tą prostą jest oś X lub oś Y to refleksję możemy zastąpić skalowaniem.

Względem osi X

Symetria osiowa względem osi X.
Jest to skalowanie (sx=1, sy=-1), czyli macierz przekształceń wygląda następująco:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(Dodatek 7 Listing100)

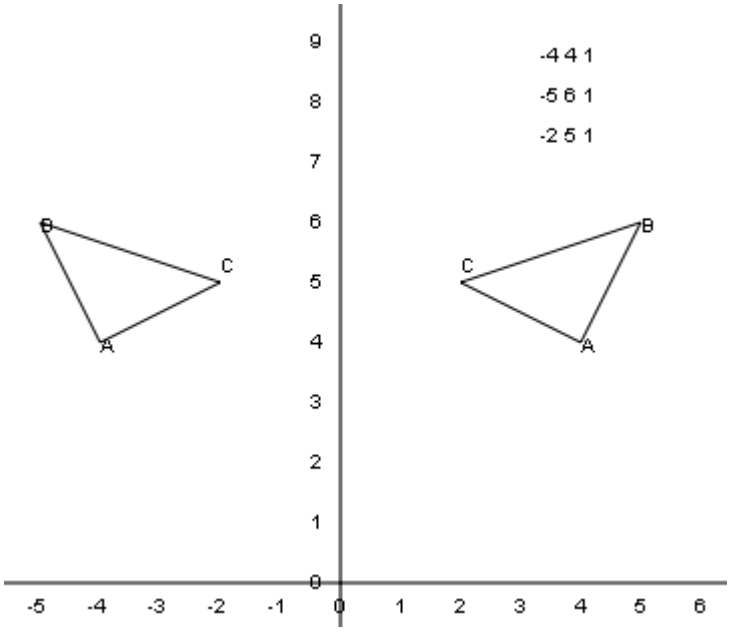


Względem osi Y

Symetria osiowa względem osi Y.
Jest to skalowanie (sx=-1, sy=1). czyli macierz przekształceń wygląda następująco:

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(Dodatek 7 Listing101)

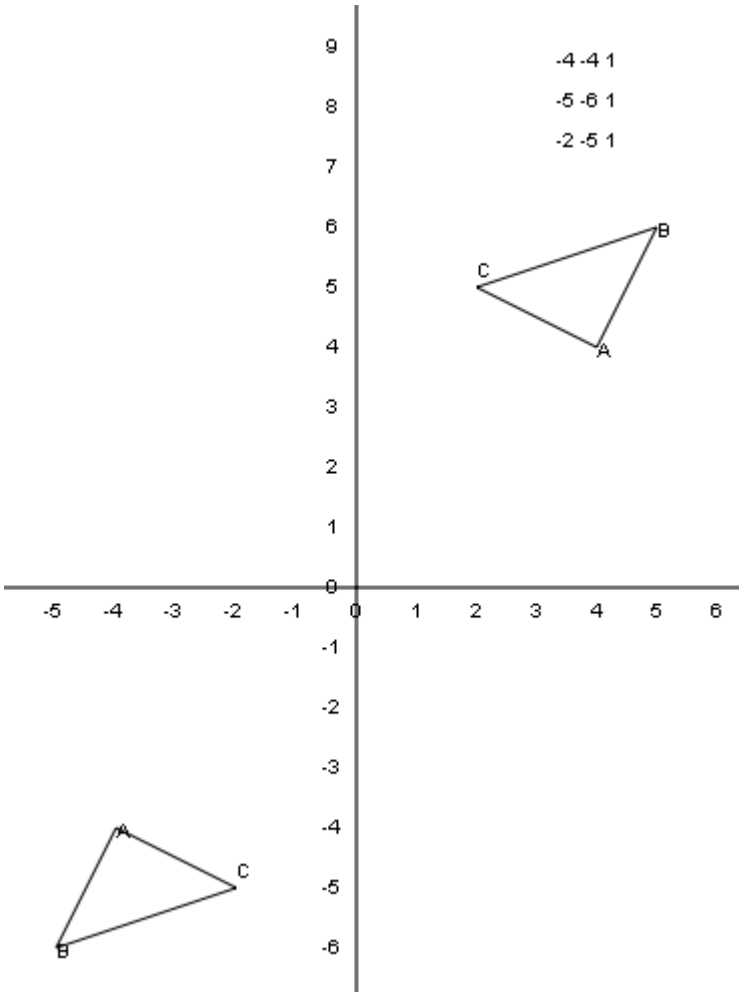


Względem osi X i osi Y

Symetria środkowa o środku w punkcie P(0,0).
Jest to skalowanie (sx=-1, sy=-1), czyli macierz obrotu wygląda następująco:

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(Dodatek 7 Listing102)



Względem prostej przechodzącej przez P(0,0)

Jeżeli równanie prostej jest typu:

y = m*x , (b=0, więc prosta przechodzi przez P(0,0) gdzie

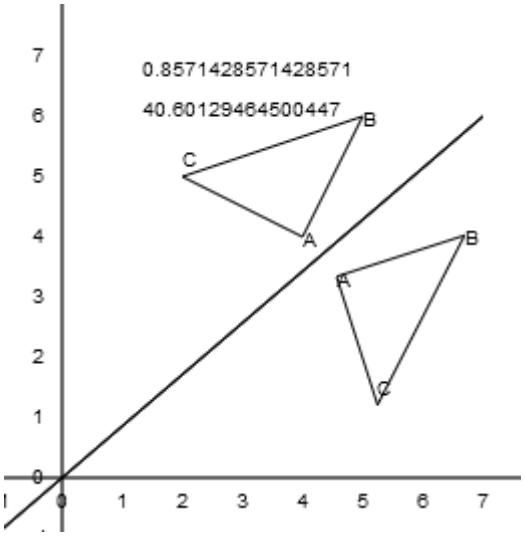
m = tgφ, gdzie

φ jest kątem pod jakim prosta przecina oś X,

to macierz refleksji wygląda następująco:

$$\begin{bmatrix} \cos 2\varphi & \sin 2\varphi & 0 \\ \sin 2\varphi & -\cos 2\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(Dodatek 7 Listing103)



Możemy użyć metody:

```
Matrix.prototype.setToReflection = function(line){
      var slope = line.slope();
      var angle = atanDeg(slope);
      var dangle = 2*angle;
      var cosa = cosDeg(dangle);
      var sina = sinDeg(dangle);
      this.array[0][0]= cosa;
      this.array[0][1]=sina;
      this.array[1][0]=sina;
      this.array[1][1]=-cosa;
};

var setToReflection = function(matrix, line){
      var slope = line.slope();
      var angle = atanDeg(slope);
      var dangle = 2*angle;
      var cosa = cosDeg(dangle);
      var sina = sinDeg(dangle);
      matrix.array[0][0]= cosa;
      matrix.array[0][1]=sina;
      matrix.array[1][0]=sina;
      matrix.array[1][1]=-cosa;
      return matrix;
};
```

Przekrzywienie (pochylenie)

Przekrzywienie jest wyrażone za pomocą macierzy:

$$\begin{bmatrix} 1 & sh_x & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + ysh_x \\ y + xsh_y \\ 1 \end{bmatrix}$$

gdzie shx oznacza przekrzywienie wzdłuż osi x, a shy przekrzywienie względem osi y.

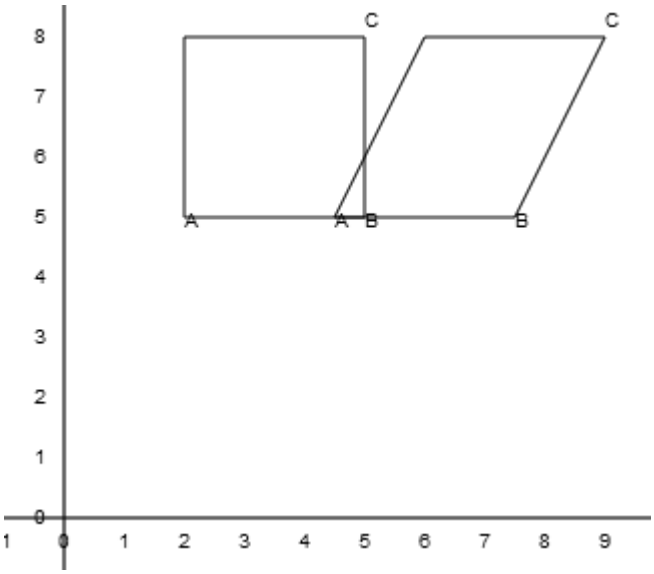
Przekrzywienie wzdłuż osi X

Przy skalowaniu shx =0.5 i shy = 0, punkt P(2,5) będzie miał nowe współrzędne P(4.5, 5) gdyż:

$$\begin{bmatrix} 1 & 0.5 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 5 \\ 1 \end{bmatrix} = \begin{bmatrix} 2*1+5*0.5 \\ 2*0+1*5 \\ 1 \end{bmatrix} = \begin{bmatrix} 4.5 \\ 5 \\ 1 \end{bmatrix}$$

Jeśli obliczymy pozostałe punkty to zyskamy obraz:

(Dodatek 7 Listing108)



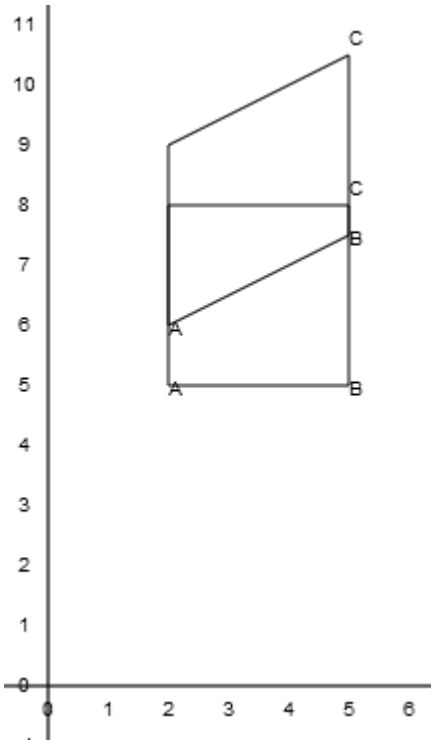
Przekrzywienie względem osi Y

Przy skalowaniu shx =0 i shy = 0.5, punkt P(2,5) będzie miał nowe współrzędne P(2, 6) gdyż:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0.5 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 5 \\ 1 \end{bmatrix} = \begin{bmatrix} 2*1+5*0 \\ 2*0.5+1*5 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 6 \\ 1 \end{bmatrix}$$

Gdy obliczymy wszystkie punkty uzyskamy obraz:

(Dodatek 7 Listing109)



Do przekrzywiania możemy użyć metod:

```
Matrix.prototype.setToShear = function(shx,shy) {
      this.array[0][1] = shx;
      this.array[1][0] = shy;
};

var setToShear = function(matrix, shx, shy) {
      matrix.array[0][1] = shx;
      matrix.array[1][0] = shy;
      return matrix;
};
```

Przekształcenia złożone

Jeżeli przyjrzałeś się uważnie przykładowi odbicia względem prostej przechodzącej przez środek układu współrzędnych zauważyłeś zapewne, że ten przykład można by wykonać dokonując trzech przekształceń po kolei:

- Rotacja o kąt φ w kierunku ruchu wskazówek zegara, tak aby linia (oś symetrii) pokryła się z osią współrzędnych
- Skalowanie (odbicie) punktu względem osi X
- Rotacja o kąt φ w kierunku przeciwnym do ruchu wskazówek zegara, tak aby linia (oś symetrii) wróciła na swoje miejsce

Obrót względem punktu, który nie jest punktem P(0,0)

(Dodatek 7 Listing104)

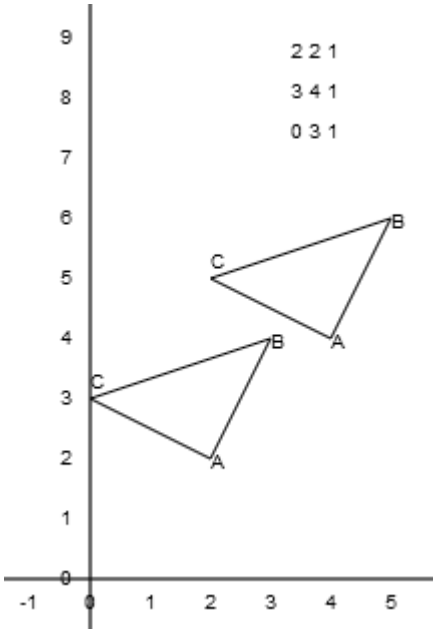
Żeby obrócić figurę względem punktu, który nie jest P(0,0), ale punktem np. P(x, y) należy:

- przesunąć punkt P(x,y) do punktu P(0,0). Jest to translacja: $\begin{bmatrix} 0 & 0 & -x \\ 0 & 0 & -y \\ 0 & 0 & 1 \end{bmatrix}$
- obrócić figurę o wskazany kąt φ, we wskazanym kierunku. Jest to rotacja:
  $\begin{bmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}$
- przesunąć punkt z P(0,0) do punktu P(x, y). Jest to translacja: $\begin{bmatrix} 0 & 0 & x \\ 0 & 0 & y \\ 0 & 0 & 1 \end{bmatrix}$

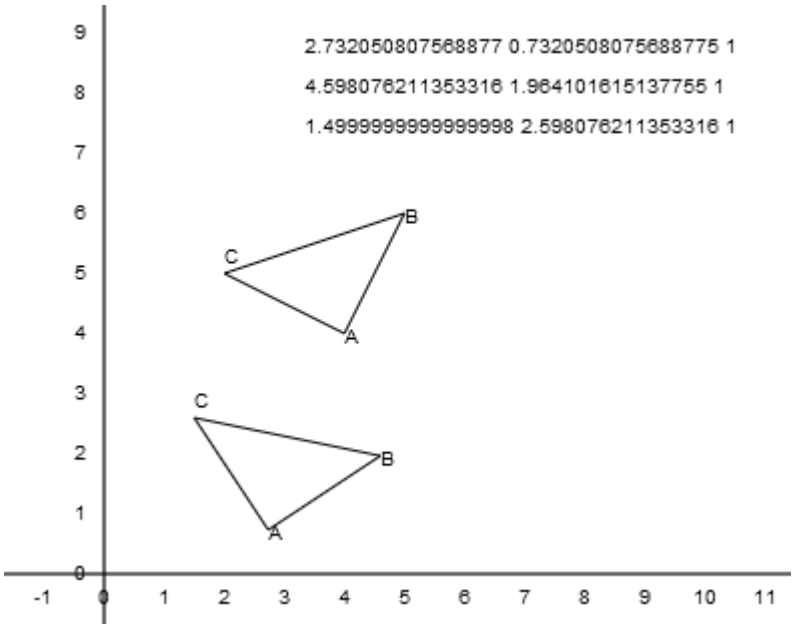
Kolejność przekształceń - oczywiście - ma znaczenie.

Dokonujemy obrotu naszego trójkąta o wierzchołkach A = (4,4), B=(5,6) i C=(2,5) o kąt 30° względem punktu P(2,2).

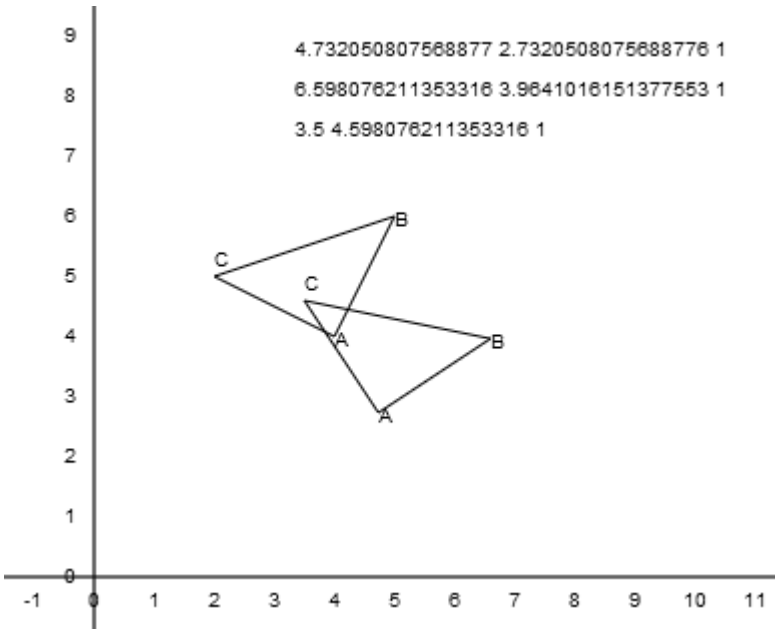
Pierwsza translacja:



Obrót względem punktu P(0,0) w kierunku ruchu wskazówek zegara.



Translacja powrotna:



Obrót w miejscu

(Dodatek 7 Listing105)

Jest to szczególny przypadek obrotu względem punktu, który nie jest punktem P(0,0). Problem polega na tym, że obroty wykonywane są na ogół w stosunku do lewego górnego rogu obszaru zajmowanego przez figurę. Jeżeli chcemy dokonać obrotu figury w miejscu musimy go dokonać w stosunku do środka ciężkości figury, czyli dokonać translacji wszystkich punktów o współrzędne punktu M, a nie o współrzędne punktów wyznaczających figurę. W przypadku większości figur ustalenie punktu ciężkości nie jest trudne.
W przypadku trójkąta o wierzchołkach $P_1(x_1,y_1)$, $P_2(x_2, y_2)$, $P_3(x_3, y_3)$ środek ciężkości przypada w punkcie:

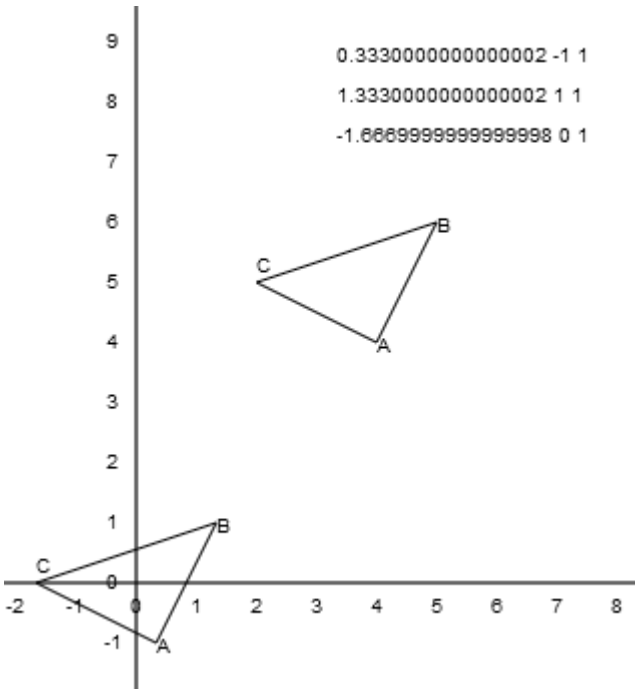
$$P = \left(\frac{x_1 + x_2 + x_3}{3}, \frac{y_1 + y_2 + y_3}{3} \right)$$

W przypadku naszego trójkąta, o wierzchołkach A = (4,4), B=(5,6) i C=(2,5) środek ciężkości figury przypada w punkcie:

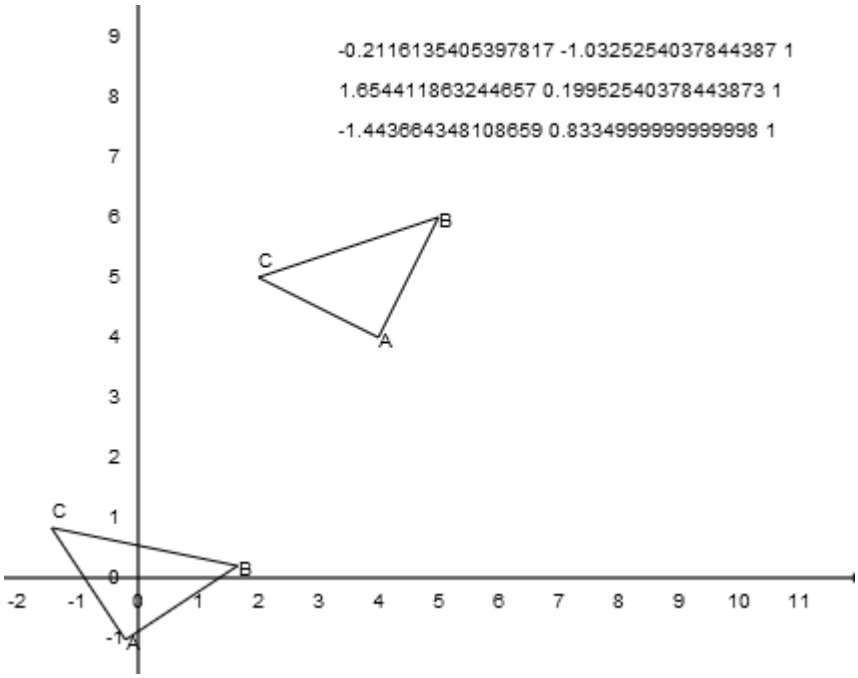
M(3.667, 5).

Dokonujemy obrotu trójkąta o 30° w miejscu.
A oto nasze trzy etapy.

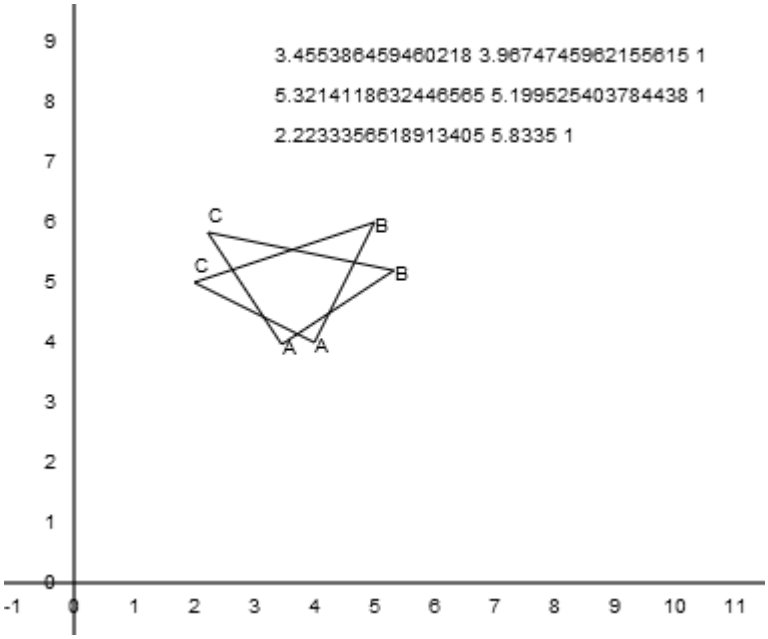
Translacja



Obrót



Translacja powrotna



Skalowanie w miejscu

(Dodatek 7 Listing106)

.

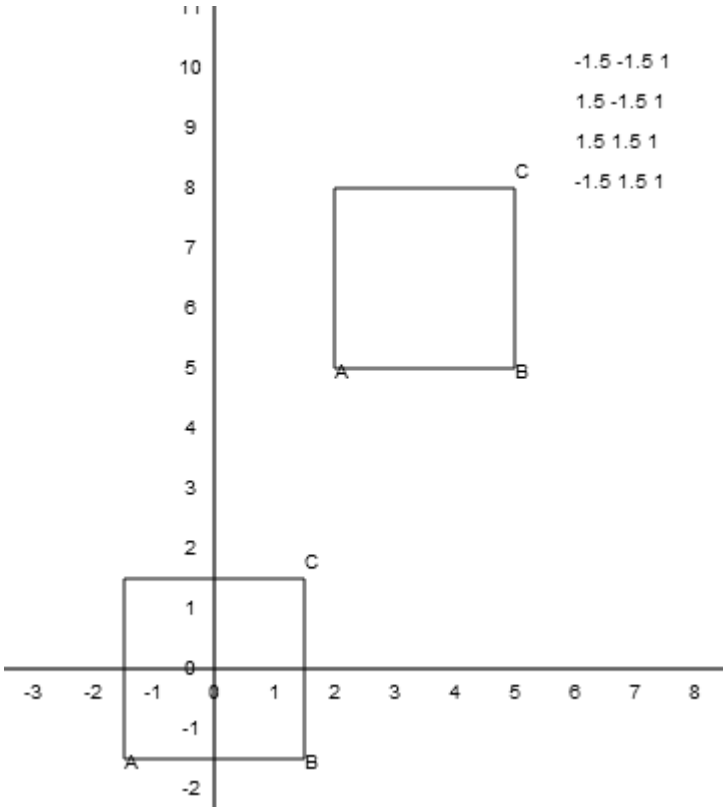
Skalowanie również wykonywane jest względem punktu P(0,0). Aby wykonać skalowanie w miejscu musimy dokonać:

- translacji figury o współrzędne środka ciężkości tej figury.
- skalowania

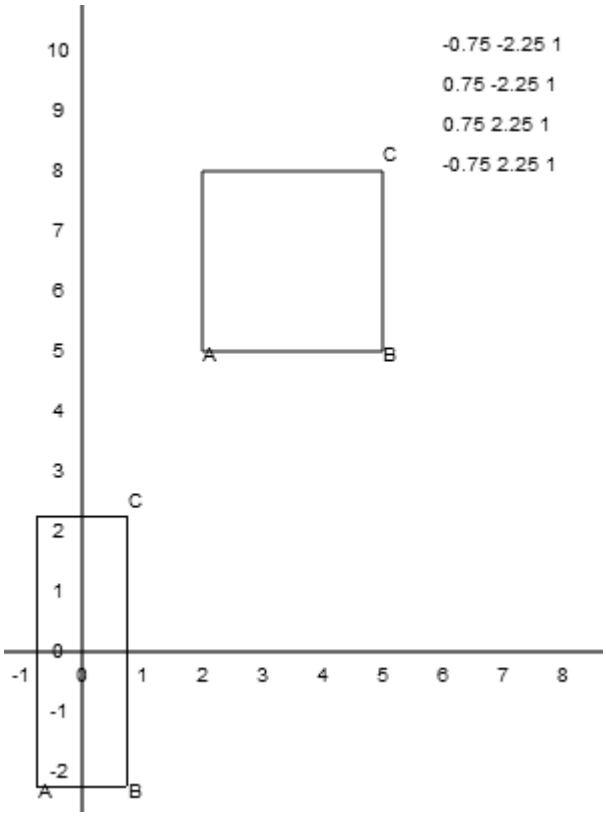
- translacji powrotnej

Weźmy nasz kwadrat z pliku *trans03.html*.
Dokonujemy skalowania w miejscu;

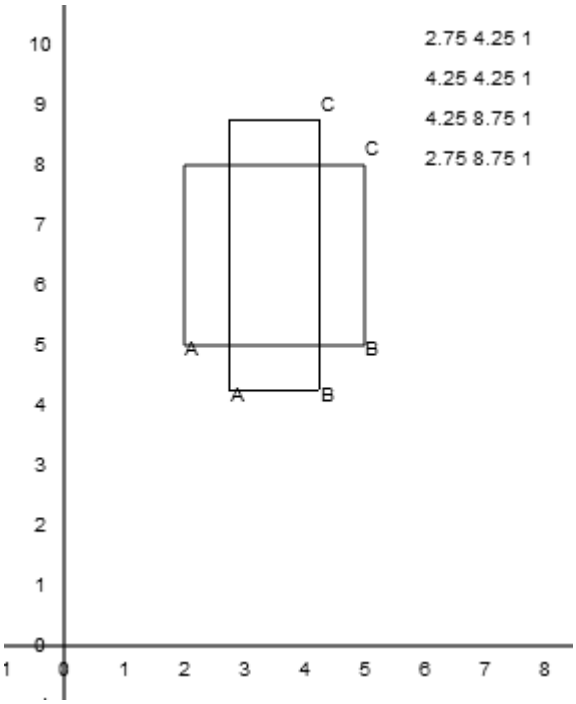
Translacja



Skalowanie



Translacja powrotna



Odbicie względem prostej nie przechodzącej przez punkt P(0,0)

Musimy rozważyć dwa przypadki:

- prosta równoległa do osi Y, która nie może być wyrażona równaniem y = mx+b

- prosta, która może być wyrażona równaniem y=mx+b, gdzie m = tgα, gdzie α oznacza kąt pod jakim prosta przecina oś X.

W przypadku prostej równoległej do osi Y musimy:
- dokonać translacji o odległość linii od osi Y, tak aby prosta pokryła się z osią Y
- dokonać odbicia punktu względem osi Y
- dokonać translacji powrotnej o odległość linii od osi Y, ale z odwrotnym znakiem.

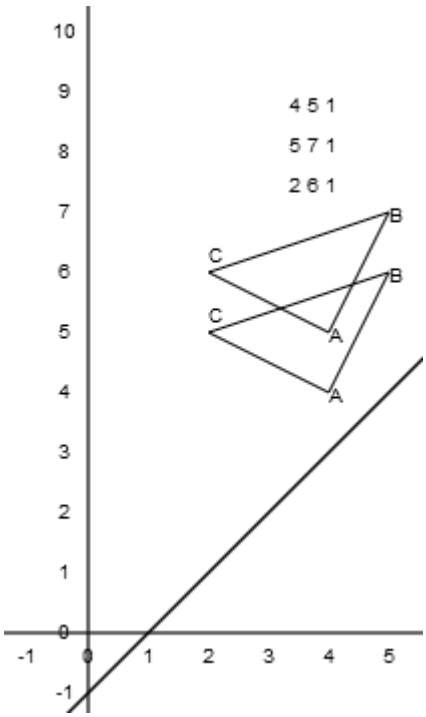
Tego przypadku nie będziemy omawiali szczegółowo, gdyż jest zbyt trywialny.

W przypadku prostej wyrażonej równaniem y = mx + b musimy:
- wykonać translację o odcinek - b, tak aby prosta przechodziła przez punkt P(0,0).

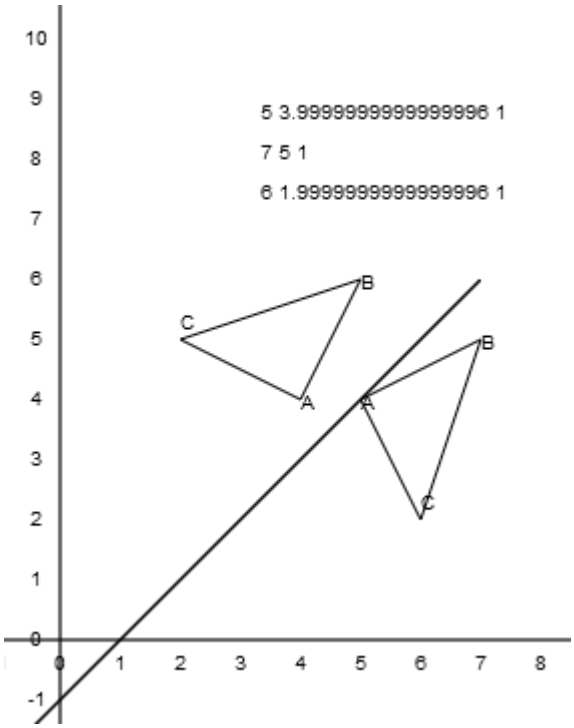
  Macierz translacji: $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -b \\ 0 & 0 & 1 \end{bmatrix}$
- Odbić punkt względem przesuniętej linii. Macierz refleksji: $\begin{bmatrix} \cos 2\varphi & \sin 2\varphi & 0 \\ \sin 2\varphi & -\cos 2\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}$
- dokonać translacji zwrotnej o odcinek b. Macierz translacji: $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & b \\ 0 & 0 & 1 \end{bmatrix}$

(Dodatek 7 Listing107)

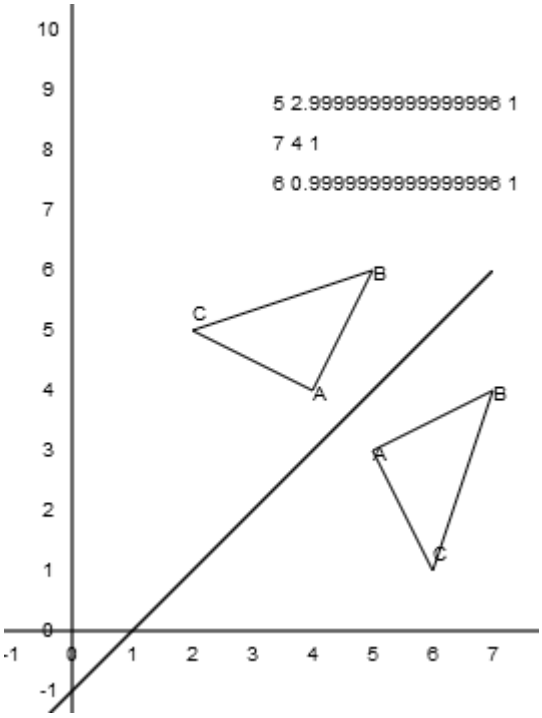
Translacja:



Odbicie przesuniętej figury względem przesuniętej prostej przechodzącej przez punkt P(0,0):



Translacja powrotna:



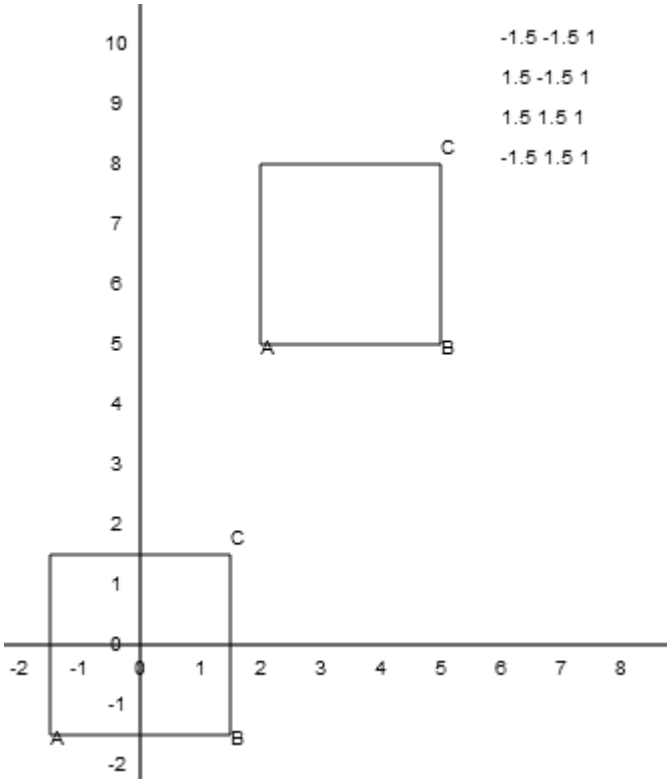
Przekrzywienie względem środka ciężkości figury

(Dodatek 7 Listing110)

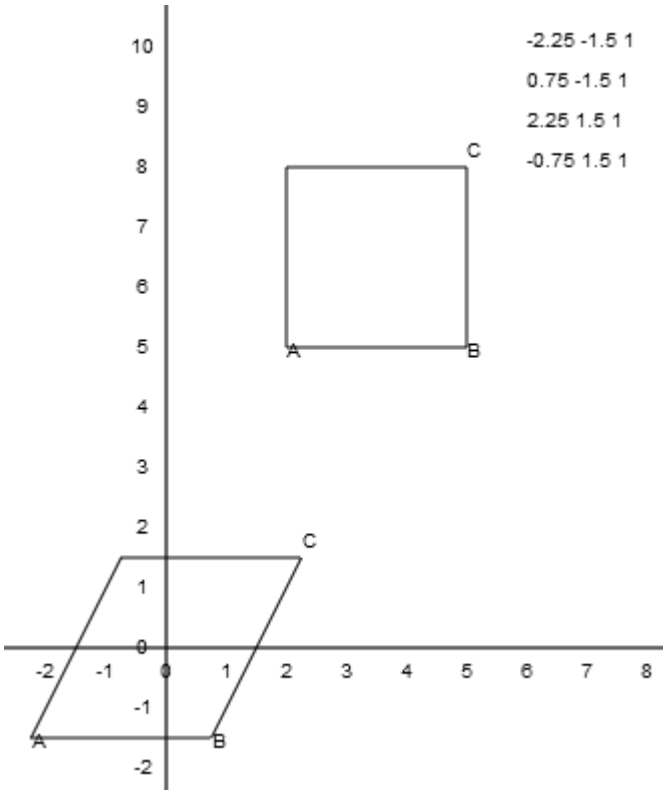
Aby dokonać transformacji należy:

- przesunąć środek ciężkości figury do punktu P(0,0)
- przekrzywić figurę
- wykonać translację zwrotną

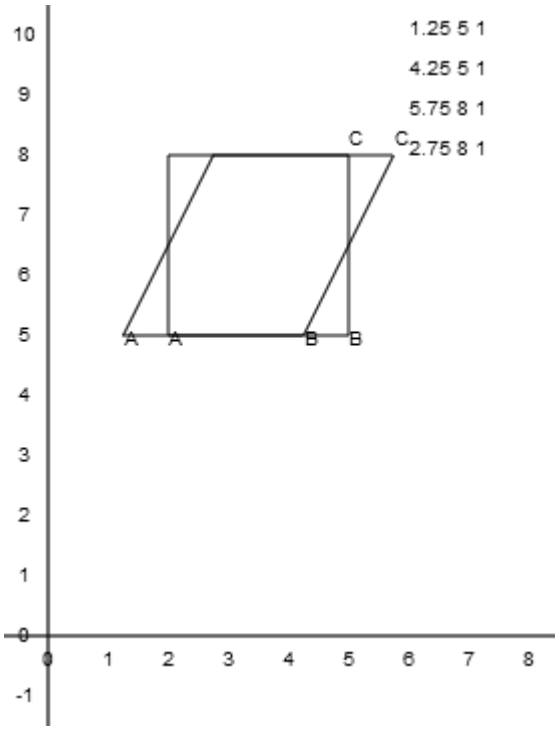
Translacja wstępna



Przekrzywienie



Translacja zwrotna



Składanie macierzy przekształceń

(Dodatek 7 Listing110)

Aby uzyskać punkt wynikowy w przypadku przekształceń złożonych dokonywaliśmy następujących operacji, które omówimy na przykładzie przekrzywienia względem środka ciężkości figury:

Ustalamy punkt do przekształcenia. Jest to jeden z punktów narożnych:

```
var pointa = [ 2, 5, 1 ];
```

Ustalamy punkt środkowy kwadratu:

```
var tpoint = [ 3.5, 6.5, 1 ];
```

Ustalamy parametry przekształcenia:

```
var shx = 0.5;
var shy = 0;
```

Tworzymy z punktu narożnego macierz kolumnową:

```
var mata = new Matrix(pointa, 1)
```

$$\begin{bmatrix} 2 \\ 5 \\ 1 \end{bmatrix}$$

Tworzymy macierz dla pierwszej translacji. Macierz jest zerowa:

```
var transa1 = new Matrix();
```

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Przekształcamy macierz zerową w jednostkową:

```
transa1.setToIdentity();
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Ustawiamy macierz pierwszej translacji:

```
transa1.setToTranslate(-tpoint[0], -tpoint[1]);
```

$$\begin{bmatrix} 1 & 0 & -3.5 \\ 0 & 1 & -6.5 \\ 0 & 0 & 1 \end{bmatrix}$$

Uzyskujemy wektor kolumnowy zawierający współrzędne punktu po translacji:

```
var aa1 = transa1.multiply2(mata);
```

$$\begin{bmatrix} -1.5 \\ -1.5 \\ 1 \end{bmatrix}$$

Tworzymy macierz dla przekrzywienia:

```
var transa2 = new Matrix();
```

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Ustawiamy macierz przekrzywienia jako jednostkową:

```
transa2.setToIdentity();
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Ustawiamy macierz przekrzywienia:

```
transa2.setToShear(shx, shy);
```

$$\begin{bmatrix} 1 & 0.5 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Tworzymy wektor kolumnowy zawierający wsporzędne punktu po przekrzywieniu:

```
var aa2 = transa2.multiply2(aa1);
```

$$\begin{bmatrix} -2.25 \\ -1.5 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0.5 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1.5 \\ -1.5 \\ 1 \end{bmatrix}$$

Tworzymy macierz translacji zwrotnej:

```
var transa3 = new Matrix();
```

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Ustawiamy ją jako jednostkową:

```
transa3.setToIdentity()
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Ustawiamy macierz translacji zwrotnej:

```
transa3.setToTranslate(tpoint[0], tpoint[1]);
```

$$\begin{bmatrix} 1 & 0 & 3.5 \\ 0 & 1 & 6.5 \\ 0 & 0 & 1 \end{bmatrix}$$

Tworzymy wektor kolumnowy zawierające końcowe współrzędne punktu:

```
var aa3 = transa3.multiply2(aa2);
```

$$\begin{bmatrix} 1.25 \\ 5 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 3.5 \\ 0 & 1 & 6.5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -2.25 \\ -1.5 \\ 1 \end{bmatrix}$$

Całość tych operacji możemy ująć skrótowo:

$$\begin{bmatrix} 1.25 \\ 5 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 3.5 \\ 0 & 1 & 6.5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0.5 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -3.5 \\ 0 & 1 & -6.5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 5 \\ 1 \end{bmatrix}$$

Czytając od prawej do lewej, czyli w kolejności wykonywania operacji mamy:

- punkt początkowy
- macierz translacji poczatkowej
- macierz przekrzywienia
- macierz translacji zwrotnej

Dodatek 6. Krzywe Beziera

Wielomiany Bernsteina

Definicja

Wielomian Bernsteina stopnia n jest definiowany następująco:

$$B_{n,k}(t) = \binom{n}{k} t^k (1-t)^{n-k}$$

gdzie:
B - jest wielomianem składowym
symbol Newtona $\binom{n}{k} = \frac{n!}{k!(n-k)!}$
parametr $t \in [0,1]$, jest liczbą z przedziału zamkniętego od 0 do 1.
n - jest stopniem wielomianu
k - jest kolejnym numerem wielomianu składowego

Wielomiany Bernsteina są funkcjami bazowymi dla krzywych Beziera.

Obliczenia

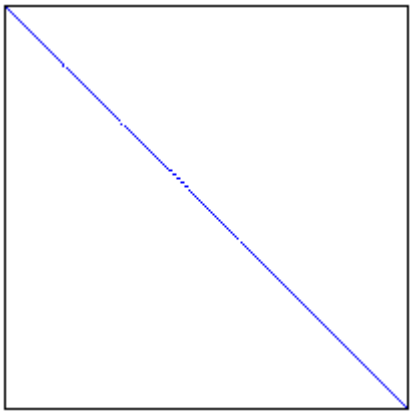
n=0

$$B_{0,0}(t) = \binom{0}{0} t^0 (1-t)^0 = 1$$

n=1

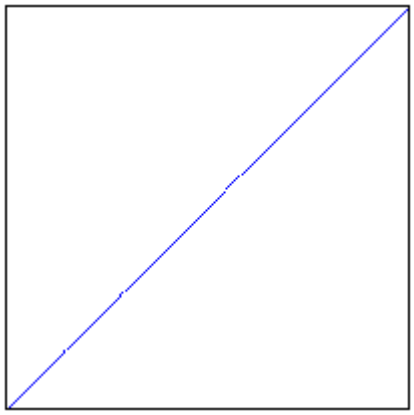
$$B_{1,0}(t) = \binom{1}{0} t^0 (1-t)^1 = 1-t$$

Wykres $B_{1,0}$ (Dodatek 7 Listing 111)
Na tym wykresie i wszystkich innych dotyczących wielomianu Bersteina t zawiera się w przedziale [0,1] i jest wykreślone na osi X. Wartości $B_{n,i}$ zawierają się w przedziale [0,1] i są wykreślone na osi Y, z $P_{0,0}$ w dolnym lewym rog wykresu.

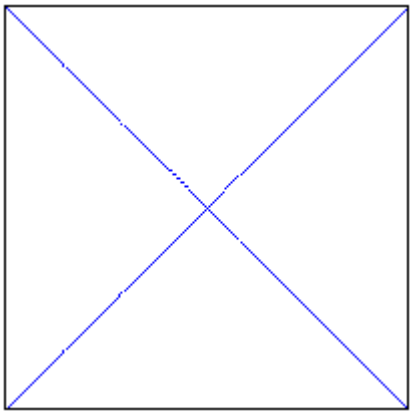


$$B_{1,1}(t) = \binom{1}{1} t^1 (1-t)^0 = t$$

Wykres $B_{1,1}$ $B_{1,0}$ (Dodatek 7 Listing 112)



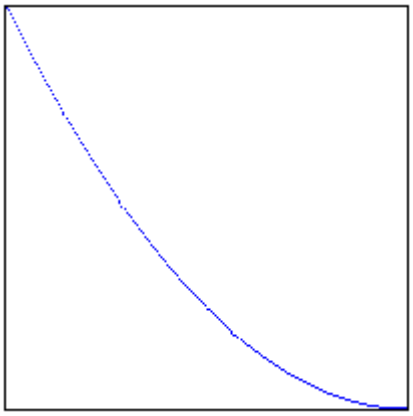
Wykres $B_{1,k}$ $B_{1,0}$ (Dodatek 7 Listing 113)



n=2

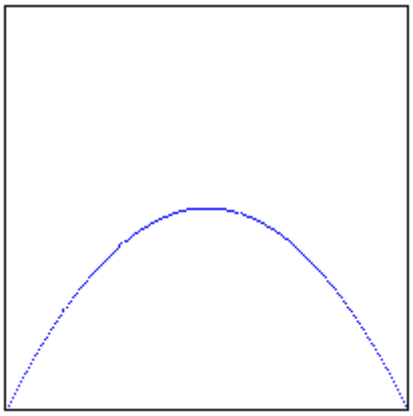
$$B_{2,0}(t) = \binom{2}{0} t^0 (1-t)^2 = (1-t)^2$$

Wykres $B_{2,0}$ $B_{1,0}$ (Dodatek 7 Listing 114)



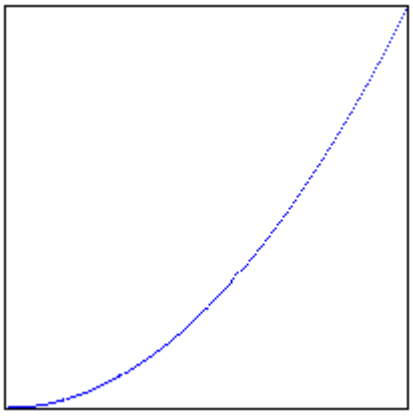
$$B_{2,1}(t) = \binom{2}{1} t^1 (1-t)^1 = 2t(1-t)$$

Wykres $B_{2,1}$ $B_{1,0}$ (Dodatek 7 Listing 115)

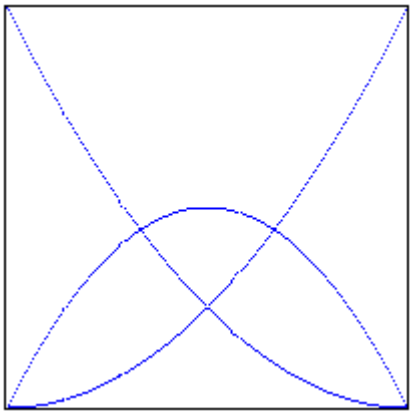


$$B_{2,2}(t) = \binom{2}{2} t^2 (1-t)^0 = t^2$$

Wykres $B_{2,2}$ $B_{1,0}$ (Dodatek 7 Listing 116)



Wykres $B_{2,k}$ $B_{1,0}$ (Dodatek 7 Listing 117)



n=3

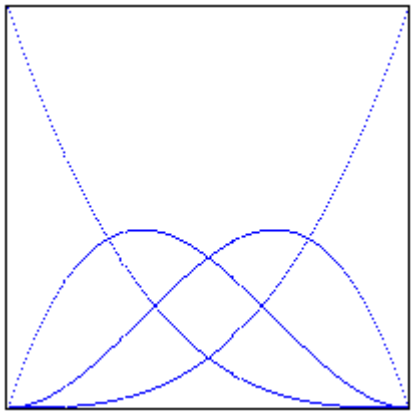
$$B_{3,0}(t) = \binom{3}{0} t^0 (1-t)^3 = (1-t)^3$$

$$B_{3,1}(t) = \binom{3}{1} t^1 (1-t)^2 = 3t(1-t)^2$$

$$B_{3,2}(t) = \binom{3}{2} t^2 (1-t)^1 = 3t^2(1-t)$$

$$B_{3,3}(t) = \binom{3}{3} t^3 (1-t)^0 = t^3$$

Wykres $B_{3,k}$ $B_{1,0}$ (Dodatek 7 Listing 118)



Algorytm

Wartość w punkcie t:

```
var bernsteinTValue=function(n, k, t){
      var btv = npok(n,k)*Math.pow(t,k)*Math.pow(1.0-t, n-k);
      if(btv<0){
            btv=0;
      }
      return btv;

};
```

Wykres wielomianu:

