Technologia i rozwiązania

HTML5 Canvas
Receptury

Ponad 80 receptur prezentujących użycie elementu canvas, które zrewolucjonizują strony WWW!

Helion

Eric Rowell

[PACKT PUBLISHING]

Tytuł oryginału: HTML5 Canvas Cookbook
Tłumaczenie: Piotr Rajca

ISBN: 978-83-246-5075-0

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: http://helion.pl (księgarnia internetowa, katalog książek)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
ftp://ftp.helion.pl/przyklady/ht5cre.zip

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/ht5cre
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

ROZDZIAŁ 1

Wprowadzenie do operacji na ścieżkach i tekstach

W tym rozdziale zostaną opisane następujące zagadnienia:

- rysowanie linii,
- rysowanie łuków,
- rysowanie krzywych kwadratowych,
- rysowanie krzywych Béziera,
- rysowanie zygzaków,
- rysowanie spiral,
- praca z tekstem,
- rysowanie trójwymiarowych tekstów z cieniem,
- wyzwolenie potęgi fraktali — rysowanie