```
var bernstein = function(n, k, liczbaPunktow) {
      var t;
      var b;
      var img1 = ctx.getImageData(0, 0, width, height);
      var idata = img1.data;
      var a = npok(n, k);
      for ( var i = 0; i < liczbaPunktow; i++) {
            t = i * 1.0 / liczbaPunktow;
            b = a * Math.pow(t, k) * Math.pow(1.0 - t, n - k);
            if (b <= 0) {
                  b = 0;
            }
            var j = 4 * index(cutDecimal(height - b * liczbaPunktow),
            cutDecimal(t
                        * liczbaPunktow), width);
            idata[j] = 0;
            idata[j + 1] = 0;
            idata[j + 2] = 255;
            idata[j + 3] = 255;
      }
      ctx.putImageData(img1, 0, 0);
};
```

Właściwości

Wielomiany Bernsteina jako funkcje mają następujące właściwości:

Właściwość 1

Są nieujemne, gdyż dla dowolnego n, k i $t \in [0,1]$ wartość $B_{n,k}(t) \geq 0$

Właściwość 2

Dla danego n i każdego $t \in [0,1]$ suma $\sum_{k=0}^{n} B_{n,k}(t) = 1$

Dla $n = 2$:
$(1-t)^2 + 2t(1-t) + t^2 = 1$

Dla $n = 3$:
$(1-t)^3 + 3t(1-t)^2 + 3t^2(1-t) + t^3 = 1$

Właściwość 3

$B_{n,0}(0) = B_{n,1}(10 = 1$

Inne sposoby obliczania

Wielomiany Bernsteina można również obliczyć stosując wzory rekurencyjne:

$B_{n,k}(t) = 0$ dla $k < 0 \,||\, k > n$

$B_{n,k}(t) = (1-t)B_{n-1,k)}(t) + tB_{n-1,k-1}(t)$ dla pozostałych k

n=0

$$B_{0,0}(t) = 1$$

n=1

$$B_{1,0}(t) = (1-t)B_{0,0}(t) + tB_{0,-1}(t) = 1-t$$
$$B_{1,1}(t) = (1-t)B_{0,1}(t) + tB_{0,0}(t) = t$$

n=2

$$B_{2,0}(t) = (1-t)B_{1,0}(t) + tB_{1,-1}(t) = (1-t)^2$$
$$B_{2,1}(t) = (1-t)B_{1,1}(t) + tB_{1,0}(t) = 2t(1-t)$$
$$B_{2,2}(t) = (1-t)B_{1,2}(t) + tB_{1,1}(t) = t^2$$

n=3

$$B_{3,0}(t) = (1-t)B_{2,0}(t) + tB_{2,-1}(t) = (1-t)^3$$
$$B_{3,1}(t) = (1-t)B_{2,1}(t) + tB_{2,0}(t) = 3t(1-t)^2$$
$$B_{3,2}(t) = (1-t)B_{2,2}(t) + tB_{2,1}(t) = 3t^2(1-t)$$
$$B_{3,3}(t) = (1-t)B_{2,3}(t) + tB_{2,2}(t) = t^3$$

Pochodne

Pochodne wielomianów można obliczyć z zależności:

$$B'_{n,k} = n\left[B_{n-1,k-1}(t) - B_{n-1,k}(t)\right]$$

przy czym wiadomo, że:

$$B_{n-1,-1}(t) = B_{n-1,n}(t) = 0$$

Pochodne obliczone według wzoru

$$B'_{0,0}(t) = 0[...] = 0$$
$$B'_{1,0}(t) = 1[B_{0,-1}(t) - B_{0,0}(t)] = 0 - 1 = -1$$
$$B'_{1,1}(t) = 1[B_{0,0}(t) - B_{0,1}(t)] = 1 - 0 = 1$$
$$B'_{2,0}(t) = 2[B_{1,-1}(t) - B_{1,0}(t) = 2(0-(1-t)) = -2(1-t)$$
$$B'_{2,1}(t) = 2[B_{1,0}(t) - B_{1,1}(t)] = 2((1-t)-t) = 2(1-2t)$$
$$B'_{2,2}(t) = 2[B_{1,1}(t) - B_{1,2}(t)] = 2(t-0) = 2t$$
$$B'_{3,0}(t) = 3[B_{2,-1}(t) - B_{2,0}(t)] = 3(0-(1-t)^2) = -3(1-t)^2$$
$$B'_{3,1}(t) = 3[B_{2,0}(t) - B_{2,1}(t)] = 3((1-t)^2 - 2t(1-t)) = 3(1-t)(1-3t)$$
$$B'_{3,2}(t) = 3[B_{2,1}(t) - B_{2,2}(t)] = 3(2t(1-t) - t^2) = 3t(2-3t)$$
$$B'_{3,3}(t) = 3[B_{2,2}(t) - B_{2,3}(t)] = 3(t^2 - 0) = 3t^2$$

Pochodne obliczone klasycznie

$B_{0,0}(t) = 1$ $\quad$ $B'_{0,0}(t) = 0$

$B_{1,0}(t) = 1 - t$ $\quad$ $B'_{1,0}(t) = -1$

$B_{1,1}(t) = t$ $\quad$ $B'_{1,1}(t) = 1$

$B_{2,0}(t) = (1-t)^2$ $\quad$ $B'_{2,0}(t) = -2(1-t)$

$B_{2,1}(t) = 2t(1-t)$ $\quad$ $B_{2,1}(t) = 2(1-2t)$

$B_{2,2}(t) = t^2$ $\quad$ $B_{2,2}(t) = 2t$

$B_{3,0}(t) = (1-t)^3$ $\quad$ $B'_{3,0}(t) = 3(1-t)^2$

$B_{3,1}(t) = 3t(1-t)^2$ $\quad$ $B'_{3,1}(t) = 3(1-t)(1-3t)$

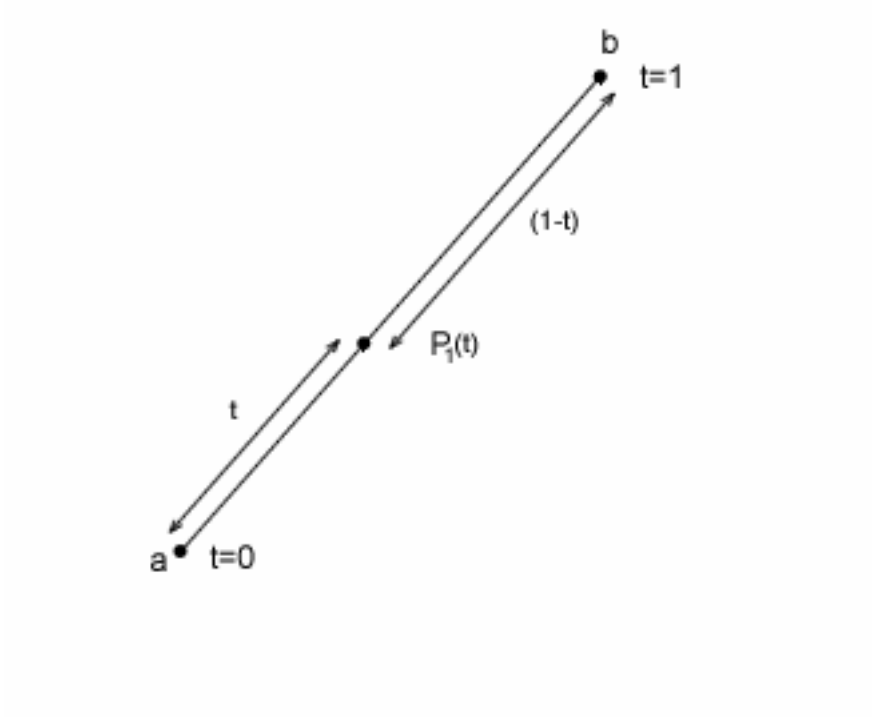
$B_{3,2}(t) = 3t^2(1-t)$ $\quad$ $B'_{3,2}(t) = 3t(2-3t)$

$B_{3,3}(t) = t^3$ $\quad$ $B'_{3,3}(t) = 3t^2$

Jak widać wyniki są identyczne, ale pochodne według wzoru są łatwiejsze do zalgorytmizowania.

Krzywa Beziera 1-go stopnia

Spójrzmy na poniższy rysunek:



Na odcinku ab leży punkt t. W punkcie a parametr $t = 0$, w punkcie b parametr $t = 1$. Możemy to zapisać w postaci równania:

$$(1-t) + t = 1$$

Możemy to wyrazić w postaci funkcji:

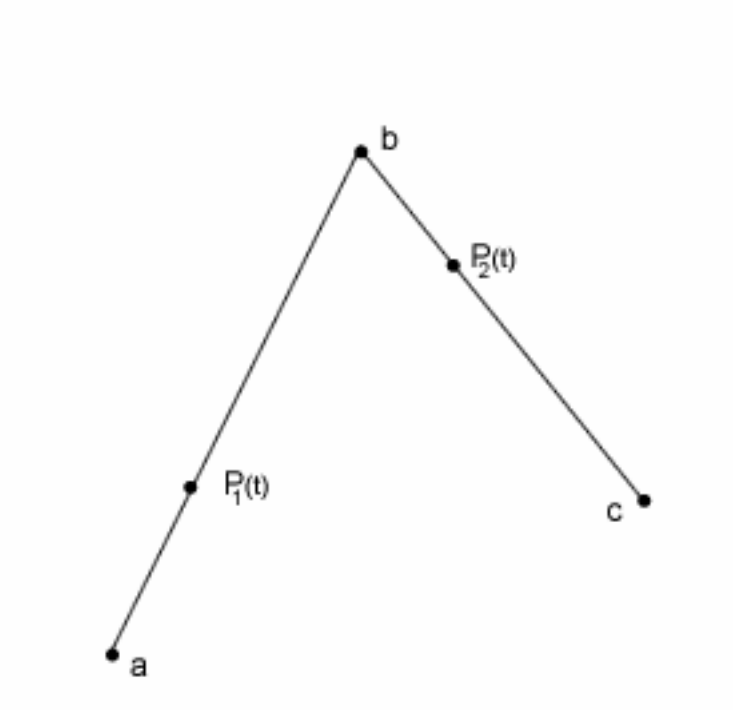
$$f(t) = tb + (1-t)a$$

Jeżeli do tego równania podstawimy $t = 1$ to otrzymamy b, jeśli podstawimy $t = 0$ to otrzymamy a.
Punkt t jest ruchomy i może znajdować się w różnych miejscach, ale równanie zawsze będzie prawidłowe.
Krzywa Beziera 1-go stopnia nie jest używana w praktyce.

Krzywa Beziera 2-go stopnia

Tworzenie krzywej

Spójrz na rysunek następny:



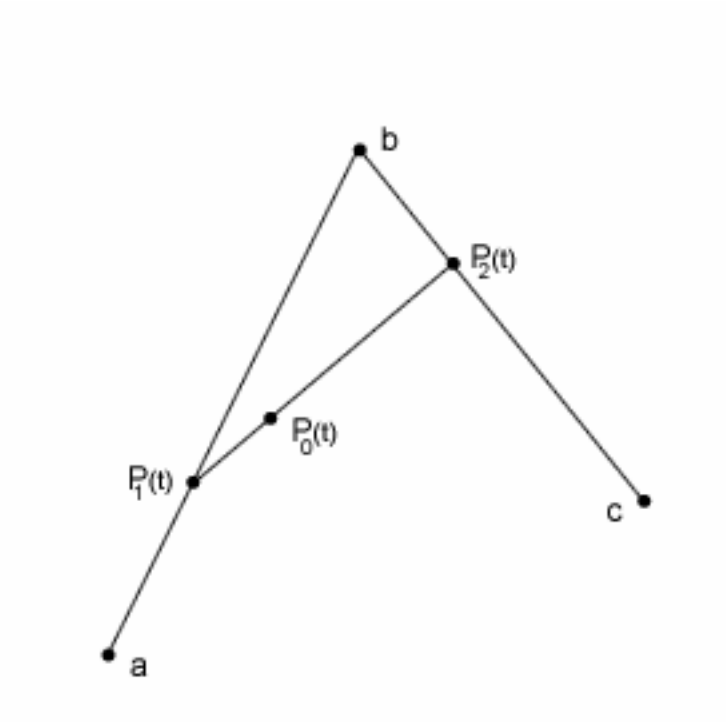
Równanie dla odcinka ab:

$$f(t) = tb + (1-t)a$$

Równanie dla odcinka bc możemy analogicznie zapisać w postaci:

$$g(t) = tc + (1-t)b$$

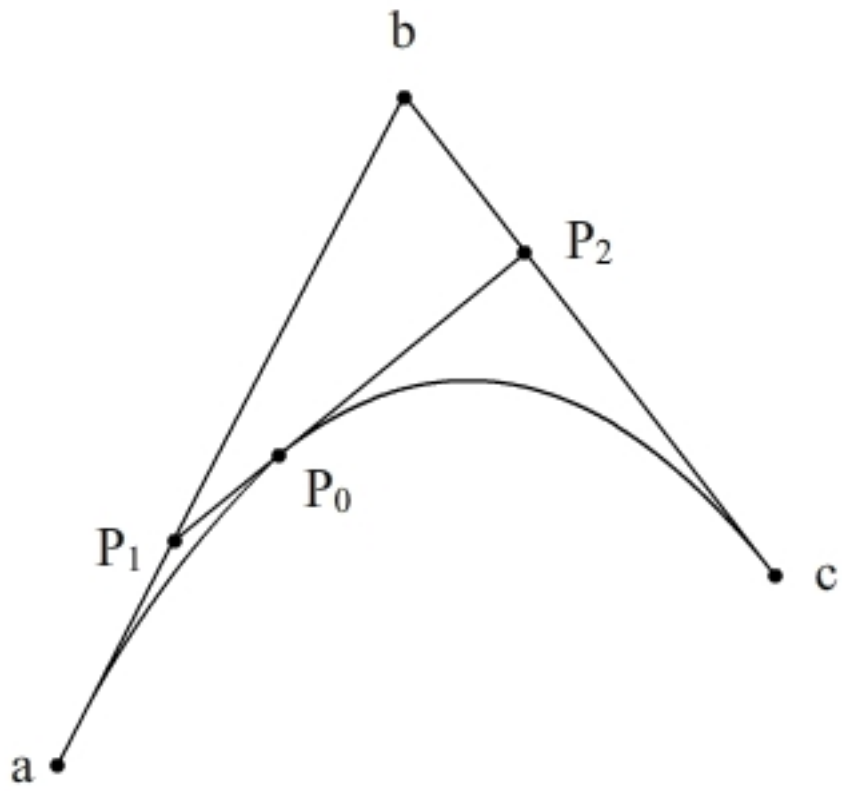
Spójrzmy na następny rysunek:



Pojawia się pytanie: jak zapisać równanie ruchu poruszającego się punktu $P_0(t)$, gdy jednocześnie poruszają się punkty $P_1(t)$ i $P_2(t)$?
Możemy je zapisać następująco:

$$h(t) = t * g(t) + (1-t) * f(t) =$$
$$t(tc + (1-t)b) + (1-t)(tb + (1-t)a) =$$
$$t^2(a - 2b + c) + t(-2a + 2b) + a$$

Otrzymaliśmy równanie krzywej Béziera, czyli równanie krzywej jaką kreśli poruszający się punkt $P_0(t)$. Jeżeli za t podstawisz 0, otrzymasz a, jeżeli podstawisz 1, otrzymasz c. Jest tak ponieważ punkty a, c i $P_0(t)$ leżą na krzywej. Wartości b nie uda ci się obliczyć, niezależnie od wartości t, gdyż punkt b nie leży na krzywej, lecz poza nią. Punkt b znajdzie się na krzywej tylko w tym szczególnym przypadku, gdy nasza krzywa będzie linią prostą. Punkt b jest punktem kontrolnym krzywej.
A tak wygląda kompletny układ wraz z kreśloną krzywą:



Obliczenia

Spróbujmy wyznaczyć położenie punktu na krzywej dla $t = 0.36$, gdy

- punkt a na krzywej ma współrzędne $a(0,1)$
- punkt kontrolny krzywej b ma współrzędne $b(4,5)$
- punkt c na krzywej ma współrzędne $c(2,3)$

$h(t) = t^2(a - 2b + c) + t(-2a + 2b) + a$

$$\begin{cases} x(0.36) = 0.36^2 * (0 - 2*4 + 2) + 0.36*(-2*0 + 2*4) + 0 = -0.7776 + 2.88 = 2.1024 \\ y(0.36) = 0.36^2 * (1 - 2*5 + 3) + 0.36*(-2*1 + 2*5) + 1 = -0.7776 + 2.88 + 1 = 3.1024 \end{cases}$$

Algorytm

```
var qbezierValue1 = function(points, t) {
      var ax = points[0].x;
      var ay = points[0].y;
      var bx = points[1].x;
      var by = points[1].y;
      var cx = points[2].x;
      var cy = points[2].y;
      var x1 = ax - 2 * bx + cx;
      var x2 = -2 * ax + 2 * bx;
      var y1 = ay - 2 * by + cy;
      var y2 = -2 * ay + 2 * by;
      var pt = t * t;
      var x = pt * x1 + t * x2 + ax;
```

```
        var y = pt * y1 + t * y2 + ay;
        return new Point(x, y);
};
```

Sprawdzenie algorytmu $B_{1,0}$ (Dodatek 7 Listing 119)

```
[2.1024, 3.1024]
```

Postać macierzowa

Równanie:

$$h(t) = t^2(a - 2b + c) + t(-2a + 2b) + a$$

możemy zapisać w postaci macierzowej:

$$P(t) = \begin{bmatrix} t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

Obliczenia

Spróbujmy wyznaczyć położenie punktu na krzywej dla $t = 0.36$, gdy

- punkt a na krzywej ma współrzędne $a(0,1)$
- punkt kontrolny krzywej b ma współrzędne $b(4,5)$
- punkt c na krzywej ma współrzędne $c(2,3)$

$$x(0.36) = \begin{bmatrix} 0.1296 & 0.36 & 1 \end{bmatrix} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 4 \\ 2 \end{bmatrix} = \begin{bmatrix} 0.1296 & 0.36 & 1 \end{bmatrix} \begin{bmatrix} -6 \\ 8 \\ 0 \end{bmatrix} = 2.1024$$

$$y(0.36) = \begin{bmatrix} 0.1296 & 0.36 & 1 \end{bmatrix} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 5 \\ 3 \end{bmatrix} = \begin{bmatrix} 0.1296 & 0.36 & 1 \end{bmatrix} \begin{bmatrix} -6 \\ 8 \\ 1 \end{bmatrix} = 3.1024$$

Algorytm

```
var qbezierValue2 = function(points, t) {
        var ax = points[0].x;
        var ay = points[0].y;
        var bx = points[1].x;
        var by = points[1].y;
        var cx = points[2].x;
        var cy = points[2].y;
        var vx = [ ax, bx, cx ];
        var vy = [ ay, by, cy ];
```

```
        var coeffs = [ 1, -2, 1, -2, 2, 0, 1, 0, 0 ];
        var ts = [ Math.pow(t, 2), t, 1 ];
        var mvx = new Matrix(vx, 1);
        var mvy = new Matrix(vy, 1);
        var mcoeffs = new Matrix(coeffs, 3);
        var mts = new Matrix(ts, 3);
        var x = (mts.multiply2(mcoeffs.multiply2(mvx))).getMN(0,0);
        var y = (mts.multiply2(mcoeffs).multiply2(mvy)).getMN(0, 0);
        return new Point(x,y);
};
```

Algorytm wygląda ładnie i działa efektywnie, ale nie jest tak efektywny jak mógłby być. Aby udoskonalić algorytm należy `Math.pow()` zastąpić przez mnożenia i zrezygnować z użycia obiektów klasy Matrix, które zawierają mnóstwo niepotrzebnych nam w tej chwili, a obciążających pamięć metod. Zamiast tego należy dokonać wstępnego przemnożenia położeń punktów przez macierz współczynników. Możemy też zrezygnować z tworzenia zmiennych `ax, ay`, etc, i pobierać dane bezpośrednio z tablicy. Nie warto jednak dokonywać optymalizacji, gdyż po dokonaniu wszelkich możliwych uproszczeń dojdziemy do algorytmu `qbezierValue1`.
Sprawdzenie algorytmu $B_{1,0}$ (Dodatek 7 Listing 120)

```
[2.1024, 3.1023999999999994]
```

Inna definicja

Krzywa Beziera stopnia n-tego może być zdefiniowana jako:

$$C(t) = \sum_{k=0}^{n} B_{n,k}(t) P_k$$

gdzie:
parametr $t \in [0,1]$
$B_{n,k}(t)$ - jest wielomianem Bernsteina n-tego stopnia
P_k - oznacza punkt kontrolny

Dla $n = 2$ mieliśmy wielomiany Bernsteina:

$$B_{2,0}(t) = (1-t)^2$$

$$B_{2,1}(t) = 2t(1-t)$$

$$B_{2,2}(t) == t^2$$

Po wstawieniu do wzoru otrzymamy:

$$C(t) = (1-t)^2 a + 2t(1-t)b + t^2 c$$

Współrzędne punktów w układzie kartezjańskim będą:

$$\begin{cases} x(t) = (1-t)^2 a_x + 2t(1-t)b_x + t^2 c_x \\ y(t) = (1-t)^2 a_y + 2t(1-t)b_y + t^2 c_y \end{cases}$$

Obliczenia

Spróbujmy wyznaczyć położenie punktu na krzywej dla $t = 0.36$, gdy

- punkt a na krzywej ma współrzędne $a(0,1)$
- punkt kontrolny krzywej b ma współrzędne $b(4,5)$
- punkt c na krzywej ma współrzędne $c(2,3)$

$$x(0.36) = (1-0.36)^2 * 0 + 2 * 0.36(1-0.36) * 4 + (0.36)^2 * 2 =$$
$$0+1.8432+0.2592=2.1024$$
$$y(0.36) = (1-0.36)^2 * 1 + 2 * 0.36(1-0.36) * 5 + (0.36)^2 * 3 =$$
$$0.4096+2.304+0.3888=3.1024$$

Algorytm

```
var qbezierValue3 = function(points, t) {
      var ax = points[0].x;
      var ay = points[0].y;
      var bx = points[1].x;
      var by = points[1].y;
      var cx = points[2].x;
      var cy = points[2].y;
      var pt1 = 1.0 - t;
      var pt2 = 2 * t * pt1;
      var pt3 = pt1 * pt1;
      var pt4 = t * t;
      var x = pt3 * ax + pt2 * bx + pt4 * cx;
      var y = pt3 * ay + pt2 * by + pt4 * cy;
      return new Point(x, y);
};
```

Ten algorytm wykonuje znacznie więcej operacji na parametrze t oraz więcej mnożeń.

Sprawdzenie algorytmu $B_{1,0}$ (Dodatek 7 Listing 121)

```
[2.1024, 3.1024]
```

Krzywa Beziera 3-go stopnia

Jeżeli dodamy jeden punkt kontrolny to nasza krzywa Béziera stanie się krzywą trzeciego stopnia. Utworzenie jej jest równie proste jak krzywej drugiego stopnia:

$$h(t) = t * g(t) + (1-t) f(t)$$

Za $f(t)$ i $g(t)$ podstawiamy nasze krzywe drugiego stopnia

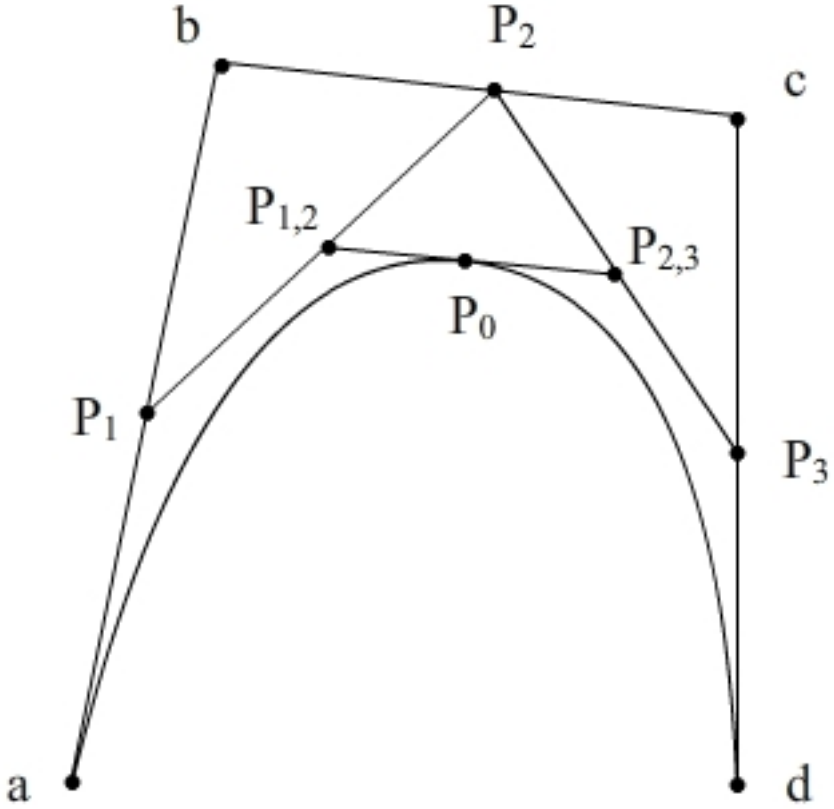
$$f(t) = t^2 (a - 2b + c + t(-2a + 2b) + a$$

$$g(t) = t^2 (b - 2c + d) + t(-2b + 2c) + b$$

i po podstawieniu i uproszczeniu otrzymamy:

$$h(t) = t^3(-a + 3b - 3c + d) + t^2(3a - 6b + 3c) + t(-3a + 3b) + a$$

gdzie:
a, d, P_0 - to punkty na krzywej
b, c - to punkty kontrolne krzywej



Obliczenia

Spróbujmy wyznaczyć położenie punktu na krzywej dla $t = 0.3$, gdy

- punkt a na krzywej ma współrzędne $a(0,1)$
- punkt kontrolny b krzywej ma współrzędne $b(2,3)$
- punkt kontrolny c krzywej ma współrzędne $c(6,7)$
- punkt d na krzywej ma współrzędne $d(4,5)$

$$h(t) = t^3(-a + 3b - 3c + d) + t^2(3a - 6b + 3c) + t(-3a + 3b) + a$$

$$x(0.3) = 0.3^3(-0 + 3*2 - 3*6 + 4) + 0.3^2(3*0 - 6*2 + 3*6) + 0.3(-3*0 + 3*2) + 0 =$$
-0.216+0.54+1.8+0=2.124

$$y(0.3) = 0.3^3(-1 + 3*3 - 3*7 + 5) + 0.3^2(3*1 - 6*3 + 3*7) + 0.3(-3*1 + 3*3) + 0 =$$
-0.216+0.54+1.8+0=3.124

Algorytm

```
var cbezierValue1 = function(points, t) {
      var ax = points[0].x;
```

```
        var ay = points[0].y;
        var bx = points[1].x;
        var by = points[1].y;
        var cx = points[2].x;
        var cy = points[2].y;
        var dx = points[3].x;
        var dy = points[3].y;
        var x1 = -ax + 3 * bx - 3 * cx + dx;
        var y1 = -ay + 3 * by - 3 * cy + dy;
        var x2 = 3 * ax - 6 * bx + 3 * cx;
        var y2 = 3 * ay - 6 * by + 3 * cy;
        var x3 = -3 * ax + 3 * bx;
        var y3 = -3 * ay + 3 * by;
        var pt2 = t * t;
        var pt3 = pt2 * t;
        var x = pt3 * x1 + pt2 * x2 + t * x3 + ax;
        var y = pt3 * y1 + pt2 * y2 + t * y3 + ay;
        return new Point(x, y);
};
```

Sprawdzenie algorytmu $B_{1,0}$ (Dodatek 7 Listing 122)

```
[2.1239999999999997, 3.1239999999999997]
```

Postać macierzowa

Równanie:

$$h(t) = t^3(-a+3b-3c+d) + t^2(3a-6b+3c) + t(-3a+3b) + a$$

możemy zapisać w postaci macierzowej:

$$P(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

Obliczenia

Spróbujmy wyznaczyć położenie punktu na krzywej dla $t = 0.3$, gdy

- punkt a na krzywej ma współrzędne $a(0,1)$
- punkt kontrolny b krzywej ma współrzędne $b(2,3)$
- punkt kontrolny c krzywej ma współrzędne $c(6,7)$
- punkt d na krzywej ma współrzędne $d(4,5)$

$$x(0.3) = \begin{bmatrix} 0.027 & 0.09 & 0.3 & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \\ 6 \\ 4 \end{bmatrix} =$$

$$=\begin{bmatrix}0.027 & 0.09 & 0.3 & 1\end{bmatrix}\begin{bmatrix}-8\\6\\6\\0\end{bmatrix}=2.124$$

$$y(0.3)=\begin{bmatrix}0.027 & 0.09 & 0.3 & 1\end{bmatrix}\begin{bmatrix}-1 & 3 & -3 & 1\\3 & -6 & 3 & 0\\-3 & 3 & 0 & 0\\1 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}1\\3\\7\\5\end{bmatrix}=$$

$$=\begin{bmatrix}0.027 & 0.09 & 0.3 & 1\end{bmatrix}\begin{bmatrix}-8\\6\\6\\1\end{bmatrix}=3.124$$

Algorytm

```
var cbezierValue2 = function(points, t) {
      var ax = points[0].x;
      var ay = points[0].y;
      var bx = points[1].x;
      var by = points[1].y;
      var cx = points[2].x;
      var cy = points[2].y;
      var dx = points[3].x;
      var dy = points[3].y;
      var vx = [ ax, bx, cx,dx ];
      var vy = [ ay, by, cy,dy ];
      var coeffs = [-1,3,-3,1,3,-6,3,0,-3,3,0,0,1,0,0,0];
      var ts = [ Math.pow(t, 3), Math.pow(t, 2), t, 1 ];
      var mvx = new Matrix(vx, 1);
      var mvy = new Matrix(vy, 1);
      var mcoeffs = new Matrix(coeffs, 4);
      var mts = new Matrix(ts, 4);
      var x = (mts.multiply2(mcoeffs.multiply2(mvx))).getMN(0,0);
      var y = (mts.multiply2(mcoeffs).multiply2(mvy)).getMN(0, 0);
      return new Point(x,y);
};
```

Sprawdzenie algorytmu $B_{1,0}$ (Dodatek 7 Listing 123)

```
[2.1239999999999997, 3.1239999999999997]
```

Inna definicja

Jak już mówiliśmy krzywa Beziera stopnia n-tego może być zdefiniowana jako:

$$C(t)=\sum_{k=0}^{n}B_{n,k}(t)P_k$$

gdzie:
parametr $t\in[0,1]$
$B_{n,k}(t)$ - jest wielomianem Bernsteina n-tego stopnia

P_k - oznacza punkt kontrolny

Dla $n = 3$ mieliśmy wielomiany Bernsteina:

$B_{3,0}(t) = (1-t)^3$

$B_{3,1}(t) = 3t(1-t)^2$

$B_{3,2}(t) = 3t^2(1-t)$

$B_{3,3}(t) == t^3$

Po wstawieniu do wzoru otrzymamy:

$$C(t) = (1-t)^3 a + 3t(1-t)^2 b + 3t^2(1-t)c + t^3 d$$

Współrzędne punktów w układzie kartezjańskim będą:

$$\begin{cases} x(t) = (1-t)^3 a_x + 3t(1-t)^2 b_x + 3t^2(1-t)c_x + t^3 d_x \\ y(x) = (1-t)^3 a_y + 3t(1-t)^2 b_y + 3t^2(1-t)c_y + t^3 d_y \end{cases}$$

Obliczenia

Spróbujmy wyznaczyć położenie punktu na krzywej dla $t = 0.3$, gdy

- punkt a na krzywej ma współrzędne $a(0,1)$
- punkt kontrolny b krzywej ma współrzędne $b(2,3)$
- punkt kontrolny c krzywej ma współrzędne $c(6,7)$
- punkt d na krzywej ma współrzędne $d(4,5)$

$$x(t) = (1-0.3)^3 * 0 + 3 * 0.3 * (1-0.3)^2 * 2 + 3 * 0.3^2 * (1-0.3) * 6 + 0.3^3 * 4 =$$
$$0+0.882+1.134+0.108 = 2.124$$

$$x(t) = (1-0.3)^3 * 1 + 3 * 0.3 * (1-0.3)^2 * 3 + 3 * 0.3^2 * (1-0.3) * 7 + 0.3^3 * 5$$
$$=0.343+1.323+1.323+0.135 = 3.124$$

Algorytm

```
var cbezierValue3 = function(points, t) {
      var ax = points[0].x;
      var ay = points[0].y;
      var bx = points[1].x;
      var by = points[1].y;
      var cx = points[2].x;
      var cy = points[2].y;
      var dx = points[3].x;
      var dy = points[3].y;
      var pt1 = 1.0 - t;
      var pt2 = pt1 * pt1;
      var pt3 = pt2 * pt1;
      var t2 = t * t;
      var t3 = t2 * t;
```

```
        var x = pt3 * ax + 3 * t * pt2 * bx + 3 * t2 * pt1 * cx + t3 * dx;
        var y = pt3 * ay + 3 * t * pt2 * by + 3 * t2 * pt1 * cy + t3 * dy;
        return new Point(x, y);
};
```

Sprawdzenie algorytmu $B_{1,0}$ (Dodatek 7 Listing 124)

```
[2.1239999999999997, 3.1239999999999997]
```

Wykresy krzywych Beziera 2-go i 3-go stopnia

Krzywa 2-go stopnia

Algorytm

Krzywa rysowana przy użyciu metody JavaScript:

```
var drawQBezier = function(points, strokeStyle, visible) {
        ctx.save();
        ctx.beginPath();
        ctx.strokeStyle = strokeStyle;
        ctx.moveTo(points[0].x, points[0].y);
        ctx.quadraticCurveTo(points[1].x, points[1].y, points[2].x, points[2].y);
        ctx.stroke();
        ctx.restore();
        if (visible) {
                drawPoints(points, strokeStyle);
        }
};

var drawPoints = function(points, style) {
        ctx.save();
        ctx.beginPath();
        ctx.fillStyle = style;
        for ( var i = 0; i < points.length; i++) {
                ctx.arc(points[i].x, points[i].y, 3, 0, 2 * Math.PI, true);
        }
        ctx.fill();
        ctx.restore();
};
```

Krzywa rysowana przy użyciu naszej metody:

```
var tValue = function(points) {
        var maxx = -1000000;
        var maxy = -1000000;
        var minx = 1000000;
        var miny = 1000000;
        for ( var i = 0; i < points.length; i++) {
                var val = points[i];
                maxx = Math.max(val.x, maxx);
                maxy = Math.max(val.y, maxy);
                minx = Math.min(val.x, minx);
                miny = Math.min(val.y, miny);
        }
        return Math.max(Math.abs(maxx) - Math.abs(minx), Math.abs(maxy)
```

```
                    - Math.abs(miny));
};

var drawMyQBezier1 = function(points, style, visible) {
      var x1 = points[0].x - 2 * points[1].x + points[2].x;
      var x2 = -2 * points[0].x + 2 * points[1].x;
      var y1 = points[0].y - 2 * points[1].y + points[2].y;
      var y2 = -2 * points[0].y + 2 * points[1].y;
      var extr = tValue(points);
      var te = 1.0 / extr;
      var img1 = ctx.getImageData(0, 0, width, height);
      var idata = img1.data;
      var t = 0;
      for ( var i = 0; i < extr; i++) {
            t = i * te;
            var pt = t * t;
            var ptx = pt * x1;
            var pty = pt * y1;
            var x = ptx + t * x2 + points[0].x;
            var y = pty + t * y2 + points[0].y;
            var j = 4 * index(cutDecimal(y), cutDecimal(x), width);
            idata[j] = 0;
            idata[j + 1] = 0;
            idata[j + 2] = 255;
            idata[j + 3] = 255;
      }
      ctx.putImageData(img1, 0, 0);
      if (visible) {
            drawPoints(points, style);
      }
};
```

Krzywa rysowana inną metodą:

```
var drawMyQBezier2 = function(points, style, visible) {
      var x1 = points[0].x - 2 * points[1].x + points[2].x;
      var x2 = -2 * points[0].x + 2 * points[1].x;
      var y1 = points[0].y - 2 * points[1].y + points[2].y;
      var y2 = -2 * points[0].y + 2 * points[1].y;
      var extr = tValue(points);
      var te = 1.0 / extr;
      var t = 0;
      ctx.save();
      ctx.beginPath();
      ctx.strokeStyle = style;
      ctx.moveTo(points[0].x, points[0].y);
      for ( var i = 0; i < extr; i++) {
            t = i * te;
            var pt = t * t;
            var ptx = pt * x1;
            var pty = pt * y1;
            var x = ptx + t * x2 + points[0].x;
            var y = pty + t * y2 + points[0].y;
            ctx.lineTo(x, y);
      }
      ctx.stroke();
      ctx.restore();
      if (visible) {
            drawPoints(points, style);
```

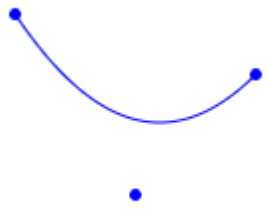
```
        }
};
```

Wykres

Krzywa rysowana przy użyciu metody JavaScript:

(Dodatek 7 Listing125)

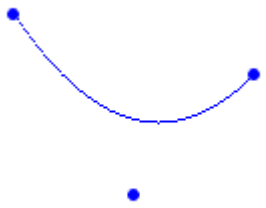
```
var points = [new Point(60,60), new Point(120,150), new Point(180,90)];
             drawQBezier(points, "blue", true);
```



Krzywa rysowana przy użyciu naszej metody:

(Dodatek 7 Listing 127)

```
var points = [new Point(60,60), new Point(120,150), new Point(180,90)];
             drawMyQBezier1(points, "blue", true);
```

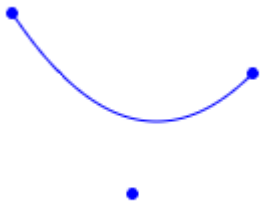


Jak widzimy krzywa jest identyczna, brakuje tylko algorytmu wygładzania krzywej.

Krzywa rysowana innym mechanizmem rysowania:

(Dodatek 7 Listing 128)

```
var points = [new Point(60,60), new Point(120,150), new Point(180,90)];
             drawMyQBezier2(points, "blue", true);
```



Tym razem krzywa jest identyczna jak ta rysowana przy użyciu rodzimych algorytmów JavaScript.

Krzywa 3-go stopnia

Algorytm

Rysowanie przy użyciu metod rodzimych JavaScript:

```
var drawCBezier = function(points, strokeStyle, visible) {
	ctx.save();
	ctx.beginPath();
	ctx.strokeStyle = strokeStyle;
	ctx.moveTo(points[0].x, points[0].y);
	ctx.bezierCurveTo(points[1].x, points[1].y, points[2].x, points[2].y,
			points[3].x, points[3].y);
	ctx.stroke();
	ctx.restore();
	if (visible) {
		drawPoints(points, strokeStyle);
	}
};
```

Rysowanie przy użyciu naszej metody:

```
var drawMyCBezier1 = function(points, style, visible) {
	var x1 = -points[0].x + 3 * points[1].x - 3 * points[2].x + points[3].x;
	var y1 = -points[0].y + 3 * points[1].y - 3 * points[2].y + points[3].y;
	var x2 = 3 * points[0].x - 6 * points[1].x + 3 * points[2].x;
	var y2 = 3 * points[0].y - 6 * points[1].y + 3 * points[2].y;
	var x3 = -3 * points[0].x + 3 * points[1].x;
	var y3 = -3 * points[0].y + 3 * points[1].y;
	var extr = tValue(points);
	var te = 1.0/extr;
	var img1 = ctx.getImageData(0, 0, width, height);
	var idata = img1.data;
	var t=0;
	for(var i=0; i<extr;i++){
		t = i*te;
		var pt2 = t * t;
		var pt3 = pt2 * t;
		var x = pt3 * x1 + pt2 * x2 + t * x3 + points[0].x;
		var y = pt3 * y1 + pt2 * y2 + t * y3 + points[0].y;
		var j = 4 * index(cutDecimal(y), cutDecimal(x), width);
		idata[j] = 0;
		idata[j + 1] = 0;
		idata[j + 2] = 255;
		idata[j + 3] = 255;
	}
	ctx.putImageData(img1,0,0);
	if(visible){
		drawPoints(points, style);
	}
};
```

Krzywa rysowana inną metodą:

```
var drawMyCBezier2 = function(points, style, visible) {
```

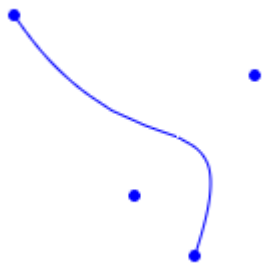
```
        var x1 = -points[0].x + 3 * points[1].x - 3 * points[2].x + points[3].x;
        var y1 = -points[0].y + 3 * points[1].y - 3 * points[2].y + points[3].y;
        var x2 = 3 * points[0].x - 6 * points[1].x + 3 * points[2].x;
        var y2 = 3 * points[0].y - 6 * points[1].y + 3 * points[2].y;
        var x3 = -3 * points[0].x + 3 * points[1].x;
        var y3 = -3 * points[0].y + 3 * points[1].y;
        var extr = tValue(points);
        var te = 1.0/extr;
        var t=0;
        ctx.save();
        ctx.beginPath();
        ctx.strokeStyle=style;
        ctx.moveTo(points[0].x, points[0].y);
        for(var i=0; i<extr;i++){
                t = i*te;
                var pt2 = t * t;
                var pt3 = pt2 * t;
                var x = pt3 * x1 + pt2 * x2 + t * x3 + points[0].x;
                var y = pt3 * y1 + pt2 * y2 + t * y3 + points[0].y;
                ctx.lineTo(x,y);
        }
        ctx.stroke();
        ctx.restore();
        if(visible){
                drawPoints(points, style);
        }
};
```

Wykres

Przy użyciu metody rodzimej JavaScript:

(Dodatek 7 Listing 126)

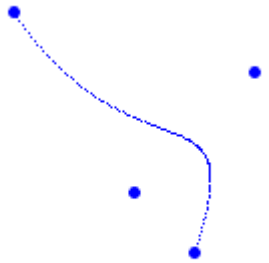
```
var points = [new Point(60,60), new Point(120,150), new Point(180,90), new
        Point(150, 180)];
drawCBezier(points, "blue", true);
```



Przy użyciu naszej metody:

(Dodatek 7 Listing 129)

```
var points = [new Point(60,60), new Point(120,150), new Point(180,90), new
        Point(150, 180)];
        drawMyCBezier1(points, "blue", true);
```

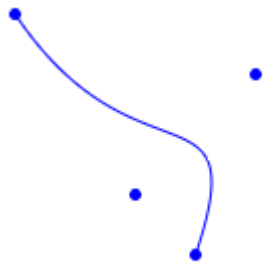


Jak widzimy krzywa jest identyczna, brakuje tylko algorytmu wygładzania krzywej.

Krzywa rysowana innym mechanizmem rysowania:

(Dodatek 7 Listing 130)

```
var points = [new Point(60,60), new Point(120,150), new Point(180,90), new
      Point(150, 180)];
      drawMyCBezier2(points, "blue", true);
```



Tym razem krzywa jest identyczna jak ta rysowana przy użyciu rodzimych algorytmów JavaScript.

Krzywe Beziera wyższych stopni

Załóżmy, że chcemy użyć krzywej Beziera 12-go stopnia. Ustalanie parametrów dla 13 punktów byłoby wyjątkowo uciążliwe i konia (trojańskiego) z rzędem temu, kto chciałby się tego podjąć, i zrobiłby to nie myląc się, bo znaczyłoby to, że chyba ma konszachty z 'ciemną stroną' matematyki :-).
Jak już mówiliśmy krzywa Beziera stopnia n-tego może być zdefiniowana jako:

$$C(t) = \sum_{k=0}^{n} B_{n,k}(t) P_k$$

gdzie:

$$B_{n,k}(t) = \binom{n}{k} t^k (1-t)^{n-k}$$

Przyjmijmy, że:
n = 12, czyli nasza krzywa jest 12 stopnia
k =13, gdyż jest liczone od 0 do n włącznie
tablica `points[a,b,c,d,e,f,g,h,i,j,k,l,m]` zawiera 13 punktów.

Musimy obliczyć kolejno dla współrzędnych x (albo y) poszczególnych punktów

$$B_{12,0}(t) = \binom{12}{0} t^0 (1-t)^{12} * a_x$$

$$B_{12,1}(t) = \binom{12}{1} t^1 (1-t)^{11} * b_x$$

....

$$B_{12,11}(t) = \binom{12}{11} t^{11} (1-t)^1 * l_x$$

$$B_{12,12}(t) = \binom{12}{12} t^{12} (1-t)^0 * m_x$$

Obliczamy sumę powyższych wartości od $B_{12,0}(t)$ do $B_{12,12}(t)$

Współrzędna x punktu $P(x, y)$ na krzywej dla danej wartości parametru t jest sumą 13 wyliczonych powyżej wartości.

Teraz należy analogicznie obliczyć współrzędną y tego punktu i powtórzyć wyliczenia dla każdej wartości t.

Dla krzywej stopnia n-tego możemy użyć poniższej funkcji:

Algorytm

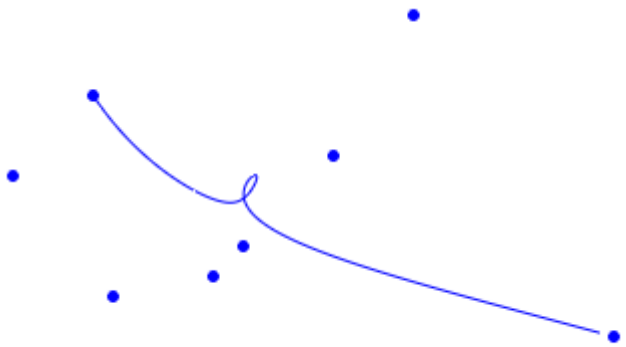
```
var drawNBezier = function(points, style, visible) {
      var n = points.length - 1;
      var extr = tValue(points);
      var te = 1.0 / extr;
      var t = 0;
      ctx.save();
      ctx.beginPath();
      ctx.strokeStyle = style;
      ctx.moveTo(points[0].x, points[0].y);
      for ( var i = 0; i < extr; i++) {
            t = i * te;
            var x = 0;
            var y = 0;
            for ( var k = 0; k < points.length; k++) {
                  var np = npok(n, k) * Math.pow(t, k) * Math.pow(1 - t, n - k);
                  x += np * points[k].x;
                  y += np * points[k].y;
            }
            ctx.lineTo(x, y);
      }
      ctx.stroke();
      ctx.restore();
      if (visible) {
            drawPoints(points, style);
      }
      ctx.restore();

};
```

Wykres

Krzywa Beziera 7-go stopnia:

(Dodatek 7 Listing 133)

```
var points = [ new Point(60, 60), new Point(120, 150), new Point(180, 90), new
Point(70, 160), new Point(220, 20), new Point(135, 135), new Point(20, 100), new
Point(320, 180)];
```



Podwyższanie stopnia krzywej

Krzywa Beziera stopnia n może być zastąpiona krzywą Beziera stopnia $n+1$ z zachowaniem kształtu krzywej.
Jeżeli np. krzywa n-go stopnia jest reprezentowana przez punkty $P_0, P_1 ... P_{n-1}, P_n$ to krzywa stopnia $n+1$ będzie reprezentowana przez punkty $Q_0, Q_1 ... Q_n, Q_{n+1}$, gdzie:

$$Q_0 = P_0$$

$$Q_k = a_k P_{k-1} + (1 - a_k) P_k \text{, gdzie } a_k = \frac{k}{n+1}$$

$$Q_{n+1} = P_n$$

Algorytm

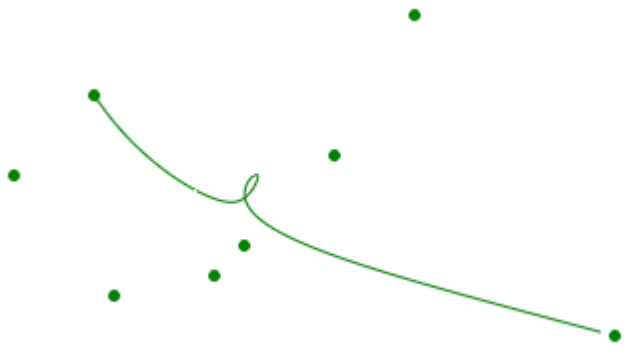
Do podwyższenia stopnia krzywej można użyć następującego algorytmu:

```
var bezierDegreeUp = function(points){
	var len = points.length;//4
	var arr = new Array();
	arr[0]=new Point(points[0].x, points[0].y);
	for(var i=1;i<len;i++){
		var a= i/len;
		var b = 1.0-a;
		arr.push(new Point(a*points[i-1].x+b*points[i].x, a*points[i-
1].y+b*points[i].y));
	}
	arr.push(new Point(points[len-1].x, points[len-1].y));
	return arr;
};
```

Wykres

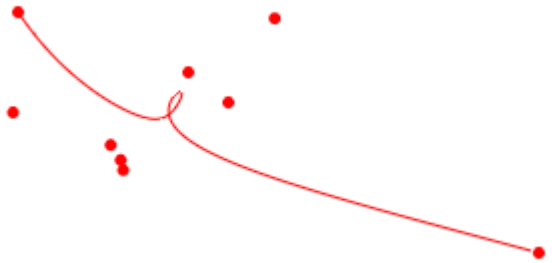
(Dodatek 7 Listing 149)

Krzywa 7-stopnia:



(Dodatek 7 Listing 150)

Krzywa 8-stopnia:



Właściwości krzywych Beziera

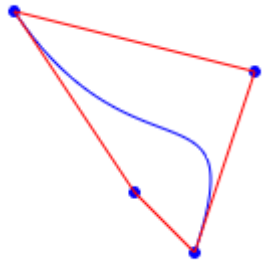
Najważniejsze właściwości krzywych Beziera:
Dla `var points` zawierającej tablicę obiektów `Point(x,y)`:

Właściwość 1

Punktem początkowym krzywej jest `points[0]`, a punktem końcowym krzywej `points[points.length-1]`. Pozostałe punkty w tablicy to punkty kontrolne krzywej, które na ogół nie leżą na krzywej.

Właściwość 2

Krzywa Beziera zawiera się w wieloboku, którego wierzchołkami są punkty umieszczone w tablicy `points`.



Właściwość 3

Styczna do krzywej w punkcie `points[0]` jest równoległa do odcinka `points[0]-points[1]`, a styczna do krzywej w punkcie `points[points.length-1]` jest równoległa do odcinka `points[points.length-1]-points[points.length-2]`.

Właściwość 4

Jeżeli `points[0] == points[length-1]` to krzywa jest zamknięta, w przeciwnym przypadku krzywa jest otwarta.

(Dodatek 7 Listing 131)

```
var points = [ new Point(60, 60), new Point(120, 150),
                          new Point(180, 90), new Point(60, 60) ];
             drawMyCBezier2(points, "blue", true);
```

Właściwość 5

Jeżeli odcinek points[0]-points[1] jest równoległy do odcinka points[length-1]-points[length-2] to połączenie w points[0]=points[length-1] jest płynne.

Nie można tego pokazać na krzywej 3-go stopnia, gdyż musielibyśmy uczynić z niej prostą.

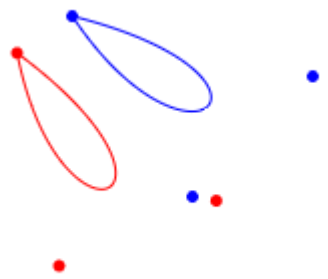
Właściwość 6

Przekształcenia (transformacje) krzywej wykonujemy poddając transformacjom punkty w tablicy `points`.

Użyjemy metody bezpośrednio przeliczającej punkty w tablicy na nowe pozycje i obracającej krzywą o podany kąt względem punktu P(0,0) osi współrzędnych. W przypadku intensywnego użycia niektórych transformacji warto było by przygotować stosowne metody dla wszystkich potrzebnych transformacji. Pozostawiamy to zadanie dla Czytelnika.:

```
var rotateBezier = function(points, angle) {
      var points1 = new Array();
      for ( var i = 0; i < points.length; i++) {
            points1.push(new Point(points[i].x * Math.cos(angle) + points[i].y
                        * Math.sin(angle), -points[i].x * Math.sin(angle) +
points[i].y
                        * Math.cos(angle)));
      }
      return points1;
};
```

(Dodatek 7 Listing 132)

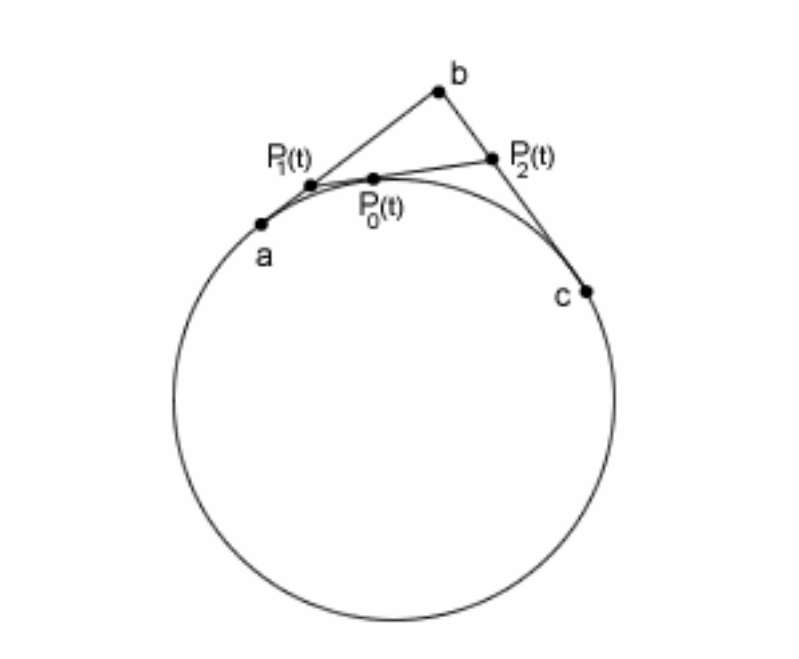


Właściwość 7

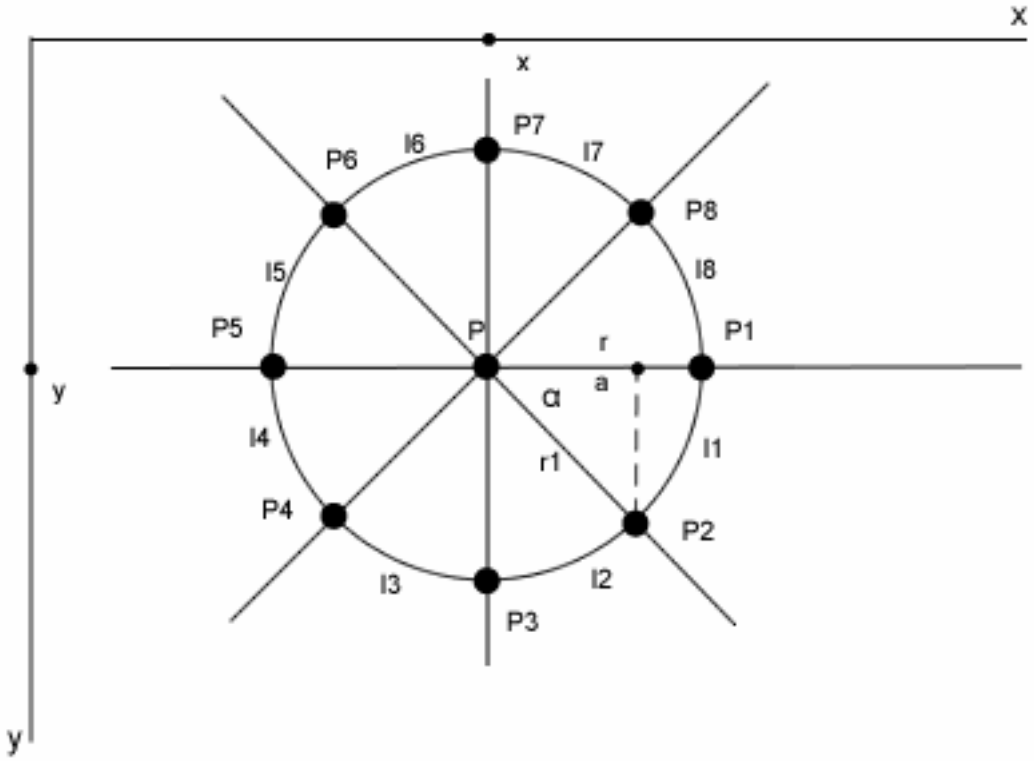
Przy użyciu jednej krzywej Beziera dowolnego stopnia nie można uzyskać takich figur stożkowych takich jak koło, elipsa, hiperbola albo parabola. Można natomiast uzyskać taką figurę składając ją z kilku krzywych Beziera.

Rysowanie koła

Spójrzmy na poniższy rysunek:



Widzimy, że punkty a, b i c wyznaczają krzywą, która w tym przypadku jest łukiem okręgu. Spróbujmy zatem wykreślić koło. Do tego aby wykreślić poprawne i gładkie koło musimy użyć co najmniej 8 krzywych Beziera:



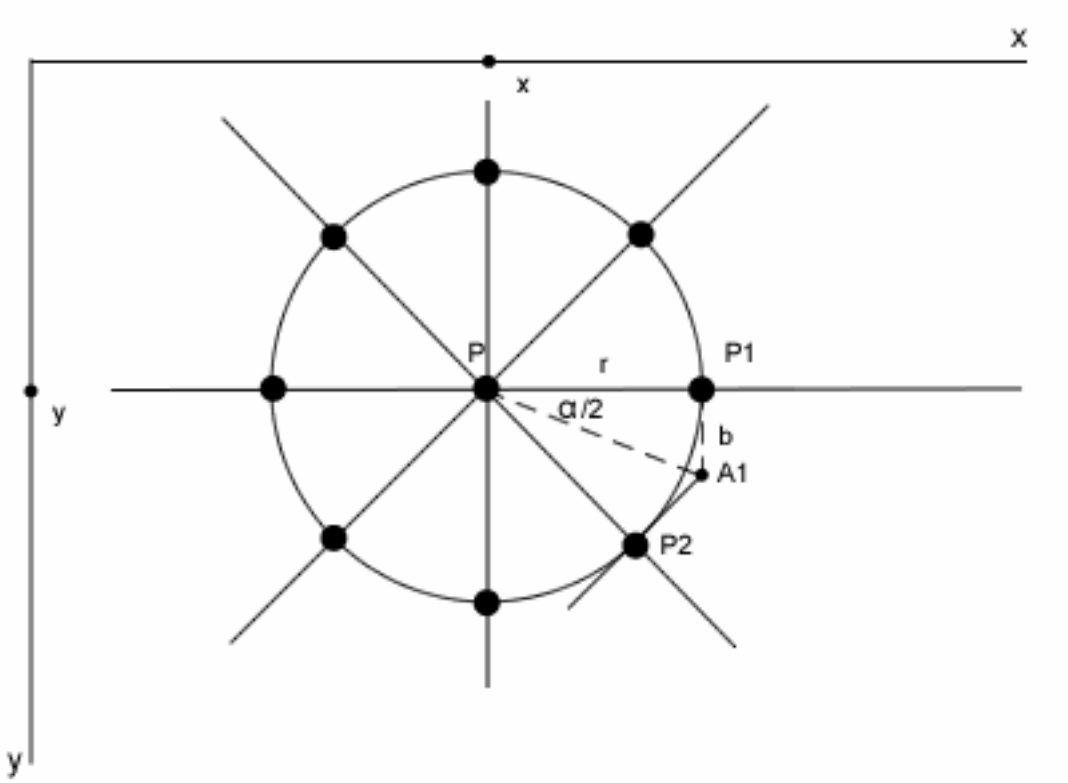
P - oznacza punkt środkowy okręgu.
r - oznacza promień okręgu.
r_1 - oznacza promień okręgu.
a - oznacza odcinek między punktem środkowym P, a rzutem punktu P_2 na promień okręgu.
P_1 do P_8 oznaczają kolejne punkty na okręgu.
l_1 do l_8 oznaczają kolejne łuki okręgu.

Położenie punktu P jest wyznaczone przez liczby (x , y) położone na osiach współrzędnych.
My oznaczymy je jako $P(startX, startY)$
Punkt P_1 położony jest w odległości r od punktu P, a zatem jego współrzędne wynoszą $P_1(startX + r, startY)$, gdyż zmienna y się nie zmienia.
Położenie punktu P_3: $P_3(startX, startY + r)$.
Położenie punktu P_5: $P_5(startX - r, startY)$.
Położenie punktu P_7: $P_7(startX, startY - r)$.
Wyznaczmy położenie punktów P_2, P_4, P_6, P_8.
Z definicji funkcji trygonometrycznych wynika, że $\frac{a}{r_1} = \cos\alpha$. Ponieważ $r_1 = r$, $\frac{a}{r} = \cos\alpha$.
Po przekształceniu: $a = r\cos\alpha$.
W związku z tym:
Położenie punktu P_2: $P_2(startX + r\cos\alpha, startY + r\cos\alpha)$.
Kąt α = 360°/8 = 45°. W radianach to odpowiednio $2\pi/8 = \pi/4$.
Przez analogię współrzędne punktów P_4, P_6, P_8 będą wynosiły odpowiednio:
Położenie punktu P_4: $P_4(startX - r\cos\alpha, startY + r\cos\alpha)$.
Położenie punku P_6: $P_6(startX - r\cos\alpha, startY - r\cos\alpha)$.
Położenie punktu P_8: $P_8(startX + r\cos\alpha, startY - r\cos\alpha)$.

Ujmijmy te dane w tabeli.

Łuk	Punkt	Współrzędne punktu kontrolnego		Współrzędne punktu na okręgu	
		A_x	A_y	P_x	P_y
	P			startX	startY
	P_1			startX + r	startY
l_1	P_2			startX + rcosα	startY + rcosα
l_2	P_3			startX	startY + r
l_3	P_4			startX – rcosα	startY + rcosα
l_4	P_5			startX – r	startY
l_5	P_6			startX – rcosα	startY – rcosα
l	P_7			startX	startY – r
l_7	P_8			startX + rcosα	startY – rcosα
l_8	P_1			startX + r	startY

Wykonaliśmy połowę naszej pracy.
Określimy teraz położenie punktów kontrolnych określających krzywiznę łuku okręgu.



Punkt kontrolny A_1 dla łuku rysowanego od punktu P_1 do punktu P_2 jest położony na przecięciu dwusiecznej (narysowanej linią przerywaną), dzielącej kąt α na połowę i prostej b wychodzącej pod kątem prostym z punktu P_1.

$$\frac{b}{r} = tg\left(\frac{\alpha}{2}\right),$$

czyli

$$b = rtg\left(\frac{\alpha}{2}\right)$$

Współrzędne naszego punktu kontrolnego to:

$$A_1(x, y) = A_1(startX + r, startY + rtg\left(\frac{\alpha}{2}\right)).$$

Obliczenia dla pozostałych punktów pozostawiam do wykonania Czytelnikowi. Dane umieścimy w naszej tabeli.

Łuk	Punkt	Współrzędne punktu kontrolnego		Współrzędne punktu na okręgu	
		A_x	A_y	P_x	P_y
	P			startX	startY
	P_1			startX + r	startY
l_1	P_2	startX + r	startY + rtg(α/2)	startX + rcosα	startY + rcosα
l_2	P_3	startX + rtg(α/2)	startY + r	startX	startY + r
l_3	P_4	startX - rtg(α/2)	startY + r	startX – rcosα	startY + rcosα
l_4	P_5	startX – r	startY + rtg(α/2)	startX – r	startY
l_5	P_6	startX – r	startY - rtg(α/2)	startX – rcosα	startY – rcosα

l_6	P_7	startX - rtg(α/2)	startY – r	startX	startY - r
l_7	P_8	startX + rtg(α/2)	startY – r	startX + rcosα	startY – rcosα
l_8	P_1	startX + r	startY - rtg(α/2)	startX + r	startY

Zmieńmy nieco dane w naszej tabeli.

Łuk	instrukcja	Współrzędne punktu kontrolnego		Współrzędne punktu na okręgu	
		cpx	xcpy	x	y
				startX	startY
	moveTo			startX + r	startY
l1	quadraticCurveTo	startX + r	starty + rtg(α/2)	startX + rcosα	startY + rcosα
l2	quadraticCurveTo	startX + rtg(α/2)	startY + r	startX	startY + r
l3	quadraticCurveTo	startX - rtg(α/2)	starty + r	startX – rcosα	startY + rcosα
l4	quadraticCurveTo	startX – r	starty + rtg(α/2)	startX - r	startY
l5	quadraticCurveTo	startX – r	starty – rtg(α/2)	startX – rcosα	startY – rcosα
l6	quadraticCurveTo	startX - rtg(α/2)	startY – r	startX	startY - r
l7	quadraticCurveTo	startX + rtg(α/2)	startY – r	startX + rcosα	startY – rcosα
l8	quadraticCurveTo	startX + r	startY – rtg(α/2)	startX + r	startY

I oto mamy funkcję:

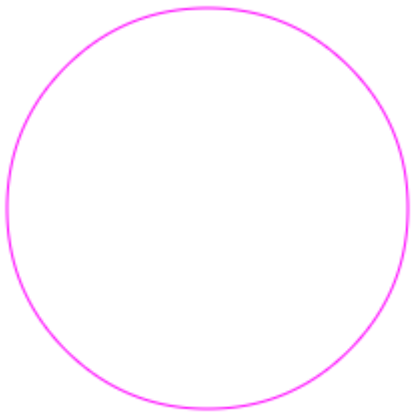
```
var drawCircle = function(startX, startY, r, style) {
        ctx.save();
        ctx.beginPath();
        ctx.strokeStyle = style;
        ctx.moveTo(startX + r, startY);
        ctx.quadraticCurveTo(startX + r, startY + r * Math.tan(Math.PI / 8), startX
                + r * Math.cos(Math.PI / 4), startY + r * Math.cos(Math.PI / 4));
        ctx.quadraticCurveTo(startX + r * Math.tan(Math.PI / 8), startY + r,
                startX, startY + r);
        ctx.quadraticCurveTo(startX - r * Math.tan(Math.PI / 8), startY + r, startX
                - r * Math.cos(Math.PI / 4), startY + r * Math.cos(Math.PI / 4));
        ctx.quadraticCurveTo(startX - r, startY + r * Math.tan(Math.PI / 8), startX
                - r, startY);
        ctx.quadraticCurveTo(startX - r, startY - r * Math.tan(Math.PI / 8), startX
                - r * Math.cos(Math.PI / 4), startY - r * Math.cos(Math.PI / 4));
        ctx.quadraticCurveTo(startX - r * Math.tan(Math.PI / 8), startY - r,
                startX, startY - r);
        ctx.quadraticCurveTo(startX + r * Math.tan(Math.PI / 8), startY - r, startX
                + r * Math.cos(Math.PI / 4), startY - r * Math.cos(Math.PI / 4));
        ctx.quadraticCurveTo(startX + r, startY - r * Math.tan(Math.PI / 8), startX
                + r, startY);
        ctx.stroke();
        ctx.restore();
};
```

Kąt α zastąpimy przez `Math.PI/4`, a kąt $\alpha/2$ przez `Math.PI/8`, funkcję *tg* przez `Math.tan()`, a funkcję cos przez `Math.cos()`. Przeliczenie kątów w stopniach na radiany jest proste. 360° = 2Π radianów. 45° = 360°/8, czyli = 2Π/8 = Π/4.

A oto nasza poprawiona funkcja:

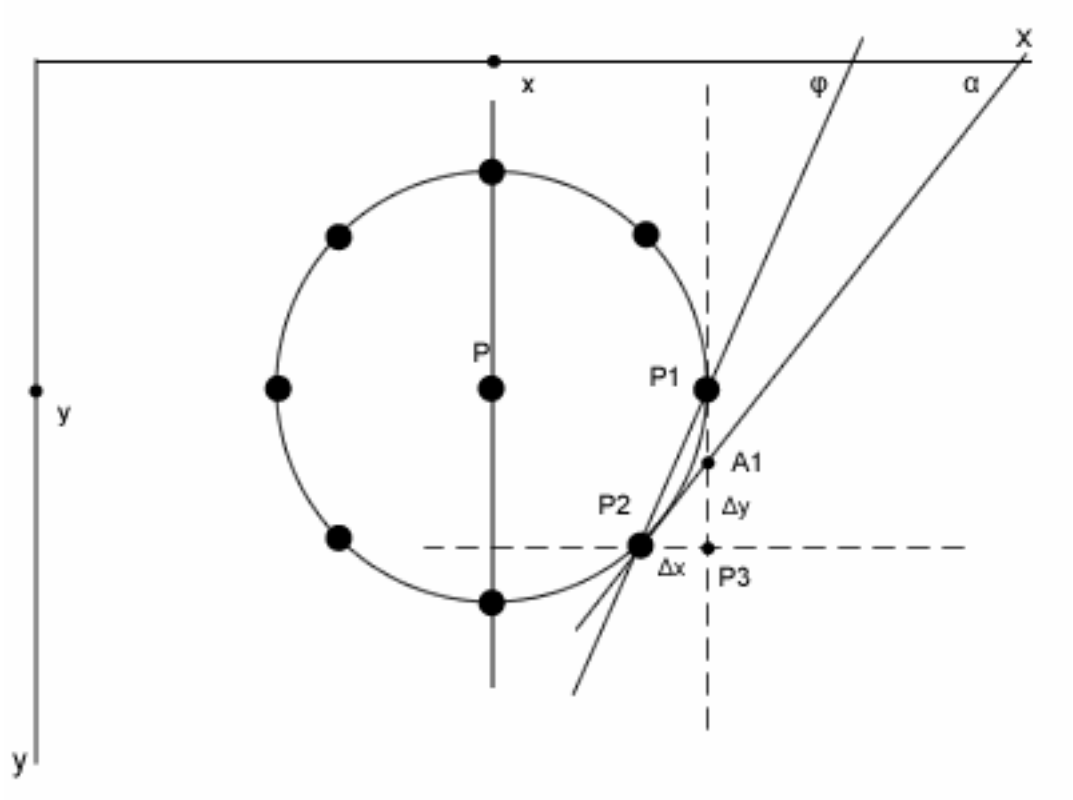
```
function drawCircle(startX, startY, r, style) {
      ctx.save();
      ctx.beginPath();
      ctx.strokeStyle=style;
      ctx.moveTo(startX + r, startY);
      ctx.quadraticCurveTo(startX + r, startY + r * Math.tan(Math.PI / 8), startX
                  + r * Math.cos(Math.PI / 4), startY + r * Math.cos(Math.PI /
4));
      ctx.quadraticCurveTo(startX + r * Math.tan(Math.PI / 8), startY + r,
                  startX, startY + r);
      ctx.quadraticCurveTo(startX - r * Math.tan(Math.PI / 8), startY + r, startX
                  - r * Math.cos(Math.PI / 4), startY + r * Math.cos(Math.PI /
4));
      ctx.quadraticCurveTo(startX - r, startY + r * Math.tan(Math.PI / 8), startX
                  - r, startY);
      ctx.quadraticCurveTo(startX - r, startY - r * Math.tan(Math.PI / 8), startX
                  - r * Math.cos(Math.PI / 4), startY - r * Math.cos(Math.PI /
4));
      ctx.quadraticCurveTo(startX - r * Math.tan(Math.PI / 8), startY - r,
                  startX, startY - r);
      ctx.quadraticCurveTo(startX + r * Math.tan(Math.PI / 8), startY - r, startX
                  + r * Math.cos(Math.PI / 4), startY - r * Math.cos(Math.PI /
4));
      ctx.quadraticCurveTo(startX + r, startY - r * Math.tan(Math.PI / 8), startX
                  + r, startY);
      ctx.stroke();
      ctx.restore();
};
```

Sprawdzenie algorytmu (Dodatek 7 Listing 137)



Jeszcze trochę matematyki

Zmieńmy niektóre elementy na naszym rysunku i przyjrzyjmy mu się ponownie.



Linia przechodząca przez punkty P_1 i P_2, nosi nazwę siecznej i tworzy z osią X kąt φ.
Linia przechodząca przez punkty A_1 i P_2 nosi nazwę stycznej - w tym przypadku - stycznej do okręgu i tworzy z osią X kąt α.
Jeśli przez punkty P i P_2 przeprowadzimy prostą, to będzie ona prostopadła do stycznej.
Na łuku koła - jeśli będziemy poruszali się od punktu P_2 do punktu P_1 osiągniemy przyrost współrzędnej x o odcinek Δx (od P_2 do P_3), a współrzędnej y o odcinek Δy (od P_3 do P_1).

Stosunek $\frac{\Delta y}{\Delta x} = tg\varphi$

Jeżeli punkt P_1 będzie poruszał się po łuku okręgu w kierunku punktu P_2 to odcinek Δx będzie dążył do zera, odcinek Δy będzie dążył do zera, sieczna będzie dążyła do stycznej, a kąt φ będzie dążył do kąta α.
Wtedy:

$$\frac{\Delta y}{\Delta x} = tg\alpha = y'$$

gdzie:
y' oznacza pierwszą pochodną.

Uogólniając można powiedzieć, że wartość pochodnej funkcji
$y = f(x)$ dla $x = x_1$
równa się tangensowi kąta nachylenia stycznej do krzywej, będącej obrazem danej funkcji, w jej punkcie o odciętej $x = x_1$.

Równanie stycznej do krzywej $y = f(x)$ w jej punkcie np. $P_2(x_1, y_1)$ jest:
$(y - y\ ) = f'(x_1)(x - x_1)$
gdzie:
x i y oznaczają współrzędne bieżące stycznej,
$f'(x_1)$ wartość pochodnej funkcji $f(x)$ dla $x = x_1$.

Właściwość 8

Wszystkie zmiany są globalne, tzn. zmiana któregokolwiek punktu w tablicy `points` prowadzi do globalnej zmiany przebiegu (kształtu) całej krzywej.

Właściwość 9

Im bardziej skomplikowany kształt tym wyższy stopień krzywej ten kształt modelującej.

Właściwość 10

Rzut perspektywiczny krzywej wielomianowej nie zawsze jest krzywą wielomianową. Ma to znaczenie w grafice komputerowej

Właściwość 11

Stopień krzywej jest bezpośrednio związany z liczbą punktów w tablicy `points`.

Właściwość 12

Krzywą Beziera, opisaną punktami `[a,b,c,d]` w tablicy `points` można przedstawić w postaci układu równań:
$P_x(t) = A_x t^3 + B_x t^2 + C_x t + D_x$
$P_y(t) = A_y t^3 + B_y t^2 + C_y t + D_y$
gdzie:
$D_x = d_x$
$C_x = 3(b_x - a_x)$
$B_x = 3(c_x - b_x) - C_x$
$A_x = d_x - a_x - C_x - B_x$
$D_y = d_y$
$C_y = 3(b_y - a_y)$
$B_y = 3(c_y - b_y) - C_y$
$A_y = d_y - a_y - C_y - B_y$

Algorytm de Casteljau

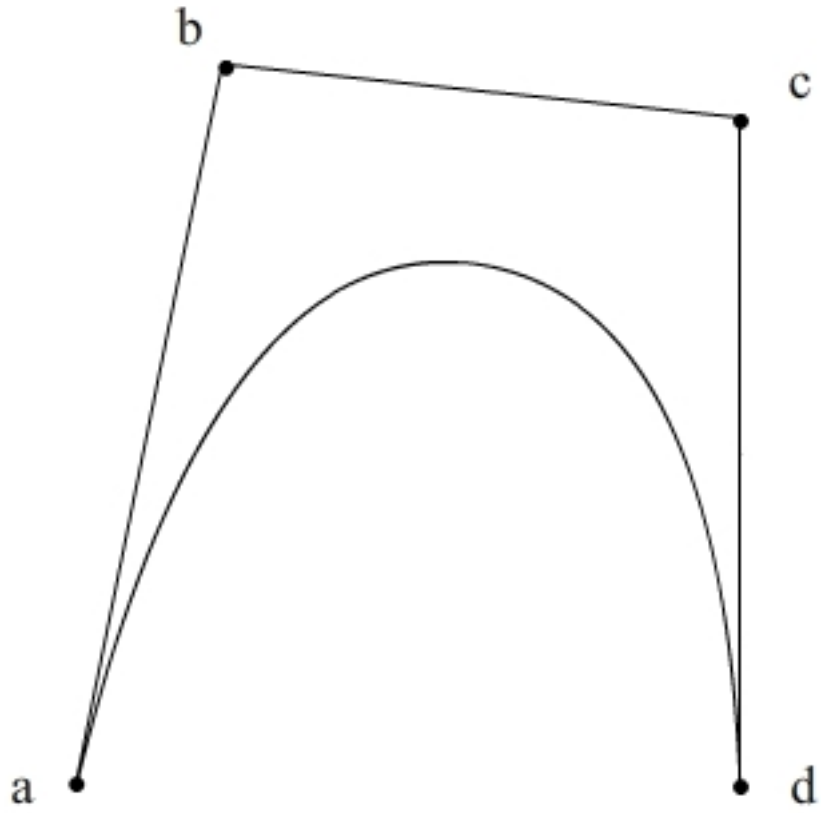
Jest algorytmem pozwalającym na:

- obliczenie położenia punktu na krzywej dowolnego stopnia dla danego t.

- podział krzywej na dwie krzywe składowe tego samego stopnia
- gładkie (płynne) łączenie dwóch krzywych

Obliczenie położenia punktu na krzywej dla danego t

Mamy 4 punkty [a,b,c,d] tworzące krzywą Beziera, a więc dwa punkty końcowe i dwa punkty kontrolne. Jest o krzywa 3-go stopnia.

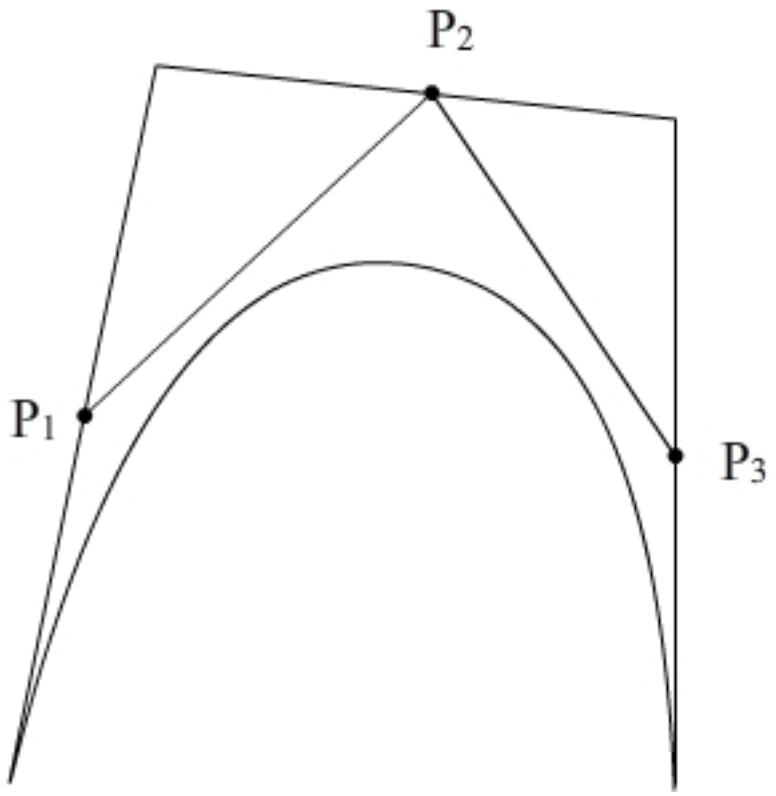


Każdy z odcinków ab, bc, cd dzielimy w proporcji t: (1-t), czyli wyznaczamy trzy punkty niższego stopnia [P_1, P_2, P_3].

$$P_1 = (1-t) * a + t * b$$

$$P_2 = (1-t) \ * b + t * c$$

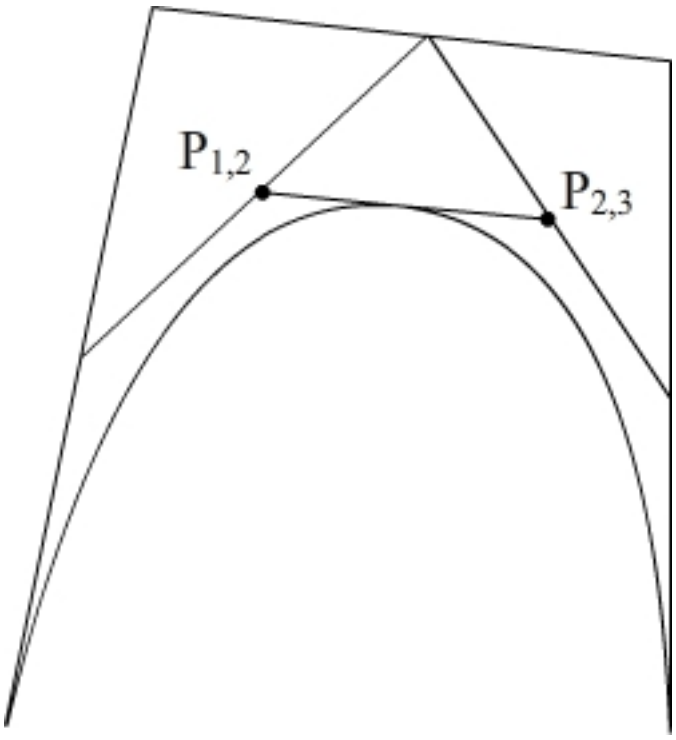
$$P_3 = (1-t) * c + t * d$$



Każdy z odcinków P_1P_2 i P_2P_1 dzielimy w proporcji t: (1-t) i otrzymujemy 2 punkty niższego stopnia [$P_{1,2}$, $P_{2,3}$]

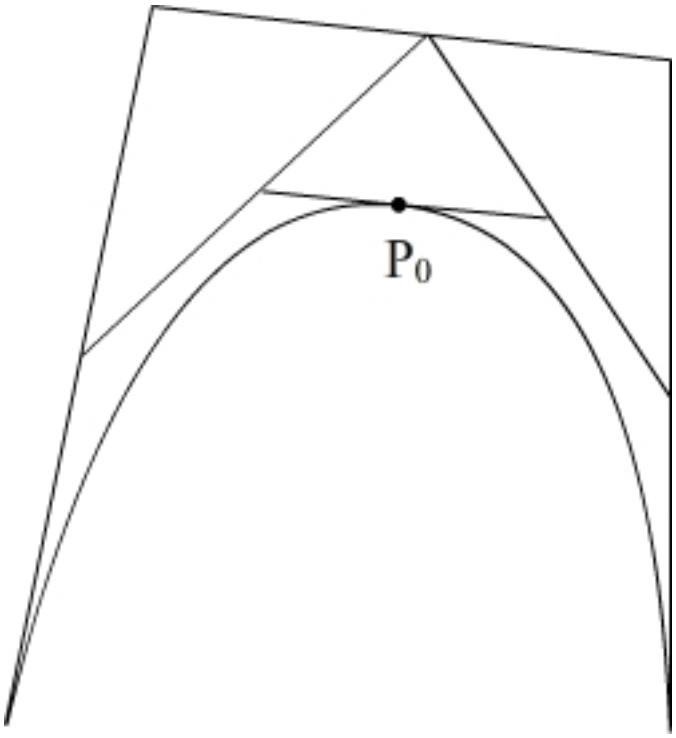
$$P_{1,2} = (1-t) * P_1 + t * P_2$$

$$P_{2,3} = (1-t) * P_2 + t * P_3$$



Odcinek $P_{1,2}$ $P_{2,3}$ dzielimy w proporcji t:(1-t) i otrzymujemy punkt [P_0] na krzywej.

$$P_0 = (1-t) * P_{1,2} + t * P_{2,3}$$



Obliczenia

Spróbujmy wyznaczyć położenie punktu na krzywej dla $t = 0.3$, gdy

- punkt a na krzywej ma współrzędne $a(0,1)$
- punkt kontrolny b krzywej ma współrzędne $b(2,3)$
- punkt kontrolny c krzywej ma współrzędne $c(6,7)$
- punkt d na krzywej ma współrzędne $d(4,5)$

$$P_1(x) = (1-t)*a_x + t*b_x = 0.7*0 + 0.3*2 = 0.6$$
$$P_1(y) = (1-t)*a_y + t*b_y = 0.7*1 + 0.3*3 = 1.6$$
$$P_2(x) = (1-t)*b_x + t*c_x = 0.7*2 + 0.3*6 = 3.2$$
$$P_2(y) = (1-t)*b_y + t*c_y = 0.7*3 = 0.3*7 = 4.2$$
$$P_3(x) = (1-t)*c_x + t*d_x = 0.7*6 + 0.3*4 = 5.4$$
$$P_3(y) = (1-t)*c_y + t*d_y = 0.7*7 + 0.3*5 = 6.4$$
$$P_{1,2}(x) = (1-t)*P_{1x} + t*P_{2x} = 0.7*0.6 + 0.3*3.2 = 1.38$$
$$P_{1,2}(y) = (1-t)*P_{1y} + t*P_{2y} = 0.7*1.6 + 0.3*4.2 = 2.38$$
$$P_{2,3}(x) = (1-t)*P_{2x} + t*P_{3x} = 0.7*3.2 + 0.3*5.4 = 3.86$$
$$P_{2,3}(y) = (1-t)*P_{2y} + t*P_{3y} = 0.7*4.2 + 0.3*6.4 = 4.86$$
$$P_0(x) = (1-t)*P_{1,2x} + t*P_{2,3x} = 0.7*1.38 + 0.3*3.86 = 2.124$$
$$P_0(y) = (1-t)*P_{1,2y} + t*P_{2,3y} = 0.7*2.38 + 0.3*4.86 = 3.124$$

Algorytm

```
var cloneArray2 = function(array) {
```

```
        var len = array.length;
        var arr = new Array(len);
        for ( var i = 0; i < len; i++) {
                arr[i] = new Point(array[i].x, array[i].y);
        }
        return arr;
};

var casteljauTValue = function(points, t) {
        var temp = cloneArray2(points);
        var len = points.length;
        var t1 = 1.0 - t;
        for ( var i = 0; i < len - 1; i++) {
                for ( var j = 0; j < len - 1 - i; j++) {
                        temp[j].x = temp[j].x * t1 + t * temp[j + 1].x;
                        temp[j].y = temp[j].y * t1 + t * temp[j + 1].y;
                }
        }
        return temp[0];
};
```

Sprawdzenie algorytmu:

(Dodatek 7 Listing 134)

```
[2.1239999999999997, 3.1239999999999997]
```

Algorytmu de Casteljau można użyć do obliczenia wszystkich t i rysowania krzywej, ale tutaj nie przedstawiamy stosownego algorytmu. Pozostawiamy to jako ćwiczenie dla Czytelnika.

Podział krzywej na dwie krzywe

Kolejne kroki algorytmu de Casteljau były następujące:

a		b		c		d
	P_1		P_2		P_3	
		$P_{1,2}$		$P_{2,3}$		
			P_0			

Krzywa Beziera 3-go stopnia o punktach [a,b,c,d] została podzielona na dwie krzywe Beziera o punktach [a, P_1, $P_{1,2}$, P_0] oraz [P_0, $P_{2,3}$, P_3, d]
Aby wyznaczyć punkty obu krzywych możemy użyć algorytmu:

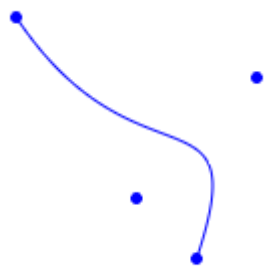
Algorytm

```
var divideCasteljau = function(points) {
        var temp = cloneArray2(points);
        var temp1 = cloneArray2(points).reverse();
        var len = points.length;
        var t = 0.3;
        var t1 = 1.0 - t;
        for ( var i = 0; i < len - 1; i++) {
                for ( var j = 0; j < len - 1 - i; j++) {
                        temp[j].x = temp[j].x * t1 + t * temp[j + 1].x;
                        temp[j].y = temp[j].y * t1 + t * temp[j + 1].y;
                        temp1[j].x = temp1[j + 1].x * t1 + t * temp1[j].x;
```

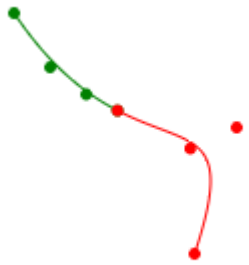
```
                    temp1[j].y = temp1[j + 1].y * t1 + t * temp1[j].y;
                }
        }
        var arr = new Array(2);
        arr[0] = temp1.reverse();
        arr[1] = temp;
        return arr;
};
```

Sprawdzenie algorytmu

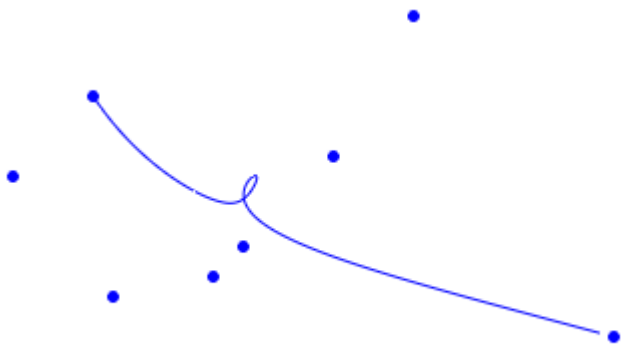
Krzywa przed podziałem:



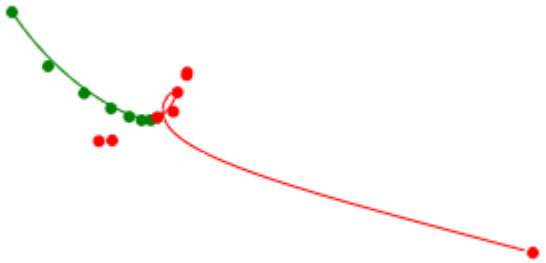
Krzywa po podziale (Dodatek 7 Listing 135)



Krzywa przed podziałem:



Krzywa po podziale (Dodatek 7 Listing 136)



Gładkie połączenie dwóch krzywych

O pojęciu 'gładkości' połączenia powiemy więcej przy krzywych B-sklejanych. Aby osiągnąć 'odpowiednią' gładkość połączenia dwóch krzywych Beziera reprezentowych przez punkty kontrolne:

$P_0, P_1, \ldots, P_{n-1}, P_n$

oraz

$Q_0, Q_1, \ldots, Q_{m-1}, Q_m$

muszą być spełnione warunki:

$P_n = Q_0$

P_{n-1}, $P_n = Q_0$, Q_1 leżą na jednej prostej

$$P_{n-1} = \frac{(m+n)Q_0 - mQ_1}{n}$$

gdzie m i n są stopniami krzywej.

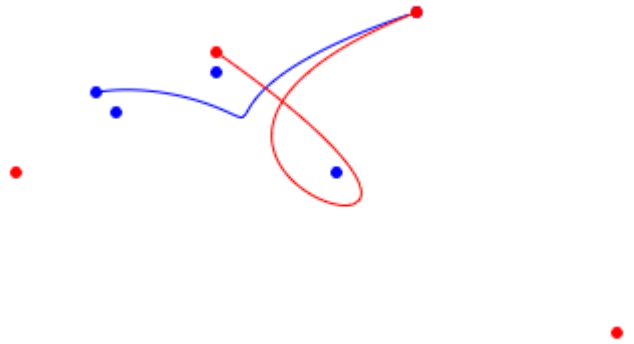
Algorytm

```
var connectBezier = function(points, qoints) {
      var n = points.length - 1;
      var m = qoints.length - 1;
      points[n - 1].x = ((m + n) * qoints[0].x - m * qoints[1].x) / n;
      points[n - 1].y = ((m + n) * qoints[0].y - m * qoints[1].y) / n;
};
```

Wykresy

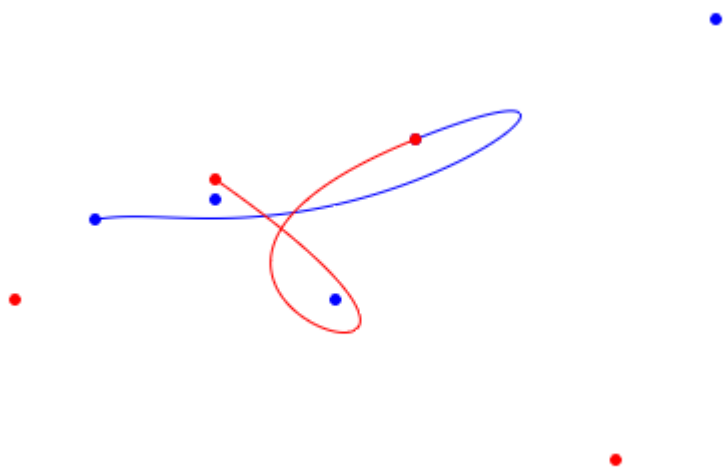
Plik: *bezier41.html*.

Połączone krzywa 4-tego i 3-stopnia:



(Dodatek 7 Listing 152)

Połączenie wygładzone:



Wymierne krzywe Beziera

Wszystkie dotychczas omawiane krzywe Beziera były krzywymi wielomianowymi. Oprócz nich stosuje się wymierne krzywe Beziera. Ich obliczanie jest nieco bardziej zawiłe.

Definicja

Wymierną krzywą Beziera *n*-tego stopnia definuje się jako:

$$C(t) = \frac{\sum_{k=0}^{n} B_{n,k}(t) w_k P_k}{\sum_{j=0}^{n} B_{n,j}(t) w_j}$$

przy $t \in [0,1]$

gdzie:

n - jest stopniem krzywej i jest o 1 mniejszy od liczby punktów
P_k - jest kolejnym punktem kontrolnym
$B_{n,k}$ i $B_{n,j}$ są wielomianami Bernsteina
w_k i w_j są wagami punktów. Może to być dowolna liczba rzeczywista. Jeżeli waga jest równa 0 to punkt, który ma tę wagę nie jest brany pod uwagę. W standardowych przypadkach $w > 0$, co powoduje, że mianownik jest zawsze większy od 0.
Powyższy wzór dla celów obliczeniowych można przekształcić w:

$$C(t) = \sum_{k=0}^{n} R_{n,k}(t) P_k$$

przy $t \in [0,1]$

gdzie:

$$R_{n,k}(t) = \frac{B_{n,k}(t) w_k}{\sum_{j=0}^{n} B_{n,j}(t) w_j}$$

gdzie $R_{n,k}(t)$ jest funkcją bazową.
Jeżeli wszystkie wagi są sobie równe i nie zerowe to $R_{n,k}(t) = B_{n,k}(t)$, a wymierna krzywa Beziera staje się wielomianową krzywą Beziera.

Funkcje bazowe wymiernych krzywych Beziera

n=0

$$R_{0,0}(t) = \frac{B_{0,0}(t) w_0}{B_{0,0}(t) w_0} = \frac{1 * w_0}{1 * w_0} = 1$$

n=1

$$R_{1,0}(t) = \frac{B_{1,0}(t) w_0}{B_{1,0}(t) w_0 + B_{1,1}(t) w_1} = \frac{(1-t) w_0}{(1-t) w_0 + t w_1}$$

$$R_{1,1}(t) = \frac{B_{1,1}(t) w_1}{B_{1,0}(t) w_0 + B_{1,1}(t) w_1} = \frac{t w_1}{(1-t) w_0 + t w_1}$$

n=2

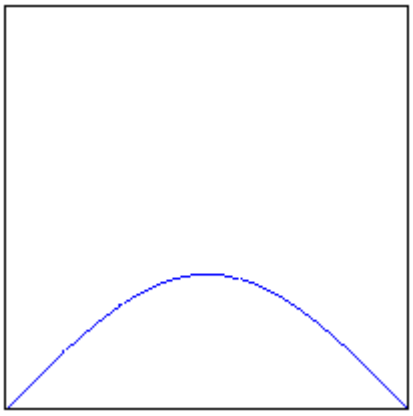
Dalej będziemy od razu zastępowali wielomiany Bernsteina ich wartościami, których obliczanie pokazaliśmy wcześniej.

$$R_{2,0}(t) = \frac{(1-t)^2 w_0}{(1-t)^2 w_0 + 2t(1-t) w_1 + t^2 w_2}$$

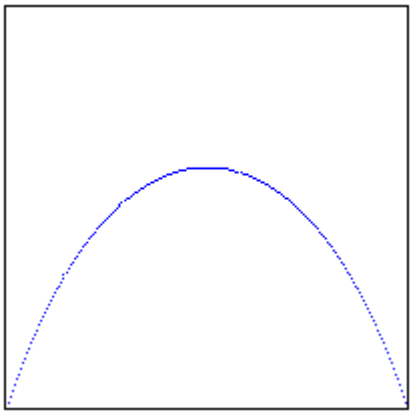
$$R_{2,1}(t) = \frac{2t(1-t)w_1}{(1-t)^2 w_0 + 2t(1-t)w_1 + t^2 w_2}$$

(Dodatek 7 Listing 138)

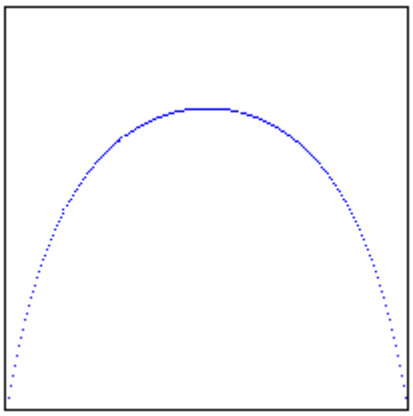
```
var weights = [1, 1, 1];
rBezier(2, 1, weights, 200);
```



```
var weights = [1, 3, 1];
rBezier(2, 1, weights, 200);
```



```
var weights = [1, 6, 1];
rBezier(2, 1, weights, 200);
```



$$R_{2,2}(t) = \frac{t^2 w_2}{(1-t)^2 w_0 + 2t(1-t)w_1 + t^2 w_2}$$

n=3

$$R_{3,0}(t) = \frac{(1-t)^3 w_0}{(1-t)^3 w_0 + 3t(1-t)^2 w_1 + 3t^2(1-t)w_2 + t^3 w_3}$$

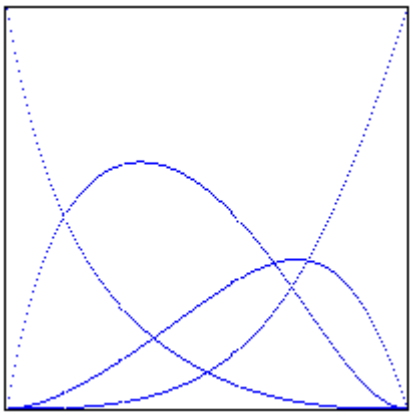
$$R_{3,1}(t) = \frac{3t(1-t)^2 w_1}{(1-t)^3 w_0 + 3t(1-t)^2 w_1 + 3t^2(1-t)w_2 + t^3 w_3}$$

$$R_{3,2}(t) = \frac{3t^2(1-t)w_2}{(1-t)^3 w_0 + 3t(1-t)^2 w_1 + 3t^2(1-t)w_2 + t^3 w_3}$$

$$R_{3,3}(t) = \frac{t^3 w_3}{(1-t)^3 w_0 + 3t(1-t)^2 w_1 + 3t^2(1-t)w_2 + t^3 w_3}$$

(Dodatek 7 Listing 139)

```
var weights = [1, 6, 3, 1];
rBezier(3, 0, weights, 200);
rBezier(3, 1, weights, 200);
rBezier(3, 2, weights, 200);
rBezier(3, 3, weights, 200);
```



Algorytm

```
var rBezierTValue2 = function(np, n, k, t, weights) {
	var arr = new Array();
	var mian = 0.0;
	var sum = 0.0;
	for ( var i = 0; i < n + 1; i++) {
		var a = np * Math.pow(t, i) * Math.pow(1.0 - t, n - i)
				* weights[i];
		if (a < 0) {
			a = 0;
		}
		arr.push(a);
		mian += a;
	}
	sum = arr[k] / mian;
```

```
        return sum;
};

var rBezier = function(n, k, weights, liczbaPunktow) {
        var t;
        var img1 = ctx.getImageData(0, 0, width, height);
        var idata = img1.data;
        var np = npok(n,k);
        for ( var i = 0; i < liczbaPunktow; i++) {
                t = i * 1.0 / liczbaPunktow;
                var b = rBezierTValue2(np,n,k,t,weights);
                if (b < 0) {
                        b = 0;
                }
                var j = 4 * index(cutDecimal(height - b * liczbaPunktow),
cutDecimal(t
                                * liczbaPunktow), width);
                idata[j] = 0;
                idata[j + 1] = 0;
                idata[j + 2] = 255;
                idata[j + 3] = 255;
        }
        ctx.putImageData(img1, 0, 0);
};
```

Wymierne krzywe Beziera 2-go stopnia

Dla tablic `points[a,b,c]` i `weights[w0,w1,w2]` i dla danego *t* położenie punktu na krzywej można obliczyć z wzoru:

$$x = R_{2,0}(t) * a_x + R_{2,1}(t) * b_x + R_{2,2}(t) * c_x$$
$$y = R_{2,0}(t) * a_y + R_{2,1}(t) * b_y + R_{2,2}(t) * c_y$$

gdzie $R_{n,k}$ są funkcjami bazowymi.

Algorytm

Dla wykreślenia krzywej możemy użyć algorytmu:

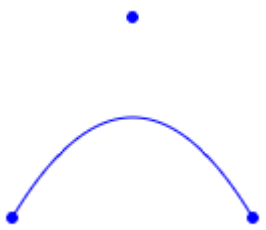
```
var drawRatBezier2 = function(points, weights, style, visible) {
        var extr = tValue(points);
        var te = 1.0 / extr;
        ctx.save();
        ctx.beginPath();
        ctx.strokeStyle = style;
        ctx.moveTo(points[0].x, points[0].y);
        for ( var i = 0; i < extr + 1; i++) {
                var t = i * te;
                var t1 = 1.0 - t;
                var m0 = t1 * t1 * weights[0];
                var m1 = 2 * t * t1 * weights[1];
                var m2 = t * t * weights[2];
                var mian = m0 + m1 + m2;
                var w0 = m0 / mian;
                var w1 = m1 / mian;
```

```
            var w2 = m2 / mian;
            var x = w0 * points[0].x + w1 * points[1].x + w2 * points[2].x;
            var y = w0 * points[0].y + w1 * points[1].y + w2 * points[2].y;
            ctx.lineTo(x, y);
        }
        ctx.lineTo(points[2].x, points[2].y);
        ctx.stroke();
        ctx.restore();
        if (visible) {
            drawPoints(points, style);
        }
};
```

Wykresy

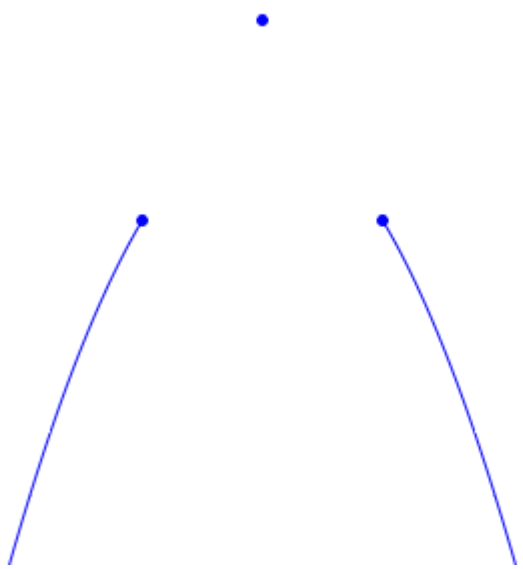
w_1 =1.0

(Dodatek 7 Listing 140)
opis: łuk paraboli



w_1=-1.0

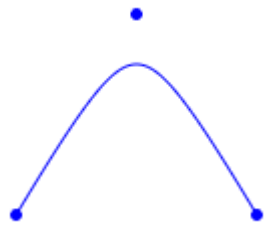
(Dodatek 7 Listing 141)
opis: dwa łuki paraboli



w1>1.0

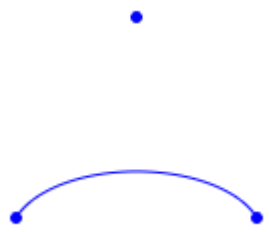
(Dodatek 7 Listing 142)

opis: łuk hiperboli



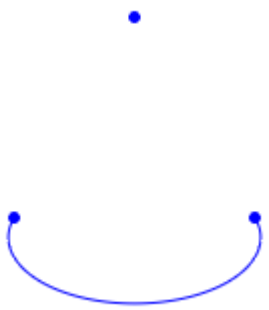
0<w1<1.0

(Dodatek 7 Listing 143)
opis: krótszy odcinek okręgu lub elipsy



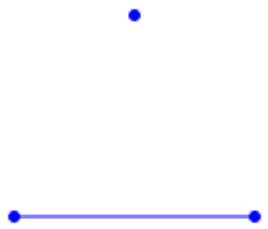
-1.0<w1<00

(Dodatek 7 Listing 144)
opis: dłuższy odcinek okręgu lub elipsy



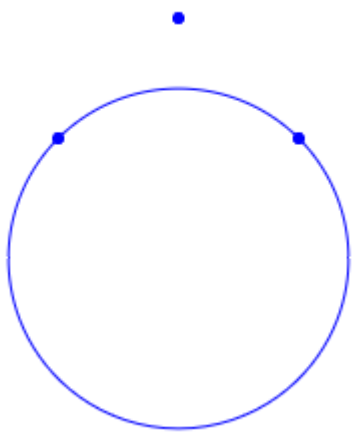
w1<0

(Dodatek 7 Listing 145)
opis: sparametryzowany odcinek



w1= $\frac{\sqrt{2}}{2}$

(Dodatek 7 Listing 146)
opis: koło



Wymierne krzywe Beziera 3-go stopnia

Dla tablic `points[a,b,c,d]` i `weights[w0, w1, w2, w3]` i dla danego *t* położenie punktu na krzywej można obliczyć z wzoru:

$$x = R_{3,0}(t) * a_x + R_{3,1}(t) * b_x + R_{3,2}(t) * c_x + R_{3,3}(t) * d_x$$
$$y = R_{3,0}(t) * a_y + R_{3,1}(t) * b_y + R_{3,2}(t) * c_y + R_{3,3}(t) * d_y$$

gdzie $R_{n,k}$ są funkcjami bazowymi.

Algorytm

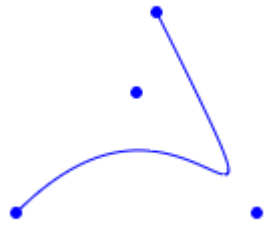
Dla wykreślenia krzywej możemy użyć algorytmu:

```
var drawRatBezier3 = function(points, weights, style, visible) {
      var extr = tValue(points);
      var te = 1.0 / extr;
      ctx.save();
      ctx.beginPath();
      ctx.strokeStyle = style;
      ctx.moveTo(points[0].x, points[0].y);
      for ( var i = 0; i < extr + 1; i++) {
            var t = i * te;
            var t1 = 1.0 - t;
            var m0 = t1 * t1 * t1 * weights[0];
            var m1 = 3 * t * t1 * t1 * weights[1];
            var m2 = 3 * t * t * t1 * weights[2];
            var m3 = t * t * t * weights[3];
            var mian = m0 + m1 + m2 + m3;
            var w0 = m0 / mian;
            var w1 = m1 / mian;
            var w2 = m2 / mian;
```

```
            var w3 = m3 / mian;
            var x = w0 * points[0].x + w1 * points[1].x + w2 * points[2].x + w3
                        * points[3].x;
            var y = w0 * points[0].y + w1 * points[1].y + w2 * points[2].y + w3
                        * points[3].y;
            ctx.lineTo(x, y);
      }
      ctx.lineTo(points[3].x, points[3].y);
      ctx.stroke();
      ctx.restore();
      if (visible) {
            drawPoints(points, style);
      }
};
```

Wykres

(Dodatek 7 Listing 147)



Wymierne krzywe Beziera n-tego stopnia

Algorytm

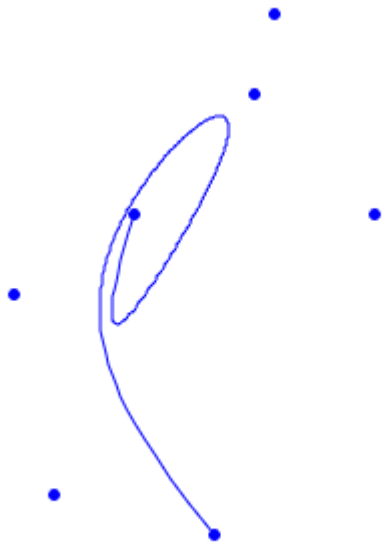
```
var drawRatBezierN = function(points, weights, style, visible) {
      var extr = tValue(points);
      var te = 1.0 / extr;
      var n = points.length - 1;
      ctx.save();
      ctx.beginPath();
      ctx.strokeStyle = style;
      ctx.moveTo(points[0].x, points[0].y);
      for ( var i = 0; i < extr + 1; i++) {
            var t = i * te;
            var t1 = 1.0 - t;
            var ms = new Array();
            for ( var j = 0; j < points.length; j++) {
                  ms.push(npok(n, j) * Math.pow(t, j) * Math.pow(t1, n - j)
                              * weights[j]);
            }
            var mian = 0.0;
            for ( var k = 0; k < ms.length; k++) {
                  mian += ms[k];
            }
            var x = 0.0;
            var y = 0.0;
            for ( var m = 0; m < points.length; m++) {
                  x += ms[m] * points[m].x;
```

```
                y += ms[m] * points[m].y;
            }
            x = cutDecimal(x / mian);
            y = cutDecimal(y / mian);
            ctx.lineTo(x, y);
            // x = 0.0;
            // y = 0.0;
        }
        ctx.lineTo(points[points.length - 1].x, points[points.length - 1].y);
        ctx.stroke();
        ctx.restore();
        if (visible) {
            drawPoints(points, style);
        }

};
```

Wykres

(Dodatek 7 Listing 148)

Wymierna krzywa Beziera 6 -go stopnia



Właściwości wymiernych krzywych Beziera

Dla `var points` zawierającej tablicę obiektów `Point(x,y)`:

Właściwość 1

Punktem początkowym krzywej jest `points[0]`, a punktem końcowym krzywej `points[points.length-1]`. Pozostałe punkty w tablicy to punkty kontrolne krzywej, które na ogół nie leżą na krzywej.

Właściwość 2

Jeżeli wszystkie wagi są niezerowe i tego samego znaku to krzywa zawiera się w wieloboku, którego wierzchołkami są punkty umieszczone w tablicy `points`.

Właściwość 3

Styczna do krzywej w punkcie `points[0]` jest równoległa do odcinka `points[0] - points[1]`, a styczna do krzywej w punkcie `points[points.length-1]` jest równoległa do odcinka `points[points.length-1] - points[points.length-2]`.

Właściwość 4

Jeżeli `points[0] == points[length-1]` to krzywa jest zamknięta, w przeciwnym przypadku krzywa jest otwarta.

Właściwość 5

Jeżeli odcinek `points[0]-points[1]` jest równoległy do odcinka `points[length-1]-points[length-2]` to połączenie w `points[0]=points[length-1]` jest płynne.

Właściwość 6

Przekształcenia (transformacje) krzywej wykonujemy poddając transformacjom punkty w tablicy `points`.

Właściwość 7

Przy użyciu jednej krzywej wymiernej można uzyskać figury stożkowe takie jak elipsa i hiperbola oraz fragmenty takich figur stożkowych takich jak koło i elipsa, przy czym koło i elipsę można złożyć z zaledwie 2 krzywych.

Właściwość 8

Wszystkie zmiany są globalne, tzn. zmiana któregokolwiek punktu w tablicy `points` prowadzi do globalnej zmiany przebiegu (kształtu) całej krzywej.

Właściwość 9

Im bardziej skomplikowany kształt tym wyższy stopień krzywej ten kształt modelującej.

Właściwość 10

Rzut perspektywiczny krzywej wymiernej jest zawsze krzywą wymierną, co ma znaczenie w grafice komputerowej.

Właściwość 11

Krzywe wymierne mają tę przewagę nad krzywymi wielomianowymi, że zapewniają lepszą kontrolę nad przebiegiem krzywej, dzięki użyciu wag punktów.

Właściwość 12

Jeżeli wszystkie wagi są równe sobie i niezerowe to krzywa staje się krzywą wielomianową.

Właściwość 13

Przemnożenie wszystkich wag przez tę samą liczbę różną od zera nie zmienia przebiegu krzywej.

Właściwość 14

Stopień krzywej jest bezpośrednio związany z liczbą punktów w tablicy `points`.

Dodatek 7. Listingi kodów uzupełniających

Listing01

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<script type="text/javascript" src="../scripts/kolory.js"></script>
<title> </title>
<script type="text/javascript">
        var $ = function(id) {
                return document.getElementById(id);
        };
        var wzrost = parseInt(175);
        var wynik1 = wzrost.toString(16);
        var wynik2 = parseInt(wynik1, 16);
        var a = 0xff;
        var wynik3 = a.toString(2);
        var rgb = 2345362;
        var wynik4 = rgb.toString(2);
        var c = rgb >> 16;
        var wynik5 = c.toString(2);
        var R = c & a;
        var wynik6 = R.toString(2);
        window.onload = function() {
                $("wynik1").firstChild.nodeValue = "wynik1: " + wynik1;
                $("wynik2").firstChild.nodeValue = "wynik2: " + wynik2
                $("wynik3").firstChild.nodeValue = "wynik3: " + wynik3;
                $("wynik4").firstChild.nodeValue = "wynik4: " + wynik4;
                $("wynik5").firstChild.nodeValue = "wynik5: " + wynik5;
                $("wynik6").firstChild.nodeValue = "wynik6: " + wynik6;
        };
</script>
</head>
<body>
        <p>
                <span id="wynik1"> </span>
        </p>
        <p>
                <span id="wynik2"> </span>
        </p>
        <p>
                <span id="wynik3"> </span>
        </p>
        <p>
                <span id="wynik4"> </span>
        </p>
        <p>
                <span id="wynik5"> </span>
        </p>
        <p>
                <span id="wynik6"> </span>
        </p>
</body>
</html>
```

Zawartość skryptu ‘kolory.js’

```
/* funkcja zmieniajaca liczbe heksadecymalna na wartosci RGB
 * Liczba może być podana jako 0x np "0xFFFFFF" lub "#FFFFFF"
 * lub "FFFFFF"
 */
var HexToRGB = function(hexa) {
	var R = -1;
	var G = -1;
	var B = -1;
	if (((hexa.indexOf("x") == -1) && (hexa.indexOf("#") == -1))) {
		R = parseInt(hexa.substr(0, 2), 16);
		G = parseInt(hexa.substr(2, 2), 16);
		B = parseInt(hexa.substr(4, 2), 16);
	} else {
		if (hexa.indexOf("#") > 0) {
			R = parseInt(hexa.substr(1, 2), 16);
			G = parseInt(hexa.substr(3, 2), 16);
			B = parseInt(hexa.substr(5, 2), 16);
		} else if (hexa.indexOf("x") > 0) {
			R = parseInt(hexa.substr(2, 2), 16);
			G = parseInt(hexa.substr(4, 2), 16);
			B = parseInt(hexa.substr(6, 2), 16);
		}
	}
	return [ R, G, B ];
};

/*
 * funkcja zmieniajaca wartosci RGB na licybe heksdecymalna. Liczby musza być
 * podane jako inty od 0 do 255 włšcznie
 */
var RGBToHex = function(r, g, b) {
	var r1 = parseInt(r).toString(16).toUpperCase();
	var g1 = parseInt(g).toString(16).toUpperCase();
	var b1 = parseInt(b).toString(16).toUpperCase();
	if (r1.length == 1) {
		r1 = "0" + r1;
	}
	if (g1.length == 1) {
		g1 = "0" + g1;
	}
	if (b1.length == 1) {
		b1 = "0" + b1;
	}
	return "0x" + r1 + g1 + b1;
};
// zwraca tablice R,G,B,A gdzie A jest liczbš float pomiedzy 0.0 - 1.0
var HexToRGBA = function(hexa) {
	var R = -1;
	var G = -1;
	var B = -1;
	var A = -1;
	if (((hexa.indexOf("x") == -1) && (hexa.indexOf("#") == -1))) {
		R = parseInt(hexa.substr(0, 2), 16);
		G = parseInt(hexa.substr(2, 2), 16);
		B = parseInt(hexa.substr(4, 2), 16);
		A = parseInt(hexa.substr(6, 2), 16);
	} else {
		if (hexa.indexOf("#") > 0) {
			R = parseInt(hexa.substr(1, 2), 16);
			G = parseInt(hexa.substr(3, 2), 16);
			B = parseInt(hexa.substr(5, 2), 16);
			A = parseInt(hexa.substr(7, 2), 16);
		} else if (hexa.indexOf("x") > 0) {
			R = parseInt(hexa.substr(2, 2), 16);
			G = parseInt(hexa.substr(4, 2), 16);
```

```
                        B = parseInt(hexa.substr(6, 2), 16);
                        A = parseInt(hexa.substr(8, 2), 16);
                }
        }
        return [ R, G, B, Math.roundToDecimal((A - 0.5) / 255, 4) ];
};
// a jest typu float pomiędzy 0.0 - 1.0
var RGBAToHex = function(r, g, b, a) {
        var r1 = parseInt(r).toString(16).toUpperCase();
        var g1 = parseInt(g).toString(16).toUpperCase();
        var b1 = parseInt(b).toString(16).toUpperCase();
        var a1 = parseInt(Math.roundToDecimal(a * 255 + 0.5), 4).toString(16)
                        .toUpperCase();
        if (r1.length == 1) {
                r1 = "0" + r1;
        }
        if (g1.length == 1) {
                g1 = "0" + g1;
        }
        if (b1.length == 1) {
                b1 = "0" + b1;
        }
        if (a1.length == 1) {
                a1 = "0" + a1;
        }
        return "0x" + r1 + g1 + b1;
};

// Przyjmuje wartoœci RGB
// zwraca tablicę HSL (h w stopniach od 0.0 - 360, pozostałe
// w %)
var RGBToHSL = function(r, g, b) {
        r /= 255, g /= 255, b /= 255;
        var max = Math.max(r, g, b), min = Math.min(r, g, b);
        var sm = max + min;
        var h = sm / 2;
        var s = sm / 2;
        var l = sm / 2;

        if (max == min) {
                h = s = 0; // achromatic
        } else {
                var d = max - min;
                s = l > 0.5 ? d / (2 - sm) : d / sm;
                switch (max) {
                case r:
                        h = (g - b) / d + (g < b ? 6 : 0);
                        break;
                case g:
                        h = (b - r) / d + 2;
                        break;
                case b:
                        h = (r - g) / d + 4;
                        break;
                }
                h *= 60.0;
        }

        return [ roundToDecimal(h, 4), roundToDecimal(s, 4), roundToDecimal(l, 4) ];
};

// h - barwa w stopniach 0.0 - 360.0
// s - nasycenie w % od 0.0 - 100%
// l - jasno ć w % od 0.0 - 100%
// zwraca tablice warto ci RGB
var HSLToRGB = function(h, s, l) {
        var r, g, b;
        h /= 360.0;
```

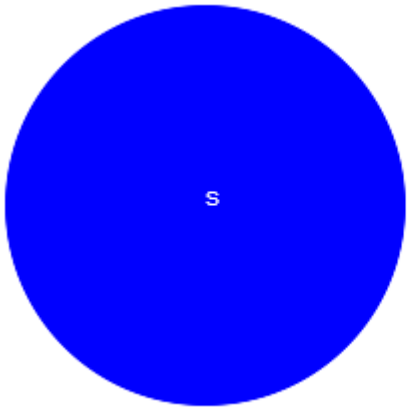
```
	if (s == 0) {
		r = g = b = l; // achromatic
	} else {
		var q = l < 0.5 ? l * (1 + s) : l + s - l * s;
		var p = 2 * l - q;
		r = HueToRGB(p, q, h + 1.0 / 3.0);
		g = HueToRGB(p, q, h);
		b = HueToRGB(p, q, h - 1.0 / 3.0);
	}

	return [ roundToDecimal(r * 255, 0), roundToDecimal(g * 255, 0),
			roundToDecimal(b * 255, 0) ];
};
// funkcja pomocnicza do obliczania nasycenia
var HueToRGB = function(a, b, c) {
	if (c < 0)
		c += 1;
	if (c > 1)
		c -= 1;
	if (c < 1.0 / 6.0)
		return a + (b - a) * 6 * c;
	if (c < 1.0 / 2.0)
		return b;
	if (c < 2.0 / 3.0)
		return a + (b - a) * (2. / 3.0 - c) * 6;
	return a;
};

var roundToDecimal = function(num, dec) {
	var multi = Math.pow(10, dec);
	return Math.round(num * multi) / multi;
};

/**
 * Zamienia wartosci R, G, B na liczbę int stosowanš w Javie
 *
 * @param r
 *		int - składnik czerwony;
 * @param g
 *		int - składnik zielony
 * @param b
 *		int - składnik niebieski
 * @return int - warto ć RGB przeliczona na int;
 */

var RGBToInt = function(r, g, b) {
	var r1 = parseInt(r).toString(16);
	var g1 = parseInt(g).toString(16);
	var b1 = parseInt(b).toString(16);
	if (r1.length == 1) {
		r1 = "0" + r1;
	}
	if (g1.length == 1) {
		g1 = "0" + g1;
	}
	if (b1.length == 1) {
		b1 = "0" + b1;
	}
	var cont = "" + r1 + g1 + b1;
	var i = parseInt(cont, 16);
	return i;
};

/**
 * Przelicza liczbę int na R, G, B
 *
 * @param rgb
 *		int - liczba int oznaczajšca kolor;
```

```
 * @return String - warto ć RGB
 */

var IntToRGB = function(rgb) {
       var R = (rgb >> 16) & 0xff;
       var G = (rgb >> 8) & 0xff;
       var B = (rgb >> 0) & 0xff;
       return new Array(R, G, B);
};
```

Listing02

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
       <canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
              przeglądarce obsługującej element &lt;canvas&gt;
              z kontekstem &#034;2d&#034;</canvas>
       <script type="text/javascript">
       var cv = document.getElementById('canvas');
       var ctx = cv.getContext('2d');
       var width = cv.width;
       var height = cv.height;
       ctx.save();
       ctx.beginPath();
       ctx.fillStyle="rgba(0,0,255,1)";
       ctx.arc(150, 150, 100,0,2*Math.PI, false);
       ctx.fill();
       ctx.restore();
       ctx.fillStyle="white";
       ctx.fillText("S", 150, 150);
       </script>
</body>
</html>
```

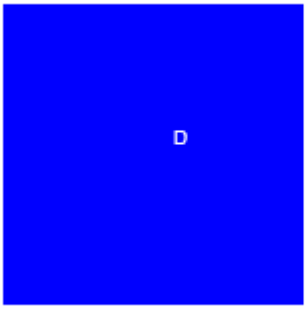
Wynik



Listing03

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		var width = cv.width;
		var height = cv.height;
		ctx.save();
		ctx.beginPath();
		ctx.fillStyle = "rgba(0,0,255,1)";
		ctx.fillRect(190, 80, 150, 150);
		ctx.restore();
		ctx.fillStyle = "white";
		ctx.fillText("D", 275, 150);
	</script>
</body>
</html>
```

Wynik

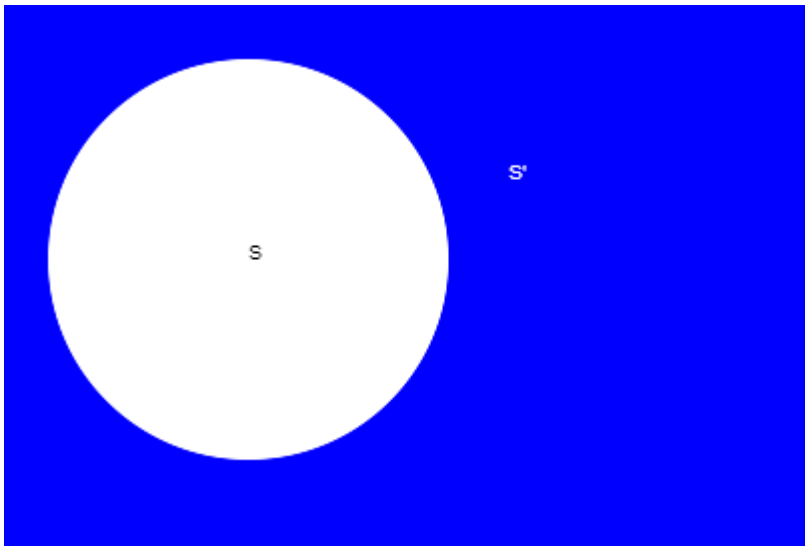


Listing04

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
```

```
        <script type="text/javascript">
                var cv = document.getElementById('canvas');
                var ctx = cv.getContext('2d');
                var width = cv.width;
                var height = cv.height;
                ctx.save();
                ctx.beginPath();
                ctx.fillStyle = "rgba(0,0,255,1)";
                ctx.fillRect(0, 0, 800, 800);
                ctx.fillStyle = "white";
                ctx.arc(150, 150, 100, 0, 2 * Math.PI, false);
                ctx.fill();
                ctx.restore();
                ctx.fillStyle = "white";
                ctx.fillText("S'", 280, 110);
                ctx.fillStyle = "black";
                ctx.fillText("S", 150, 150);
        </script>
</body>
</html>
```

Wynik

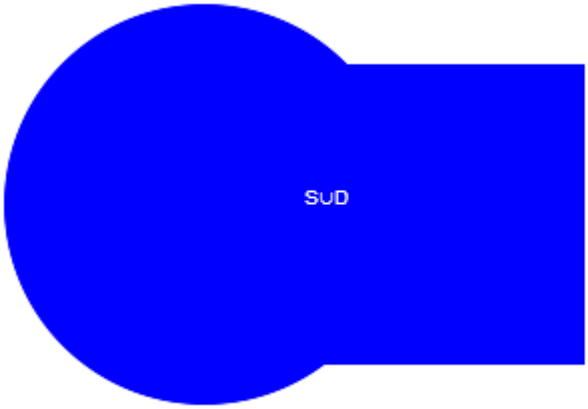


Listing05

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
        <canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
                przeglądarce obsługującej element &lt;canvas&gt;
                z kontekstem &#034;2d&#034;</canvas>
        <script type="text/javascript">
                var cv = document.getElementById('canvas');
                var ctx = cv.getContext('2d');
                var width = cv.width;
                var height = cv.height;
                ctx.save();
                ctx.beginPath();
                ctx.globalCompositeOperation = "source-over";
```

```
            ctx.fillStyle = "rgba(0,0,255,1)";
            ctx.arc(150, 150, 100, 0, 2 * Math.PI, false);
            ctx.fillRect(190, 80, 150, 150);
            ctx.fill();
            ctx.restore();
            ctx.fillStyle = "white";
            ctx.fillText("S\u222AD", 200, 150);
      </script>
</body>
</html>
```

Wynik

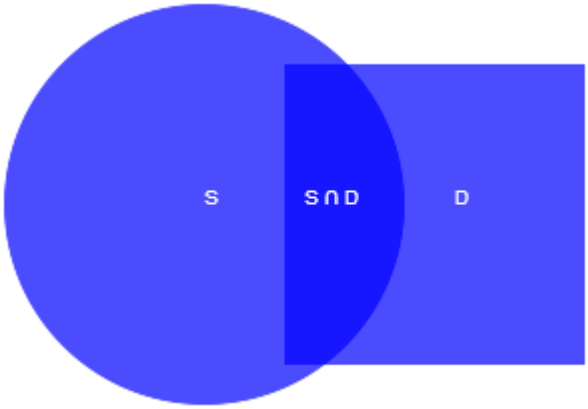


Listing06

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
            z kontekstem &#034;2d&#034;</canvas>
      <script type="text/javascript">
            var cv = document.getElementById('canvas');
            var ctx = cv.getContext('2d');
            var width = cv.width;
            var height = cv.height;
            ctx.save();
            ctx.beginPath();
            ctx.globalCompositeOperation = "source-over";
            ctx.fillStyle = "rgba(0,0,255,0.7)";
            ctx.arc(150, 150, 100, 0, 2 * Math.PI, false);
            ctx.fillRect(190, 80, 150, 150);
            ctx.fill();
            ctx.restore();
            ctx.fillStyle = "white";
            ctx.fillText("S \u2229 D", 200, 150);
            ctx.fillText("S", 150, 150);
            ctx.fillText("D", 275, 150);
      </script>
</body>
```

```
</html>
```

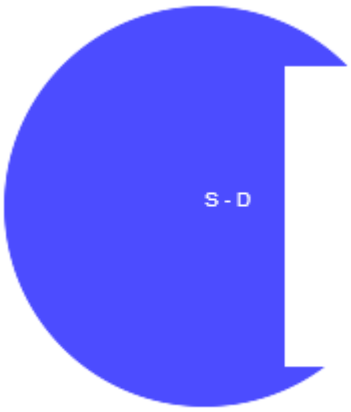
Wynik



Listing07

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		var width = cv.width;
		var height = cv.height;
		ctx.save();
		ctx.beginPath();
		//ctx.globalCompositeOperation="source-over";
		ctx.fillStyle = "rgba(0,0,255,0.7)";
		ctx.arc(150, 150, 100, 0, 2 * Math.PI, false);
		ctx.fill();
		ctx.fillStyle = "white";
		ctx.fillRect(190, 80, 150, 150);
		ctx.restore();
		ctx.fillStyle = "white";
		ctx.fillText("S - D", 150, 150);
	</script>
</body>
</html>
```

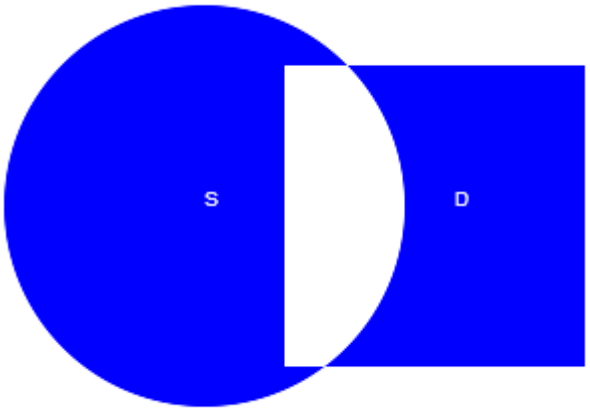
Wynik



Listing08

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		var width = cv.width;
		var height = cv.height;
		ctx.save();
		ctx.beginPath();
		ctx.globalCompositeOperation = "xor";
		ctx.fillStyle = "rgba(0,0,255,1)";
		ctx.arc(150, 150, 100, 0, 2 * Math.PI, false);
		ctx.fillRect(150, 80, 150, 150);
		ctx.fill();
		ctx.restore();
		ctx.fillStyle = "white";
		ctx.fillText("S \u2229 D", 200, 150);
		ctx.fillText("S", 150, 150);
		ctx.fillText("D", 275, 150);
	</script>
</body>
</html>
```

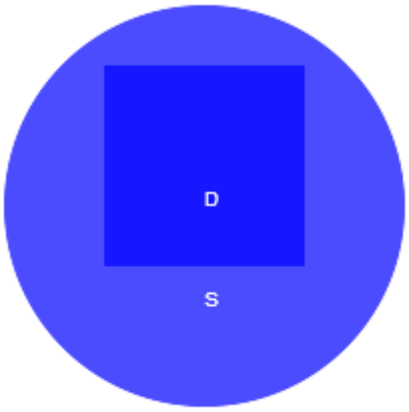
Wynik



Listing09

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		var width = cv.width;
		var height = cv.height;
		ctx.save();
		ctx.beginPath();
		//ctx.globalCompositeOperation="xor";
		ctx.fillStyle = "rgba(0,0,255,0.7)";
		ctx.arc(150, 150, 100, 0, 2 * Math.PI, false);
		ctx.fillRect(100, 80, 100, 100);
		ctx.fill();
		ctx.restore();
		ctx.fillStyle = "white";
		ctx.fillText("S", 150, 200);
		ctx.fillText("D", 150, 150);
	</script>
</body>
</html>
```

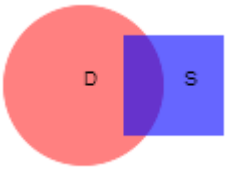
Wynik



Listing10

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		var width = cv.width;
		var height = cv.height;
		ctx.save();
		ctx.beginPath();
		ctx.fillStyle = "rgba(255,0,0,0.5)";//destination
		ctx.arc(150, 100, 40, 0, 2 * Math.PI, false);
		ctx.fill();
		ctx.globalCompositeOperation = "source-over";
		ctx.fillStyle = "rgba(0,0,255,0.6)"; //source
		ctx.fillRect(170, 75, 50, 50);
		ctx.restore();
		ctx.fillStyle = "black";
		ctx.fillText("D", 150, 100);
		//ctx.fillText("S \u2229 D",205, 150);
		ctx.fillText("S", 200, 100);
	</script>
</body>
</html>
```

Wynik



Listing11

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css>

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		var width = cv.width;
		var height = cv.height;
		ctx.save();
		ctx.beginPath();
		ctx.fillStyle = "rgba(51,0,153,0.8)"; //source
		ctx.fillRect(170, 75, 50, 50);
		ctx.restore();
	</script>
</body>
</html>
```

Wynik

Listing12

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/rachzb.js"></script>
	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		var width = cv.width;
		var height = cv.height;
		ctx.save();
		ctx.beginPath();
		var col1 = "rgba(255,0,0,0.5)";//destination
		var col2 = "rgba(0,0,255,0.6)"; //source
		ctx.fillStyle = kolor(col1, col2, "source-over");
		ctx.fillRect(270, 75, 50, 50);
```

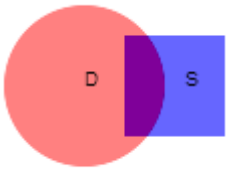
```
            ctx.fillText(ctx.fillStyle, 200, 170);
            ctx.restore();
      </script>
</body>
</html>
```

Wynik

Listing13

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
            z kontekstem &#034;2d&#034;</canvas>
      <script type="text/javascript">
      var cv = document.getElementById('canvas');
      var ctx = cv.getContext('2d');
      var width = cv.width;
      var height = cv.height;
      ctx.save();
      ctx.beginPath();
      ctx.fillStyle="rgba(255,0,0,0.5)";//destination
      ctx.arc(150, 100, 40,0,2*Math.PI, false);
      ctx.fill();
      ctx.globalCompositeOperation="lighter";
      ctx.fillStyle="rgba(0,0,255,0.6)";      //source
      ctx.fillRect(170,75, 50,50);
      ctx.restore();
      ctx.fillStyle="black";
      ctx.fillText("D", 150, 100);
      //ctx.fillText("S \u2229 D",205, 150);
      ctx.fillText("S", 200, 100);


      </script>
</body>
</html>
```

Wynik



Listing14

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		var width = cv.width;
		var height = cv.height;
		ctx.save();
		ctx.beginPath();
		ctx.fillStyle = "rgba(255,0,0,0.5)";//destination
		ctx.arc(150, 100, 40, 0, 2 * Math.PI, false);
		ctx.fill();
		ctx.globalCompositeOperation = "destination-atop";
		ctx.fillStyle = "rgba(0,0,255,0.6)"; //source
		ctx.fillRect(170, 75, 50, 50);
		ctx.restore();
		ctx.fillStyle = "black";
		ctx.fillText("D", 150, 100);
		//ctx.fillText("S \u2229 D",205, 150);
		ctx.fillText("S", 200, 100);
	</script>
</body>
</html>
```

Wynik

Listing`5

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
```

```
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		var width = cv.width;
		var height = cv.height;
		ctx.save();
		ctx.beginPath();
		ctx.fillStyle = "rgba(255,0,0,0.5)";//destination
		ctx.arc(150, 100, 40, 0, 2 * Math.PI, false);
		ctx.fill();
		ctx.globalCompositeOperation = "destination-in";
		ctx.fillStyle = "rgba(0,0,255,0.6)"; //source
		ctx.fillRect(170, 75, 50, 50);
		ctx.restore();
		ctx.fillStyle = "black";
		ctx.fillText("D", 150, 100);
		//ctx.fillText("S \u2229 D",205, 150);
		ctx.fillText("S", 200, 100);
	</script>
</body>
</html>
```

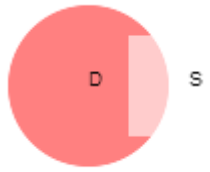
Wynik

Listing`6

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		var width = cv.width;
		var height = cv.height;
		ctx.save();
		ctx.beginPath();
		ctx.fillStyle = "rgba(255,0,0,0.5)";//destination
		ctx.arc(150, 100, 40, 0, 2 * Math.PI, false);
		ctx.fill();
		ctx.globalCompositeOperation = "destination-out";
		ctx.fillStyle = "rgba(0,0,255,0.6)"; //source
		ctx.fillRect(170, 75, 50, 50);
		ctx.restore();
		ctx.fillStyle = "black";
		ctx.fillText("D", 150, 100);
```

```
            //ctx.fillText("S \u2229 D",205, 150);
            ctx.fillText("S", 200, 100);
      </script>
</body>
</html>
```

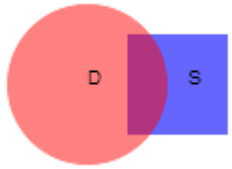
Wynik



Listing17

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
            z kontekstem &#034;2d&#034;</canvas>
      <script type="text/javascript">
      var cv = document.getElementById('canvas');
      var ctx = cv.getContext('2d');
      var width = cv.width;
      var height = cv.height;
      ctx.save();
      ctx.beginPath();
      ctx.fillStyle="rgba(255,0,0,0.5)";//destination
      ctx.arc(150, 100, 40,0,2*Math.PI, false);
      ctx.fill();
      ctx.globalCompositeOperation="destination-over";
      ctx.fillStyle="rgba(0,0,255,0.6)";      //source
      ctx.fillRect(170,75, 50,50);
      ctx.restore();
      ctx.fillStyle="black";
      ctx.fillText("D", 150, 100);
      //ctx.fillText("S \u2229 D",205, 150);
      ctx.fillText("S", 200, 100);


      </script>
</body>
</html>
```

Wynik



Listing18

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript">
	var cv = document.getElementById('canvas');
	var ctx = cv.getContext('2d');
	var width = cv.width;
	var height = cv.height;
	ctx.save();
	ctx.beginPath();
	ctx.fillStyle="rgba(255,0,0,0.5)";//destination
	ctx.arc(150, 100, 40,0,2*Math.PI, false);
	ctx.fill();
	ctx.globalCompositeOperation="copy";
	ctx.fillStyle="rgba(0,0,255,0.6)";	//source
	ctx.fillRect(170,75, 50,50);
	ctx.restore();
	ctx.fillStyle="black";
	ctx.fillText("D", 150, 100);
	//ctx.fillText("S \u2229 D",205, 150);
	ctx.fillText("S", 200, 100);


	</script>
</body>
</html>
```

Wynik

Listing19

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
```

```
            var width = cv.width;
            var height = cv.height;
            ctx.save();
            ctx.beginPath();
            ctx.fillStyle = "rgba(255,0,0,0.5)";//destination
            ctx.arc(150, 100, 40, 0, 2 * Math.PI, false);
            ctx.fill();
            ctx.globalCompositeOperation = "source-atop";
            ctx.fillStyle = "rgba(0,0,255,0.6)"; //source
            ctx.fillRect(170, 75, 50, 50);
            ctx.restore();
            ctx.fillStyle = "black";
            ctx.fillText("D", 150, 100);
            //ctx.fillText("S \u2229 D",205, 150);
            ctx.fillText("S", 200, 100);
      </script>
</body>
</html>
```

Wynik

Listing20

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
            z kontekstem &#034;2d&#034;</canvas>
      <script type="text/javascript">
            var cv = document.getElementById('canvas');
            var ctx = cv.getContext('2d');
            var width = cv.width;
            var height = cv.height;
            ctx.save();
            ctx.beginPath();
            ctx.fillStyle = "rgba(255,0,0,0.5)";//destination
            ctx.arc(150, 100, 40, 0, 2 * Math.PI, false);
            ctx.fill();
            ctx.globalCompositeOperation = "source-in";
            ctx.fillStyle = "rgba(0,0,255,0.6)"; //source
            ctx.fillRect(170, 75, 50, 50);
            ctx.restore();
            ctx.fillStyle = "black";
            ctx.fillText("D", 150, 100);
            //ctx.fillText("S \u2229 D",205, 150);
            ctx.fillText("S", 200, 100);
      </script>
</body>
</html>
```

Wynik

Listing21

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript">
	var cv = document.getElementById('canvas');
	var ctx = cv.getContext('2d');
	var width = cv.width;
	var height = cv.height;
	ctx.save();
	ctx.beginPath();
	ctx.fillStyle="rgba(255,0,0,0.5)";//destination
	ctx.arc(150, 100, 40,0,2*Math.PI, false);
	ctx.fill();
	ctx.globalCompositeOperation="source-out";
	ctx.fillStyle="rgba(0,0,255,0.6)";	//source
	ctx.fillRect(170,75, 50,50);
	ctx.restore();
	ctx.fillStyle="black";
	ctx.fillText("D", 150, 100);
	//ctx.fillText("S \u2229 D",205, 150);
	ctx.fillText("S", 200, 100);


	</script>
</body>
</html>
```

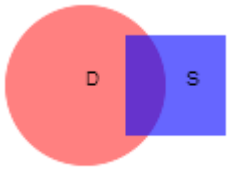
Wynik

Listing22

```
<!DOCTYPE html>
<html>
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		var width = cv.width;
		var height = cv.height;
		ctx.save();
		ctx.beginPath();
		ctx.fillStyle = "rgba(255,0,0,0.5)";//destination
		ctx.arc(150, 100, 40, 0, 2 * Math.PI, false);
		ctx.fill();
		ctx.globalCompositeOperation = "source-over";
		ctx.fillStyle = "rgba(0,0,255,0.6)"; //source
		ctx.fillRect(170, 75, 50, 50);
		ctx.restore();
		ctx.fillStyle = "black";
		ctx.fillText("D", 150, 100);
		//ctx.fillText("S \u2229 D",205, 150);
		ctx.fillText("S", 200, 100);
	</script>
</body>
</html>
```

Wynik

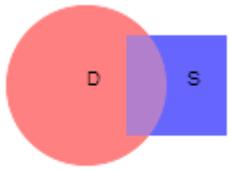


Listing23

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		var width = cv.width;
		var height = cv.height;
		ctx.save();
		ctx.beginPath();
		ctx.fillStyle = "rgba(255,0,0,0.5)";//destination
		ctx.arc(150, 100, 40, 0, 2 * Math.PI, false);
		ctx.fill();
```

```
            ctx.globalCompositeOperation = "xor";
            ctx.fillStyle = "rgba(0,0,255,0.6)"; //source
            ctx.fillRect(170, 75, 50, 50);
            ctx.restore();
            ctx.fillStyle = "black";
            ctx.fillText("D", 150, 100);
            //ctx.fillText("S \u2229 D",205, 150);
            ctx.fillText("S", 200, 100);
      </script>
</body>
</html>
```

Wynik



Zawartość skryptu rachzb.js

```
var kolor = function(rgba1, rgba2, rule) {
      var adst = rgba1.split(",");
      var asrc = rgba2.split(",");
      var src = parseFloat(asrc[3]);
      var dst = parseFloat(adst[3]);
      var fs = 0;
      var fd = 0;
      switch (rule) {
      case "copy":
            fs = 0;
            fd = 1.0;
            break;
      case "destination-atop":
            fs = 1.0 - dst;
            fd = src;
            break;
      case "destination-in":
            fs = 0;
            fd = src;
            break;
      case "destination-out":
            fs = 0;
            fd = 1.0 - src;
            break;
      case "destination-over":
            fs = 1.0 - dst;
            fd = 1.0;
            break;
      case "lighter":
            fs = 1.0;
            fd = 0;
            break;
      case "source-atop":
            fs = dst;
            fd = 1.0 - src;
            break;
      case "source-in":
            fs = dst;
            fd = 0;
            break;
```

```
        case "source-out":
              fs = 1.0 - dst;
              fd = 0;
              break;
        case "source-over":
              fs = 1.0;
              fd = 1.0 - src;
              break;
        case "xor":
              fs = 1.0 - dst;
              fd = 1.0 - src;
              break;
        }
        var a = fs * src + fd * dst;
        var r = fs * parseInt(asrc[0].substring(5)) * src + fd
                    * parseInt(adst[0].substring(5)) * dst;
        var g = fs * parseInt(asrc[1]) * src + fd * parseInt(adst[1]) * dst;
        var b = fs * parseInt(asrc[2]) * src + fd * parseInt(adst[2]) * dst;
        return "rgba(" + Math.round(r) + "," + Math.round(g) + "," + Math.round(b)
                    + "," + a + ")";

};
```

Listing24

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
        <canvas id="canvas" width="800" height="800">Zawartość możesz
zobaczyć w
                przeglądarce obsługującej element &lt;canvas&gt;
                z kontekstem &#034;2d&#034;</canvas>
        <script type="text/javascript" src="../scripts/arrow.js"></script>
        <script type="text/javascript" src="../scripts/axes.js"></script>
        <script type="text/javascript">
                drawAxes("cartesian");
        </script>
</body>
</html>
```

Zawartość skryptu arrow.js

```
var cv = document.getElementById('canvas');
var ctx = cv.getContext('2d');
/**
 * Tworzy obiekt typu Arrow
 *
 * @param endX
 *        float - współrzędna x tylnego końca strzałki
 * @param endY
 *        float - współrzędna y tylnego końca strzałki
 * @param arrowLength
 *        float - długość strzałki
```

```
 * @param arrowWidth
 *          float - grubość strzałki
 * @param arrowAngle
 *          float - kat pod ktorym jest skierowana strzalka katy podawane sa w
 *          stopniach, kat zerowy to polozenie godziny 3.00, katy wzrastaja w
 *          kierunku przeciwnym do wskazowek zegara
 * @param brudLength
 *          float - długość grota mierzona wzdłuż osi strzałki
 * @param brudAngle
 *          float - kąt między osią strzałki a brzegiem grota
 * @param close
 *          boolean -
 *          <ul>
 *          <li>true - grot bez wcięcia *
 *          <li>false - grot z wcięciem
 *          </ul>
 */
function drawArrow(endX, endY, arrowLength, arrowWidth, arrowAngle, brudLength,
                brudAngle, close, fillStyle) {
        this.endX = endX;
        this.endY = endY;
        this.arrowLength = arrowLength;
        this.arrowAngle = arrowAngle;
        this.brudLength = brudLength;
        this.brudAngle = brudAngle;
        this.close = close;
        this.fillStyle = fillStyle;
        var cosa = Math.cos(degToRad(arrowAngle));
        var sina = Math.sin(degToRad(arrowAngle));
        var half = arrowWidth / 2.0;
        var cosah = cosa * half;
        var sinah = sina * half;
        var startX = endX + arrowLength * Math.cos(degToRad(-arrowAngle));
        var startY = endY + arrowLength * Math.sin(degToRad(-arrowAngle));
        var middleX = endX + (arrowLength - brudLength)
                        * Math.cos(degToRad(-arrowAngle));// x7
        var middleY = endY + (arrowLength - brudLength)
                        * Math.sin(degToRad(-arrowAngle));// y7
        var cosb = Math.cos(degToRad(brudAngle));
        var sinb = Math.sin(degToRad(brudAngle));
        var w2 = half * cosb / sinb;
        var mmiddleX = endX + (arrowLength - w2) * Math.cos(degToRad(-arrowAngle));//
        var mmiddleY = endY + (arrowLength - w2) * Math.sin(degToRad(-arrowAngle));//
        var w3 = brudLength - w2;
        var w4 = w3 * sinb / cosb;
        var w5 = w4 + half;
        // -
        var x1 = endX - sinah;
        var y1 = endY - cosah;
        var x2 = endX + sinah;
```

```
        var y2 = endY + cosah;
        var x5 = middleX + sinah;
        var y5 = middleY + cosah;
        var x6 = middleX - sinah;
        var y6 = middleY - cosah;
        var x8 = mmiddleX + sinah;
        var y8 = mmiddleY + cosah;
        var x9 = mmiddleX - sinah;
        var y9 = mmiddleY - cosah;
        var x10 = middleX + sina * w5;
        var y10 = middleY + cosa * w5;
        var x11 = middleX - sina * w5;
        var y11 = middleY - cosa * w5;
        ctx.save();
        ctx.beginPath();
        ctx.fillStyle = this.fillStyle;
        ctx.strokeStyle = this.fillStyle;
        ctx.lineWidth = arrowWidth;
        if (this.close) {
                ctx.moveTo(x1, y1);
                ctx.lineTo(x2, y2);
                ctx.lineTo(x5, y5);
                ctx.lineTo(x10, y10);
                ctx.lineTo(x8, y8);
                ctx.lineTo(startX, startY);
                ctx.lineTo(x9, y9);
                ctx.lineTo(x11, y11);
                ctx.lineTo(x6, y6);
                ctx.closePath();
                ctx.fill();
                ctx.stroke();
        } else {
                ctx.moveTo(x1, y1);
                ctx.lineTo(x2, y2);
                ctx.lineTo(x8, y8);
                ctx.lineTo(startX, startY);
                ctx.lineTo(x9, y9);
                ctx.lineTo(x1, y1);
                ctx.fill();
                ctx.moveTo(x11, y11);
                ctx.lineTo(x9, y9);
                ctx.moveTo(x10, y10);
                ctx.lineTo(x8, y8);
                ctx.stroke();
        }
        ctx.restore();
};
/*
 * funkcja zamienia stopnie na radiany
 */
```

```
// k¹t deg podajemy w stopniach
var degToRad = function(deg) {
	return Math.PI * deg / 180.0;
};

/*
 * funkcja zamienia radiany na stopnie k¹t rad podajemy w radianach
 */
var radToDeg = function(rad) {
	return rad * 180.0 / Math.PI;
};
var arrowLength = function(x1, y1, x2, y2) {
	var distance = 0.0;
	var x3 = x1 - x2;
	var y3 = y1 - y2;
	distance = Math.sqrt(x3 * x3 + y3 * y3);
	return distance;
};
var atan2Deg = function(yy, xx) {
	return Math.atan2(yy, xx) * 180.0 / Math.PI;
};
var arrowAngle = function(x1, y1, x2, y2) {
	var x3 = x1 - x2;
	var y3 = y1 - y2;
	return atan2Deg(x3, y3);
};
```

Zawartość skryptu axes.js

```
var cv = document.getElementById('canvas');
var ctx = cv.getContext('2d');
var w = cv.width;
var h = cv.height;
var x0 = w / 2.0;
var y0 = h / 2.0;
var distx = x0 / 13;
var disty = y0 / 13;
var col = "black";
var col1 = "red";
var lw = 0.5;
var bw = 12;
var bh = 10;
var cl = true;
var dist = 30;
var m = 30;
var corr = 150;

function drawAxes(axes) {
	ctx.save();
	ctx.beginPath();
```

```
ctx.fillStyle = "black";
ctx.textBaseline = "middle";
ctx.textAlign = "center";
if (axes == "complex" || axes == "cartesian") {
	ctx.fillText("X", w - distx / 2, y0);
	ctx.fillText("-X", 0 + distx / 2, y0);
	ctx.fillText("Y", x0, 0 + disty / 2, y0);
	ctx.fillText("-Y", x0, h - disty / 2);
	for ( var i = 0; i < 12; i++) {
		ctx.fillText(i.toString(), x0 + i * m, y0 + 12);
	}
	for ( var k = 11; k > 0; k--) {
		ctx.fillText((-k).toString(), 99 + (10 - k) * m, y0 + 12);
	}
	if (axes == "complex") {
		ctx.fillStyle = "red";
		col1 = "red";
	}
	if (axes == "cartesian") {
		ctx.fillStyle = "black";
		col1 = "black";
	}
	for ( var j = 0; j < 12; j++) {
		ctx.fillText(j.toString(), x0 - 12, y0 - j * m);
	}
	for ( var n = 11; n > 0; n--) {
		ctx.fillText((-n).toString(), x0 - 12, y0 + n * m);
	}

	drawArrow(x0, y0, x0 - distx, lw, 0, bw, bh, cl, col);
	drawArrow(x0, y0, x0 - distx, lw, 180, bw, bh, cl, col);
	drawArrow(x0, y0, y0 - disty, lw, 90, bw, bh, cl, col1);
	drawArrow(x0, y0, y0 - disty, lw, 270, bw, bh, cl, col1);
}
if (axes == "js") {

	ctx.fillText("X", w - 12, 15);
	ctx.fillText("Y", 15, h - 12);
	for ( var i = 0; i < 24; i++) {
		ctx.fillText(i.toString(), 15 + i * m, 8);
	}

	for ( var j = 23; j > 0; j--) {
		ctx.fillText(j.toString(), 7, 15 + j * m);
	}
	drawArrow(15, 15, w - 40, lw, 0, bw, bh, cl, col);
	drawArrow(15, 15, w - 40, lw, 270, bw, bh, cl, col);

}
ctx.restore();
```

```
};

function drawAxesG(axes, gameMatrix) {
	var min = gameMatrix.gameMin();
	var max = gameMatrix.gameMax();
	var diff = Math.abs(min) + Math.abs(max);
	diff = Math.ceil(diff / 23.0);
	var y5 = 400;
	ctx.save();
	ctx.beginPath();
	ctx.fillStyle = "black";
	ctx.textBaseline = "middle";
	ctx.textAlign = "center";
	if (axes == "games") {
		// left
		for ( var n = 11; n > -12; n--) {
			ctx.fillText((-n * diff).toString(), x0 - 12 - corr, y5 + n * m);
		}
		drawArrow(x0 - corr, y0, y0 - disty, lw, 90, bw, bh, cl, col);
		drawArrow(x0 - corr, y0, y0 - disty, lw, 270, bw, bh, cl, col);
		// right
		for ( var n = 11; n > -12; n--) {
			ctx.fillText((-n * diff).toString(), x0 + 12 + corr, y5 + n * m);
		}
		drawArrow(x0 + corr, y0, y0 - disty, lw, 90, bw, bh, cl, col);
		drawArrow(x0 + corr, y0, y0 - disty, lw, 270, bw, bh, cl, col);
	}
	if (gameMatrix.rows == 2) {
		ctx.beginPath();
		for ( var i = 0; i < gameMatrix.cols; i++) {
			ctx.moveTo(x0 - corr, y5 - ((gameMatrix.array[0][i]) / diff) * m);
			ctx.lineTo(x0 + corr, y5 - ((gameMatrix.array[1][i]) / diff) * m);
		}
		ctx.stroke();
	} else if (gameMatrix.cols == 2) {
		ctx.beginPath();
		for ( var i = 0; i < gameMatrix.rows; i++) {
			ctx.moveTo(x0 - corr, y5 - ((gameMatrix.array[i][0]) / diff) * m);
			ctx.lineTo(x0 + corr, y5 - ((gameMatrix.array[i][1]) / diff) * m);
		}
		ctx.stroke();
	}
	ctx.restore();
};
```

Listing25

```
<!DOCTYPE html>
<html>
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
        <canvas id="canvas" width="800" height="800">Zawartość możesz
zobaczyć w
                przeglądarce obsługującej element &lt;canvas&gt;
                z kontekstem &#034;2d&#034;</canvas>
        <script type="text/javascript" src="../scripts/arrow.js"></script>
        <script type="text/javascript" src="../scripts/axes.js"></script>
        <script type="text/javascript">
                drawAxes("js");
        </script>
</body>
</html>
```

Listing26

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href=".//styles/canvas2d.css">

<title> </title>
</head>
<body>
        <canvas id="canvas" width="800" height="800">Zawartość możesz
zobaczyć w
                przeglądarce obsługującej element &lt;canvas&gt;
                z kontekstem &#034;2d&#034;</canvas>
        <script type="text/javascript" src="../scripts/arrow.js"></script>
        <script type="text/javascript" src="../scripts/axes.js"></script>
        <script type="text/javascript" src="../scripts/line.js"></script>

        <script type="text/javascript">
                var cv = document.getElementById('canvas');
                var ctx = cv.getContext('2d');
                drawAxes("cartesian");
                drawVectorf(4.0, 3.0, 8.0, 5.0, "cartesian", "green");
        </script>
</body>
</html>
```

Zawartość skryptu line.js

```
var cv = document.getElementById('canvas');
var ctx = cv.getContext('2d');
var w = cv.width;
var h = cv.height;
var x0 = w / 2.0;
var y0 = h / 2.0;
var lw = 0.5;

/*
```

```
 * Wektor swobodny. Jesli nie podano zadnych argumentow tworzony jest wektor
 * zerowy. Jezli podano 2 argumenty musza to byc wspolrzedne x i y punktu
 */
var Vector2f = function(x, y) {
	switch (arguments.length) {
	case 0:
		this.x = 0.0;
		this.y = 0.0;
		break;
	case 2:
		this.x = arguments[0];
		this.y = arguments[1];
		break;
	}
};
/*
 * Oblicza odleglosc punktu podanego jako Vector2f lub w postaci (x,y) od tego
 * wektora
 */
Vector2f.prototype.distance = function() {
	var distance = 0.0;
	switch (arguments.length) {
	case 1:
		if (arguments[0] instanceof Vector2f) {
			var x = this.x - arguments[0].x;
			var y = this.y - arguments[0].y;
			distance = Math.sqrt(x * x + y * y);
		}
		break;
	case 2:
		var x1 = this.x - arguments[0];
		var y1 = this.y - arguments[0];
		distance = Math.sqrt(x1 * x1 + y1 * y1);
		break;
	}

	return distance;
};
/*
 * Vektor pomocniczy przyjmujący parametry rownania prostej Ax + By + C = 0 jako
 * a,b,c
 */
var Vector3f = function() {
	switch (arguments.length) {
	case 0:
		this.a = 0.0;
		this.b = 0.0;
		this.c = 0.0;
		break;
	case 3:
```

```
                this.a = arguments[0];
                this.b = arguments[1];
                this.c = arguments[2];
                break;
        }
};
Vector2f.prototype.toString = function() {
        return "V=[" + this.x + ", " + this.y + "]";
};

var atan2Deg = function(yy, xx) {
        return Math.atan2(yy, xx) * 180.0 / Math.PI;
};
function drawVectorf(x1, y1, x2, y2, axes, fillStyle) {
        ctx.save();
        ctx.beginPath();
        ctx.fillStyle = fillStyle;
        var radius = 0;
        var angle = 0;
        var vect = new Vector2f(x2 - x1, y2 - y1);
        radius = Math.sqrt(Math.pow(vect.y, 2) + Math.pow(vect.x, 2));
        angle = atan2Deg(vect.y, vect.x);

        if (axes == "complex" || axes == "cartesian") {
                drawArrow(x0 + x1 * 30, y0 - y1 * 30, radius * 30, lw, angle, bw, bh,
                                cl, fillStyle);

        }
        if (axes == "js") {
                drawArrow(x1 + 15, y1 + 15, radius * 30, lw, -angle, bw, bh, cl,
                                fillStyle);

        }
        if (axes == "norm") {
                drawArrow(x1, y1, arrowLength(x1, y1, x2, y2), lw, -arrowAngle(x1, y1,
                                x2, y2), bw, bh, cl, fillStyle);
        }
        ctx.restore();
};

/*
 * Tworzymy linie używając dwóch wektorów lub 4 wspolrzednych x1,y1,x2,y2,
 * punktow przez ktore przechodzi linia
 */

var Line = function() {
        switch (arguments.length) {
        case 2:
                if ((arguments[0] instanceof Vector2f)
                                && (arguments[1] instanceof Vector2f)) {
```

```
                this.start = arguments[0];
                this.end = arguments[1];
                this.length = this.start.distance(this.end);
                this.cosinus = (this.end.x - this.start.x) / this.length;
                this.sinus = (this.end.y - this.start.y) / this.length;
            }
            break;
        case 4:
            this.start = new Vector2f(arguments[0], arguments[1]);
            this.end = new Vector2f(arguments[2], arguments[3]);
            this.length = this.start.distance(this.end);
            this.cosinus = (arguments[0] - arguments[2]) / this.length;
            this.sinus = (arguments[1] - arguments[3]) / this.length;
            break;
        }
};
// mozna podac:
// 1. axes, fillStyle, Line
// 2. axes, fillStyle, Vector2f, Vector2f
// 3. axes, fillStyle, x1,y1,x2, y2
function drawLine() {
        ctx.save();
        ctx.beginPath();
        var x1 = 0.0;
        var y1 = 0.0;
        var x2 = 0.0;
        var y2 = 0.0;
        var axes = arguments[0];
        ctx.strokeStyle = arguments[1];
        switch (arguments.length) {
        case 3:
            if (arguments[2] instanceof Line) {
                x1 = arguments[2].start.x;
                y1 = arguments[2].start.y;
                x2 = arguments[2].end.x;
                y2 = arguments[2].end.y;
            }
            break;
        case 4:
            if (arguments[2] instanceof Vector2f
                    && arguments[3] instanceof Vector2f) {
                x1 = arguments[2].x;
                y1 = arguments[2].y;
                x2 = arguments[3].x;
                y2 = arguments[3].y;
            }
            break;
        case 6:
            x1 = arguments[2];
            y1 = arguments[3];
```

```
                x2 = arguments[4];
                y2 = arguments[5];
                break;
        }
        if (axes == "complex" || axes == "cartesian") {
                ctx.moveTo(x0 + x1 * 30, y0 - y1 * 30);
                ctx.lineTo(x0 + x2 * 30, y0 - y2 * 30);
                ctx.stroke();

        }
        if (axes == "js") {
                ctx.moveTo(15 + x1 * 30, 15 + y1 * 30);
                ctx.lineTo(15 + x2 * 30, 15 + y2 * 30);
                ctx.stroke();

        }
        if (axes = "norm") {
                ctx.moveTo(x1, y1);
                ctx.lineTo(x2, y2);
                ctx.stroke();
        }
        ctx.restore();
};
/**
 * Oblicza współczynnik nachylenia tej linii
 *
 * @return zwraca współczynnik nachylenia linii
 */
Line.prototype.slope = function() {
        var xx1 = this.start.x - this.end.x;
        var yy1 = this.start.y - this.end.y;
        if (xx1 != 0.0) {
                return yy1 / xx1;
        } else {
                throw new Error("Nie moge dzielic przez 0");
        }
};

/**
 * Oblicza wskazaną liczbę punktów na tej linii
 *
 * @param liczbaPunktow
 *          int - liczba obliczanych punktów point1, point 2, wektory 2f
 *          zwraca tablice Vector2f;
 */
var pointsOnLine = function(nPoints, point1, point2) {
        var tabl = new Array(nPoints);
        var delta = 1.0 / (nPoints - 1.0);
        var point = null;
        var ax = point1.x;
```

```
		var ay = point1.y;
		var bx = point2.x;
		var by = point2.y;
		for ( var i = 0; i < nPoints; i++) {
			var t = i * delta;
			var tt = 1.0 - t;
			var x = t * bx + tt * ax;
			var y = t * by + tt * ay;
			point = new Vector2f(x, y);
			tabl[i] = point;
		}
		return tabl;
};
var pointsOnLine = function(nPoints, line) {
		var tabl = new Array(nPoints);
		var delta = 1.0 / (nPoints - 1.0);
		var point = null;
		var ax = line.start.x;
		var ay = line.start.y;
		var bx = line.end.x;
		var by = line.end.y;
		for ( var i = 0; i < nPoints; i++) {
			var t = i * delta;
			var tt = 1.0 - t;
			var x = t * bx + tt * ax;
			var y = t * by + tt * ay;
			point = new Vector2f(x, y);
			tabl[i] = point;
		}
		return tabl;
};

Line.prototype.toString = function() {
		return "[" + "(" + this.start.x + ", " + this.start.y + ")" + ";" + "("
				+ this.end.x + ", " + this.end.y + ")" + "]";
};

/**
 * Oblicza współczynnik nachylenia linii przechodzącej przez dwa dane punkty
 *
 * @param point1
 *		Vector2d - pierwszy punkt
 * @param point2
 *		Vector2d - drugi punkt
 * @return double - zwraca współczynnik nachylenia linii
 */
var slope = function(point1, point2) {
		var xx2 = 0.0;
		var yy2 = 0.0;
		if (point1 instanceof Vector2f && point2 instanceof Vector2f) {
```

```
                xx2 = point1.x - point2.x;
                yy2 = point1.y - point2.y;
        }
        if (xx2 != 0.0) {
                return yy2 / xx2;
        } else {
                throw new Error("linia nie może być równoległa do osi Y!");

        }
};

/**
 * Sprawdza czy dwie wskazane linie przecinają się
 *
 * @param line1
 *          Line - pierwsza linia
 * @param line2
 *          Line - druga linia
 * @return boolean - zwraca <code>true</code> jeśli linie przecinają się albo
 *        <code>false</code> jeśli sa rownolegle
 */
var areIntersect = function(line1, line2) {
        var m1 = line1.slope();
        var m2 = line2.slope();
        return (m1 != m2);
};
/*
 * Sprawdza CZY LINIE SA PROSTOPADLE
 */
var areNormal = function(line1, line2) {
        var m1 = line1.slope();
        var m2 = line2.slope();
        return m1 * m2 == -1;
};
/**
 * Zwraca współczynniki równania linii w postaci Vectora3d
 *
 * @param line
 *          Line - badana linia
 * @return Vector3D - zawierający współczynniki równania liniowego
 *        Vector3d.getX() = a, Vector3d.getY() = b, Vector3d.getZ() = c
 *        otrzymujemy równanie linii AX + BY + C = 0;
 */
var findEquation = function(line) {
        var m = line.slope();
        var A = -m;
        var B = 1;
        var C = m * line.start.x - line.start.y;
        return (new Vector3f(A, B, C));
};
```

```
var findb = function(line) {
	var m = line.slope();
	var C = m * line.start.x - line.start.y;
	return -C;
};
/**
 * Oblicza i zwraca punkt przecięcia dwóch linii podanych jako obiekty typu Line
 *
 * @param line1
 *		Line - linia 1
 * @param line2
 *		Line - linia 2
 * @return Vector2d - zwraca punkt przecięcia wskazanych linii lub
 *	<code>null</code> jeśli punkt przecięcia nie istnieje
 */
var interceptPointL = function(line1, line2) {
	var xx = 0;
	var yy = 0;
	if (areIntersect(line1, line2)) {
		var v1 = findEquation(line1);
		var v2 = findEquation(line2);
		xx = (v2.b * v1.c - v1.b * v2.c) / (v2.a * v1.b - v1.a * v2.b);
		yy = (-v2.c - v2.a * xx) / v2.b;
	} else {
		return null;
	}
	return (new Vector2f(xx, yy));
};

/**
 * Tworzy obiekt typu Line z równania linii, jeśli równanie linii wyrażone jest
 * w postaci aX + bY + c;
 *
 * @param a
 *		double - wspólczynnik a równania linii;
 * @param b
 *		double - współczynnik b równania linii
 * @param c
 *		double - współczynnik c równania linii
 * @return Line - obiekt linii wyrażonej równaniem o podanych wspólczynnikach
 */
var findLine = function(a, b, c) {
	var x1 = 0;
	var x2 = 3;
	var y1 = -c / b - a / b * x1;
	var y2 = -c / b - a / b * x2;
	return new Line(new Vector2f(x1, y1), new Vector2f(x2, y2));
};

/**
```

```
 * Oblicza i zwraca punkt przecięcia się dwóch linii podanych w postaci Vectora.
 * Jeżeli równanie linii jest wyrażone jako aX + bY + c = 0, to parametrem
 * równania będzie new Vector(a,b,c)
 *
 * @param v1
 *          Vector3d - wektor 1
 * @param v2
 *          Vector3d - wektor 2
 * @return Vector2d - zwraca punkt przecięcia, jeśli istnieje lub
 *       <code>null</code> jeśli nie istnieje
 */
var interceptPointE = function(v1, v2) {
        var xx = 0;
        var yy = 0;
        var line1 = findLine(v1.a, v1.b, v1.c);
        var line2 = findLine(v2.a, v2.b, v2.c);
        if (areIntersect(line1, line2)) {
                xx = (v2.b * v1.c - v1.b * v2.c) / (v2.a * v1.b - v1.a * v2.b);
                yy = (-v2.c - v2.a * xx) / v2.b;
        } else {
                return null;
        }
        return (new Vector2f(xx, yy));
};
/*
 * Zwraca odleglosc pomiedzy punktami okreslonymi albo jako wektory swobodne
 * albo wspolrzedne x1,x2,y1,y2
 */
var distance = function() {
        var distance = 0.0;
        switch (arguments.length) {
        case 2:
                if (argument[0] instanceof Vector2f && arguments[1] instanceof Vector2f) {
                        var x = arguments[0].x - arguments[1].x;
                        var y = arguments[0].y - arguments[1].y;
                        distance = Math.sqrt(x * x + y * y);
                }
                break;

        case 4:
                var x1 = arguments[0] - arguments[2];
                var y1 = arguments[1] - arguments[3];
                distance = Math.sqrt(x1 * x1 + y1 * y1);
                break;
        }
        return distance;

};
var distance2 = function(line1, line2) {
        var v1 = findEquation(line1);
```

```
	var v2 = findEquation(line2);
	var dist = Math.abs(v2.c - v1.c) / Math.sqrt(v1.a * v1.a + v1.b * v1.b);
	return dist;
};
var distance3 = function(line, point) {
	var v = null;
	var d = 0.0;
	if ((line instanceof Line) && (point instanceof Vector2f)) {
		v = findEquation(line);
		d = (v.a * point.x + v.b * point.y + v.c)
				/ (Math.sqrt(v.a * v.a + v.b * v.b));
	}
	return d;
};
// Zwraca kat w stopniach
var angleBetween = function(line1, line2) {
	var m1 = line1.slope();
	var m2 = line2.slope();
	var angle = (m2 - m1) / (1 + m1 * m2);
	return atanDeg(angle);
};
var radToDeg = function(rad) {
	return rad * 180.0 / Math.PI;
};
var atanDeg = function(ratio) {
	return Math.atan(ratio) * 180.0 / Math.PI;

};
var lineThruPoint = function(line, point) {
	var line1 = null;
	if ((line instanceof Line) && (point instanceof Vector2f)) {
		var v3f = findEquation(line);
		line1 = findLine(v3f.b, -v3f.a, v3f.a * point.y - v3f.b * point.x);
	}
	return line1;
};
var lineParallelThruPoint = function(line, point) {
	var line1 = null;
	if ((line instanceof Line) && (point instanceof Vector2f)) {
		var v3f = findEquation(line);
		line1 = findLine(v3f.b, v3f.a, v3f.a * point.y, v3f.b * point.x);
	}
	return line1;
};
```

Listing27

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

```
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
        <canvas id="canvas" width="800" height="800">Zawartość możesz
zobaczyć w
                przeglądarce obsługującej element &lt;canvas&gt;
                z kontekstem &#034;2d&#034;</canvas>
        <script type="text/javascript" src="../scripts/arrow.js"></script>
        <script type="text/javascript" src="../scripts/axes.js"></script>
        <script type="text/javascript" src="../scripts/line.js"></script>

        <script type="text/javascript">
                var cv = document.getElementById('canvas');
                var ctx = cv.getContext('2d');
                drawAxes("cartesian");
                drawVectorf(4.0, 3.0, 8.0, 5.0, "cartesian", "green");
                ctx.beginPath();
                ctx.lineWidth = 1;
                ctx.moveTo(400, 400 - 6 * 30);
                ctx.lineTo(400 + 3 * 30, 400);
                ctx.stroke();
        </script>
</body>
</html>
```

Listing28

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
        <canvas id="canvas" width="800" height="800">Zawartość możesz
zobaczyć w
                przeglądarce obsługującej element &lt;canvas&gt;
                z kontekstem &#034;2d&#034;</canvas>
        <script type="text/javascript" src="../scripts/arrow.js"></script>
        <script type="text/javascript" src="../scripts/axes.js"></script>
        <script type="text/javascript" src="../scripts/line.js"></script>

        <script type="text/javascript">
                var cv = document.getElementById('canvas');
                var ctx = cv.getContext('2d');
                drawAxes("cartesian");
                ctx.beginPath();
                ctx.lineWidth = 1;
                ctx.moveTo(400, 400 - 6 * 30);
                ctx.lineTo(400 + 3 * 30, 400);
                ctx.stroke();
                ctx.fillText("b", 402, 312);
                ctx.fillText("a", 432, 396);
                ctx.fillText("\u03B1", 492, 396);
                ctx.fillText("\u03B2", 476, 396);
                ctx.stroke();
```

```
        </script>
</body>
</html>
```

Listing29

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
        <canvas id="canvas" width="800" height="800">Zawartość możesz
zobaczyć w
                przeglądarce obsługującej element &lt;canvas&gt;
                z kontekstem &#034;2d&#034;</canvas>
        <script type="text/javascript" src="../scripts/arrow.js"></script>
        <script type="text/javascript" src="../scripts/axes.js"></script>
        <script type="text/javascript" src="../scripts/line.js"></script>

        <script type="text/javascript">
                var cv = document.getElementById('canvas');
                var ctx = cv.getContext('2d');
                drawAxes("cartesian");
                var line = new Line(2, 4, 4, 7);
                drawLine("cartesian", "black", line);
                ctx.fillText(line.slope(), 40, 40)
                ctx.fillText(slope(new Vector2f(2, 4), new Vector2f(4, 7)),
40, 60);
        </script>
</body>
</html>
```

Listing30

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
        <canvas id="canvas" width="800" height="800">Zawartość możesz
zobaczyć w
                przeglądarce obsługującej element &lt;canvas&gt;
                z kontekstem &#034;2d&#034;</canvas>
        <script type="text/javascript" src="../scripts/arrow.js"></script>
        <script type="text/javascript" src="../scripts/axes.js"></script>
        <script type="text/javascript" src="../scripts/line.js"></script>

        <script type="text/javascript">
                var cv = document.getElementById('canvas');
                var ctx = cv.getContext('2d');
                drawAxes("cartesian");
                var line = new Line(2, 4, 4, 7);
```

```
                drawLine("cartesian", "black", line);
                var vect = findEquation(line);
                ctx.fillText("" + vect.a + "x + " + vect.b + "y " + vect.c +
" = 0",
                                440, 140)
        </script>
</body>
</html>
```

Listing31

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
        <canvas id="canvas" width="800" height="800">Zawartość możesz
zobaczyć w
                przeglądarce obsługującej element &lt;canvas&gt;
                z kontekstem &#034;2d&#034;</canvas>
        <script type="text/javascript" src="../scripts/arrow.js"></script>
        <script type="text/javascript" src="../scripts/axes.js"></script>
        <script type="text/javascript" src="../scripts/line.js"></script>

        <script type="text/javascript">
                var cv = document.getElementById('canvas');
                var ctx = cv.getContext('2d');
                drawAxes("cartesian");
                var line = findLine(-1.5, 1, -1);
                drawLine("cartesian", "black", line);
                ctx.fillText(line.slope(), 440, 140)
                var vect = findEquation(line);
                ctx.fillText("" + vect.a + "x + " + vect.b + "y " + vect.c +
" = 0",
                                440, 160)
        </script>
</body>
</html>
```

Listing32

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
        <canvas id="canvas" width="800" height="800">Zawartość możesz
zobaczyć w
                przeglądarce obsługującej element &lt;canvas&gt;
                z kontekstem &#034;2d&#034;</canvas>
        <script type="text/javascript" src="../scripts/arrow.js"></script>
        <script type="text/javascript" src="../scripts/axes.js"></script>
```

```
        <script type="text/javascript" src="../scripts/line.js"></script>

        <script type="text/javascript">
                var cv = document.getElementById('canvas');
                var ctx = cv.getContext('2d');
                drawAxes("cartesian");
                var line = findLine(-1.5, 1, -1);
                drawLine("cartesian", "black", line);
                ctx.fillText(line.slope(), 440, 140)
                var line1 = findLine(1.5, -1, 2);
                drawLine("cartesian", "black", line1);
                ctx.fillText(line1.slope(), 440, 160);
                ctx.fillText("przecinają się: " + areIntersect(line, line1),
440, 180);
        </script>
</body>
</html>
```

Listing33

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
        <canvas id="canvas" width="800" height="800">Zawartość możesz
zobaczyć w
                przeglądarce obsługującej element &lt;canvas&gt;
                z kontekstem &#034;2d&#034;</canvas>
        <script type="text/javascript" src="../scripts/arrow.js"></script>
        <script type="text/javascript" src="../scripts/axes.js"></script>
        <script type="text/javascript" src="../scripts/line.js"></script>

        <script type="text/javascript">
                var cv = document.getElementById('canvas');
                var ctx = cv.getContext('2d');
                drawAxes("cartesian");
                var line = findLine(-1.5, 1, -1);
                drawLine("cartesian", "black", line);
                var line1 = findLine(1.5, -1, 2);
                drawLine("cartesian", "black", line1);
                ctx.fillText("odległość :" + distance2(line, line1), 440,
160);
        </script>
</body>
</html>
```

Listing34

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
```

```
</head>
<body>
        <canvas id="canvas" width="800" height="800">Zawartość możesz
zobaczyć w
                przeglądarce obsługującej element &lt;canvas&gt;
                z kontekstem &#034;2d&#034;</canvas>
        <script type="text/javascript" src="../scripts/arrow.js"></script>
        <script type="text/javascript" src="../scripts/axes.js"></script>
        <script type="text/javascript" src="../scripts/line.js"></script>

        <script type="text/javascript">
                var cv = document.getElementById('canvas');
                var ctx = cv.getContext('2d');
                drawAxes("cartesian");
                var line = findLine(-1.5, 1, -1);
                drawLine("cartesian", "black", line);
                ctx.fillText(line.slope(), 440, 140);
                var line1 = findLine(1, 1.5, 2);
                drawLine("cartesian", "black", line1);
                ctx.fillText(line1.slope(), 440, 160);
                ctx.fillText(areNormal(line, line1), 440, 180);
        </script>
</body>
</html>
```

Listing35

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
        <canvas id="canvas" width="800" height="800">Zawartość możesz
zobaczyć w
                przeglądarce obsługującej element &lt;canvas&gt;
                z kontekstem &#034;2d&#034;</canvas>
        <script type="text/javascript" src="../scripts/arrow.js"></script>
        <script type="text/javascript" src="../scripts/axes.js"></script>
        <script type="text/javascript" src="../scripts/line.js"></script>

        <script type="text/javascript">
                var cv = document.getElementById('canvas');
                var ctx = cv.getContext('2d');
                drawAxes("cartesian");
                var line = findLine(2.5, 1, -1);
                drawLine("cartesian", "black", line);
                var line1 = findLine(3, 6, 5);
                drawLine("cartesian", "black", line1);
                var angle = angleBetween(line, line1);
                ctx.fillText(angle, 440, 180);
        </script>
</body>
</html>
```

Listing36

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz
zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/arrow.js"></script>
	<script type="text/javascript" src="../scripts/axes.js"></script>
	<script type="text/javascript" src="../scripts/line.js"></script>

	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		drawAxes("cartesian");
		var line = findLine(2.5, 1, -1);
		drawLine("cartesian", "black", line);
		var line1 = findLine(3, 6, 5);
		drawLine("cartesian", "black", line1);
		var v = interceptPointL(line, line1);
		ctx.fillText(v.x + " " + v.y, 440, 180);
	</script>
</body>
</html>
```

Listing37

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz
zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/arrow.js"></script>
	<script type="text/javascript" src="../scripts/axes.js"></script>
	<script type="text/javascript" src="../scripts/line.js"></script>

	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		drawAxes("cartesian");
		var vect1 = new Vector3f(2.5, 1, -1);
		var vect2 = new Vector3f(3, 6, 5);
		var v = interceptPointE(vect1, vect2);
		ctx.fillText(v.x + " " + v.y, 440, 180);
	</script>
</body>
```

```
</html>
```

Listing38

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
        <canvas id="canvas" width="800" height="800">Zawartość możesz
zobaczyć w
                przeglądarce obsługującej element &lt;canvas&gt;
                z kontekstem &#034;2d&#034;</canvas>
        <script type="text/javascript" src="../scripts/arrow.js"></script>
        <script type="text/javascript" src="../scripts/axes.js"></script>
        <script type="text/javascript" src="../scripts/line.js"></script>

        <script type="text/javascript">
                var cv = document.getElementById('canvas');
                var ctx = cv.getContext('2d');
                drawAxes("cartesian");
                var line = findLine(2.5, 1, -1);
                drawLine("cartesian", "black", line);
                var point = new Vector2f(7, 7);
                var dist = distance3(line, point);
                ctx.fillText(dist, 440, 180);
        </script>
</body>
</html>
```

Listing39

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
        <canvas id="canvas" width="800" height="800">Zawartość możesz
zobaczyć w
                przeglądarce obsługującej element &lt;canvas&gt;
                z kontekstem &#034;2d&#034;</canvas>
        <script type="text/javascript" src="../scripts/arrow.js"></script>
        <script type="text/javascript" src="../scripts/axes.js"></script>
        <script type="text/javascript" src="../scripts/line.js"></script>

        <script type="text/javascript">
                var cv = document.getElementById('canvas');
                var ctx = cv.getContext('2d');
                drawAxes("cartesian");
                var line = new Line(-3, 8, 4, -4);
                drawLine("cartesian", "black", line);
                var point = new Vector2f(7, 7);
```

```
                var line1 = lineThruPoint(line, point);
                drawLine("cartesian", "red", line1);
                var vect = findEquation(line1);
                ctx.fillText(vect.a + " " + vect.b + " " + vect.c, 440,
180);
        </script>
</body>
</html
```

Listing40

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
        <canvas id="canvas" width="800" height="800">Zawartość możesz
zobaczyć w
                przeglądarce obsługującej element &lt;canvas&gt;
                z kontekstem &#034;2d&#034;</canvas>
        <script type="text/javascript" src="../scripts/arrow.js"></script>
        <script type="text/javascript" src="../scripts/axes.js"></script>
        <script type="text/javascript" src="../scripts/line.js"></script>

        <script type="text/javascript">
                var cv = document.getElementById('canvas');
                var ctx = cv.getContext('2d');
                drawAxes("cartesian");
                var line = new Line(-3, 8, 4, -4);
                drawLine("cartesian", "black", line);
                var points = pointsOnLine(10, line);
                for ( var i = 0; i < points.length; i++) {
                        ctx.fillText(points[i], 440, 140 + i * 20);

                }
        </script>
</body>
</html>
```

Listing41

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
            z kontekstem &#034;2d&#034;</canvas>
      <script type="text/javascript" src="../scripts/arrow.js"></script>
      <script type="text/javascript" src="../scripts/axes.js"></script>
```

```
        <script type="text/javascript" src="../scripts/linear.js"></script>

        <script type="text/javascript">
                drawAxes("cartesian");
                drawVector(new Vector2d(7.5, 3.5), "cartesian", "black");
        </script>
</body>

</html>
```

Zawartość skryptu linear.js

```
var w = cv.width;
var h = cv.height;
var x0 = w / 2.0;
var y0 = h / 2.0;
var lw = 0.5;
var bw = 12;
var bh = 10;
var cl = true;

var Vector2d = function(x, y) {
        this.x = x;
        this.y = y;
};
var Vector3d = function(x, y, z) {
        this.x = x;
        this.y = y;
        this.z = z;
};
Vector2d.prototype.toString = function() {
        return "V=[" + this.x + ", " + this.y + "]";
};
Vector3d.prototype.toString = function() {
        return "V=[" + this.x + ", " + this.y + ", " + this.z + "]";
};
Vector2d.prototype.equals = function(vect) {
        if (vect instanceof Vector2d) {
                return this.x == vect.x && this.y == vect.y;
        }
        return false;
};
Vector3d.prototype.equals = function(vect) {
        if (vect instanceof Vector3d) {
                return this.x == vect.x && this.y == vect.y && this.z == vect.z;
        }
        return false;
};
// dodaje podany wektor do tego wektora. Nie jest tworzony nowy wektor.
Vector2d.prototype.add = function(vector) {
        if (vector instanceof Vector2d) {
                this.x += vector.x;
                this.y += vector.y;
        }
};

// dodanie dwoch wektorow
Vector3d.prototype.add = function(vect) {
```

```
        if (vect instanceof Vector3d) {
                this.x += vect.x;
                this.y += vect.y;
                this.z += vect.z;
        }
};

// dodaje ten wektor do podanego wektora. Zwraca nowy wektor
Vector2d.prototype.addN = function(vector) {
        if (vector instanceof Vector2d) {
                var xx = vector.x + this.x;
                var yy = vector.y + this.y;
                return (new Vector2d(xx, yy));
        }
        return null;
};
// dodanie dwoch wektorow z utworzeniem nowego
Vector3d.prototype.addN = function(vect) {
        if (vect instanceof Vector3d) {
        }
        var xx = vect.x + this.x;
        var yy = vect.y + this.y;
        var zz = vect.z + this.z;
        return (new Vector3d(xx, yy, zz));
};
// odejmuje podany wektor od tego wektora. Nie jest tworzony nowy wektor
Vector2d.prototype.substract = function(vector) {
        if (vector instanceof Vector2d) {
                this.x -= vector.x;
                this.y -= vector.y;
        }
};
// odjecie dwoch wektorow
Vector3d.prototype.substract = function(vect) {
        if (vect instanceof Vector3d) {
                this.x -= vect.x;
                this.y -= vect.y;
                this.z -= vect.z;
        }
};

// odejmuje podany wektor od wektora tworzac i zwracajac nowy wektor
Vector2d.prototype.substractN = function(vector) {
        if (vector instanceof Vector2d) {
                var xx = this.x - vector.x;
                var yy = this.y - vector.y;
                return (new Vector2d(xx, yy));
        }
        return null;
};
// odjecie dwoch wektorow z utworzeniem nowego
Vector3d.prototype.substractN = function(vect) {
        if (vect instanceof "Vector3d") {
                var xx = vect.x - this.x;
                var yy = vect.y - this.y;
                var zz = vect.z - this.z;
                return new Vector3d(xx, yy, zz);
        }
        return null;
```

```
};
// skaluje wektor do podanej liczby. Nie tworzy nowego wektora
Vector2d.prototype.scale = function(num) {
	this.x *= num;
	this.y *= num;
};
// skalowanie wektora do podanej liczby
Vector3d.prototype.scale = function(num) {
	this.x *= num;
	this.y *= num;
	this.z *= num;
};

// skaluje wektor do podanej liczby. Zwraca nowy wektor
Vector2d.prototype.scaleN = function(num) {
	var xx = this.x * num;
	var yy = this.y * num;
	return (new Vector2d(xx, yy));
};
// skalowanie wektora do podanej liczby z utworzeniem nowego
Vector3d.prototype.scaleN = function(num) {
	var xx = this.x * num;
	var yy = this.y * num;
	var zz = this.z * num;
	return (new Vector3D(xx, yy, zz));
};
// bada i zwraca dlugosc tego wektora
Vector2d.prototype.getLength = function() {
	var xx = Math.pow(this.x, 2);
	var yy = Math.pow(this.y, 2);
	return Math.sqrt(xx + yy);
};

// zbadanie dlugosci wektora
Vector3d.prototype.getLength = function() {
	var xx = Math.pow(this.x, 2);
	var yy = Math.pow(this.y, 2);
	var zz = Math.pow(this.z, 2);
	return Math.sqrt(xx + yy + zz);
};

// ustawia dlugosc tego wektora
Vector2d.prototype.setLength = function(length) {
	var r = this.getLength();
	if (r) {
		this.scale(length / r);
	} else {
		this.x = length;
	}
};
// ustawienie nowej dlugosci wektora i stosowne skalowanie wektora
Vector3d.prototype.setLength = function(num) {
	var rr = this.length;
	if (rr) {
		this.scale(num / rr);
	} else {
		this.x = num;
	}
};
```

```
// okresla i podaje kat tego wektora (wsp. nachylenia) w stopniach
Vector2d.prototype.getAngle = function() {
	var angle = atan2Deg(this.y, this.x);
	return angle;
};
// zmienia kat tego wektora na podany kat (oba w stopniach
Vector2d.prototype.setAngle = function(angle) {
	var r = this.getLength();
	this.x = r * cosDeg(angle);
	this.y = r * sinDeg(angle);
};
// zmienia x i y tak, aby wektor mial dlugosc = 1;
Vector2d.prototype.normalize = function() {
	var len = this.getLength();
	this.x /= len;
	this.y /= len;
};
Vector2d.prototype.normalizeN = function() {
	var len = this.getLength();
	return new Vector2d(this.x /= len, this.y /= len);
};
// normalizacja wektora
Vector3d.prototype.normalize = function() {
	var len = this.length;
	return (new Vector3d(this.x / len, this.y / len, this.z / len));
};
// tworzy iloczyn skalarny dwoch wektorow. Nie zwraca nowego wektora
// jesli iloczyn = 0 wektory s	prostopad	e
Vector2d.prototype.dot = function(vector) {
	var x1 = 0;
	var x2 = 0;
	if (vector instanceof Vector2d) {
		x1 = this.x * vector.x;
		x2 = this.y * vector.y;
	}
	return (x1 + x2);
};

// obliczenie iloczynu skalarnego 2 wektorow
// jesli iloczyn = 0, wektory sa prostopadle;
Vector3d.prototype.dot = function(vect) {
	if (vect instanceof Vector3d) {
		var x1 = this.x * vect.x;
		var x2 = this.y * vect.y;
		var x3 = this.z * vect.z;
		return (x1 + x2 + x3);
	}
};
// tworzy i zwraca wektor normalny (prostopadly) do danego wektora;
// trzeba podac kierunek
Vector2d.prototype.normal = function(direction) {
	var x1 = 0;
	var y1 = 0;
	switch (direction) {
	case "left":
		x1 = -this.y;
		y1 = this.x;
		break;
	case "right":
```

```
            x1 = this.y;
            y1 = -this.x;
            break;
        }
        return (new Vector2d(x1, y1));
};

// sprawdza czy podany wektor jest (prostopadly) do wektora
Vector2d.prototype.isNormalTo = function(vector) {
        if (vector instanceof Vector2d) {
            return (this.dot(vect) == 0);
        }
        return false;
};

// podaje kat miedzy dwoma wektorami w stopniach
Vector2d.prototype.angleBetween = function(vector) {
        if (vector instanceof Vector2d) {
            var d = this.dot(vector);
            var c = d / (this.getLength() * vector.getLength());
            return acosDeg(c);
        }
};

// obliczenie kata miedzy wektorami
Vector3d.prototype.angleBetween = function(vect) {
        if (vect instanceof Vector3d) {
            var d = this.dot(vect);
            var c = d / (this.getLength() * vect.getLength());
            return acosDeg(c);
        }
};
// iloczyn wektorowy
Vector2d.prototype.cross = function(vect) {
        if (vect instanceof Vector2d) {
            var z = this.x * vect.y - this.y * vect.x;
            return (new Vector2d(0, z));
        }
        return null;
};
// iloczyn wektorowy
// wytworzenie nowego wektora z 2 wektorow
// nowy wektor jest prostopadly do kazdego z obu wektorow
Vector3d.prototype.cross = function(vect) {
        if (vect instanceof Vector3d) {
            var cx = this.y * vect.z - this.z * vect.y;
            var cy = this.z * vect.x - this.x * vect.z;
            var cz = this.x * vect.y - this.y * vect.x;
            return (new Vector3d(cx, cy, cz));
        }
        return null;
};
Vector3d.prototype.surface = function(vect) {
        if (vect instanceof Vector3d) {
            var temp1 = this.cross(vect);
            var temp2 = temp1.normalize();
            return temp2;
        }
        return null;
```

```
};
// radius = r = promien
// angle = angle = kat
var Polar = function(radius, angle) {
	this.radius = radius;
	this.angle = angle;
};

Polar.prototype.toString = function() {
	return ("[r: " + this.radius + ",  \u03C6" + ": " + this.angle + "
\u00BA]");
};

Polar.prototype.equals = function(polar) {
	if (polar instanceof Polar) {
		return this.radius == polar.radius && this.angle == polar.angle;
	}
	return false;
};

/*
 * przeksztala wspolrzedne kartezjanskie na polarne (biegunowe) zwraca obiekt
 * typu Polar; katy podawane sa w stopniach Przyjmuje albo obiekt typu Vector
 * albo par□ wsp□lrzednych kartezjanskich x, y cartToPolar(vector)
 * cartToPolar(x, y);
 */
var cartToPolar = function(args) {
	var polar = null;
	switch (arguments.length) {
	case 1:
		if (arguments[0] instanceof Vector2d) {
			var radius = Math.sqrt(Math.pow(arguments[0].y, 2)
				+ Math.pow(arguments[0].x, 2));
			var angle = atan2Deg(arguments[0].y, arguments[0].x);
			polar = new Polar(radius, angle);
		}
		break;
	case 2:
		if ((typeof arguments[0] == "number")
				&& (typeof arguments[1] == "number")) {
			var radius1 = Math.sqrt(Math.pow(arguments[1], 2)
				+ Math.pow(arguments[0], 2));
			var angle1 = atan2Deg(arguments[1], arguments[0]);
			polar = new Polar(radius1, angle1);
		}
		break;
	}
	return polar;
};

/*
 * przeksztalca wspolrzedne polarne (biegunowe) na kartezjanskie. katy podawane
 * sa w stopniach. Przyjmuje albo obiekt Polar albo pare liczb oznaczajacych
 * wspolrzedne polarne. zwraca obiekt Vector2d
 */
var polarToCart = function(args) {
	var vector = null;
	switch (arguments.length) {
	case 1:
```

```
            if (arguments[0] instanceof Polar) {
                var xx = arguments[0].radius * cosDeg(arguments[0].angle);
                var yy = arguments[0].radius * sinDeg(arguments[0].angle);
                vector = new Vector2d(xx, yy);
            }
            break;
        case 2:
            if ((typeof arguments[0] == "number")
                        && (typeof arguments[1] == "number")) {
                var xx1 = arguments[0] * cosDeg(arguments[1]);
                var yy1 = arguments[0] * sinDeg(arguments[1]);
                vector = new Vector2d(xx1, yy1);
            }
            break;
        }
        return vector;
};
// funkcje przepisane z pliku trygon.js
var atan2Deg = function(yy, xx) {
        return Math.atan2(yy, xx) * 180.0 / Math.PI;
};
var acosDeg = function(ratio) {
        return Math.acos(ratio) * 180.0 / Math.PI;
};
var sinDeg = function(angleDeg) {
        return Math.sin(angleDeg * Math.PI / 180.0);
};
var cosDeg = function(angleDeg) {
        return Math.cos(angleDeg * Math.PI / 180.0);
};
function drawVector(vector, axes, fillStyle) {
        ctx.save();
        ctx.beginPath();
        ctx.fillStyle = fillStyle;
        var radius = 0;
        var angle = 0;
        if (vector instanceof Vector2d) {
            radius = Math.sqrt(Math.pow(vector.y, 2) + Math.pow(vector.x, 2));
            angle = atan2Deg(vector.y, vector.x);
        }
        if (axes == "complex" || axes == "cartesian") {
            drawArrow(x0, y0, radius * 30, lw, angle, bw, bh, cl, fillStyle);

        }
        if (axes == "js") {
            drawArrow(15, 15, radius * 30, lw, -angle, bw, bh, cl, fillStyle);

        }
        if (axes == "norm") {
            drawArrow(0, 0, radius, lw, angle, bw, bh, cl, fillStyle);
        }
        ctx.restore();

};
function drawPolar(polar, axes, fillStyle) {
        ctx.save();
        ctx.beginPath();
        ctx.fillStyle = fillStyle;
        if (polar instanceof Polar) {
```

```
            if (axes == "complex" || axes == "cartesian") {
                  drawArrow(x0, y0, polar.radius * 30, lw, polar.angle, bw, bh,
cl,
                              fillStyle);

            }
            if (axes == "js") {
                  drawArrow(15, 15, polar.radius * 30, lw, -polar.angle, bw, bh,
cl,
                              fillStyle);

            }
            if (axes = "norm") {
                  drawArrow(0, 0, polar.radius, lw, -polar.angle, bw, bh, cl,
                              fillStyle);
            }
      }
      ctx.restore();

};
/*
 * funkcja zaokragla podan⌂ liczb⌂ do okre⌂lonej liczby miejsc po przecinku
 */
var roundToDecimal = function(num, decim) {
      switch (arguments.length) {
      case 1:
            dec = 0;
            num = arguments[0];
            break;
      case 2:
            dec = arguments[1];
            num = arguments[0];
            break;
      }
      var multi = Math.pow(10, dec);
      return Math.round(num * multi) / multi;
};
```

Listing42

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
            z kontekstem &#034;2d&#034;</canvas>
      <script type="text/javascript" src="../scripts/arrow.js"></script>
      <script type="text/javascript" src="../scripts/axes.js"></script>
      <script type="text/javascript" src="../scripts/linear.js"></script>
```

```
	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		drawAxes("cartesian");
		var vect = new Vector2d(7.5, 3.5);
		drawVector(vect, "cartesian", "black");
		ctx.fillText(vect.toString(), 50, 50);
	</script>
</body>

</html>
```

Listing43

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/arrow.js"></script>
	<script type="text/javascript" src="../scripts/axes.js"></script>
	<script type="text/javascript" src="../scripts/linear.js"></script>

	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		var vect = new Vector2d(7.5, 3.5);
		var vect1 = new Vector2d(7.5, 3.5);
		var vect2 = new Vector2d(6, 4);
		ctx.fillText(vect.equals(vect1), 50, 50);
		ctx.fillText(vect.equals(vect2), 50, 70);
	</script>
</body>

</html>
```

Listing44

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
```

```
            z kontekstem &#034;2d&#034;</canvas>
      <script type="text/javascript" src="../scripts/arrow.js"></script>
      <script type="text/javascript" src="../scripts/axes.js"></script>
      <script type="text/javascript" src="../scripts/linear.js"></script>

      <script type="text/javascript">
            drawAxes("cartesian");
            drawPolar(new Polar(5, 30), "cartesian", "black");
      </script>
</body>

</html>
```

Listing45

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
            z kontekstem &#034;2d&#034;</canvas>
      <script type="text/javascript" src="../scripts/arrow.js"></script>
      <script type="text/javascript" src="../scripts/axes.js"></script>
      <script type="text/javascript" src="../scripts/linear.js"></script>

      <script type="text/javascript">
            var cv = document.getElementById('canvas');
            var ctx = cv.getContext('2d');
            drawAxes("cartesian");
            var polar = new Polar(6, 30);
            drawPolar(polar, "cartesian", "black");
            ctx.fillText(polar.toString(), 50, 50);
      </script>
</body>

</html>
```

Listing46

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
            z kontekstem &#034;2d&#034;</canvas>
```

```
	<script type="text/javascript" src="../scripts/arrow.js"></script>
	<script type="text/javascript" src="../scripts/axes.js"></script>
	<script type="text/javascript" src="../scripts/linear.js"></script>

	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		var pol = new Polar(6, 30);
		var pol1 = new Polar(6, 30);
		var pol2 = new Polar(5, 28);
		ctx.fillText(pol.equals(pol1), 50, 50);
		ctx.fillText(pol.equals(pol2), 50, 70);
	</script>
</body>

</html>
```

Listing47

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/arrow.js"></script>
	<script type="text/javascript" src="../scripts/axes.js"></script>
	<script type="text/javascript" src="../scripts/linear.js"></script>

	<script type="text/javascript">
		drawAxes("cartesian");
		var x = 7.5;
		var y = 3.5
		var vect = new Vector2d(x, y);
		var polar1 = cartToPolar(vect);
		drawPolar(polar1, "cartesian", "black");
	</script>
</body>

</html>
```

Listing48

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
```

```
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/arrow.js"></script>
	<script type="text/javascript" src="../scripts/axes.js"></script>
	<script type="text/javascript" src="../scripts/linear.js"></script>

	<script type="text/javascript">
		drawAxes("cartesian");
		var x = 7.5;
		var y = 3.5;
		var polar1 = cartToPolar(x, y);
		drawPolar(polar1, "cartesian", "black");
	</script>
</body>

</html>
```

Listing49

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/arrow.js"></script>
	<script type="text/javascript" src="../scripts/axes.js"></script>
	<script type="text/javascript" src="../scripts/linear.js"></script>

	<script type="text/javascript">
		drawAxes("cartesian");
		var r = 6;
		var a = 30;
		var polar = new Polar(r, a);
		var vect = polarToCart(polar);
		drawVector(vect, "cartesian", "black");
	</script>
</body>

</html>
```

Listing50

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">
```

```
<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/arrow.js"></script>
	<script type="text/javascript" src="../scripts/axes.js"></script>
	<script type="text/javascript" src="../scripts/linear.js"></script>

	<script type="text/javascript">
		drawAxes("cartesian");
		var r = 6;
		var a = 30;
		var vect = polarToCart(6, 30);
		drawVector(vect, "cartesian", "black");
	</script>
</body>

</html>
```

Listing51

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/arrow.js"></script>
	<script type="text/javascript" src="../scripts/axes.js"></script>
	<script type="text/javascript" src="../scripts/linear.js"></script>

	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		drawAxes("cartesian");
		var x = 7.5;
		var y = 3.5
		var vect = new Vector2d(x, y);
		var length = roundToDecimal(vect.getLength(), 4);
		drawVector(vect, "cartesian", "black");
		ctx.fillText(length.toString(), 50, 50);
	</script>
</body>

</html>
```

Listing52

```
<!DOCTYPE html>
```

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/arrow.js"></script>
	<script type="text/javascript" src="../scripts/axes.js"></script>
	<script type="text/javascript" src="../scripts/linear.js"></script>

	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		drawAxes("cartesian");
		var x = 7.5;
		var y = 3.5
		var vect = new Vector2d(x, y);
		vect.setLength(5.25);
		drawVector(vect, "cartesian", "black");
		ctx.fillText(vect.getLength().toString(), 50, 50);
	</script>
</body>

</html>
```

Listing53

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/arrow.js"></script>
	<script type="text/javascript" src="../scripts/axes.js"></script>
	<script type="text/javascript" src="../scripts/linear.js"></script>

	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		drawAxes("cartesian");
		var x = 7.5;
		var y = 3.5
		var vect = new Vector2d(x, y);
		vect.setAngle(40);
		drawVector(vect, "cartesian", "black");
		ctx.fillText(vect.getAngle().toString(), 50, 50);
```

```
        </script>
</body>

</html>
```

Listing54

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
        <canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
                przeglądarce obsługującej element &lt;canvas&gt;
                z kontekstem &#034;2d&#034;</canvas>
        <script type="text/javascript" src="../scripts/arrow.js"></script>
        <script type="text/javascript" src="../scripts/axes.js"></script>
        <script type="text/javascript" src="../scripts/linear.js"></script>

        <script type="text/javascript">
        var cv = document.getElementById('canvas');
        var ctx = cv.getContext('2d');
                drawAxes("cartesian");
                var vect = new Vector2d(7.5, 3.5);
                drawVector(vect,"cartesian", "black");
                var vect1 = new Vector2d(4,3);
                drawVector(vect1,"cartesian", "black");
                vect.add(vect1);
                drawVector(vect, "cartesian", "red");
                ctx.fillStyle="red"
                ctx.fillText(vect.toString(), 50, 50);
        </script>
</body>

</html>
```

Listing55

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
        <canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
                przeglądarce obsługującej element &lt;canvas&gt;
                z kontekstem &#034;2d&#034;</canvas>
        <script type="text/javascript" src="../scripts/arrow.js"></script>
        <script type="text/javascript" src="../scripts/axes.js"></script>
        <script type="text/javascript" src="../scripts/linear.js"></script>
```

```
	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		drawAxes("cartesian");
		var vect = new Vector2d(7.5, 3.5);
		drawVector(vect, "cartesian", "black");
		var vect1 = new Vector2d(4, 3);
		drawVector(vect1, "cartesian", "black");
		var vect2 = vect.addN(vect1);
		drawVector(vect2, "cartesian", "red");
		ctx.fillStyle = "red"
		ctx.fillText(vect2.toString(), 50, 50);
	</script>
</body>

</html>
```

Listing56

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/arrow.js"></script>
	<script type="text/javascript" src="../scripts/axes.js"></script>
	<script type="text/javascript" src="../scripts/linear.js"></script>

	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		drawAxes("cartesian");
		var vect = new Vector2d(7.5, 3.5);
		drawVector(vect, "cartesian", "black");
		var vect1 = new Vector2d(4, 3);
		drawVector(vect1, "cartesian", "black");
		vect.substract(vect1);
		drawVector(vect, "cartesian", "red");
		ctx.fillStyle = "red"
		ctx.fillText(vect.toString(), 50, 50);
	</script>
</body>

</html>
```

Listing57

```
<!DOCTYPE html>
<html>
```

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/arrow.js"></script>
	<script type="text/javascript" src="../scripts/axes.js"></script>
	<script type="text/javascript" src="../scripts/linear.js"></script>

	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		drawAxes("cartesian");
		var vect = new Vector2d(7.5, 3.5);
		drawVector(vect, "cartesian", "black");
		var vect1 = new Vector2d(4, 3);
		drawVector(vect1, "cartesian", "black");
		var vect2 = vect.substractN(vect1);
		drawVector(vect2, "cartesian", "red");
		ctx.fillStyle = "red"
		ctx.fillText(vect2.toString(), 50, 50);
	</script>
</body>

</html>
```

Listing58

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/arrow.js"></script>
	<script type="text/javascript" src="../scripts/axes.js"></script>
	<script type="text/javascript" src="../scripts/linear.js"></script>

	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		drawAxes("cartesian");
		var vect = new Vector2d(7.5, 3.5);
		drawVector(vect, "cartesian", "black");
		vect.scale(0.75);
		drawVector(vect, "cartesian", "red");
		ctx.fillStyle = "red"
```

```
            ctx.fillText(vect.toString(), 50, 50);
      </script>
</body>

</html>
```

Listing59

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
            z kontekstem &#034;2d&#034;</canvas>
      <script type="text/javascript" src="../scripts/arrow.js"></script>
      <script type="text/javascript" src="../scripts/axes.js"></script>
      <script type="text/javascript" src="../scripts/linear.js"></script>

      <script type="text/javascript">
            var cv = document.getElementById('canvas');
            var ctx = cv.getContext('2d');
            drawAxes("cartesian");
            var vect = new Vector2d(7.5, 3.5);
            drawVector(vect, "cartesian", "black");
            var vect2 = vect.scaleN(0.75);
            drawVector(vect2, "cartesian", "red");
            ctx.fillStyle = "red"
            ctx.fillText(vect2.toString(), 50, 50);
      </script>
</body>

</html>
```

Listing60

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
            z kontekstem &#034;2d&#034;</canvas>
      <script type="text/javascript" src="../scripts/arrow.js"></script>
      <script type="text/javascript" src="../scripts/axes.js"></script>
      <script type="text/javascript" src="../scripts/linear.js"></script>
```

```
	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		drawAxes("cartesian");
		var vect = new Vector2d(7.5, 3.5);
		drawVector(vect, "cartesian", "black");
		vect.normalize();
		drawVector(vect, "cartesian", "red");
		ctx.fillStyle = "red"
		ctx.fillText(vect.toString(), 50, 50);
	</script>
</body>

</html>
```

Listing61

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/arrow.js"></script>
	<script type="text/javascript" src="../scripts/axes.js"></script>
	<script type="text/javascript" src="../scripts/linear.js"></script>

	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		drawAxes("cartesian");
		var vect = new Vector2d(7.5, 3.5);
		drawVector(vect, "cartesian", "black");
		var vect2 = vect.normalizeN();
		drawVector(vect2, "cartesian", "red");
		ctx.fillStyle = "red"
		ctx.fillText(vect2.toString(), 50, 50);
	</script>
</body>

</html>
```

Listing62

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
```

```
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/arrow.js"></script>
	<script type="text/javascript" src="../scripts/axes.js"></script>
	<script type="text/javascript" src="../scripts/linear.js"></script>

	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		drawAxes("cartesian");
		var vect = new Vector2d(7.5, 3.5);
		drawVector(vect, "cartesian", "black");
		var vect2 = new Vector2d(4, 3);
		drawVector(vect2, "cartesian", "black");
		ctx.fillStyle = "red"
		ctx.fillText(vect.dot(vect2), 450, 150);
	</script>
</body>

</html>
```

Listing63

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/arrow.js"></script>
	<script type="text/javascript" src="../scripts/axes.js"></script>
	<script type="text/javascript" src="../scripts/linear.js"></script>

	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		drawAxes("cartesian");
		var vect = new Vector2d(7.5, 3.5);
		drawVector(vect, "cartesian", "black");
		var vect2 = vect.normal("left");
		drawVector(vect2, "cartesian", "red");
		ctx.fillStyle = "red"
		ctx.fillText(vect2.toString(), 250, 150);
	</script>
</body>

</html>
```

Listing64

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/arrow.js"></script>
	<script type="text/javascript" src="../scripts/axes.js"></script>
	<script type="text/javascript" src="../scripts/linear.js"></script>

	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		drawAxes("cartesian");
		var vect = new Vector2d(7.5, 3.5);
		drawVector(vect, "cartesian", "black");
		var vect2 = vect.normal("right");
		drawVector(vect2, "cartesian", "red");
		ctx.fillStyle = "red"
		var isNormal = vect2.isNormalTo(vect);
		ctx.fillText(isNormal, 250, 150);
	</script>
</body>

</html>
```

Listing65

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/arrow.js"></script>
	<script type="text/javascript" src="../scripts/axes.js"></script>
	<script type="text/javascript" src="../scripts/linear.js"></script>

	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		drawAxes("cartesian");
		var vect1 = new Vector2d(7.5, 3.5);
		drawVector(vect1, "cartesian", "black");
		var vect2 = new Vector2d(3, 4);
		drawVector(vect2, "cartesian", "black");
```

```
            ctx.fillStyle = "red"
            ctx.fillText(vect1.angleBetween(vect2), 450, 150);
      </script>
</body>

</html>
```

Listing66

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<script src="../scripts/matrix.js"></script>
<title> </title>
<script type="text/javascript">
      var $ = function(id) {
            return document.getElementById(id);
      };
      var a = new Matrix();
      var wynik1 = a;
      var wynik2 = "";
      var wynik3 = "";
      var wynik4 = "";
      var wynik5 = "";

      window.onload = function() {
            $("wynik1").firstChild.nodeValue = wynik1;
            $("wynik2").firstChild.nodeValue = wynik2;
            $("wynik3").firstChild.nodeValue = wynik3;
            $("wynik4").firstChild.nodeValue = wynik4;
            $("wynik5").firstChild.nodeValue = wynik5;
      };
</script>
</head>
<body>
      <p>
            <span id="wynik1"> </span> <span id="wynik2"> </span> <span
                  id="wynik3"> </span> <span id="wynik4"> </span> <span
                  id="wynik5"> </span>
      </p>
</body>

</html>
```

Zawartość skryptu matrix.js

```
/*
 * Je►eli nie podano ►adnych parametrow tworzona jest macierz 3x3 wypelniona
 * zerami.
 * Jezeli podano jeden argument to moze to byc tablica dwuwymiarowa albo
 * macierz klasy Matrix. Dane z nich sa przejmowane do tej macierzy.
 * Jesli sa dwa argumenty:
 * albo podana jest tablica 1-wymiarowa, a jako drugi argument - liczba kolumn
 * albo podae sa m i n oznaczajace kolejno liczbe rzedow i liczbe kolumn
 * w macierzy. Macierz wypelniana jest zerami.
```

```
 */
var Matrix = function() {
	switch (arguments.length) {
	case 0:
		this.m = 3;
		this.n = 3;
		this.array = new Array(this.m);
		for ( var i = 0; i < this.m; i++) {
			this.array[i] = new Array(this.n);
		}
		this.setToValue(0);
		break;
	case 1:
		if (arguments[0] instanceof Array) {
			this.array = arguments[0];
			this.m = this.array.length;
			this.n = this.array[0].length;
		} else if (arguments[0] instanceof Matrix) {
			this.array = arguments[0].array;
			this.m = arguments[0].m;
			this.n = arguments[0].n;
		}

		break;
	case 2:
		if (arguments[0] instanceof Array) {
			this.array = oneToTwo(arguments[0], arguments[1]);
			this.m = this.array.length;
			this.n = this.array[0].length;
		}
		if (typeof arguments[0] == "number") {
			this.m = arguments[0];
			this.n = arguments[1];
			this.array = new Array(this.m);
			for ( var i = 0; i < this.m; i++) {
				this.array[i] = new Array(this.n);
			}
			this.setToValue(0);
		}
		break;
	}

};
/*
 * Ustawia wszystkie warto►ci w macierzy na warto►► s. Najcz►ciej jest to 0.
 */
Matrix.prototype.setToValue = function(s) {
	for ( var i = 0; i < this.m; i++) {
		for ( var j = 0; j < this.n; j++) {
			this.array[i][j] = s;
		}
	}
};
Matrix.prototype.setToIdentity = function() {
	if (this.isSquared()) {
		for ( var i = 0; i < this.m; i++) {
			for ( var j = 0; j < this.n; j++) {
				this.array[i][j] = (i == j ? 1.0 : 0.0);
			}
```

```
            }
        }
};

Matrix.prototype.equals = function(matrix) {
        if (!(matrix instanceof Matrix)) {
                return false;
        }
        if ((this.m != matrix.m) || (this.n != matrix.n)) {
                return false;
        }

        for ( var i = 0; i < this.m; i++) {
                for ( var j = 0; j < this.n; j++) {
                        if (this.array[i][j] != matrix.array[i][j]) {
                                return false;
                        }
                }
        }
        return true;
};
Matrix.prototype.toString = function() {
        var st = "";
        for ( var i = 0; i < this.m; i++) {
                for ( var j = 0; j < this.n; j++) {
                        st = st + this.array[i][j] + "\n";
                }
        }
        return st;
};
Matrix.prototype.getArray = function() {
        return this.array;
};

Matrix.prototype.setArray = function(array) {
        if (array instanceof Array) {
                this.array = array;
                this.m = this.array.length;
                this.n = this.array[0].length;
        }
};
// kopiuje tablice tej macierzy i zwraca nowa identyczna tablcie
Matrix.prototype.cloneArray = function() {
        var ar = new Array(this.m);
        for ( var k = 0; k < this.m; k++) {
                ar[k] = new Array(this.n);
        }
        for ( var i = 0; i < this.m; i++) {
                for ( var j = 0; j < this.n; j++) {
                        ar[i][j] = this.array[i][j];
                }
        }
        return ar;
};

Matrix.prototype.getM = function() {
        return this.m;
};
```

```
Matrix.prototype.getN = function() {
	return this.n;
};

Matrix.prototype.getMN = function(m, n) {
	return this.array[m][n];
};

Matrix.prototype.setMN = function(m, n, s) {
	this.array[m][n] = s;
};
Matrix.prototype.isSquared = function() {
	if (this.m == this.n) {
		return true;
	}
	return false;
};

Matrix.prototype.checkSize = function(matrix) {
	if (matrix instanceof Matrix) {
		if (matrix.m != this.m || matrix.n != this.n) {
			throw new Error("Nieprawid►owe wymiary.");
		}
	}

};
Matrix.prototype.transpose = function() {
	var temp1 = new Array(this.n);
	for ( var k = 0; k < this.n; k++) {
		temp1[k] = new Array(this.m);
	}
	for ( var i = 0; i < this.m; i++) {
		for ( var j = 0; j < this.n; j++) {
			temp1[j][i] = this.array[i][j];
		}
	}
	this.setArray(temp1);
};

Matrix.prototype.transpose2 = function() {
	var temp = new Matrix(this.n, this.m);
	var temp1 = temp.getArray();
	for ( var i = 0; i < this.m; i++) {
		for ( var j = 0; j < this.n; j++) {
			temp1[j][i] = this.array[i][j];
		}
	}
	return temp;
};
Matrix.prototype.add = function(matrix) {
	this.checkSize(matrix);
	for ( var i = 0; i < this.m; i++) {
		for ( var j = 0; j < this.n; j++) {
			this.array[i][j] += matrix.array[i][j];
		}
	}
};

Matrix.prototype.add2 = function(matrix) {
```

```
        this.checkSize(matrix);
        var temp = new Matrix(this.m, this.n);
        var temp1 = temp.getArray();
        for ( var i = 0; i < this.m; i++) {
                for ( var j = 0; j < this.n; j++) {
                        temp1[i][j] = this.array[i][j] + matrix.array[i][j];
                }
        }
        return temp;
};

Matrix.prototype.substract = function(matrix) {
        this.checkSize(matrix);
        for ( var i = 0; i < this.m; i++) {
                for ( var j = 0; j < this.n; j++) {
                        this.array[i][j] = this.array[i][j] - matrix.array[i][j];
                }
        }
};

Matrix.prototype.substract2 = function(matrix) {
        this.checkSize(matrix);
        var temp = new Matrix(this.m, this.n);
        var temp1 = temp.getArray();
        for ( var i = 0; i < this.m; i++) {
                for ( var j = 0; j < this.n; j++) {
                        temp1[i][j] = this.array[i][j] - matrix.array[i][j];
                }
        }
        return temp;
};

Matrix.prototype.multiplys = function(s) {
        for ( var i = 0; i < this.m; i++) {
                for ( var j = 0; j < this.n; j++) {
                        this.array[i][j] = s * this.array[i][j];
                }
        }
};

Matrix.prototype.multiplys2 = function(s) {
        var temp = new Matrix(this.m, this.n);
        var temp1 = temp.getArray();
        for ( var i = 0; i < this.m; i++) {
                for ( var j = 0; j < this.n; j++) {
                        temp1[i][j] = s * this.array[i][j];
                }
        }
        return temp;
};

Matrix.prototype.multiply = function(matrix) {
        if (matrix.m != this.n) {
                throw new Error("Nieprawid►owe wymiary macierzy.");
        }
        var temp1 = new Array(this.m);
        for ( var p = 0; p < this.m; p++) {
                temp1[p] = new Array(matrix.n);
        }
```

```
        var bc = new Array(this.n);
        for ( var j = 0; j < matrix.n; j++) {
                for ( var k = 0; k < this.n; k++) {
                        bc[k] = matrix.array[k][j];
                }
                for ( var i = 0; i < this.m; i++) {
                        var ai = this.array[i];
                        var s = 0.0;
                        for ( var r = 0; r < this.n; r++) {
                                s += ai[r] * bc[r];
                        }
                        temp1[i][j] = s;
                }
        }
        this.array = temp1;
        this.n = matrix.n;
};

Matrix.prototype.multiply2 = function(matrix) {
        if (matrix.m != this.n) {
                throw new Error("Nieprawid►owe wymiary macierzy");
        }
        var temp = new Matrix(this.m, matrix.n);
        var temp1 = new Array(this.m);
        for ( var p = 0; p < this.m; p++) {
                temp1[p] = new Array(matrix.n);
        }
        var bc = new Array(this.n);
        for ( var j = 0; j < matrix.n; j++) {
                for ( var k = 0; k < this.n; k++) {
                        bc[k] = matrix.array[k][j];
                }
                for ( var i = 0; i < this.m; i++) {
                        var ai = this.array[i];
                        var s = 0.0;
                        for ( var r = 0; r < this.n; r++) {
                                s += ai[r] * bc[r];
                        }
                        temp1[i][j] = s;
                }
        }
        temp.array = temp1;
        return temp;
};
var oneToTwo = function(tab, cols) {
        var rows = tab.length / cols;
        var temparr = new Array(rows);
        for ( var k = 0; k < rows; k++) {
                temparr[k] = new Array(cols);
        }
        for ( var i = 0; i < tab.length; i++) {
                var x = cutDecimal(i / cols);
                var y = i % cols;
                temparr[x][y] = tab[i];
        }
        return temparr;
};
var cutDecimal = function(nr) {
        var temp = nr.toString();
```

```
        var temp1 = temp.indexOf(".");
        var temp2 = "";
        if (temp1 > -1) {
                temp2 = temp.substr(0, temp1);
        } else {
                temp2 = temp;
        }
        return parseInt(temp2);
};
Matrix.prototype.setToTranslate = function(dx, dy) {
        this.array[0][2] = dx;
        this.array[1][2] = dy;
};
var setToTranslate = function(matrix, dx, dy) {
        matrix.array[0][2] = dx;
        matrix.array[1][2] = dy;
        return matrix;
};
Matrix.prototype.setToScale = function(sx, sy) {
        this.array[0][0] = sx;
        this.array[1][1] = sy;
};
var setToScale = function(matrix, sx, sy) {
        matrix.array[0][0] = sx;
        matrix.array[1][1] = sy;
        return matrix;
};
Matrix.prototype.setToRotation = function(angleDeg) {
        var angleRad = degToRad(angleDeg);
        var t1 = Math.cos(angleRad);
        var t2 = Math.sin(angleRad);
        this.array[0][0] = t1;
        this.array[0][1] = t2;
        this.array[1][0] = -t2;
        this.array[1][1] = t1;
};
var setToRotation = function(matrix, angleDeg) {
        var angleRad = degToRad(angleDeg);
        var t1 = Math.cos(angleRad);
        var t2 = Math.sin(angleRad);
        matrix.array[0][0] = t1;
        matrix.array[0][1] = t2;
        matrix.array[1][0] = -t2;
        matrix.array[1][1] = t1;
        return matrix;
};
// oblicza wyznacznik tej macierzy
Matrix.prototype.detSarrus = function() {
        var a = this.array[0][0];
        var b = this.array[0][1];
        var e = this.array[0][2];
        var c = this.array[1][0];
        var d = this.array[1][1];
        var f = this.array[1][2];
        var g = this.array[2][0];
        var h = this.array[2][1];
        var i = this.array[2][2];
        return ((a * d * i + b * f * g + e * c * h) - (e * d * g + a * f * h + b
                    * c * i));
```

```
};
// oblicza wyznacznik podanej macierzy
var detSarrus = function(matrix) {
	var sarrus = 0.0;
	switch (matrix.m) {
	case 1:
		sarrus = matrix.array[0][0];
		break;
	case 2:
		sarrus = matrix.array[0][0] * matrix.array[1][1] - matrix.array[0][1]
				* matrix.array[1][0];
		break;
	case 3:
		var a = matrix.array[0][0];
		var b = matrix.array[0][1];
		var e = matrix.array[0][2];
		var c = matrix.array[1][0];
		var d = matrix.array[1][1];
		var f = matrix.array[1][2];
		var g = matrix.array[2][0];
		var h = matrix.array[2][1];
		var i = matrix.array[2][2];
		sarrus = ((a * d * i + b * f * g + e * c * h) - (e * d * g + a * f *
h + b
				* c * i));
		break;

	}
	return sarrus;
};
// Oblicza podmacierz macierzy. row i col
// zaczynaja sie od 1
var submatrix = function(matrix, row, col) {
	var i1 = row - 1;
	var i2 = col - 1;
	var ar = matrix.cloneArray();
	ar = shortenArray(ar, i1);
	for ( var i = 0; i < ar.length; i++) {
		ar[i] = shortenArray(ar[i], i2);
	}
	var mat = new Matrix(ar);
	return mat;
};
// row, col zaczynaja sie od 1
var minor = function(matrix, row, col) {
	var sub = submatrix(matrix, row, col);
	return detSarrus(sub);
};
// skraca tablice pojedyncza wycinajac z niej podany element
var shortenArray = function(array, index) {
	for ( var i = index + 1; i < array.length; i++) {
		array[i - 1] = array[i];
	}
	array.pop();
	return array;
};
// row, col zaczynaja sie od 1
var algComplement = function(matrix, row, col) {
	return ac = minor(matrix, row, col) * Math.pow(-1, row + col);
```

```
};
// klonuje pojedyncza tablice
var cloneArray = function(array) {
	var b = new Array();
	return b.concat(array);
};
function cosDeg(angleDeg) {
	return Math.cos(angleDeg * Math.PI / 180);
};
function sinDeg(angleDeg) {
	return Math.sin(angleDeg * Math.PI / 180);
}
Matrix.prototype.setToReflection = function(line) {
	var slope = line.slope();
	var angle = atanDeg(slope);
	var dangle = 2 * angle;
	var cosa = cosDeg(dangle);
	var sina = sinDeg(dangle);
	this.array[0][0] = cosa;
	this.array[0][1] = sina;
	this.array[1][0] = sina;
	this.array[1][1] = -cosa;
};
var setToReflection = function(matrix, line) {
	var slope = line.slope();
	var angle = atanDeg(slope);
	var dangle = 2 * angle;
	var cosa = cosDeg(dangle);
	var sina = sinDeg(dangle);
	matrix.array[0][0] = cosa;
	matrix.array[0][1] = sina;
	matrix.array[1][0] = sina;
	matrix.array[1][1] = -cosa;
	return matrix;
};
Matrix.prototype.setToShear = function(shx, shy) {
	this.array[0][1] = shx;
	this.array[1][0] = shy;
};
var setToShear = function(matrix, shx, shy) {
	matrix.array[0][1] = shx;
	matrix.array[1][0] = shy;
	return matrix;
};
Matrix.prototype.reverte = function() {
	var matrixac = new Matrix(3, 3);
	for ( var i = 0; i < 3; i++) {
		for ( var j = 0; j < 3; j++) {
			ac = algComplement(this, i + 1, j + 1);
			matrixac.setMN(i, j, ac);
		}
	}
	var matrix1 = matrixac.transpose2();
	var det = this.detSarrus();
	var matrixt = matrix1.multiplys2(1.0 / det);
	return matrixt;

};
```

Listing67

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<script src="../scripts/matrix.js"></script>
<title> </title>
<script type="text/javascript">
	var $ = function(id) {
		return document.getElementById(id);
	};
	var a = [[1,2,3],[4,5,6],[7,8,9]];
	var wynik1 = new Matrix(a);
	var wynik2="";
	var wynik3="";
	var wynik4="";
	var wynik5="";

	window.onload = function() {
		$("wynik1").firstChild.nodeValue = wynik1;
		$("wynik2").firstChild.nodeValue = wynik2;
		$("wynik3").firstChild.nodeValue = wynik3;
		$("wynik4").firstChild.nodeValue = wynik4;
		$("wynik5").firstChild.nodeValue = wynik5;
	};
</script>
</head>
<body>
	<p>
		<span id="wynik1"> </span>
		<span id="wynik2"> </span>
		<span id="wynik3"> </span>
		<span id="wynik4"> </span>
		<span id="wynik5"> </span>
	</p>
</body>

</html>
```

Listing68

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<script src="../scripts/matrix.js"></script>
<title> </title>
<script type="text/javascript">
	var $ = function(id) {
		return document.getElementById(id);
	};
	var a = [ [ 1, 2, 3 ], [ 4, 5, 6 ], [ 7, 8, 9 ] ];
	var b = new Matrix(a);
	var wynik1 = new Matrix(b);
	var wynik2 = "";
```

```
        var wynik3 = "";
        var wynik4 = "";
        var wynik5 = "";

        window.onload = function() {
                $("wynik1").firstChild.nodeValue = wynik1;
                $("wynik2").firstChild.nodeValue = wynik2;
                $("wynik3").firstChild.nodeValue = wynik3;
                $("wynik4").firstChild.nodeValue = wynik4;
                $("wynik5").firstChild.nodeValue = wynik5;
        };
</script>
</head>
<body>
        <p>
                <span id="wynik1"> </span> <span id="wynik2"> </span> <span
                        id="wynik3"> </span> <span id="wynik4"> </span> <span
                        id="wynik5"> </span>
        </p>
</body>

</html>
```

Listing69

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<script src="../scripts/matrix.js"></script>
<title> </title>
<script type="text/javascript">
        var $ = function(id) {
                return document.getElementById(id);
        };
        var a = [ 1, 2, 3, 4, 5, 6, 7, 8, 9 ];
        var b = new Matrix(a, 3);
        var wynik1 = b;
        var wynik2 = "";
        var wynik3 = "";
        var wynik4 = "";
        var wynik5 = "";

        window.onload = function() {
                $("wynik1").firstChild.nodeValue = wynik1;
                $("wynik2").firstChild.nodeValue = wynik2;
                $("wynik3").firstChild.nodeValue = wynik3;
                $("wynik4").firstChild.nodeValue = wynik4;
                $("wynik5").firstChild.nodeValue = wynik5;
        };
</script>
</head>
<body>
        <p>
                <span id="wynik1"> </span> <span id="wynik2"> </span> <span
                        id="wynik3"> </span> <span id="wynik4"> </span> <span
                        id="wynik5"> </span>
```

```
	</p>
</body>

</html>
```

Listing70

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<script src="../scripts/matrix.js"></script>
<title> </title>
<script type="text/javascript">
	var $ = function(id) {
		return document.getElementById(id);
	};
	var a = new Matrix(3, 3);
	var wynik1 = a;
	var wynik2 = "";
	var wynik3 = "";
	var wynik4 = "";
	var wynik5 = "";

	window.onload = function() {
		$("wynik1").firstChild.nodeValue = wynik1;
		$("wynik2").firstChild.nodeValue = wynik2;
		$("wynik3").firstChild.nodeValue = wynik3;
		$("wynik4").firstChild.nodeValue = wynik4;
		$("wynik5").firstChild.nodeValue = wynik5;
	};
</script>
</head>
<body>
	<p>
		<span id="wynik1"> </span> <span id="wynik2"> </span> <span
			id="wynik3"> </span> <span id="wynik4"> </span> <span
			id="wynik5"> </span>
	</p>
</body>

</html>
```

Listing71

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<script src="../scripts/matrix.js"></script>
<title> </title>
<script type="text/javascript">
	var $ = function(id) {
		return document.getElementById(id);
	};
```

```
        var a = new Matrix(3, 3);
        a.setToIdentity();
        var wynik1 = a;
        var wynik2 = "";
        var wynik3 = "";
        var wynik4 = "";
        var wynik5 = "";

        window.onload = function() {
                $("wynik1").firstChild.nodeValue = wynik1;
                $("wynik2").firstChild.nodeValue = wynik2;
                $("wynik3").firstChild.nodeValue = wynik3;
                $("wynik4").firstChild.nodeValue = wynik4;
                $("wynik5").firstChild.nodeValue = wynik5;
        };
</script>
</head>
<body>
        <p>
                <span id="wynik1"> </span> <span id="wynik2"> </span> <span
                        id="wynik3"> </span> <span id="wynik4"> </span> <span
                        id="wynik5"> </span>
        </p>
</body>

</html>
```

Listing72

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<script src="../scripts/matrix.js"></script>
<title> </title>
<script type="text/javascript">
        var $ = function(id) {
                return document.getElementById(id);
        };
        var a = new Matrix(3, 3);
        a.setToIdentity();
        var b = new Matrix(3, 3);
        b.setToIdentity();
        var wynik1 = a.equals(b);
        var wynik2 = "";
        var wynik3 = "";
        var wynik4 = "";
        var wynik5 = "";

        window.onload = function() {
                $("wynik1").firstChild.nodeValue = wynik1;
                $("wynik2").firstChild.nodeValue = wynik2;
                $("wynik3").firstChild.nodeValue = wynik3;
                $("wynik4").firstChild.nodeValue = wynik4;
                $("wynik5").firstChild.nodeValue = wynik5;
        };
</script>
```

```
</head>
<body>
	<p>
		<span id="wynik1"> </span> <span id="wynik2"> </span> <span
			id="wynik3"> </span> <span id="wynik4"> </span> <span
			id="wynik5"> </span>
	</p>
</body>

</html>
```

Listing73

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<script src="../scripts/matrix.js"></script>
<title> </title>
<script type="text/javascript">
	var $ = function(id) {
		return document.getElementById(id);
	};
	var a = [ [ 1, 2, 3 ], [ 4, 5, 6 ], [ 7, 8, 9 ] ];
	var aa = new Matrix(a);
	var b = [ [ 9, 8, 7 ], [ 6, 5, 4 ], [ 3, 2, 1 ] ];
	var bb = new Matrix(b)
	aa.add(bb);
	var wynik1 = aa;
	var wynik2 = "";
	var wynik3 = "";
	var wynik4 = "";
	var wynik5 = "";

	window.onload = function() {
		$("wynik1").firstChild.nodeValue = wynik1;
		$("wynik2").firstChild.nodeValue = wynik2;
		$("wynik3").firstChild.nodeValue = wynik3;
		$("wynik4").firstChild.nodeValue = wynik4;
		$("wynik5").firstChild.nodeValue = wynik5;
	};
</script>
</head>
<body>
	<p>
		<span id="wynik1"> </span> <span id="wynik2"> </span> <span
			id="wynik3"> </span> <span id="wynik4"> </span> <span
			id="wynik5"> </span>
	</p>
</body>

</html>
```

Listing74

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

```
<script src="../scripts/matrix.js"></script>
<title> </title>
<script type="text/javascript">
	var $ = function(id) {
		return document.getElementById(id);
	};
	var a = [ [ 1, 2, 3 ], [ 4, 5, 6 ], [ 7, 8, 9 ] ];
	var aa = new Matrix(a);
	var b = [ [ 9, 8, 7 ], [ 6, 5, 4 ], [ 3, 2, 1 ] ];
	var bb = new Matrix(b)
	var c = aa.add2(bb);
	var wynik1 = c;
	var wynik2 = "";
	var wynik3 = "";
	var wynik4 = "";
	var wynik5 = "";

	window.onload = function() {
		$("wynik1").firstChild.nodeValue = wynik1;
		$("wynik2").firstChild.nodeValue = wynik2;
		$("wynik3").firstChild.nodeValue = wynik3;
		$("wynik4").firstChild.nodeValue = wynik4;
		$("wynik5").firstChild.nodeValue = wynik5;
	};
</script>
</head>
<body>
	<p>
		<span id="wynik1"> </span> <span id="wynik2"> </span> <span
			id="wynik3"> </span> <span id="wynik4"> </span> <span
			id="wynik5"> </span>
	</p>
</body>

</html>
```

Listing75

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<script src="../scripts/matrix.js"></script>
<title> </title>
<script type="text/javascript">
	var $ = function(id) {
		return document.getElementById(id);
	};
	var a = [ [ 1, 2, 3 ], [ 4, 5, 6 ], [ 7, 8, 9 ] ];
	var aa = new Matrix(a);
	var b = [ [ 9, 8, 7 ], [ 6, 5, 4 ], [ 3, 2, 1 ] ];
	var bb = new Matrix(b)
	aa.substract(bb);
	var wynik1 = aa;
	var wynik2 = "";
	var wynik3 = "";
```

```
        var wynik4 = "";
        var wynik5 = "";

        window.onload = function() {
                $("wynik1").firstChild.nodeValue = wynik1;
                $("wynik2").firstChild.nodeValue = wynik2;
                $("wynik3").firstChild.nodeValue = wynik3;
                $("wynik4").firstChild.nodeValue = wynik4;
                $("wynik5").firstChild.nodeValue = wynik5;
        };
</script>
</head>
<body>
        <p>
                <span id="wynik1"> </span> <span id="wynik2"> </span> <span
                        id="wynik3"> </span> <span id="wynik4"> </span> <span
                        id="wynik5"> </span>
        </p>
</body>

</html>
```

Listing76

```
!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<script src="../scripts/matrix.js"></script>
<title> </title>
<script type="text/javascript">
        var $ = function(id) {
                return document.getElementById(id);
        };
        var a = [ [ 1, 2, 3 ], [ 4, 5, 6 ], [ 7, 8, 9 ] ];
        var aa = new Matrix(a);
        var b = [ [ 9, 8, 7 ], [ 6, 5, 4 ], [ 3, 2, 1 ] ];
        var bb = new Matrix(b)
        var c = aa.substract2(bb);
        var wynik1 = c;
        var wynik2 = "";
        var wynik3 = "";
        var wynik4 = "";
        var wynik5 = "";

        window.onload = function() {
                $("wynik1").firstChild.nodeValue = wynik1;
                $("wynik2").firstChild.nodeValue = wynik2;
                $("wynik3").firstChild.nodeValue = wynik3;
                $("wynik4").firstChild.nodeValue = wynik4;
                $("wynik5").firstChild.nodeValue = wynik5;
        };
</script>
</head>
<body>
        <p>
                <span id="wynik1"> </span> <span id="wynik2"> </span> <span
```

```
                    id="wynik3"> </span> <span id="wynik4"> </span> <span
                    id="wynik5"> </span>
        </p>
</body>

</html>
```

Listing77

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<script src="../scripts/matrix.js"></script>
<title> </title>
<script type="text/javascript">
        var $ = function(id) {
                return document.getElementById(id);
        };
        var a = [ [ 1, 2, 3 ], [ 4, 5, 6 ], [ 7, 8, 9 ] ];
        var aa = new Matrix(a);
        var s = 2;
        aa.multiplys(s);
        var wynik1 = aa;
        var wynik2 = "";
        var wynik3 = "";
        var wynik4 = "";
        var wynik5 = "";

        window.onload = function() {
                $("wynik1").firstChild.nodeValue = wynik1;
                $("wynik2").firstChild.nodeValue = wynik2;
                $("wynik3").firstChild.nodeValue = wynik3;
                $("wynik4").firstChild.nodeValue = wynik4;
                $("wynik5").firstChild.nodeValue = wynik5;
        };
</script>
</head>
<body>
        <p>
                <span id="wynik1"> </span> <span id="wynik2"> </span> <span
                    id="wynik3"> </span> <span id="wynik4"> </span> <span
                    id="wynik5"> </span>
        </p>
</body>

</html>
```

Listing78

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<script src="../scripts/matrix.js"></script>
```

```
<title> </title>
<script type="text/javascript">
	var $ = function(id) {
		return document.getElementById(id);
	};
	var a = [[1,2,3],[4,5,6],[7,8,9]];
	var aa = new Matrix(a);
	var s = 2;
	var bb = aa.multiplys2(s);
	var wynik1 = bb;
	var wynik2="";
	var wynik3="";
	var wynik4="";
	var wynik5="";

	window.onload = function() {
		$("wynik1").firstChild.nodeValue = wynik1;
		$("wynik2").firstChild.nodeValue = wynik2;
		$("wynik3").firstChild.nodeValue = wynik3;
		$("wynik4").firstChild.nodeValue = wynik4;
		$("wynik5").firstChild.nodeValue = wynik5;
	};
</script>
</head>
<body>
	<p>
		<span id="wynik1"> </span>
		<span id="wynik2"> </span>
		<span id="wynik3"> </span>
		<span id="wynik4"> </span>
		<span id="wynik5"> </span>
	</p>
</body>

</html>
```

Listing79

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<script src="../scripts/matrix.js"></script>
<title> </title>
<script type="text/javascript">
	var $ = function(id) {
		return document.getElementById(id);
	};
	var a = [ [ 1 ], [ 2 ], [ 3 ] ];
	var aa = new Matrix(a);
	var b = [ [ 4, 5, 6 ] ];
	var bb = new Matrix(b)
	aa.multiply(bb);
	var wynik1 = aa;
	var wynik2 = "";
	var wynik3 = "";
	var wynik4 = "";
```

```
	var wynik5 = "";

	window.onload = function() {
		$("wynik1").firstChild.nodeValue = wynik1;
		$("wynik2").firstChild.nodeValue = wynik2;
		$("wynik3").firstChild.nodeValue = wynik3;
		$("wynik4").firstChild.nodeValue = wynik4;
		$("wynik5").firstChild.nodeValue = wynik5;
	};
</script>
</head>
<body>
	<p>
		<span id="wynik1"> </span> <span id="wynik2"> </span> <span
			id="wynik3"> </span> <span id="wynik4"> </span> <span
			id="wynik5"> </span>
	</p>
</body>

</html>
```

Listing80

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<script src="../scripts/matrix.js"></script>
<title> </title>
<script type="text/javascript">
	var $ = function(id) {
		return document.getElementById(id);
	};
	var a = [ [ 1, 2, 3 ], [ 4, 5, 6 ], [ 7, 8, 9 ] ];
	var aa = new Matrix(a);
	var b = [ [ 2, 2, 2 ], [ 2, 2, 2 ], [ 2, 2, 2 ] ];
	var bb = new Matrix(b)
	var cc = aa.multiply2(bb);
	var wynik1 = cc;
	var wynik2 = "";
	var wynik3 = "";
	var wynik4 = "";
	var wynik5 = "";

	window.onload = function() {
		$("wynik1").firstChild.nodeValue = wynik1;
		$("wynik2").firstChild.nodeValue = wynik2;
		$("wynik3").firstChild.nodeValue = wynik3;
		$("wynik4").firstChild.nodeValue = wynik4;
		$("wynik5").firstChild.nodeValue = wynik5;
	};
</script>
</head>
<body>
	<p>
		<span id="wynik1"> </span> <span id="wynik2"> </span> <span
			id="wynik3"> </span> <span id="wynik4"> </span> <span
```

```
                    id="wynik5"> </span>
        </p>
</body>

</html>
```

Listing81

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<script src="../scripts/matrix.js"></script>
<title> </title>
<script type="text/javascript">
        var $ = function(id) {
                return document.getElementById(id);
        };
        var a = [ [ 1, 2, 3 ], [ 4, 5, 6 ], [ 7, 8, 9 ] ];
        var aa = new Matrix(a);
        aa.transpose();
        var wynik1 = aa;
        var wynik2 = "";
        var wynik3 = "";
        var wynik4 = "";
        var wynik5 = "";

        window.onload = function() {
                $("wynik1").firstChild.nodeValue = wynik1;
                $("wynik2").firstChild.nodeValue = wynik2;
                $("wynik3").firstChild.nodeValue = wynik3;
                $("wynik4").firstChild.nodeValue = wynik4;
                $("wynik5").firstChild.nodeValue = wynik5;
        };
</script>
</head>
<body>
        <p>
                <span id="wynik1"> </span> <span id="wynik2"> </span> <span
                        id="wynik3"> </span> <span id="wynik4"> </span> <span
                        id="wynik5"> </span>
        </p>
</body>

</html>
```

Listing82

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<script src="../scripts/matrix.js"></script>
<title> </title>
<script type="text/javascript">
```

```
        var $ = function(id) {
                return document.getElementById(id);
        };
        var a = [ [ 1, 2, 3 ], [ 4, 5, 6 ], [ 7, 8, 9 ] ];
        var aa = new Matrix(a);
        var c = aa.transpose2();
        var wynik1 = c;
        var wynik2 = "";
        var wynik3 = "";
        var wynik4 = "";
        var wynik5 = "";

        window.onload = function() {
                $("wynik1").firstChild.nodeValue = wynik1;
                $("wynik2").firstChild.nodeValue = wynik2;
                $("wynik3").firstChild.nodeValue = wynik3;
                $("wynik4").firstChild.nodeValue = wynik4;
                $("wynik5").firstChild.nodeValue = wynik5;
        };
</script>
</head>
<body>
        <p>
                <span id="wynik1"> </span> <span id="wynik2"> </span> <span
                        id="wynik3"> </span> <span id="wynik4"> </span> <span
                        id="wynik5"> </span>
        </p>
</body>

</html>
```

Listing83

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
        <canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
                przeglądarce obsługującej element &lt;canvas&gt;
                z kontekstem &#034;2d&#034;</canvas>
        <script type="text/javascript" src="../scripts/arrow.js"></script>
        <script type="text/javascript" src="../scripts/axes.js"></script>
        <script type="text/javascript" src="../scripts/linear.js"></script>

        <script type="text/javascript">
                var cv = document.getElementById('canvas');
                var ctx = cv.getContext('2d');
                drawAxes("cartesian");
                var vect = new Vector2d(4, 4);
                drawVector(vect, "cartesian", "black");
                var vect1 = new Vector2d(4, 1);
                drawVector(vect1, "cartesian", "black");
                var vect2 = vect.addN(vect1);
```

```
            drawVector(vect2, "cartesian", "red");
            var s = vect1.getLength() * vect2.getLength()
                        * sinDeg(vect1.angleBetween(vect2));
            ctx.fillText(s, 460, 180);
      </script>
</body>

</html>
```

Listing84

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
            z kontekstem &#034;2d&#034;</canvas>
      <script type="text/javascript" src="../scripts/arrow.js"></script>
      <script type="text/javascript" src="../scripts/axes.js"></script>
      <script type="text/javascript" src="../scripts/linear.js"></script>

      <script type="text/javascript">
            var cv = document.getElementById('canvas');
            var ctx = cv.getContext('2d');
            drawAxes("cartesian");
            var vect1 = new Vector2d(1, -1);
            drawVector(vect1, "cartesian", "black");
            var vect2 = new Vector2d(3, 4);
            drawVector(vect2, "cartesian", "black");
            var vect3 = new Vector2d(-2, 5);
            drawVector(vect3, "cartesian", "black");
            var xx = new Vector2d(vect1.x - vect2.x, vect1.y - vect2.y)
            var x = xx.getLength();
            ctx.fillText("" + x, 440, 180);
            var yy = new Vector2d(vect2.x - vect3.x, vect2.y - vect3.y);
            var y = yy.getLength();
            ctx.fillText("" + y, 440, 200);
            var zz = new Vector2d(vect3.x - vect1.x, vect3.y - vect1.y);
            var z = zz.getLength();
            ctx.fillText("" + z, 440, 220);
            var p = (x + y + z) / 2;
            ctx.fillText("" + p + " ", 440, 240);
            var s = Math.sqrt(p * (p - x) * (p - y) * (p - z));
            ctx.fillText("" + s + " ", 440, 260);
      </script>
</body>

</html>
```

Listing85

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/arrow.js"></script>
	<script type="text/javascript" src="../scripts/axes.js"></script>
	<script type="text/javascript" src="../scripts/linear.js"></script>

	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		var vect1 = new Vector2d(4, 4);
		var vect2 = new Vector2d(4, 1);
		var d = vect1.cross(vect2);
		ctx.fillText(d.getLength(), 440, 180);
		var vect3 = new Vector3d(4, 4, 0);
		var vect4 = new Vector3d(4, 1, 0);
		var d1 = vect3.cross(vect4);
		ctx.fillText(d1.getLength(), 440, 200);
	</script>
</body>

</html>
```

Listing86

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/arrow.js"></script>
	<script type="text/javascript" src="../scripts/axes.js"></script>
	<script type="text/javascript" src="../scripts/linear.js"></script>

	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		var vect1 = new Vector3d(1, -1, 0);
		var vect2 = new Vector3d(3, 4, 0);
		var vect3 = new Vector3d(-2, 5, 0);
		var c1 = vect1.cross(vect2);
		var c2 = vect2.cross(vect3);
```

```
            var c3 = vect3.cross(vect1);
            var cl1 = c1.getLength() * 0.5;
            var cl2 = c2.getLength() * 0.5;
            var cl3 = c3.getLength() * 0.5;
            ctx.fillText(cl1, 440, 200);
            ctx.fillText(cl2, 440, 220);
            ctx.fillText(cl3, 440, 240);
      </script>
</body>

</html>
```

Listing87

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
            z kontekstem &#034;2d&#034;</canvas>
      <script type="text/javascript" src="../scripts/arrow.js"></script>
      <script type="text/javascript" src="../scripts/axes.js"></script>
      <script type="text/javascript" src="../scripts/linear.js"></script>
      <script type="text/javascript" src="../scripts/matrix.js"></script>

      <script type="text/javascript">
            var cv = document.getElementById('canvas');
            var ctx = cv.getContext('2d');
            var ar = [ [ 2, 2, 3 ], [ 4, 5, 6 ], [ 7, 8, 9 ] ];
            var matrix1 = new Matrix(ar);
            var submatrix11 = submatrix(matrix1, 3, 3);
            ctx.fillText(submatrix11, 440, 200);
      </script>
</body>

</html>
```

Listing88

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
            z kontekstem &#034;2d&#034;</canvas>
```

```
	<script type="text/javascript" src="../scripts/arrow.js"></script>
	<script type="text/javascript" src="../scripts/axes.js"></script>
	<script type="text/javascript" src="../scripts/linear.js"></script>
	<script type="text/javascript" src="../scripts/matrix.js"></script>

	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		var ar = [ [ 2, 2, 3 ], [ 4, 5, 6 ], [ 7, 8, 9 ] ];
		var matrix = new Matrix(ar);
		var min = minor(matrix, 3, 3);
		ctx.fillText(min, 440, 200);
	</script>
</body>

</html>
```

Listing89

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/arrow.js"></script>
	<script type="text/javascript" src="../scripts/axes.js"></script>
	<script type="text/javascript" src="../scripts/linear.js"></script>
	<script type="text/javascript" src="../scripts/matrix.js"></script>

	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		var ar = [ [ 2, 2, 3 ], [ 4, 5, 6 ], [ 7, 8, 9 ] ];
		var matrix = new Matrix(ar);
		var ac = algComplement(matrix, 2, 2);
		ctx.fillText(ac, 440, 200);
	</script>
</body>

</html>
```

Listing90

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
```

```
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/arrow.js"></script>
	<script type="text/javascript" src="../scripts/axes.js"></script>
	<script type="text/javascript" src="../scripts/linear.js"></script>
	<script type="text/javascript" src="../scripts/matrix.js"></script>

	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		var ar = [ [ 2, 2, 3 ], [ 4, 5, 6 ], [ 7, 8, 9 ] ];
		var matrix = new Matrix(ar);
		var matrixac = new Matrix(3, 3);
		for ( var i = 0; i < 3; i++) {
			for ( var j = 0; j < 3; j++) {
				ac = algComplement(matrix, i + 1, j + 1);
				matrixac.setMN(i, j, ac);
			}
		}

		ctx.fillText(matrixac, 440, 200);
		var det = matrix.detSarrus();
		ctx.fillText(det, 440, 220);
	</script>
</body>

</html>
```

Listing91

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/arrow.js"></script>
	<script type="text/javascript" src="../scripts/axes.js"></script>
	<script type="text/javascript" src="../scripts/linear.js"></script>
	<script type="text/javascript" src="../scripts/matrix.js"></script>

	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		var ar = [ [ 2, 2, 3 ], [ 4, 5, 6 ], [ 7, 8, 9 ] ];
		var matrix = new Matrix(ar);
		var matrixac = new Matrix(3, 3);
		for ( var i = 0; i < 3; i++) {
			for ( var j = 0; j < 3; j++) {
```

```
                        ac = algComplement(matrix, i + 1, j + 1);
                        matrixac.setMN(i, j, ac);
                    }
                }
                var matrix1 = matrixac.transpose2();
                ctx.fillText(matrix1, 440, 200);
                ctx.fillText(matrix.detSarrus(), 440, 220);
        </script>
</body>

</html>
```

Listing92

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
        <canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
                przeglądarce obsługującej element &lt;canvas&gt;
                z kontekstem &#034;2d&#034;</canvas>
        <script type="text/javascript" src="../scripts/arrow.js"></script>
        <script type="text/javascript" src="../scripts/axes.js"></script>
        <script type="text/javascript" src="../scripts/linear.js"></script>
        <script type="text/javascript" src="../scripts/matrix.js"></script>

        <script type="text/javascript">
                var cv = document.getElementById('canvas');
                var ctx = cv.getContext('2d');
                var ar = [ [ 2, 2, 3 ], [ 4, 5, 6 ], [ 7, 8, 9 ] ];
                var matrix = new Matrix(ar);
                var matrixac = new Matrix(3, 3);
                for ( var i = 0; i < 3; i++) {
                        for ( var j = 0; j < 3; j++) {
                        ac = algComplement(matrix, i + 1, j + 1);
                        matrixac.setMN(i, j, ac);
                    }
                }
                var matrix1 = matrixac.transpose2();
                var det = matrix.detSarrus();
                var matrixt = matrix1.multiplys2(1 / det);
                ctx.fillText(matrixt, 440, 220);
        </script>
</body>

</html>
```

Listing93

```
<!DOCTYPE html>
<html>
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/arrow.js"></script>
	<script type="text/javascript" src="../scripts/axes.js"></script>
	<script type="text/javascript" src="../scripts/linear.js"></script>
	<script type="text/javascript" src="../scripts/matrix.js"></script>

	<script type="text/javascript">
	var cv = document.getElementById('canvas');
	var ctx = cv.getContext('2d');
		var ar = [[2,2,3],[4,5,6],[7,8,9]];
		var matrix = new Matrix(ar);
		var matrixac = new Matrix(3, 3);
		for(var i=0; i< 3;i++){
			for(var j=0; j<3;j++){
				ac=algComplement(matrix, i+1, j+1);
				matrixac.setMN(i,j, ac);
			}
		}
		var matrix1 = matrixac.transpose2();
		var det = matrix.detSarrus();
		var matrixt = matrix1.multiplys2(1/det);
		var finalm = matrix.multiply2(matrixt);
		ctx.fillText(finalm, 440, 220);




	</script>
</body>

</html>
```

Listing94

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/arrow.js"></script>
	<script type="text/javascript" src="../scripts/axes.js"></script>
	<script type="text/javascript" src="../scripts/linear.js"></script>
	<script type="text/javascript" src="../scripts/matrix.js"></script>
```

```
	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		var ar1 = [ [ 3, 2, 0 ], [ 5, -3, 0 ], [ 0, 0, 1 ] ];
		var ar2 = [ [ 17 ], [ 3 ], [ 1 ] ];
		var matrix1 = new Matrix(ar1);
		var matrix2 = new Matrix(ar2);
		var matrix3 = matrix1.reverte();
		var matrix4 = matrix3.multiply2(matrix2);
		ctx.fillText(matrix4, 20, 20);
	</script>
</body>

</html>
```

Listing95

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/arrow.js"></script>
	<script type="text/javascript" src="../scripts/axes.js"></script>
	<script type="text/javascript" src="../scripts/line.js"></script>

	<script type="text/javascript">
	var cv = document.getElementById('canvas');
	var ctx = cv.getContext('2d');
		drawAxes("cartesian");
		var line1 = findLine(3,2,-17);
		drawLine("cartesian", "black", line1);
		var line2 = findLine(5,-3,-3);
		drawLine("cartesian", "black", line2);
	</script>
</body>

</html>
```

Listing96

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
```

```
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/arrow.js"></script>
	<script type="text/javascript" src="../scripts/axes.js"></script>
	<script type="text/javascript" src="../scripts/matrix.js"></script>
	<script type="text/javascript">
		var drawTriangle = function(x1, y1, x2, y2, x3, y3) {
			ctx.beginPath();
			ctx.moveTo(x0 + x1 * 30, y0 - y1 * 30);
			ctx.fillText("A", x0 + x1 * 30, y0 - y1 * 30 + 5);
			ctx.lineTo(x0 + x2 * 30, y0 - y2 * 30);
			ctx.fillText("B", x0 + x2 * 30, y0 - y2 * 30 + 5);
			ctx.lineTo(x0 + x3 * 30, y0 - y3 * 30);
			ctx.fillText("C", x0 + x3 * 30, y0 - y3 * 30 - 5);
			ctx.closePath();
			ctx.stroke();
		}
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		drawAxes("cartesian");
		drawTriangle(4, 4, 5, 6, 2, 5);
		var pointa = [ 4, 4, 1 ];
		var mata = new Matrix(pointa, 1);
		var transa = new Matrix();
		transa.setToIdentity();
		var dx = 3;
		var dy = 4;
		transa.setToTranslate(dx, dy);
		var aa = transa.multiply2(mata);
		ctx.fillText(aa.toString(), 440, 140);
		//-
		var pointb = [ 5, 6, 1 ];
		var matb = new Matrix(pointb, 1);
		var transb = new Matrix();
		transb.setToIdentity();
		transb.setToTranslate(dx, dy);
		var bb = transb.multiply2(matb);
		ctx.fillText(bb.toString(), 440, 160);
		//-
		var pointc = [ 2, 5, 1 ];
		var matc = new Matrix(pointc, 1)
		var transc = new Matrix();
		transc.setToIdentity();
		transc.setToTranslate(dx, dy);
		var cc = transc.multiply2(matc);
		ctx.fillText(cc.toString(), 440, 180);
		//-
		drawTriangle(aa.getMN(0, 0), aa.getMN(1, 0), bb.getMN(0, 0),
bb.getMN(
				1, 0), cc.getMN(0, 0), cc.getMN(1, 0));
	</script>
</body>

</html>
```

Listing97

```
!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/arrow.js"></script>
	<script type="text/javascript" src="../scripts/axes.js"></script>
	<script type="text/javascript" src="../scripts/matrix.js"></script>
	<script type="text/javascript">
		var drawTriangle = function(x1, y1, x2, y2, x3, y3) {
			ctx.beginPath();
			ctx.moveTo(x0 + x1 * 30, y0 - y1 * 30);
			ctx.fillText("A", x0 + x1 * 30, y0 - y1 * 30 + 5);
			ctx.lineTo(x0 + x2 * 30, y0 - y2 * 30);
			ctx.fillText("B", x0 + x2 * 30, y0 - y2 * 30 + 5);
			ctx.lineTo(x0 + x3 * 30, y0 - y3 * 30);
			ctx.fillText("C", x0 + x3 * 30, y0 - y3 * 30 - 5);
			ctx.closePath();
			ctx.stroke();
		}
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		drawAxes("cartesian");
		drawTriangle(4, 4, 5, 6, 2, 5);
		var pointa = [ 4, 4, 1 ];
		var mata = new Matrix(pointa, 1)
		var transa = new Matrix();
		transa.setToIdentity();
		var sx = 0.5;
		var sy = 1.5;
		transa.setToScale(sx, sy);
		var aa = transa.multiply2(mata);
		ctx.fillText(aa.toString(), 500, 140);
		//-
		var pointb = [ 5, 6, 1 ];
		var matb = new Matrix(pointb, 1)
		var transb = new Matrix();
		transb.setToIdentity();
		transb.setToScale(sx, sy);
		var bb = transb.multiply2(matb);
		ctx.fillText(bb.toString(), 500, 160);
		//-
		var pointc = [ 2, 5, 1 ];
		var matc = new Matrix(pointc, 1)
		var transc = new Matrix();
		transc.setToIdentity();
		transc.setToScale(sx, sy);
		var cc = transc.multiply2(matc);
		ctx.fillText(cc.toString(), 500, 180);
		//-
		drawTriangle(aa.getMN(0, 0), aa.getMN(1, 0), bb.getMN(0, 0),
bb.getMN(
```

```
                        1, 0), cc.getMN(0, 0), cc.getMN(1, 0));
      </script>
</body>

</html>
```

Listing98

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
            z kontekstem &#034;2d&#034;</canvas>
      <script type="text/javascript" src="../scripts/arrow.js"></script>
      <script type="text/javascript" src="../scripts/axes.js"></script>
      <script type="text/javascript" src="../scripts/matrix.js"></script>
      <script type="text/javascript">
            var drawRectangle = function(x1, y1, x2, y2, x3, y3, x4, y4) {
                  ctx.beginPath();
                  ctx.moveTo(x0 + x1 * 30, y0 - y1 * 30);
                  ctx.fillText("A", x0 + x1 * 30, y0 - y1 * 30 + 5);
                  ctx.lineTo(x0 + x2 * 30, y0 - y2 * 30);
                  ctx.fillText("B", x0 + x2 * 30, y0 - y2 * 30 + 5);
                  ctx.lineTo(x0 + x3 * 30, y0 - y3 * 30);
                  ctx.fillText("C", x0 + x3 * 30, y0 - y3 * 30 - 5);
                  ctx.lineTo(x0 + x4 * 30, y0 - y4 * 30);
                  ctx.closePath();
                  ctx.stroke();
            }
            var cv = document.getElementById('canvas');
            var ctx = cv.getContext('2d');
            drawAxes("cartesian");
            drawRectangle(2, 5, 5, 5, 5, 8, 2, 8);
            var pointa = [ 2, 5, 1 ];
            var mata = new Matrix(pointa, 1);
            var transa = new Matrix();
            transa.setToIdentity();
            var sx = 0.5;
            var sy = 1.5;
            transa.setToScale(sx, sy);
            var aa = transa.multiply2(mata);

            //-
            var pointb = [ 5, 5, 1 ];
            var matb = new Matrix(pointb, 1);
            var transb = new Matrix();
            transb.setToIdentity();
            transb.setToScale(sx, sy);
            var bb = transb.multiply2(matb);

            //-
```

```
            var pointc = [ 5, 8, 1 ];
            var matc = new Matrix(pointc, 1)
            var transc = new Matrix();
            transc.setToIdentity();
            transc.setToScale(sx, sy);
            var cc = transc.multiply2(matc);

            //-
            var pointd = [ 2, 8, 1 ];
            var matd = new Matrix(pointd, 1)
            var transd = new Matrix();
            transd.setToIdentity();
            transd.setToScale(sx, sy);
            var dd = transc.multiply2(matd);

            //-
            drawRectangle(aa.getMN(0, 0), aa.getMN(1, 0), bb.getMN(0, 0),
bb.getMN(
                        1, 0), cc.getMN(0, 0), cc.getMN(1, 0), dd.getMN(0,0),
dd.getMN(1,0));

      </script>
</body>

</html>
```

Listing99

```
!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
            z kontekstem &#034;2d&#034;</canvas>
      <script type="text/javascript" src="../scripts/arrow.js"></script>
      <script type="text/javascript" src="../scripts/axes.js"></script>
      <script type="text/javascript" src="../scripts/matrix.js"></script>
      <script type="text/javascript">
            var drawTriangle = function(x1, y1, x2, y2, x3, y3) {
                  ctx.beginPath();
                  ctx.moveTo(x0 + x1 * 30, y0 - y1 * 30);
                  ctx.fillText("A", x0 + x1 * 30, y0 - y1 * 30 + 5);
                  ctx.lineTo(x0 + x2 * 30, y0 - y2 * 30);
                  ctx.fillText("B", x0 + x2 * 30, y0 - y2 * 30 + 5);
                  ctx.lineTo(x0 + x3 * 30, y0 - y3 * 30);
                  ctx.fillText("C", x0 + x3 * 30, y0 - y3 * 30 - 5);
                  ctx.closePath();
                  ctx.stroke();
            }
            var cv = document.getElementById('canvas');
            var ctx = cv.getContext('2d');
            drawAxes("cartesian");
```

```
            drawTriangle(4, 4, 5, 6, 2, 5);
            var pointa = [ 4, 4, 1 ];
            var mata = new Matrix(pointa, 1)
            var transa = new Matrix();
            transa.setToIdentity();
            var angle = 30;
            transa.setToRotation(angle);
            var aa = transa.multiply2(mata);
            ctx.fillText(aa.toString(), 500, 140);
            //-
            var pointb = [ 5, 6, 1 ];
            var matb = new Matrix(pointb, 1)
            var transb = new Matrix();
            transb.setToIdentity();
            transb.setToRotation(angle);
            var bb = transb.multiply2(matb);
            ctx.fillText(bb.toString(), 500, 160);
            //-
            var pointc = [ 2, 5, 1 ];
            var matc = new Matrix(pointc, 1)
            var transc = new Matrix();
            transc.setToIdentity();
            transc.setToRotation(angle);
            var cc = transc.multiply2(matc);
            ctx.fillText(cc.toString(), 500, 180);
            //-
            drawTriangle(aa.getMN(0, 0), aa.getMN(1, 0), bb.getMN(0, 0),
bb.getMN(
                        1, 0), cc.getMN(0, 0), cc.getMN(1, 0));

      </script>
</body>

</html>
```

Listing100

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
            z kontekstem &#034;2d&#034;</canvas>
      <script type="text/javascript" src="../scripts/arrow.js"></script>
      <script type="text/javascript" src="../scripts/axes.js"></script>
      <script type="text/javascript" src="../scripts/matrix.js"></script>
      <script type="text/javascript">
            var drawTriangle = function(x1, y1, x2, y2, x3, y3) {
                  ctx.beginPath();
                  ctx.moveTo(x0 + x1 * 30, y0 - y1 * 30);
                  ctx.fillText("A", x0 + x1 * 30, y0 - y1 * 30 + 5);
                  ctx.lineTo(x0 + x2 * 30, y0 - y2 * 30);
```

```
                ctx.fillText("B", x0 + x2 * 30, y0 - y2 * 30 + 5);
                ctx.lineTo(x0 + x3 * 30, y0 - y3 * 30);
                ctx.fillText("C", x0 + x3 * 30, y0 - y3 * 30 - 5);
                ctx.closePath();
                ctx.stroke();
            }
            var cv = document.getElementById('canvas');
            var ctx = cv.getContext('2d');
            drawAxes("cartesian");
            drawTriangle(4, 4, 5, 6, 2, 5);
            var pointa = [ 4, 4, 1 ];
            var mata = new Matrix(pointa, 1)
            var transa = new Matrix();
            transa.setToIdentity();
            var angle = 30;
            transa.setToScale(1, -1);
            var aa = transa.multiply2(mata);
            ctx.fillText(aa.toString(), 500, 140);
            //-
            var pointb = [ 5, 6, 1 ];
            var matb = new Matrix(pointb, 1)
            var transb = new Matrix();
            transb.setToIdentity();
            transb.setToScale(1, -1);
            var bb = transb.multiply2(matb);
            ctx.fillText(bb.toString(), 500, 160);
            //-
            var pointc = [ 2, 5, 1 ];
            var matc = new Matrix(pointc, 1)
            var transc = new Matrix();
            transc.setToIdentity();
            transc.setToScale(1, -1);
            var cc = transc.multiply2(matc);
            ctx.fillText(cc.toString(), 500, 180);
            //-
            drawTriangle(aa.getMN(0, 0), aa.getMN(1, 0), bb.getMN(0, 0),
bb.getMN(
                        1, 0), cc.getMN(0, 0), cc.getMN(1, 0));
        </script>
</body>

</html>
```

Listing101

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
        <canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
                przeglądarce obsługującej element &lt;canvas&gt;
                z kontekstem &#034;2d&#034;</canvas>
        <script type="text/javascript" src="../scripts/arrow.js"></script>
```

```
        <script type="text/javascript" src="../scripts/axes.js"></script>
        <script type="text/javascript" src="../scripts/matrix.js"></script>
        <script type="text/javascript">
                var drawTriangle = function(x1, y1, x2, y2, x3, y3) {
                        ctx.beginPath();
                        ctx.moveTo(x0 + x1 * 30, y0 - y1 * 30);
                        ctx.fillText("A", x0 + x1 * 30, y0 - y1 * 30 + 5);
                        ctx.lineTo(x0 + x2 * 30, y0 - y2 * 30);
                        ctx.fillText("B", x0 + x2 * 30, y0 - y2 * 30 + 5);
                        ctx.lineTo(x0 + x3 * 30, y0 - y3 * 30);
                        ctx.fillText("C", x0 + x3 * 30, y0 - y3 * 30 - 5);
                        ctx.closePath();
                        ctx.stroke();
                }
                var cv = document.getElementById('canvas');
                var ctx = cv.getContext('2d');
                drawAxes("cartesian");
                drawTriangle(4, 4, 5, 6, 2, 5);
                var pointa = [ 4, 4, 1 ];
                var mata = new Matrix(pointa, 1)
                var transa = new Matrix();
                transa.setToIdentity();
                var angle = 30;
                transa.setToScale(-1, 1);
                var aa = transa.multiply2(mata);
                ctx.fillText(aa.toString(), 500, 140);
                //-
                var pointb = [ 5, 6, 1 ];
                var matb = new Matrix(pointb, 1)
                var transb = new Matrix();
                transb.setToIdentity();
                transb.setToScale(-1, 1);
                var bb = transb.multiply2(matb);
                ctx.fillText(bb.toString(), 500, 160);
                //-
                var pointc = [ 2, 5, 1 ];
                var matc = new Matrix(pointc, 1)
                var transc = new Matrix();
                transc.setToIdentity();
                transc.setToScale(-1, 1);
                var cc = transc.multiply2(matc);
                ctx.fillText(cc.toString(), 500, 180);
                //-
                drawTriangle(aa.getMN(0, 0), aa.getMN(1, 0), bb.getMN(0, 0),
bb.getMN(
                        1, 0), cc.getMN(0, 0), cc.getMN(1, 0));
        </script>
</body>

</html>
```

Listing102

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">
```

```
<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/arrow.js"></script>
	<script type="text/javascript" src="../scripts/axes.js"></script>
	<script type="text/javascript" src="../scripts/matrix.js"></script>
	<script type="text/javascript">
		var drawTriangle = function(x1, y1, x2, y2, x3, y3) {
			ctx.beginPath();
			ctx.moveTo(x0 + x1 * 30, y0 - y1 * 30);
			ctx.fillText("A", x0 + x1 * 30, y0 - y1 * 30 + 5);
			ctx.lineTo(x0 + x2 * 30, y0 - y2 * 30);
			ctx.fillText("B", x0 + x2 * 30, y0 - y2 * 30 + 5);
			ctx.lineTo(x0 + x3 * 30, y0 - y3 * 30);
			ctx.fillText("C", x0 + x3 * 30, y0 - y3 * 30 - 5);
			ctx.closePath();
			ctx.stroke();
		}
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		drawAxes("cartesian");
		drawTriangle(4, 4, 5, 6, 2, 5);
		var pointa = [ 4, 4, 1 ];
		var mata = new Matrix(pointa, 1)
		var transa = new Matrix();
		transa.setToIdentity();
		var angle = 30;
		transa.setToScale(-1, -1);
		var aa = transa.multiply2(mata);
		ctx.fillText(aa.toString(), 500, 140);
		//-
		var pointb = [ 5, 6, 1 ];
		var matb = new Matrix(pointb, 1)
		var transb = new Matrix();
		transb.setToIdentity();
		transb.setToScale(-1, -1);
		var bb = transb.multiply2(matb);
		ctx.fillText(bb.toString(), 500, 160);
		//-
		var pointc = [ 2, 5, 1 ];
		var matc = new Matrix(pointc, 1)
		var transc = new Matrix();
		transc.setToIdentity();
		transc.setToScale(-1, -1);
		var cc = transc.multiply2(matc);
		ctx.fillText(cc.toString(), 500, 180);
		//-
		drawTriangle(aa.getMN(0, 0), aa.getMN(1, 0), bb.getMN(0, 0),
bb.getMN(
				1, 0), cc.getMN(0, 0), cc.getMN(1, 0));
	</script>
</body>

</html>
```

Listing103

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/arrow.js"></script>
	<script type="text/javascript" src="../scripts/axes.js"></script>
	<script type="text/javascript" src="../scripts/matrix.js"></script>
	<script type="text/javascript" src="../scripts/line.js"></script>
	<script type="text/javascript">
		var drawTriangle = function(x1, y1, x2, y2, x3, y3) {
			ctx.beginPath();
			ctx.moveTo(x0 + x1 * 30, y0 - y1 * 30);
			ctx.fillText("A", x0 + x1 * 30, y0 - y1 * 30 + 5);
			ctx.lineTo(x0 + x2 * 30, y0 - y2 * 30);
			ctx.fillText("B", x0 + x2 * 30, y0 - y2 * 30 + 5);
			ctx.lineTo(x0 + x3 * 30, y0 - y3 * 30);
			ctx.fillText("C", x0 + x3 * 30, y0 - y3 * 30 - 5);
			ctx.closePath();
			ctx.stroke();
		}
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		drawAxes("cartesian");
		//-
		//var line = new Line(-4, -5, 7, 6);
		var line = new Line(-7,-6,7,6)
		drawLine("cartesian", "black", line);
		var slope = line.slope();
		ctx.fillText(slope, 440, 200);
		var angle = atanDeg(slope);
		ctx.fillText(angle, 440, 220);
		//-
		drawTriangle(4, 4, 5, 6, 2, 5);
		var pointa = [ 4, 4, 1 ];
		var mata = new Matrix(pointa, 1)
		var transa = new Matrix();
		transa.setToIdentity();
		var angle = 30;
		transa.setToReflection(line);
		var aa = transa.multiply2(mata);

		//-
		var pointb = [ 5, 6, 1 ];
		var matb = new Matrix(pointb, 1)
		var transb = new Matrix();
		transb.setToIdentity();
		transb.setToReflection(line);
		var bb = transb.multiply2(matb);
```

```
            //-
            var pointc = [ 2, 5, 1 ];
            var matc = new Matrix(pointc, 1)
            var transc = new Matrix();
            transc.setToIdentity();
            transc.setToReflection(line);
            var cc = transc.multiply2(matc);

            //-
            drawTriangle(aa.getMN(0, 0), aa.getMN(1, 0), bb.getMN(0, 0),
bb.getMN(
                        1, 0), cc.getMN(0, 0), cc.getMN(1, 0));

      </script>
</body>

</html>
```

Listing104

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
            z kontekstem &#034;2d&#034;</canvas>
      <script type="text/javascript" src="../scripts/arrow.js"></script>
      <script type="text/javascript" src="../scripts/axes.js"></script>
      <script type="text/javascript" src="../scripts/matrix.js"></script>
      <script type="text/javascript">
            var drawTriangle = function(x1, y1, x2, y2, x3, y3) {
                  ctx.beginPath();
                  ctx.moveTo(x0 + x1 * 30, y0 - y1 * 30);
                  ctx.fillText("A", x0 + x1 * 30, y0 - y1 * 30 + 5);
                  ctx.lineTo(x0 + x2 * 30, y0 - y2 * 30);
                  ctx.fillText("B", x0 + x2 * 30, y0 - y2 * 30 + 5);
                  ctx.lineTo(x0 + x3 * 30, y0 - y3 * 30);
                  ctx.fillText("C", x0 + x3 * 30, y0 - y3 * 30 - 5);
                  ctx.closePath();
                  ctx.stroke();
            }
            var cv = document.getElementById('canvas');
            var ctx = cv.getContext('2d');
            drawAxes("cartesian");
            drawTriangle(4, 4, 5, 6, 2, 5);
            var tpoint = [2,2,1];
            var angle = 30;
            //-
            var pointa = [ 4, 4, 1 ];
            var mata = new Matrix(pointa, 1)
            var transa1 = new Matrix();
            transa1.setToIdentity();
```

```
            var transa2 = new Matrix();
            transa2.setToIdentity();
            var transa3 = new Matrix();
            transa3.setToIdentity()
            transa1.setToTranslate(-tpoint[0], -tpoint[1]);
            var aa1 = transa1.multiply2(mata);
            transa2.setToRotation(angle);
            var aa2 = transa2.multiply2(aa1);
            transa3.setToTranslate(tpoint[0], tpoint[1]);
            var aa3 = transa3.multiply2(aa2);
            ctx.fillText(aa3.toString(), 500, 140);
            //-
            var pointb = [ 5, 6, 1 ];
            var matb = new Matrix(pointb, 1)
            var transb1 = new Matrix();
            transb1.setToIdentity();
            var transb2 = new Matrix();
            transb2.setToIdentity();
            var transb3 = new Matrix();
            transb3.setToIdentity();
            transb1.setToTranslate(-tpoint[0], -tpoint[1]);
            var bb1 = transb1.multiply2(matb);
            transb2.setToRotation(angle);
            var bb2 = transb2.multiply2(bb1);
            transb3.setToTranslate(tpoint[0], tpoint[1]);
            var bb3 = transb3.multiply2(bb2);
            ctx.fillText(bb3.toString(), 500, 160);
            //-
            var pointc = [ 2, 5, 1 ];
            var matc = new Matrix(pointc, 1)
            var transc1 = new Matrix();
            transc1.setToIdentity();
            var transc2 = new Matrix();
            transc2.setToIdentity();
            var transc3 = new Matrix();
            transc3.setToIdentity();
            transc1.setToTranslate(-tpoint[0], -tpoint[1]);
            var cc1 = transc1.multiply2(matc);
            transc2.setToRotation(angle);
            var cc2 = transc2.multiply2(cc1);
            transc3.setToTranslate(tpoint[0], tpoint[1]);
            var cc3 = transc3.multiply2(cc2);
            ctx.fillText(cc3.toString(), 500, 180);
            //-
            drawTriangle(aa3.getMN(0, 0), aa3.getMN(1, 0), bb3.getMN(0, 0),
bb3.getMN(
                        1, 0), cc3.getMN(0, 0), cc3.getMN(1, 0));

      </script>
</body>

</html>
```

Listing105

```
<!DOCTYPE html>
<html>
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/arrow.js"></script>
	<script type="text/javascript" src="../scripts/axes.js"></script>
	<script type="text/javascript" src="../scripts/matrix.js"></script>
	<script type="text/javascript">
		var drawTriangle = function(x1, y1, x2, y2, x3, y3) {
			ctx.beginPath();
			ctx.moveTo(x0 + x1 * 30, y0 - y1 * 30);
			ctx.fillText("A", x0 + x1 * 30, y0 - y1 * 30 + 5);
			ctx.lineTo(x0 + x2 * 30, y0 - y2 * 30);
			ctx.fillText("B", x0 + x2 * 30, y0 - y2 * 30 + 5);
			ctx.lineTo(x0 + x3 * 30, y0 - y3 * 30);
			ctx.fillText("C", x0 + x3 * 30, y0 - y3 * 30 - 5);
			ctx.closePath();
			ctx.stroke();
		}
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		drawAxes("cartesian");
		drawTriangle(4, 4, 5, 6, 2, 5);
		var tpoint = [ 3.667, 5, 1 ];
		var angle = 30;
		//-
		var pointa = [ 4, 4, 1 ];
		var mata = new Matrix(pointa, 1)
		var transa1 = new Matrix();
		transa1.setToIdentity();
		var transa2 = new Matrix();
		transa2.setToIdentity();
		var transa3 = new Matrix();
		transa3.setToIdentity()
		transa1.setToTranslate(-tpoint[0], -tpoint[1]);
		var aa1 = transa1.multiply2(mata);
		transa2.setToRotation(angle);
		var aa2 = transa2.multiply2(aa1);
		transa3.setToTranslate(tpoint[0], tpoint[1]);
		var aa3 = transa3.multiply2(aa2);
		ctx.fillText(aa3.toString(), 500, 140);
		//-
		var pointb = [ 5, 6, 1 ];
		var matb = new Matrix(pointb, 1)
		var transb1 = new Matrix();
		transb1.setToIdentity();
		var transb2 = new Matrix();
		transb2.setToIdentity();
		var transb3 = new Matrix();
		transb3.setToIdentity();
		transb1.setToTranslate(-tpoint[0], -tpoint[1]);
		var bb1 = transb1.multiply2(matb);
		transb2.setToRotation(angle);
		var bb2 = transb2.multiply2(bb1);
```

```
            transb3.setToTranslate(tpoint[0], tpoint[1]);
            var bb3 = transb3.multiply2(bb2);
            ctx.fillText(bb3.toString(), 500, 160);
            //-
            var pointc = [ 2, 5, 1 ];
            var matc = new Matrix(pointc, 1)
            var transc1 = new Matrix();
            transc1.setToIdentity();
            var transc2 = new Matrix();
            transc2.setToIdentity();
            var transc3 = new Matrix();
            transc3.setToIdentity();
            transc1.setToTranslate(-tpoint[0], -tpoint[1]);
            var cc1 = transc1.multiply2(matc);
            transc2.setToRotation(angle);
            var cc2 = transc2.multiply2(cc1);
            transc3.setToTranslate(tpoint[0], tpoint[1]);
            var cc3 = transc3.multiply2(cc2);
            ctx.fillText(cc3.toString(), 500, 180);
            //-
            drawTriangle(aa3.getMN(0, 0), aa3.getMN(1, 0), bb3.getMN(0, 0), bb3
                        .getMN(1, 0), cc3.getMN(0, 0), cc3.getMN(1, 0));
      </script>
</body>

</html>
```

Listing106

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
            z kontekstem &#034;2d&#034;</canvas>
      <script type="text/javascript" src="../scripts/arrow.js"></script>
      <script type="text/javascript" src="../scripts/axes.js"></script>
      <script type="text/javascript" src="../scripts/matrix.js"></script>
      <script type="text/javascript">
            var drawRectangle = function(x1, y1, x2, y2, x3, y3, x4, y4) {
                  ctx.beginPath();
                  ctx.moveTo(x0 + x1 * 30, y0 - y1 * 30);
                  ctx.fillText("A", x0 + x1 * 30, y0 - y1 * 30 + 5);
                  ctx.lineTo(x0 + x2 * 30, y0 - y2 * 30);
                  ctx.fillText("B", x0 + x2 * 30, y0 - y2 * 30 + 5);
                  ctx.lineTo(x0 + x3 * 30, y0 - y3 * 30);
                  ctx.fillText("C", x0 + x3 * 30, y0 - y3 * 30 - 5);
                  ctx.lineTo(x0 + x4 * 30, y0 - y4 * 30);
                  ctx.closePath();
                  ctx.stroke();
            }
            var cv = document.getElementById('canvas');
```

```
var ctx = cv.getContext('2d');
drawAxes("cartesian");
drawRectangle(2, 5, 5, 5, 5, 8, 2, 8);
var tpoint = [ 3.5, 6.5, 1 ];
var sx = 0.5;
var sy = 1.5;
var pointa = [ 2, 5, 1 ];
var mata = new Matrix(pointa, 1)
var transa1 = new Matrix();
transa1.setToIdentity();
var transa2 = new Matrix();
transa2.setToIdentity();
var transa3 = new Matrix();
transa3.setToIdentity()
transa1.setToTranslate(-tpoint[0], -tpoint[1]);
var aa1 = transa1.multiply2(mata);
transa2.setToScale(sx, sy);
var aa2 = transa2.multiply2(aa1);
transa3.setToTranslate(tpoint[0], tpoint[1]);
var aa3 = transa3.multiply2(aa2);
ctx.fillText(aa3.toString(), 580, 100);
//-
var pointb = [ 5, 5, 1 ];
var matb = new Matrix(pointb, 1)
var transb1 = new Matrix();
transb1.setToIdentity();
var transb2 = new Matrix();
transb2.setToIdentity();
var transb3 = new Matrix();
transb3.setToIdentity();
transb1.setToTranslate(-tpoint[0], -tpoint[1]);
var bb1 = transb1.multiply2(matb);
transb2.setToScale(sx, sy);
var bb2 = transb2.multiply2(bb1);
transb3.setToTranslate(tpoint[0], tpoint[1]);
var bb3 = transb3.multiply2(bb2);
ctx.fillText(bb3.toString(), 580, 120);
//-
var pointc = [ 5, 8, 1 ];
var matc = new Matrix(pointc, 1)
var transc1 = new Matrix();
transc1.setToIdentity();
var transc2 = new Matrix();
transc2.setToIdentity();
var transc3 = new Matrix();
transc3.setToIdentity();
transc1.setToTranslate(-tpoint[0], -tpoint[1]);
var cc1 = transc1.multiply2(matc);
transc2.setToScale(sx, sy);
var cc2 = transc2.multiply2(cc1);
transc3.setToTranslate(tpoint[0], tpoint[1]);
var cc3 = transc3.multiply2(cc2);
ctx.fillText(cc3.toString(), 580, 140);
//-
var pointd = [ 2, 8, 1 ];
var matd = new Matrix(pointd, 1)
var transd1 = new Matrix();
transd1.setToIdentity();
var transd2 = new Matrix();
```

```
            transd2.setToIdentity();
            var transd3 = new Matrix();
            transd3.setToIdentity();
            transd1.setToTranslate(-tpoint[0], -tpoint[1]);
            var dd1 = transd1.multiply2(matd);
            transd2.setToScale(sx, sy);
            var dd2 = transd2.multiply2(dd1);
            transd3.setToTranslate(tpoint[0], tpoint[1]);
            var dd3 = transd3.multiply2(dd2);
            ctx.fillText(dd3.toString(), 580, 160);
            //-
            drawRectangle(aa3.getMN(0, 0), aa3.getMN(1, 0), bb3.getMN(0, 0),
bb3.getMN(
                        1, 0), cc3.getMN(0, 0), cc3.getMN(1, 0), dd3.getMN(0,0),
dd3.getMN(1,0));

      </script>
</body>

</html>
```

Listing107

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
            z kontekstem &#034;2d&#034;</canvas>
      <script type="text/javascript" src="../scripts/arrow.js"></script>
      <script type="text/javascript" src="../scripts/axes.js"></script>
      <script type="text/javascript" src="../scripts/matrix.js"></script>
      <script type="text/javascript" src="../scripts/line.js"></script>
      <script type="text/javascript">
            var drawTriangle = function(x1, y1, x2, y2, x3, y3) {
                  ctx.beginPath();
                  ctx.moveTo(x0 + x1 * 30, y0 - y1 * 30);
                  ctx.fillText("A", x0 + x1 * 30, y0 - y1 * 30 + 5);
                  ctx.lineTo(x0 + x2 * 30, y0 - y2 * 30);
                  ctx.fillText("B", x0 + x2 * 30, y0 - y2 * 30 + 5);
                  ctx.lineTo(x0 + x3 * 30, y0 - y3 * 30);
                  ctx.fillText("C", x0 + x3 * 30, y0 - y3 * 30 - 5);
                  ctx.closePath();
                  ctx.stroke();
            }
            var cv = document.getElementById('canvas');
            var ctx = cv.getContext('2d');
            drawAxes("cartesian");
            drawTriangle(4, 4, 5, 6, 2, 5);
            var line = new Line(-4, -5, 7, 6);
      var line1 = lineParallelThruPoint(line,new Vector2f(0,0));
            drawLine("cartesian", "black", line);
```

```
            var slope = line.slope();
            var b = findb(line);
            //-
            var pointa = [ 4, 4, 1 ];
            var mata = new Matrix(pointa, 1)
            var transa1 = new Matrix();
            transa1.setToIdentity();
            var transa2 = new Matrix();
            transa2.setToIdentity();
            var transa3 = new Matrix();
            transa3.setToIdentity()
            transa1.setToTranslate(0, -b);
            var aa1 = transa1.multiply2(mata);
            transa2.setToReflection(line1);
            var aa2 = transa2.multiply2(aa1);
            transa3.setToTranslate(0, b);
            var aa3 = transa3.multiply2(aa2);
            ctx.fillText(aa3.toString(), 500, 140);
            //-
            var pointb = [ 5, 6, 1 ];
            var matb = new Matrix(pointb, 1)
            var transb1 = new Matrix();
            transb1.setToIdentity();
            var transb2 = new Matrix();
            transb2.setToIdentity();
            var transb3 = new Matrix();
            transb3.setToIdentity();
            transb1.setToTranslate(0, -b);
            var bb1 = transb1.multiply2(matb);
            transb2.setToReflection(line1);
            var bb2 = transb2.multiply2(bb1);
            transb3.setToTranslate(0, b);
            var bb3 = transb3.multiply2(bb2);
            ctx.fillText(bb3.toString(), 500, 160);
            //-
            var pointc = [ 2, 5, 1 ];
            var matc = new Matrix(pointc, 1)
            var transc1 = new Matrix();
            transc1.setToIdentity();
            var transc2 = new Matrix();
            transc2.setToIdentity();
            var transc3 = new Matrix();
            transc3.setToIdentity();
            transc1.setToTranslate(0, -b);
            var cc1 = transc1.multiply2(matc);
            transc2.setToReflection(line1);
            var cc2 = transc2.multiply2(cc1);
            transc3.setToTranslate(0, b);
            var cc3 = transc3.multiply2(cc2);
            ctx.fillText(cc3.toString(), 500, 180);
            //-
            drawTriangle(aa3.getMN(0, 0), aa3.getMN(1, 0), bb3.getMN(0, 0),
bb3.getMN(
                        1, 0), cc3.getMN(0, 0), cc3.getMN(1, 0));

      </script>
</body>

</html>
```

Listing108

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/arrow.js"></script>
	<script type="text/javascript" src="../scripts/axes.js"></script>
	<script type="text/javascript" src="../scripts/matrix.js"></script>
	<script type="text/javascript">
		var drawRectangle = function(x1, y1, x2, y2, x3, y3, x4, y4) {
			ctx.beginPath();
			ctx.moveTo(x0 + x1 * 30, y0 - y1 * 30);
			ctx.fillText("A", x0 + x1 * 30, y0 - y1 * 30 + 5);
			ctx.lineTo(x0 + x2 * 30, y0 - y2 * 30);
			ctx.fillText("B", x0 + x2 * 30, y0 - y2 * 30 + 5);
			ctx.lineTo(x0 + x3 * 30, y0 - y3 * 30);
			ctx.fillText("C", x0 + x3 * 30, y0 - y3 * 30 - 5);
			ctx.lineTo(x0 + x4 * 30, y0 - y4 * 30);
			ctx.closePath();
			ctx.stroke();
		}
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		drawAxes("cartesian");
		drawRectangle(2, 5, 5, 5, 5, 8, 2, 8);
		var pointa = [ 2, 5, 1 ];
		var mata = new Matrix(pointa, 1);
		var transa = new Matrix();
		transa.setToIdentity();
		var sx = 0.5;
		var sy = 0;
		transa.setToShear(sx, sy);
		var aa = transa.multiply2(mata);

		//-
		var pointb = [ 5, 5, 1 ];
		var matb = new Matrix(pointb, 1);
		var transb = new Matrix();
		transb.setToIdentity();
		transb.setToShear(sx, sy);
		var bb = transb.multiply2(matb);

		//-
		var pointc = [ 5, 8, 1 ];
		var matc = new Matrix(pointc, 1)
		var transc = new Matrix();
		transc.setToIdentity();
		transc.setToShear(sx, sy);
		var cc = transc.multiply2(matc);
```

```
            //-
            var pointd = [ 2, 8, 1 ];
            var matd = new Matrix(pointd, 1)
            var transd = new Matrix();
            transd.setToIdentity();
            transd.setToShear(sx, sy);
            var dd = transc.multiply2(matd);

            //-
            drawRectangle(aa.getMN(0, 0), aa.getMN(1, 0), bb.getMN(0, 0),
bb.getMN(
                        1, 0), cc.getMN(0, 0), cc.getMN(1, 0), dd.getMN(0,0),
dd.getMN(1,0));

      </script>
</body>

</html>
```

Listing109

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
            z kontekstem &#034;2d&#034;</canvas>
      <script type="text/javascript" src="../scripts/arrow.js"></script>
      <script type="text/javascript" src="../scripts/axes.js"></script>
      <script type="text/javascript" src="../scripts/matrix.js"></script>
      <script type="text/javascript">
            var drawRectangle = function(x1, y1, x2, y2, x3, y3, x4, y4) {
                  ctx.beginPath();
                  ctx.moveTo(x0 + x1 * 30, y0 - y1 * 30);
                  ctx.fillText("A", x0 + x1 * 30, y0 - y1 * 30 + 5);
                  ctx.lineTo(x0 + x2 * 30, y0 - y2 * 30);
                  ctx.fillText("B", x0 + x2 * 30, y0 - y2 * 30 + 5);
                  ctx.lineTo(x0 + x3 * 30, y0 - y3 * 30);
                  ctx.fillText("C", x0 + x3 * 30, y0 - y3 * 30 - 5);
                  ctx.lineTo(x0 + x4 * 30, y0 - y4 * 30);
                  ctx.closePath();
                  ctx.stroke();
            }
            var cv = document.getElementById('canvas');
            var ctx = cv.getContext('2d');
            drawAxes("cartesian");
            drawRectangle(2, 5, 5, 5, 5, 8, 2, 8);
            var pointa = [ 2, 5, 1 ];
            var mata = new Matrix(pointa, 1);
            var transa = new Matrix();
            transa.setToIdentity();
            var sx = 0;
```

```
            var sy = 0.5;
            transa.setToShear(sx, sy);
            var aa = transa.multiply2(mata);

            //-
            var pointb = [ 5, 5, 1 ];
            var matb = new Matrix(pointb, 1);
            var transb = new Matrix();
            transb.setToIdentity();
            transb.setToShear(sx, sy);
            var bb = transb.multiply2(matb);

            //-
            var pointc = [ 5, 8, 1 ];
            var matc = new Matrix(pointc, 1)
            var transc = new Matrix();
            transc.setToIdentity();
            transc.setToShear(sx, sy);
            var cc = transc.multiply2(matc);

            //-
            var pointd = [ 2, 8, 1 ];
            var matd = new Matrix(pointd, 1)
            var transd = new Matrix();
            transd.setToIdentity();
            transd.setToShear(sx, sy);
            var dd = transc.multiply2(matd);

            //-
            drawRectangle(aa.getMN(0, 0), aa.getMN(1, 0), bb.getMN(0, 0),
bb.getMN(
                        1, 0), cc.getMN(0, 0), cc.getMN(1, 0), dd.getMN(0, 0),
dd
                        .getMN(1, 0));
      </script>
</body>

</html>
```

Listing110

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
            z kontekstem &#034;2d&#034;</canvas>
      <script type="text/javascript" src="../scripts/arrow.js"></script>
      <script type="text/javascript" src="../scripts/axes.js"></script>
      <script type="text/javascript" src="../scripts/matrix.js"></script>
      <script type="text/javascript">
            var drawRectangle = function(x1, y1, x2, y2, x3, y3, x4, y4) {
```

```
        ctx.beginPath();
        ctx.moveTo(x0 + x1 * 30, y0 - y1 * 30);
        ctx.fillText("A", x0 + x1 * 30, y0 - y1 * 30 + 5);
        ctx.lineTo(x0 + x2 * 30, y0 - y2 * 30);
        ctx.fillText("B", x0 + x2 * 30, y0 - y2 * 30 + 5);
        ctx.lineTo(x0 + x3 * 30, y0 - y3 * 30);
        ctx.fillText("C", x0 + x3 * 30, y0 - y3 * 30 - 5);
        ctx.lineTo(x0 + x4 * 30, y0 - y4 * 30);
        ctx.closePath();
        ctx.stroke();
}
var cv = document.getElementById('canvas');
var ctx = cv.getContext('2d');
drawAxes("cartesian");
drawRectangle(2, 5, 5, 5, 5, 8, 2, 8);
var tpoint = [ 3.5, 6.5, 1 ];
var shx = 0.5;
var shy = 0;
var pointa = [ 2, 5, 1 ];
var mata = new Matrix(pointa, 1)
var transa1 = new Matrix();
transa1.setToIdentity();
var transa2 = new Matrix();
transa2.setToIdentity();
var transa3 = new Matrix();
transa3.setToIdentity()
transa1.setToTranslate(-tpoint[0], -tpoint[1]);
var aa1 = transa1.multiply2(mata);
transa2.setToShear(shx, shy);
var aa2 = transa2.multiply2(aa1);
transa3.setToTranslate(tpoint[0], tpoint[1]);
var aa3 = transa3.multiply2(aa2);
ctx.fillText(aa3.toString(), 580, 100);
//-
var pointb = [ 5, 5, 1 ];
var matb = new Matrix(pointb, 1)
var transb1 = new Matrix();
transb1.setToIdentity();
var transb2 = new Matrix();
transb2.setToIdentity();
var transb3 = new Matrix();
transb3.setToIdentity();
transb1.setToTranslate(-tpoint[0], -tpoint[1]);
var bb1 = transb1.multiply2(matb);
transb2.setToShear(shx, shy);
var bb2 = transb2.multiply2(bb1);
transb3.setToTranslate(tpoint[0], tpoint[1]);
var bb3 = transb3.multiply2(bb2);
ctx.fillText(bb3.toString(), 580, 120);
//-
var pointc = [ 5, 8, 1 ];
var matc = new Matrix(pointc, 1)
var transc1 = new Matrix();
transc1.setToIdentity();
var transc2 = new Matrix();
transc2.setToIdentity();
var transc3 = new Matrix();
transc3.setToIdentity();
transc1.setToTranslate(-tpoint[0], -tpoint[1]);
```

```
            var cc1 = transc1.multiply2(matc);
            transc2.setToShear(shx, shy);
            var cc2 = transc2.multiply2(cc1);
            transc3.setToTranslate(tpoint[0], tpoint[1]);
            var cc3 = transc3.multiply2(cc2);
            ctx.fillText(cc3.toString(), 580, 140);
            //-
            var pointd = [ 2, 8, 1 ];
            var matd = new Matrix(pointd, 1)
            var transd1 = new Matrix();
            transd1.setToIdentity();
            var transd2 = new Matrix();
            transd2.setToIdentity();
            var transd3 = new Matrix();
            transd3.setToIdentity();
            transd1.setToTranslate(-tpoint[0], -tpoint[1]);
            var dd1 = transd1.multiply2(matd);
            transd2.setToShear(shx, shy);
            var dd2 = transd2.multiply2(dd1);
            transd3.setToTranslate(tpoint[0], tpoint[1]);
            var dd3 = transd3.multiply2(dd2);
            ctx.fillText(dd3.toString(), 580, 160);
            //-
            drawRectangle(aa3.getMN(0, 0), aa3.getMN(1, 0), bb3.getMN(0, 0), bb3
                        .getMN(1, 0), cc3.getMN(0, 0), cc3.getMN(1, 0),
                        dd3.getMN(0, 0), dd3.getMN(1, 0));
      </script>
</body>

</html>
```

Listing111

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="200" height="200">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
            z kontekstem &#034;2d&#034;</canvas>
      <script type="text/javascript" src="../scripts/bezier.js"></script>
      <script type="text/javascript">
            bernstein(1, 0, 200);
      </script>
</body>

</html>
```

Zawartość skryptu bezier.js

```
var cv = document.getElementById('canvas');
var ctx = cv.getContext('2d');
```

```
var width = cv.width;
var height = cv.height;
var factorial = function(n) {
	if (n > 20) {
		throw new Error("zbyt du►a liczba");
	}
	if (n == 0) {
		return 1;
	}
	var sum = 1;
	for ( var i = 1; i < n + 1; i++) {
		sum *= i;
	}
	return sum;
};
var npok = function(n, k) {
	if (k > n) {
		return 0;
	} else if (k == 0 || k == n) {
		return 1;
	} else if (k == 1 || k == (n - 1)) {
		return n;
	} else {
		var b = factorial(k);
		var c = factorial(n - k);
		var a = b;
		for ( var i = k + 1; i < n + 1; i++) {
			a *= i;
		}
		return a / (b * c);
	}
};
var index = function(row, col, cols) {
	return row * cols + col;
};
var cutDecimal = function(nr) {
	var temp = nr.toString();
	var temp1 = temp.indexOf(".");
	var temp2 = "";
	if (temp1 > -1) {
		temp2 = temp.substr(0, temp1);
	} else {
		temp2 = temp;
	}
	return parseInt(temp2);
};
var bernsteinTValue = function(n, k, t) {
	var btv = npok(n, k) * Math.pow(t, k) * Math.pow(1.0 - t, n - k);
	if (btv < 0) {
		btv = 0;
	}
	return btv;
};
var bernstein = function(n, k, liczbaPunktow) {
	var t;
	var b;
	var img1 = ctx.getImageData(0, 0, width, height);
	var idata = img1.data;
	var a = npok(n, k);
```

```
	for ( var i = 0; i < liczbaPunktow; i++) {
		t = i * 1.0 / liczbaPunktow;
		b = a * Math.pow(t, k) * Math.pow(1.0 - t, n - k);
		if (b <= 0) {
			b = 0;
		}
		var j = 4 * index(cutDecimal(height - b * liczbaPunktow),
cutDecimal(t
				* liczbaPunktow), width);
		idata[j] = 0;
		idata[j + 1] = 0;
		idata[j + 2] = 255;
		idata[j + 3] = 255;
	}
	ctx.putImageData(img1, 0, 0);
};
var Point = function(x, y) {
	this.x = x;
	this.y = y;
};
Point.prototype.toString = function() {
	return "[" + this.x + ", " + this.y + "]";
};
var qbezierValue1 = function(points, t) {
	var ax = points[0].x;
	var ay = points[0].y;
	var bx = points[1].x;
	var by = points[1].y;
	var cx = points[2].x;
	var cy = points[2].y;
	var x1 = ax - 2 * bx + cx;
	var x2 = -2 * ax + 2 * bx;
	var y1 = ay - 2 * by + cy;
	var y2 = -2 * ay + 2 * by;
	var pt = t * t;
	var x = pt * x1 + t * x2 + ax;
	var y = pt * y1 + t * y2 + ay;
	return new Point(x, y);
};
var qbezierValue2 = function(points, t) {
	var ax = points[0].x;
	var ay = points[0].y;
	var bx = points[1].x;
	var by = points[1].y;
	var cx = points[2].x;
	var cy = points[2].y;
	var vx = [ ax, bx, cx ];
	var vy = [ ay, by, cy ];
	var coeffs = [ 1, -2, 1, -2, 2, 0, 1, 0, 0 ];
	var ts = [ Math.pow(t, 2), t, 1 ];
	var mvx = new Matrix(vx, 1);
	var mvy = new Matrix(vy, 1);
	var mcoeffs = new Matrix(coeffs, 3);
	var mts = new Matrix(ts, 3);
	var x = (mts.multiply2(mcoeffs.multiply2(mvx))).getMN(0, 0);
	var y = (mts.multiply2(mcoeffs).multiply2(mvy)).getMN(0, 0);
	return new Point(x, y);
};
var qbezierValue3 = function(points, t) {
```

```
        var ax = points[0].x;
        var ay = points[0].y;
        var bx = points[1].x;
        var by = points[1].y;
        var cx = points[2].x;
        var cy = points[2].y;
        var pt1 = 1.0 - t;
        var pt2 = 2 * t * pt1;
        var pt3 = pt1 * pt1;
        var pt4 = t * t;
        var x = pt3 * ax + pt2 * bx + pt4 * cx;
        var y = pt3 * ay + pt2 * by + pt4 * cy;
        return new Point(x, y);
    };
    var cbezierValue1 = function(points, t) {
        var ax = points[0].x;
        var ay = points[0].y;
        var bx = points[1].x;
        var by = points[1].y;
        var cx = points[2].x;
        var cy = points[2].y;
        var dx = points[3].x;
        var dy = points[3].y;
        var x1 = -ax + 3 * bx - 3 * cx + dx;
        var y1 = -ay + 3 * by - 3 * cy + dy;
        var x2 = 3 * ax - 6 * bx + 3 * cx;
        var y2 = 3 * ay - 6 * by + 3 * cy;
        var x3 = -3 * ax + 3 * bx;
        var y3 = -3 * ay + 3 * by;
        var pt2 = t * t;
        var pt3 = pt2 * t;
        var x = pt3 * x1 + pt2 * x2 + t * x3 + ax;
        var y = pt3 * y1 + pt2 * y2 + t * y3 + ay;
        return new Point(x, y);
    };
    var cbezierValue2 = function(points, t) {
        var ax = points[0].x;
        var ay = points[0].y;
        var bx = points[1].x;
        var by = points[1].y;
        var cx = points[2].x;
        var cy = points[2].y;
        var dx = points[3].x;
        var dy = points[3].y;
        var vx = [ ax, bx, cx, dx ];
        var vy = [ ay, by, cy, dy ];
        var coeffs = [ -1, 3, -3, 1, 3, -6, 3, 0, -3, 3, 0, 0, 1, 0, 0, 0 ];
        var ts = [ Math.pow(t, 3), Math.pow(t, 2), t, 1 ];
        var mvx = new Matrix(vx, 1);
        var mvy = new Matrix(vy, 1);
        var mcoeffs = new Matrix(coeffs, 4);
        var mts = new Matrix(ts, 4);
        var x = (mts.multiply2(mcoeffs.multiply2(mvx))).getMN(0, 0);
        var y = (mts.multiply2(mcoeffs).multiply2(mvy)).getMN(0, 0);
        return new Point(x, y);
    };
    var cbezierValue3 = function(points, t) {
        var ax = points[0].x;
        var ay = points[0].y;
```

```
        var bx = points[1].x;
        var by = points[1].y;
        var cx = points[2].x;
        var cy = points[2].y;
        var dx = points[3].x;
        var dy = points[3].y;
        var pt1 = 1.0 - t;
        var pt2 = pt1 * pt1;
        var pt3 = pt2 * pt1;
        var t2 = t * t;
        var t3 = t2 * t;
        var x = pt3 * ax + 3 * t * pt2 * bx + 3 * t2 * pt1 * cx + t3 * dx;
        var y = pt3 * ay + 3 * t * pt2 * by + 3 * t2 * pt1 * cy + t3 * dy;
        return new Point(x, y);
};
var drawQBezier = function(points, strokeStyle, visible) {
        ctx.save();
        ctx.beginPath();
        ctx.strokeStyle = strokeStyle;
        ctx.moveTo(points[0].x, points[0].y);
        ctx.quadraticCurveTo(points[1].x, points[1].y, points[2].x, points[2].y);
        ctx.stroke();
        ctx.restore();
        if (visible) {
                drawPoints(points, strokeStyle);
        }
};
var drawCBezier = function(points, strokeStyle, visible) {
        ctx.save();
        ctx.beginPath();
        ctx.strokeStyle = strokeStyle;
        ctx.moveTo(points[0].x, points[0].y);
        ctx.bezierCurveTo(points[1].x, points[1].y, points[2].x, points[2].y,
                        points[3].x, points[3].y);
        ctx.stroke();
        ctx.restore();
        if (visible) {
                drawPoints(points, strokeStyle);
        }
};
var drawPoints = function(points, style) {
        ctx.save();
        ctx.beginPath();
        ctx.fillStyle = style;
        for ( var i = 0; i < points.length; i++) {
                ctx.arc(points[i].x, points[i].y, 3, 0, 2 * Math.PI, true);
        }
        ctx.fill();
        ctx.restore();
};
// -

var tValue = function(points) {
        var maxx = -1000000;
        var maxy = -1000000;
        var minx = 1000000;
        var miny = 1000000;
        for ( var i = 0; i < points.length; i++) {
                var val = points[i];
```

```
            maxx = Math.max(val.x, maxx);
            maxy = Math.max(val.y, maxy);
            minx = Math.min(val.x, minx);
            miny = Math.min(val.y, miny);
        }
        return Math.max(Math.abs(maxx) - Math.abs(minx), Math.abs(maxy)
                    - Math.abs(miny));
};
var drawMyQBezier1 = function(points, style, visible) {
        var x1 = points[0].x - 2 * points[1].x + points[2].x;
        var x2 = -2 * points[0].x + 2 * points[1].x;
        var y1 = points[0].y - 2 * points[1].y + points[2].y;
        var y2 = -2 * points[0].y + 2 * points[1].y;
        var extr = tValue(points);
        var te = 1.0 / extr;
        var img1 = ctx.getImageData(0, 0, width, height);
        var idata = img1.data;
        var t = 0;
        for ( var i = 0; i < extr; i++) {
            t = i * te;
            var pt = t * t;
            var ptx = pt * x1;
            var pty = pt * y1;
            var x = ptx + t * x2 + points[0].x;
            var y = pty + t * y2 + points[0].y;
            var j = 4 * index(cutDecimal(y), cutDecimal(x), width);
            idata[j] = 0;
            idata[j + 1] = 0;
            idata[j + 2] = 255;
            idata[j + 3] = 255;
        }
        ctx.putImageData(img1, 0, 0);
        if (visible) {
            drawPoints(points, style);
        }
};
var drawMyQBezier2 = function(points, style, visible) {
        var x1 = points[0].x - 2 * points[1].x + points[2].x;
        var x2 = -2 * points[0].x + 2 * points[1].x;
        var y1 = points[0].y - 2 * points[1].y + points[2].y;
        var y2 = -2 * points[0].y + 2 * points[1].y;
        var extr = tValue(points);
        var te = 1.0 / extr;
        var t = 0;
        ctx.save();
        ctx.beginPath();
        ctx.strokeStyle = style;
        ctx.moveTo(points[0].x, points[0].y);
        for ( var i = 0; i < extr; i++) {
            t = i * te;
            var pt = t * t;
            var ptx = pt * x1;
            var pty = pt * y1;
            var x = ptx + t * x2 + points[0].x;
            var y = pty + t * y2 + points[0].y;
            ctx.lineTo(x, y);
        }
        ctx.stroke();
        ctx.restore();
```

```
        if (visible) {
                drawPoints(points, style);
        }
};
var drawMyCBezier1 = function(points, style, visible) {
        var x1 = -points[0].x + 3 * points[1].x - 3 * points[2].x + points[3].x;
        var y1 = -points[0].y + 3 * points[1].y - 3 * points[2].y + points[3].y;
        var x2 = 3 * points[0].x - 6 * points[1].x + 3 * points[2].x;
        var y2 = 3 * points[0].y - 6 * points[1].y + 3 * points[2].y;
        var x3 = -3 * points[0].x + 3 * points[1].x;
        var y3 = -3 * points[0].y + 3 * points[1].y;
        var extr = tValue(points);
        var te = 1.0 / extr;
        var img1 = ctx.getImageData(0, 0, width, height);
        var idata = img1.data;
        var t = 0;
        for ( var i = 0; i < extr; i++) {
                t = i * te;
                var pt2 = t * t;
                var pt3 = pt2 * t;
                var x = pt3 * x1 + pt2 * x2 + t * x3 + points[0].x;
                var y = pt3 * y1 + pt2 * y2 + t * y3 + points[0].y;
                var j = 4 * index(cutDecimal(y), cutDecimal(x), width);
                idata[j] = 0;
                idata[j + 1] = 0;
                idata[j + 2] = 255;
                idata[j + 3] = 255;
        }
        ctx.putImageData(img1, 0, 0);
        if (visible) {
                drawPoints(points, style);
        }
};
var drawMyCBezier2 = function(points, style, visible) {
        var x1 = -points[0].x + 3 * points[1].x - 3 * points[2].x + points[3].x;
        var y1 = -points[0].y + 3 * points[1].y - 3 * points[2].y + points[3].y;
        var x2 = 3 * points[0].x - 6 * points[1].x + 3 * points[2].x;
        var y2 = 3 * points[0].y - 6 * points[1].y + 3 * points[2].y;
        var x3 = -3 * points[0].x + 3 * points[1].x;
        var y3 = -3 * points[0].y + 3 * points[1].y;
        var extr = tValue(points);
        var te = 1.0 / extr;
        var t = 0;
        ctx.save();
        ctx.beginPath();
        ctx.strokeStyle = style;
        ctx.moveTo(points[0].x, points[0].y);
        for ( var i = 0; i < extr; i++) {
                t = i * te;
                var pt2 = t * t;
                var pt3 = pt2 * t;
                var x = pt3 * x1 + pt2 * x2 + t * x3 + points[0].x;
                var y = pt3 * y1 + pt2 * y2 + t * y3 + points[0].y;
                ctx.lineTo(x, y);
        }
        ctx.stroke();
        ctx.restore();
        if (visible) {
                drawPoints(points, style);
```

```
        }
        ctx.restore();
};
var rotateBezier = function(points, angle) {
        var points1 = new Array();
        for ( var i = 0; i < points.length; i++) {
                points1.push(new Point(points[i].x * Math.cos(angle) + points[i].y
                                * Math.sin(angle), -points[i].x * Math.sin(angle) +
points[i].y
                                * Math.cos(angle)));
        }
        return points1;
};
var drawNBezier = function(points, style, visible) {
        var n = points.length - 1;
        var extr = tValue(points);
        var te = 1.0 / extr;
        var t = 0;
        ctx.save();
        ctx.beginPath();
        ctx.strokeStyle = style;
        ctx.moveTo(points[0].x, points[0].y);
        for ( var i = 0; i < extr; i++) {
                t = i * te;
                var x = 0;
                var y = 0;
                for ( var k = 0; k < points.length; k++) {
                        var np = npok(n, k) * Math.pow(t, k) * Math.pow(1 - t, n - k);
                        x += np * points[k].x;
                        y += np * points[k].y;
                }
                ctx.lineTo(x, y);
        }
        ctx.stroke();
        ctx.restore();
        if (visible) {
                drawPoints(points, style);
        }
        ctx.restore();
};

var cloneArray2 = function(array) {
        var len = array.length;
        var arr = new Array(len);
        for ( var i = 0; i < len; i++) {
                arr[i] = new Point(array[i].x, array[i].y);
        }
        return arr;
};
var casteljauTValue = function(points, t) {
        var temp = cloneArray2(points);
        var len = points.length;
        var t1 = 1.0 - t;
        for ( var i = 0; i < len - 1; i++) {
                for ( var j = 0; j < len - 1 - i; j++) {
                        temp[j].x = temp[j].x * t1 + t * temp[j + 1].x;
                        temp[j].y = temp[j].y * t1 + t * temp[j + 1].y;
                }
        }
```

```
        return temp[0];
};
var divideCasteljau = function(points) {
        var temp = cloneArray2(points);
        var temp1 = cloneArray2(points).reverse();
        var len = points.length;
        var t = 0.3;
        var t1 = 1.0 - t;
        for ( var i = 0; i < len - 1; i++) {
                for ( var j = 0; j < len - 1 - i; j++) {
                        temp[j].x = temp[j].x * t1 + t * temp[j + 1].x;
                        temp[j].y = temp[j].y * t1 + t * temp[j + 1].y;
                        temp1[j].x = temp1[j + 1].x * t1 + t * temp1[j].x;
                        temp1[j].y = temp1[j + 1].y * t1 + t * temp1[j].y;
                }
        }
        var arr = new Array(2);
        arr[0] = temp1.reverse();
        arr[1] = temp;
        return arr;
};
var drawCircle = function(startX, startY, r, style) {
        ctx.save();
        ctx.beginPath();
        ctx.strokeStyle = style;
        ctx.moveTo(startX + r, startY);
        ctx.quadraticCurveTo(startX + r, startY + r * Math.tan(Math.PI / 8), startX
                        + r * Math.cos(Math.PI / 4), startY + r * Math.cos(Math.PI /
4));
        ctx.quadraticCurveTo(startX + r * Math.tan(Math.PI / 8), startY + r,
                        startX, startY + r);
        ctx.quadraticCurveTo(startX - r * Math.tan(Math.PI / 8), startY + r, startX
                        - r * Math.cos(Math.PI / 4), startY + r * Math.cos(Math.PI /
4));
        ctx.quadraticCurveTo(startX - r, startY + r * Math.tan(Math.PI / 8), startX
                        - r, startY);
        ctx.quadraticCurveTo(startX - r, startY - r * Math.tan(Math.PI / 8), startX
                        - r * Math.cos(Math.PI / 4), startY - r * Math.cos(Math.PI /
4));
        ctx.quadraticCurveTo(startX - r * Math.tan(Math.PI / 8), startY - r,
                        startX, startY - r);
        ctx.quadraticCurveTo(startX + r * Math.tan(Math.PI / 8), startY - r, startX
                        + r * Math.cos(Math.PI / 4), startY - r * Math.cos(Math.PI /
4));
        ctx.quadraticCurveTo(startX + r, startY - r * Math.tan(Math.PI / 8), startX
                        + r, startY);
        ctx.stroke();
        ctx.restore();
};
var rBezierTValue = function(n, k, t, weights) {
        var arr = new Array();
        var mian = 0.0;
        var sum = 0.0;
        for ( var i = 0; i < n + 1; i++) {
                var a = npok(n, i) * Math.pow(t, i) * Math.pow(1.0 - t, n - i)
                                * weights[i];
                if (a < 0) {
                        a = 0;
                }
```

```
                arr.push(a);
                mian += a;
        }
        sum = arr[k] / mian;
        return sum;
};
var rBezierTValue2 = function(np, n, k, t, weights) {
        var arr = new Array();
        var mian = 0.0;
        var sum = 0.0;
        for ( var i = 0; i < n + 1; i++) {
                var a = np * Math.pow(t, i) * Math.pow(1.0 - t, n - i) * weights[i];
                if (a < 0) {
                        a = 0;
                }
                arr.push(a);
                mian += a;
        }
        sum = arr[k] / mian;
        return sum;
};
var rBezier = function(n, k, weights, liczbaPunktow) {
        var t;
        var img1 = ctx.getImageData(0, 0, width, height);
        var idata = img1.data;
        var np = npok(n, k);
        for ( var i = 0; i < liczbaPunktow; i++) {
                t = i * 1.0 / liczbaPunktow;
                var b = rBezierTValue2(np, n, k, t, weights);
                if (b < 0) {
                        b = 0;
                }
                var j = 4 * index(cutDecimal(height - b * liczbaPunktow),
cutDecimal(t
                                * liczbaPunktow), width);
                idata[j] = 0;
                idata[j + 1] = 0;
                idata[j + 2] = 255;
                idata[j + 3] = 255;
        }
        ctx.putImageData(img1, 0, 0);
};
var drawRatBezier2 = function(points, weights, style, visible) {
        var extr = tValue(points);
        var te = 1.0 / extr;
        ctx.save();
        ctx.beginPath();
        ctx.strokeStyle = style;
        ctx.moveTo(points[0].x, points[0].y);
        for ( var i = 0; i < extr + 1; i++) {
                var t = i * te;
                var t1 = 1.0 - t;
                var m0 = t1 * t1 * weights[0];
                var m1 = 2 * t * t1 * weights[1];
                var m2 = t * t * weights[2];
                var mian = m0 + m1 + m2;
                var w0 = m0 / mian;
                var w1 = m1 / mian;
                var w2 = m2 / mian;
```

```
            var x = w0 * points[0].x + w1 * points[1].x + w2 * points[2].x;
            var y = w0 * points[0].y + w1 * points[1].y + w2 * points[2].y;
            ctx.lineTo(x, y);
        }
        ctx.lineTo(points[2].x, points[2].y);
        ctx.stroke();
        ctx.restore();
        if (visible) {
            drawPoints(points, style);
        }
};
var drawRatBezier3 = function(points, weights, style, visible) {
        var extr = tValue(points);
        var te = 1.0 / extr;
        ctx.save();
        ctx.beginPath();
        ctx.strokeStyle = style;
        ctx.moveTo(points[0].x, points[0].y);
        for ( var i = 0; i < extr; i++) {
            var t = i * te;
            var t1 = 1.0 - t;
            var m0 = t1 * t1 * t1 * weights[0];
            var m1 = 3 * t * t1 * t1 * weights[1];
            var m2 = 3 * t * t * t1 * weights[2];
            var m3 = t * t * t * weights[3];
            var mian = m0 + m1 + m2 + m3;
            var w0 = m0 / mian;
            var w1 = m1 / mian;
            var w2 = m2 / mian;
            var w3 = m3 / mian;
            var x = w0 * points[0].x + w1 * points[1].x + w2 * points[2].x + w3
                        * points[3].x;
            var y = w0 * points[0].y + w1 * points[1].y + w2 * points[2].y + w3
                        * points[3].y;
            ctx.lineTo(x, y);
        }
        ctx.lineTo(points[3].x, points[3].y);
        ctx.stroke();
        ctx.restore();
        if (visible) {
            drawPoints(points, style);
        }
};

var drawRatBezierN = function(points, weights, style, visible) {
        var extr = tValue(points);
        var te = 1.0 / extr;
        var n = points.length - 1;
        ctx.save();
        ctx.beginPath();
        ctx.strokeStyle = style;
        ctx.moveTo(points[0].x, points[0].y);
        for ( var i = 0; i < extr + 1; i++) {
            var t = i * te;
            var t1 = 1.0 - t;
            var ms = new Array();
            for ( var j = 0; j < points.length; j++) {
                ms.push(npok(n, j) * Math.pow(t, j) * Math.pow(t1, n - j)
                            * weights[j]);
```

```
            }
            var mian = 0.0;
            for ( var k = 0; k < ms.length; k++) {
                  mian += ms[k];
            }
            var x = 0.0;
            var y = 0.0;
            for ( var m = 0; m < points.length; m++) {
                  x += ms[m] * points[m].x;
                  y += ms[m] * points[m].y;
            }
            x = cutDecimal(x / mian);
            y = cutDecimal(y / mian);
            ctx.lineTo(x, y);
            // x = 0.0;
            // y = 0.0;
      }
      ctx.lineTo(points[points.length - 1].x, points[points.length - 1].y);
      ctx.stroke();
      ctx.restore();
      if (visible) {
            drawPoints(points, style);
      }
};
var bezierDegreeUp = function(points) {
      var len = points.length;// 4
      var arr = new Array();
      arr[0] = new Point(points[0].x, points[0].y);
      for ( var i = 1; i < len; i++) {
            var a = i / len;
            var b = 1.0 - a;
            arr.push(new Point(a * points[i - 1].x + b * points[i].x, a
                        * points[i - 1].y + b * points[i].y));
      }
      arr.push(new Point(points[len - 1].x, points[len - 1].y));
      return arr;
};
var connectBezier = function(points, qoints) {
      var n = points.length - 1;
      var m = qoints.length - 1;
      points[n - 1].x = ((m + n) * qoints[0].x - m * qoints[1].x) / n;
      points[n - 1].y = ((m + n) * qoints[0].y - m * qoints[1].y) / n;

};
```

Listing112

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="200" height="200">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
```

```
            z kontekstem &#034;2d&#034;</canvas>
      <script type="text/javascript" src="../scripts/bezier.js"></script>
      <script type="text/javascript">
            bernstein(1, 1, 200);
      </script>
</body>

</html>
```

Listing113

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="200" height="200">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
            z kontekstem &#034;2d&#034;</canvas>
      <script type="text/javascript" src="../scripts/bezier.js"></script>
      <script type="text/javascript">
            bernstein(1, 0, 200);
            bernstein(1, 1, 200);
      </script>
</body>

</html>
```

Listing114

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="200" height="200">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
            z kontekstem &#034;2d&#034;</canvas>
      <script type="text/javascript" src="../scripts/bezier.js"></script>
      <script type="text/javascript">
            bernstein(2, 0, 200);
      </script>
</body>

</html>
```

Listing115

```
<!DOCTYPE html>
```

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="200" height="200">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/bezier.js"></script>
	<script type="text/javascript">
		bernstein(2, 1, 200);
	</script>
</body>

</html>
```

Listing116

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="200" height="200">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/bezier.js"></script>
	<script type="text/javascript">
		bernstein(2, 2, 200);
	</script>
</body>

</html>
```

Listing117

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="200" height="200">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/bezier.js"></script>
	<script type="text/javascript">
```

```
            bernstein(2, 0, 200);
            bernstein(2, 1, 200);
            bernstein(2, 2, 200);
      </script>
</body>

</html>
```

Listing118

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="200" height="200">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
            z kontekstem &#034;2d&#034;</canvas>
      <script type="text/javascript" src="../scripts/bezier.js"></script>
      <script type="text/javascript">
            bernstein(3, 0, 200);
            bernstein(3, 1, 200);
            bernstein(3, 2, 200);
            bernstein(3, 3, 200);
      </script>
</body>

</html>
```

Listing119

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
            z kontekstem &#034;2d&#034;</canvas>
      <script type="text/javascript" src="../scripts/bezier.js"></script>
      <script type="text/javascript" src="../scripts/matrix.js"></script>
      <script type="text/javascript">
            var cv = document.getElementById('canvas');
            var ctx = cv.getContext('2d');
            var points = [new Point(0,1), new Point(4,5), new Point(2,3)];
            var t = 0.36;
            ctx.fillText(qbezierValue1(points, t),20, 20);
```

```
	</script>


</body>

</html>
```

Listing120

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/bezier.js"></script>
	<script type="text/javascript" src="../scripts/matrix.js"></script>
	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		var points = [ new Point(0, 1), new Point(4, 5), new Point(2, 3) ];
		var t = 0.36;
		ctx.fillText(qbezierValue2(points, t), 20, 20);
	</script


</body>

</html>
```

Listing121

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/bezier.js"></script>
	<script type="text/javascript" src="../scripts/matrix.js"></script>
	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		var points = [ new Point(0, 1), new Point(4, 5), new Point(2, 3) ];
```

```
            var t = 0.36;
            ctx.fillText(qbezierValue3(points, t), 20, 20);
      </script>


</body>

</html>
```

Listing122

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
            z kontekstem &#034;2d&#034;</canvas>
      <script type="text/javascript" src="../scripts/bezier.js"></script>
      <script type="text/javascript" src="../scripts/matrix.js"></script>
      <script type="text/javascript">
            var cv = document.getElementById('canvas');
            var ctx = cv.getContext('2d');
            var points = [ new Point(0, 1), new Point(2, 3), new Point(6, 7),
                        new Point(4, 5) ];
            var t = 0.3;
            ctx.fillText(cbezierValue1(points, t), 20, 20);
      </script>


</body>

</html>
```

Listing123

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
            z kontekstem &#034;2d&#034;</canvas>
      <script type="text/javascript" src="../scripts/bezier.js"></script>
      <script type="text/javascript" src="../scripts/matrix.js"></script>
      <script type="text/javascript">
            var cv = document.getElementById('canvas');
```

```
            var ctx = cv.getContext('2d');
            var points = [ new Point(0, 1), new Point(2, 3), new Point(6, 7),
                        new Point(4, 5) ];
            var t = 0.3;
            ctx.fillText(cbezierValue2(points, t), 20, 20);
      </script>


</body>

</html>
```

Listing124

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
            z kontekstem &#034;2d&#034;</canvas>
      <script type="text/javascript" src="../scripts/bezier.js"></script>
      <script type="text/javascript" src="../scripts/matrix.js"></script>
      <script type="text/javascript">
            var cv = document.getElementById('canvas');
            var ctx = cv.getContext('2d');
            var points = [ new Point(0, 1), new Point(2, 3), new Point(6, 7),
                        new Point(4, 5) ];
            var t = 0.3;
            ctx.fillText(cbezierValue3(points, t), 20, 20);
      </script>


</body>

</html>
```

Listing125

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="500" height="500">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
            z kontekstem &#034;2d&#034;</canvas>
      <script type="text/javascript" src="../scripts/bezier.js"></script>
```

```
	<script type="text/javascript">
		var points = [ new Point(60, 60), new Point(120, 150),
				new Point(180, 90) ];
		drawQBezier(points, "blue", true);
	</script>


</body>

</html>
```

Listing126

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="500" height="500">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/bezier.js"></script>
	<script type="text/javascript">
		var points = [ new Point(60, 60), new Point(120, 150),
				new Point(180, 90), new Point(150, 180) ];
		drawCBezier(points, "blue", true);
	</script>


</body>

</html>
```

Listing127

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="500" height="500">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/bezier.js"></script>
	<script type="text/javascript">
		var points = [ new Point(60, 60), new Point(120, 150),
				new Point(180, 90) ];
		drawMyQBezier1(points, "blue", true);
	</script>
```

```
</body>

</html>
```

Listing128

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="500" height="500">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/bezier.js"></script>
	<script type="text/javascript">
		var points = [ new Point(60, 60), new Point(120, 150),
				new Point(180, 90) ];
		drawMyQBezier2(points, "blue", true);
	</script>


</body>

</html>
```

Listing129

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="500" height="500">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/bezier.js"></script>
	<script type="text/javascript">
		var points = [ new Point(60, 60), new Point(120, 150),
				new Point(180, 90), new Point(150, 180) ];
		drawMyCBezier1(points, "blue", true);
	</script>


</body>

</html>
```

Listing130

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="500" height="500">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
            z kontekstem &#034;2d&#034;</canvas>
      <script type="text/javascript" src="../scripts/bezier.js"></script>
      <script type="text/javascript">
            var points = [ new Point(60, 60), new Point(120, 150),
                        new Point(180, 90), new Point(150, 180) ];
            drawMyCBezier2(points, "blue", true);
      </script>


</body>

</html>
```

Listing131

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="500" height="500">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
            z kontekstem &#034;2d&#034;</canvas>
      <script type="text/javascript" src="../scripts/bezier.js"></script>
      <script type="text/javascript">
            var points = [ new Point(60, 60), new Point(120, 150),
                        new Point(180, 90), new Point(60, 60) ];
            drawMyCBezier2(points, "blue", true);
      </script>


</body>

</html>
```

Listing132

```
<!DOCTYPE html>
<html>
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="500" height="500">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/bezier.js"></script>
	<script type="text/javascript">
		var points = [ new Point(60, 60), new Point(120, 150),
				new Point(180, 90), new Point(60, 60) ];
		drawMyCBezier2(points, "blue", true);
		var points1 = rotateBezier(points, -Math.PI / 8.0);
		drawMyCBezier2(points1, "red", true)
	</script>


</body>

</html>
```

Listing133

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="500" height="500">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/bezier.js"></script>
	<script type="text/javascript">
		var points = [ new Point(60, 60), new Point(120, 150),
				new Point(180, 90), new Point(70, 160), new Point(220,
20),
				new Point(135, 135), new Point(20, 100), new Point(320,
180) ];
		drawNBezier(points, "blue", true);
	</script>


</body>

</html>
```

Listing134

```
<!DOCTYPE html>
<html>
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="800" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/bezier.js"></script>
	<script type="text/javascript" src="../scripts/matrix.js"></script>
	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		var points = [ new Point(0, 1), new Point(2, 3), new Point(6, 7),
				new Point(4, 5) ];
		var t = 0.3;
		ctx.fillText(casteljauTValue(points, t), 20, 20);
	</script>


</body>

</html>
```

Listing135

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="1200" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/bezier.js"></script>
	<script type="text/javascript" src="../scripts/matrix.js"></script>
	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		var points = [ new Point(60, 60), new Point(120, 150),
				new Point(180, 90), new Point(150, 180) ];
		var arr = divideCasteljau(points);
		var points1 = arr[0];
		var points2 = arr[1];
		//	drawMyCBezier2(points, "blue", true);
		drawMyCBezier2(points1, "green", true);
		drawMyCBezier2(points2, "red", true);
	</script>


</body>

</html>
```

Listing136

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="1200" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/bezier.js"></script>
	<script type="text/javascript" src="../scripts/matrix.js"></script>
	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		var points = [ new Point(60, 60), new Point(120, 150),
				new Point(180, 90), new Point(70, 160), new Point(220,
20),
				new Point(135, 135), new Point(20, 100), new Point(320,
180) ];
		var arr = divideCasteljau(points);
		var points1 = arr[0];
		var points2 = arr[1];
		drawNBezier(points1, "green", true);
		drawNBezier(points2, "red", true);
	</script>


</body>

</html>
```

Listing137

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="1200" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/bezier.js"></script>
	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		drawCircle(150, 150, 100, "magenta");
	</script>
```

```
</body>

</html>
```

Listing138

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="200" height="200">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/bezier.js"></script>
	<script type="text/javascript">
		var weights = [ 1, 6, 1 ];
		rBezier(2, 1, weights, 200);
	</script>
</body>

</html>
```

Listing139

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="200" height="200">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/bezier.js"></script>
	<script type="text/javascript">
		var weights = [ 1, 6, 3, 1 ];
		rBezier(3, 0, weights, 200);
		rBezier(3, 1, weights, 200);
		rBezier(3, 2, weights, 200);
		rBezier(3, 3, weights, 200);
	</script>
</body>

</html>
```

Listing140

```
<!DOCTYPE html>
<html>
```

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="500" height="500">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/bezier.js"></script>
	<script type="text/javascript">
		var points = [ new Point(60, 160), new Point(120, 60),
				new Point(180, 160) ];
		var weights = [ 1, 1, 1 ]
		drawRatBezier2(points, weights, "blue", true);
	</script>


</body>

</html>
```

Listing141

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="500" height="500">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/bezier.js"></script>
	<script type="text/javascript">
		var points = [ new Point(160, 160), new Point(220, 60),
				new Point(280, 160) ];
		var weights = [ 1, -1, 1 ]
		drawRatBezier2(points, weights, "blue", true);
	</script>


</body>

</html>
```

Listing142

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">
```

```
<title> </title>
</head>
<body>
	<canvas id="canvas" width="500" height="500">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/bezier.js"></script>
	<script type="text/javascript">
		var points = [ new Point(160, 160), new Point(220, 60),
				new Point(280, 160) ];
		var weights = [ 1, 3, 1 ]
		drawRatBezier2(points, weights, "blue", true);
	</script>


</body>

</html>
```

Listing143

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="500" height="500">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/bezier.js"></script>
	<script type="text/javascript">
		var points = [ new Point(160, 160), new Point(220, 60),
				new Point(280, 160) ];
		var weights = [ 1, 0.3, 1 ]
		drawRatBezier2(points, weights, "blue", true);
	</script>


</body>

</html>
```

Listing144

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
```

```
<body>
	<canvas id="canvas" width="500" height="500">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/bezier.js"></script>
	<script type="text/javascript">
		var points = [ new Point(160, 160), new Point(220, 60),
				new Point(280, 160) ];
		var weights = [ 1, -0.3, 1 ]
		drawRatBezier2(points, weights, "blue", true);
	</script>


</body>

</html>
```

Listing145

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="500" height="500">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/bezier.js"></script>
	<script type="text/javascript">
		var points = [ new Point(160, 160), new Point(220, 60),
				new Point(280, 160) ];
		var weights = [ 1, 0, 1 ]
		drawRatBezier2(points, weights, "blue", true);
	</script>


</body>

</html>
```

Listing146

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="500" height="500">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
```

```
            z kontekstem &#034;2d&#034;</canvas>
      <script type="text/javascript" src="../scripts/bezier.js"></script>
      <script type="text/javascript">
            var points = [ new Point(160, 160), new Point(220, 100),
                        new Point(280, 160) ];
            var weights = [ 1, Math.sqrt(2) / 2.0, 1 ];
            var weights1 = [ 1, -Math.sqrt(2) / 2.0, 1 ];
            drawRatBezier2(points, weights, "blue", true);
            drawRatBezier2(points, weights1, "blue", true);
      </script>


</body>

</html>
```

Listing147

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="500" height="500">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
            z kontekstem &#034;2d&#034;</canvas>
      <script type="text/javascript" src="../scripts/bezier.js"></script>
      <script type="text/javascript">
            var points = [ new Point(160, 160), new Point(220, 100),
                        new Point(280, 160), new Point(230, 60) ];
            var weights = [ 1, 3, 6, 1 ];
            drawRatBezier3(points, weights, "blue", true);
      </script>


</body>

</html>
```

Listing148

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="500" height="500">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
            z kontekstem &#034;2d&#034;</canvas>
```

```
	<script type="text/javascript" src="../scripts/bezier.js"></script>
	<script type="text/javascript">
		var points = [ new Point(160, 160), new Point(120, 300),
				new Point(220, 100), new Point(280, 160), new Point(230,
60),
				new Point(100, 200), new Point(200, 320) ];
		var weights = [ 1, 3, 6, 1, 8, 6, 1 ];
		drawRatBezierN(points, weights, "blue", true);
	</script>


</body>

</html>
```

Listing149

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="1200" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/bezier.js"></script>
	<script type="text/javascript" src="../scripts/matrix.js"></script>
	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		var points = [ new Point(60, 60), new Point(120, 150),
				new Point(180, 90), new Point(70, 160), new Point(220,
20),
				new Point(135, 135), new Point(20, 100), new Point(320,
180) ];
		drawNBezier(points, "green", true);
	</script>


</body>

</html>
```

Listing150

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
```

```
<body>
	<canvas id="canvas" width="1200" height="800">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/bezier.js"></script>
	<script type="text/javascript" src="../scripts/matrix.js"></script>
	<script type="text/javascript">
		var cv = document.getElementById('canvas');
		var ctx = cv.getContext('2d');
		var points = [ new Point(60, 60), new Point(120, 150),
				new Point(180, 90), new Point(70, 160), new Point(220,
20),
				new Point(135, 135), new Point(20, 100), new Point(320,
180) ];
		var points2 = bezierDegreeUp(points);
		drawNBezier(points2, "red", true);
	</script>


</body>

</html>
```

Listing151

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
	<canvas id="canvas" width="500" height="500">Zawartość możesz zobaczyć w
		przeglądarce obsługującej element &lt;canvas&gt;
		z kontekstem &#034;2d&#034;</canvas>
	<script type="text/javascript" src="../scripts/bezier.js"></script>
	<script type="text/javascript">
		var points = [ new Point(60, 160), new Point(120, 150), new
Point(180, 200), new Point(70, 170), new Point(220, 120) ];
		var qoints = [ new Point(220, 120), new Point(20, 200), new
Point(320, 280), new Point(120, 140) ];
		drawNBezier(points, "blue", true);
		drawNBezier(qoints, "red", true);
	</script>


</body>

</html>
```

Listing152

```
<!DOCTYPE html>
<html>
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="../styles/canvas2d.css">

<title> </title>
</head>
<body>
      <canvas id="canvas" width="500" height="500">Zawartość możesz zobaczyć w
            przeglądarce obsługującej element &lt;canvas&gt;
            z kontekstem &#034;2d&#034;</canvas>
      <script type="text/javascript" src="../scripts/bezier.js"></script>
      <script type="text/javascript">
            var points = [ new Point(60, 160), new Point(120, 150),
                        new Point(180, 200), new Point(70, 170), new Point(220,
120) ];
            var qoints = [ new Point(220, 120), new Point(20, 200),
                        new Point(320, 280), new Point(120, 140) ];
            connectBezier(points, qoints);
            drawNBezier(points, "blue", true);
            drawNBezier(qoints, "red", true);
      </script


</body>

</html>
```

Dodatek 8. Palety kolorów

Paleta 16 kolorów nazwanych

W czasach 16-bitowych obliczeń w informatyce stosowano 16 kolorów mających swoje nazwy.

Standard gwarantuje, że nazwy te będą zawsze rozpoznawane.

Tab. Kolory nazwane (X)HTML

Nazwa EN	Nazwa PL	Kolor	HEX	R	G	B
black	czarny		#000000	0	0	0;
silver	srebrny		#C0C0C0	192	192	192
gray	szary		#808080	128	128	128
white	biały		#FFFFFF	255	255	255
maroon	rdzawoczerwony (wiśniowy)		#800000	128	0	0
red	czerwony		#FF0000	255	0	0
purple	fioletowy		#808080	128	0	128
fuchsia	fuksja		#FF00FF	255	0	255
green	zielony		#008000	0	128	0
lime	limonkowy		#00FF00	0	255	0
olive	oliwkowy		#808000	128	128	0
yellow	żółty		#FFFF00	255	255	0
navy	granatowy		#000080	0	0	128
blue	niebieski		#0000FF	0	0	255
teal	morski		#008080	0	128	128
aqua	cyan		#00FFFF	0	255	255

Paleta Web Safe Colors

W czasach gdy powstały pierwsze graficzne przeglądarki internetowe większość użytkowników komputerów korzystała z 8 bitowej palety kolorów. Aby zapewnić jednakowe wyświetlanie się kolorów na różnych komputerach przy tak ograniczonych możliwościach sprzętowych, opracowano specjalną paletę 216 kolorów, która miała wyglądać identycznie na wszystkich komputerach. Jej szczególną cechą jest to że każda składowa koloru RGB może przyjąć tylko jedną z sześciu wartości szesnastkowych: 0x00, 0x33, 0x66, 0x99, 0xCC lub 0xFF.

Aktualnie ma raczej znaczenie historyczne, chociaż ciągle jest obecna jako opcja w wielu programach graficznych.

Tab. Paleta Web Safe Colors

Kolor	HEX	R	G	B
	#000000	0	0	0

#000033	0	0	51
#000066	0	0	102
#000099	0	0	153
#0000CC	0	0	204
#0000FF	0	0	255
#003300	0	51	0
#003333	0	51	51
#003366	0	51	102
#003399	0	51	153
#0033CC	0	51	204
#0033FF	0	51	255
#006600	0	102	0
#006633	0	102	51
#006666	0	102	102
#006699	0	102	153
#0066CC	0	102	204
#0066FF	0	102	255
#009900	0	153	0
#009933	0	153	51
#009966	0	153	102
#009999	0	153	153
#0099CC	0	153	204
#0099FF	0	153	255
#00CC00	0	204	0
#00CC33	0	204	51
#00CC66	0	204	102
#00CC99	0	204	153
#00CCCC	0	204	204
#00CCFF	0	204	255
#00FF00	0	255	0
#00FF33	0	255	51
#00FF66	0	255	102
#00FF99	0	255	153
#00FFCC	0	255	204
#00FFFF	0	255	255
#330000	51	0	0
#330033	51	0	51
#330066	51	0	102
#330099	51	0	153
#3300CC	51	0	204
#3300FF	51	0	255

#333300	51	51	0
#333333	51	51	51
#333366	51	51	102
#333399	51	51	153
#3333CC	51	51	204
#3333FF	51	51	255
#336600	51	102	0
#336633	51	102	51
#336666	51	102	102
#336699	51	102	153
#3366CC	51	102	204
#3366FF	51	102	255
#339900	51	153	0
#339933	51	153	51
#339966	51	153	102
#339999	51	153	153
#3399CC	51	153	204
#3399FF	51	153	255
#33CC00	51	204	0
#33CC33	51	204	51
#33CC66	51	204	102
#33CC99	51	204	153
#33CCCC	51	204	204
#33CCFF	51	204	255
#33FF00	51	255	0
#33FF33	51	255	51
#33FF66	51	255	102
#33FF99	51	255	153
#33FFCC	51	255	204
#33FFFF	51	255	255
#660000	102	0	0
#660033	102	0	51
#660066	102	0	102
#660099	102	0	153
#6600CC	102	0	204
#6600FF	102	0	255
#663300	102	51	0
#663333	102	51	51
#663366	102	51	102
#663399	102	51	153
#6633CC	102	51	204

#6633FF	102	51	255
#666600	102	102	0
#666633	102	102	51
#666666	102	102	102
#666699	102	102	153
#6666CC	102	102	204
#6666FF	102	102	255
#669900	102	153	0
#669933	102	153	51
#669966	102	153	102
#669999	102	153	153
#6699CC	102	153	204
#6699FF	102	153	255
#66CC00	102	204	0
#66CC33	102	204	51
#66CC66	102	204	102
#66CC99	102	204	153
#66CCCC	102	204	204
#66CCFF	102	204	255
#66FF00	102	255	0
#66FF33	102	255	51
#66FF66	102	255	102
#66FF99	102	255	153
#66FFCC	102	255	204
#66FFFF	102	255	255
#990000	153	0	0
#990033	153	0	51
#990066	153	0	102
#990099	153	0	153
#9900CC	153	0	204
#9900FF	153	0	255
#993300	153	51	0
#993333	153	51	51
#993366	153	51	102
#993399	153	51	153
#9933CC	153	51	204
#9933FF	153	51	255
#996600	153	102	0
#996633	153	102	51
#996666	153	102	102
#996699	153	102	153

#9966CC	153	102	204
#9966FF	153	102	255
#999900	153	153	0
#999933	153	153	51
#999966	153	153	102
#999999	153	153	153
#9999CC	153	153	204
#9999FF	153	153	255
#99CC00	153	204	0
#99CC33	153	204	51
#99CC66	153	204	102
#99CC99	153	204	153
#99CCCC	153	204	204
#99CCFF	153	204	255
#99FF00	153	255	0
#99FF33	153	255	51
#99FF66	153	255	102
#99FF99	153	255	153
#99FFCC	153	255	204
#99FFFF	153	255	255
#CC0000	204	0	0
#CC0033	204	0	51
#CC0066	204	0	102
#CC0099	204	0	153
#CC00CC	204	0	204
#CC00FF	204	0	255
#CC3300	204	51	0
#CC3333	204	51	51
#CC3366	204	51	102
#CC3399	204	51	153
#CC33CC	204	51	204
#CC33FF	204	51	255
#CC6600	204	102	0
#CC6633	204	102	51
#CC6666	204	102	102
#CC6699	204	102	153
#CC66CC	204	102	204
#CC66FF	204	102	255
#CC9900	204	153	0
#CC9933	204	153	51
#CC9966	204	153	102

#CC9999	204	153	153
#CC99CC	204	153	204
#CC99FF	204	153	255
#CCCC00	204	204	0
#CCCC33	204	204	51
#CCCC66	204	204	102
#CCCC99	204	204	153
#CCCCCC	204	204	204
#CCCCFF	204	204	255
#CCFF00	204	255	0
#CCFF33	204	255	51
#CCFF66	204	255	102
#CCFF99	204	255	153
#CCFFCC	204	255	204
#CCFFFF	204	255	255
#FF0000	255	0	0
#FF0033	255	0	51
#FF0066	255	0	102
#FF0099	255	0	153
#FF00CC	255	0	204
#FF00FF	255	0	255
#FF3300	255	51	0
#FF3333	255	51	51
#FF3366	255	51	102
#FF3399	255	51	153
#FF33CC	255	51	204
#FF33FF	255	51	255
#FF6600	255	102	0
#FF6633	255	102	51
#FF6666	255	102	102
#FF6699	255	102	153
#FF66CC	255	102	204
#FF66FF	255	102	255
#FF9900	255	153	0
#FF9933	255	153	51
#FF9966	255	153	102
#FF9999	255	153	153
#FF99CC	255	153	204
#FF99FF	255	153	255
#FFCC00	255	204	0
#FFCC33	255	204	51

	#FFCC66	255	204	102
	#FFCC99	255	204	153
	#FFCCCC	255	204	204
	#FFCCFF	255	204	255
	#FFFF00	255	255	0
	#FFFF33	255	255	51
	#FFFF66	255	255	102
	#FFFF99	255	255	153
	#FFFFCC	255	255	204
	#FFFFFF	255	255	255

Rozszerzona paleta kolorów nazwanych

Do standardu CSS dodano poniższe kolory mające nazwy. W skryptach używa się nazw pisanych wyłącznie małymi literami.

Starsze przeglądarki mogą nie rozpoznawać nazw. W nowszych, mimo że wszystkie aktualnie obsługują te nazwy - nie ma żadnej gwarancji, że będą też to robić w przyszłości.

W przypadku tych kolorów bezpieczniej jest używać liczb niż nazw.

W tej palecie jest - oczywiście - 16 kolorów palety standardowej

Nazwa HTML	Kolor	HEX	R	G	B
IndianRed		#CD5C5C	205	92	92
LightCoral		#F08080	240	128	128
Salmon		#FA8072	250	128	114
DarkSalmon		#E9967A	233	150	122
LightSalmon		#FFA07A	255	160	122
Crimson		#DC143C	220	20	60
Red		#FF0000	255	0	0
FireBrick		#B22222	178	34	34
DarkRed		#8B0000	139	0	0
Pink		#FFC0CB	255	192	203
LightPink		#FFB6C1	255	182	193
HotPink		#FF69B4	255	105	180
DeepPink		#FF1493	255	20	147
MediumVioletRed		#C71585	199	21	133
PaleVioletRed		#DB7093	219	112	147
LightSalmon		#FFA07A	255	160	122
Coral		#FF7F50	255	127	80
Tomato		#FF6347	255	99	71
OrangeRed		#FF4500	255	69	0

DarkOrange	#FF8C00	255	140	0
Orange	#FFA500	255	165	0
Gold	#FFD700	255	215	0
Yellow	#FFFF00	255	255	0
LightYellow	#FFFFE0	255	255	224
LemonChiffon	#FFFACD	255	250	205
LightGoldenrodYellow	#FAFAD2	250	250	210
PapayaWhip	#FFEFD5	255	239	213
Moccasin	#FFE4B5	255	228	181
PeachPuff	#FFDAB9	255	218	185
PaleGoldenrod	#EEE8AA	238	232	170
Khaki	#F0E68C	240	230	140
DarkKhaki	#BDB76B	189	183	107
Lavender	#E6E6FA	230	230	250
Thistle	#D8BFD8	216	191	216
Plum	#DDA0DD	221	160	221
Violet	#EE82EE	238	130	238
Orchid	#DA70D6	218	112	214
Fuchsia	#FF00FF	255	0	255
Magenta	#FF00FF	255	0	255
MediumOrchid	#BA55D3	186	85	211
MediumPurple	#9370DB	147	112	219
BlueViolet	#8A2BE2	138	43	226
DarkViolet	#9400D3	148	0	211
DarkOrchid	#9932CC	153	50	204
DarkMagenta	#8B008B	139	0	139
Purple	#800080	128	0	128
Indigo	#4B0082	75	0	130
SlateBlue	#6A5ACD	106	90	205
DarkSlateBlue	#483D8B	72	61	139
MediumSlateBlue	#7B68EE	123	104	238
GreenYellow	#ADFF2F	173	255	47
Chartreuse	#7FFF00	127	255	0
LawnGreen	#7CFC00	124	252	0
Lime	#00FF00	0	255	0
LimeGreen	#32CD32	50	205	50
PaleGreen	#98FB98	152	251	152
LightGreen	#90EE90	144	238	144
MediumSpringGreen	#00FA9A	0	250	154
SpringGreen	#00FF7F	0	255	127
MediumSeaGreen	#3CB371	60	179	113

SeaGreen	#2E8B57	46	139	87
ForestGreen	#228B22	34	139	34
Green	#008000	0	128	0
DarkGreen	#006400	0	100	0
YellowGreen	#9ACD32	154	205	50
OliveDrab	#6B8E23	107	142	35
Olive	#808000	128	128	0
DarkOliveGreen	#556B2F	85	107	47
MediumAquamarine	#66CDAA	102	205	170
DarkSeaGreen	#8FBC8F	143	188	143
LightSeaGreen	#20B2AA	32	178	170
DarkCyan	#008B8B	0	139	139
Teal	#008080	0	128	128
Aqua	#00FFFF	0	255	255
Cyan	#00FFFF	0	255	255
LightCyan	#E0FFFF	224	255	255
PaleTurquoise	#AFEEEE	175	238	238
Aquamarine	#7FFFD4	127	255	212
Turquoise	#40E0D0	64	224	208
MediumTurquoise	#48D1CC	72	209	204
DarkTurquoise	#00CED1	0	206	209
CadetBlue	#5F9EA0	95	158	160
SteelBlue	#4682B4	70	130	180
LightSteelBlue	#B0C4DE	176	196	222
PowderBlue	#B0E0E6	176	224	230
LightBlue	#ADD8E6	173	216	230
SkyBlue	#87CEEB	135	206	235
LightSkyBlue	#87CEFA	135	206	250
DeepSkyBlue	#00BFFF	0	191	255
DodgerBlue	#1E90FF	30	144	255
CornflowerBlue	#6495ED	100	149	237
MediumSlateBlue	#7B68EE	123	104	238
RoyalBlue	#4169E1	65	105	225
Blue	#0000FF	0	0	255
MediumBlue	#0000CD	0	0	205
DarkBlue	#00008B	0	0	139
Navy	#000080	0	0	128
MidnightBlue	#191970	25	25	112
Cornsilk	#FFF8DC	255	248	220
BlanchedAlmond	#FFEBCD	255	235	205
Bisque	#FFE4C4	255	228	196

NavajoWhite	#FFDEAD	255	222	173
Wheat	#F5DEB3	245	222	179
BurlyWood	#DEB887	222	184	135
Tan	#D2B48C	210	180	140
RosyBrown	#BC8F8F	188	143	143
SandyBrown	#F4A460	244	164	96
Goldenrod	#DAA520	218	165	32
DarkGoldenrod	#B8860B	184	134	11
Peru	#CD853F	205	133	63
Chocolate	#D2691E	210	105	30
SaddleBrown	#8B4513	139	69	19
Sienna	#A0522D	160	82	45
Brown	#A52A2A	165	42	42
Maroon	#800000	128	0	0
White	#FFFFFF	255	255	255
Snow	#FFFAFA	255	250	250
Honeydew	#F0FFF0	240	255	240
MintCream	#F5FFFA	245	255	250
Azure	#F0FFFF	240	255	255
AliceBlue	#F0F8FF	240	248	255
GhostWhite	#F8F8FF	248	248	255
WhiteSmoke	#F5F5F5	245	245	245
Seashell	#FFF5EE	255	245	238
Beige	#F5F5DC	245	245	220
OldLace	#FDF5E6	253	245	230
FloralWhite	#FFFAF0	255	250	240
Ivory	#FFFFF0	255	255	240
AntiqueWhite	#FAEBD7	250	235	215
Linen	#FAF0E6	250	240	230
LavenderBlush	#FFF0F5	255	240	245
MistyRose	#FFE4E1	255	228	225
Gainsboro	#DCDCDC	220	220	220
LightGrey	#D3D3D3	211	211	211
Silver	#C0C0C0	192	192	192
DarkGray	#A9A9A9	169	169	169
Gray	#808080	128	128	128
DimGray	#696969	105	105	105
LightSlateGray	#778899	119	136	153
SlateGray	#708090	112	128	144
DarkSlateGray	#2F4F4F	47	79	79
Black	#000000	0	0	0

Paleta kolorów z polskimi nazwami

Nazwa	Kolor	Hex	R	G	B
akwamaryna		#7FFFD4	127	255	212
alabastrowy		#FAFFFA	240	255	240
amarantowy		#E61C66	230	28	102
ametystowy		#9966CC	153	102	204
antracytowy		#364135	54	65	53
atramentowy		#003153	0	49	83
żółty yellow		#F9E04B	249	224	75
bananowy		#FEFE33	254	254	51
beżowy		#C2B280	194	178	128
biały		#FFFFFF	255	255	255
biskupi		#FF0080	255	0	128
błękit królewski		#4169E1	65	105	225
błękit paryski		#0027C2	0	39	194
błękit pruski		#003153	0	49	83
błękit Thénarda		#0300FD	3	0	253
błękit Turnbulla		#082E79	8	46	121
bordowy, bordo		#800000	128	0	0
brązowy		#964B00	150	75	0
brunatny		#70201F	112	32	31
brzoskwiniowy		#FFCC99	255	204	153
burakowy		#730029	98	0	44
burgund		#600201	96	2	1
bursztynowy		#FFBF00	255	191	0
bury		#6B5636	107	86	54
ceglasty		#E96B39	233	107	57
chabrowy		#3300CC	51	0	204
chamois		#FFFFD2	255	255	210
cielisty		#FFE5B4	255	229	180
cyjan		#00FFFF	0	255	255
cyjan (druk)		#00B7EB	0	180	247
cyklamen		#A2007B	162	0	123
cynamonowy		#9D5B03	157	91	3
cynobrowy, cynober		#E34234	227	66	52
cytrynowy		#FFFF00	255	255	0
czarny		#000000	0	0	0
czekoladowy		#8B4726	139	71	38
czerwień wzrokowa		#FC0300	252	3	0

czerwień żelazowa	#801818	128	24	24
czerwony	#FF0000	255	0	0
écru	#F5F5DC	245	245	220
eozyna	#C35C6F	195	92	111
feldgrau	#4D5D53	77	93	83
fioletowy	#B803FF	184	3	255
fiołkowy	#EE82EE	238	130	238
fuksja	#FF00FF	255	0	255
gołębi	#C6CECE	198	206	206
grafitowy	#36454F	54	69	79
granatowy	#000080	0	0	128
groszkowy	#CADC79	202	220	121
grynszpan	#00A693	0	166	147
heban, hebanowy	#3D2B1F	61	43	31
herbaciany	#CC5D2B	204	93	43
indygo	#4B0082	75	0	130
jagodowy	#660066	128	0	128
jaśminowy	#CDEBA7	205	235	167
kakaowy	#906030	144	96	48
kanarkowy	#FFFF33	255	255	51
kardynalski	#FF2400	168	8	8
karmazynowy	#DC143C	220	20	60
karminowy	#841839	132	24	57
kasztanowy	#4D1F1C	77	31	28
kawowy	#2C1B01	44	27	1
khaki	#C3B091	195	176	145
kobaltowy	#19247C	25	36	124
koniakowy	#C73C07	199	60	7
koralowy	#FF7F50	255	127	80
kość słoniowa	#FFFFF0	255	255	240
kremowy	#FFFDD0	255	253	208
krwisty	#CF2929	207	41	41
lawendowy	#C9A2BF	201	162	191
lapis-lazuli	#3366CC	51	102	204
lazurowy	#007FFF	0	127	255
lila	#C8A2C8	200	162	200
lila róż	#DEB4CB	222	180	203
liliowy	#DD9ECD	221	158	205
limonkowy	#93F600	147	246	0
łososiowy	#FA8072	250	128	114
magenta (HTML)	#FF00FF	255	0	255

magenta (druk)	#FF0090	255	0	144
mahoń, mahoniowy	#C04000	192	64	0
majtkowy	#E4BECF	228	190	207
makowy	#E21E13	226	30	19
malachitowy	#006633	0	102	51
malinowy	#EB0165	235	1	101
marchewkowy	#FFC000	255	192	0
marengo	#5B5D74	91	93	116
miedziany	#B87333	184	115	51
miętowy	#BCE27F	188	226	127
minia	#B22222	178	34	34
miodowy	#E3AF34	213	173	66
mleczny	#FFFFF0	255	255	240
modry	#4169E1	65	105	225
morelowy	#E9967B	233	150	123
morski	#008080	0	128	128
mosiądz	#D4C279	212	194	121
mysi	#E3E7E6	227	231	230
niebieski	#0000FF	0	0	255
niebieskofioletowy	#9F9FDF	159	159	223
oberżynowy	#532338	83	35	56
ochra	#CC7722	204	119	34
oliwkowy	#808000	128	128	0
oranż	#FFA500	255	165	0
orzechowy	#BD9460	189	148	96
palisander	#532F28	83	47	40
patynowy	#6EBE9F	110	190	159
pąsowy	#EDB5AE	237	181	174
perłowy	#FAFAE7	250	250	231
piaskowy	#F3ECDB	243	236	219
pistacjowy	#9FFB88	159	251	136
piwny	#FDE456	253	228	86
platynowy	#E5E4E2	229	228	226
płowy	#E1D9AC	225	217	172
pomarańczowy	#FE7F00	255	165	0
popielaty	#DBE3DE	219	227	222
porcelanowy	#FAF7E5	250	247	229
poziomkowy	#CF2F2F	207	47	47
purpurowy	#800080	128	0	128
rdzawy	#800000	128	0	0
róż indyjski	#C34444	195	68	68

róż pompejański	#B80000	184	0	0
róż wenecki	#F3B387	243	179	135
różowy	#FFC0CB	255	192	203
rubinowy	#D41B56	212	27	86
rudy	#CD5700	205	87	0
sadza angielska	#3A3431	58	52	49
seledynowy	#ACE1AF	172	225	174
sepia	#AC7A33	172	122	51
siarkowy	#EDFA8E	237	250	142
siny	#8AA4B7	138	164	183
siwy	#DED5D0	222	213	208
sjena palona	#E97451	233	116	81
słomkowy	#FFFF99	255	255	153
smolisty	#0C2419	12	36	25
spiżowy	#CD7F32	205	127	50
srebrny	#C0C0C0	192	192	192
stalowy	#7A7D80	122	125	128
stare złoto	#CFB53B	207	181	59
szafirowy	#082567	8	37	103
szafranowy	#F4C430	244	196	48
szary	#808080	128	128	128
szkarłatny żołądź	#AD1111	173	17	17
szmaragdowy	#19A56F	25	165	111
śliwkowy	#50344F	80	52	79
świniowy	#FFE1BE	255	225	190
tabaczkowy	#866423	134	100	35
tango	#870121	135	1	33
truskawkowy	#CF2942	207	41	66
turkusowy	#30D5C8	48	213	200
tycjan	#FF8000	255	128	0
ugier	#DA7C20	218	124	32
ultramaryna	#002D6E	0	45	110
umbra	#73542F	115	84	47
winny	#660033	102	0	51
wiśniowy	#800000	128	0	0
woskowy	#E2E1A3	226	225	163
wrzosowy	#DBB0EF	219	176	239
zieleń butelkowa	#326647	50	102	71
zieleń jaskrawa	#00FF00	0	255	0
zieleń Veronese'a	#33CC66	51	204	102
zieleń wiosenna	#C0D727	192	215	39

zieleń zgniła		#6E6F2F	110 111 47
zielony		#008000	0 128 0
złocisty		#FFC125	255 193 37
złoty		#FFD700	255 215 0
żółty		#FFFF00	255 255 0
żółty (druk)		#FFEF00	255 239 0

Paleta kolorów nazwanych HSL

Nie można jej zaprezentować ponieważ Word nie rozpoznaje kolorów w tym trybie. Oto treść którą musisz umieścić w pliku html aby obejrzeć paletę kolorów:

```
<!DOCTYPE html>
<html lang="pl">

<head>
<meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="styles/canvas.css" />
</head>

<body>
	<h1>Paleta nazwanych kolorów HSL</h1>
	<p>Dla wszystkich kolorów 'saturation=100%', a 'lightness="50%'</p>
	<table>
		<tr>
			<th>Nazwa</th>
			<th>Kolor</th>
			<th>Kąt barwy w &deg;</th>
		</tr>
		<tr >
			<td>red</td>
			<td style="background: hsl(0, 100%, 50%);"> </td>
			<td>0</td>
		</tr>
		<tr >
			<td>orangish red</td>
			<td style="background: hsl(7.5, 100%, 50%);"> </td>
			<td>7.5</td>
		</tr>
		<tr >
			<td>red orange  albo orange red </td>
			<td style="background: hsl(15, 100%, 50%);"> </td>
			<td>15</td>
		<tr >
			<td>reddish orange</td>
			<td style="background: hsl(22.5, 100%, 50%);"> </td>
			<td>22.5</td>
		</tr>
		<tr >
			<td>orange</td>
			<td style="background: hsl(30, 100%, 50%);"> </td>
			<td>30</td>
		</tr>
		<tr >
			<td>yellowish orange</td>
			<td style="background: hsl(37.5, 100%, 50%);"> </td>
			<td>37.5</td>
		</tr>
		<tr >
			<td>orange yellow albo yellow orange</td>
			<td style="background: hsl(45, 100%, 50%);"> </td>
			<td>45</td>
```

```
</tr>
<tr >
      <td>orangish yellow</td>
      <td style="background: hsl(52.5, 100%, 50%);"> </td>
      <td>52.5</td>
</tr>
<tr >
      <td>yellow</td>
      <td style="background: hsl(60, 100%, 50%);"> </td>
      <td>60</td>
</tr>
<tr >
      <td>greenish yellow</td>
      <td style="background: hsl(75, 100%, 50%);"> </td>
      <td>75</td>
</tr>
<tr >
      <td>yellow green albo green yellow</td>
      <td style="background: hsl(90, 100%, 50%);"> </td>
      <td>90</td>
</tr>
<tr >
      <td>yellowish green</td>
      <td style="background: hsl(105, 100%, 50%);"> </td>
      <td>105</td>
</tr>
<tr >
      <td>green</td>
      <td style="background: hsl(120, 100%, 50%);"> </td>
      <td>120</td>
</tr>
<tr >
      <td>bluish green</td>
      <td style="background: hsl(150, 100%, 50%);"> </td>
      <td>150</td>
</tr>
<tr >
      <td>green blue albo blue green</td>
      <td style="background: hsl(180, 100%, 50%);"> </td>
      <td>180</td>
</tr>
<tr >
      <td>greenish blue</td>
      <td style="background: hsl(210, 100%, 50%);"> </td>
      <td>210</td>
</tr>
<tr >
      <td>blue</td>
      <td style="background: hsl(240, 100%, 50%);"> </td>
      <td>240</td>
</tr>
<tr >
      <td>purplish blue</td>
      <td style="background: hsl(255, 100%, 50%);"> </td>
      <td>255</td>
</tr>
<tr >
      <td>blue purple albo purple blue</td>
      <td style="background: hsl(270, 100%, 50%);"> </td>
      <td>270</td>
</tr>
<tr >
      <td>bluish purple</td>
      <td style="background: hsl(285, 100%, 50%);"> </td>
      <td>285</td>
</tr>
<tr >
      <td>purple</td>
```

```
                <td style="background: hsl(300, 100%, 50%);"> </td>
                <td>300</td>
            </tr>
            <tr >
                <td>reddish purple</td>
                <td style="background: hsl(315, 100%, 50%);"> </td>
                <td>315</td>
            </tr>
            <tr >
                <td>red purple albo purple red</td>
                <td style="background: hsl(330, 100%, 50%);"> </td>
                <td>330</td>
            </tr>
            <tr >
                <td>purplish red</td>
                <td style="background: hsl(345, 100%, 50%);"> </td>
                <td>345</td>
            </tr>
        </table>


</body>
</html>
```

Paleta kolorów HSL

Nie można jej zaprezentować ponieważ Word nie rozpoznaje kolorów w tym trybie. Oto treść którą musisz umieścić w pliku html aby obejrzeć paletę kolorów:
Treść strony html przedstawiającej paletę liczy ok. 80 stron maszynopisu, dlatego nie przedstawiamy jej tutaj. Możesz ją wygenerować automatycznie używając klasy Java, której kod przedstawiony jest w jednym z rozdziałów: ‘Pomocnicze kody Java’.

Spis treści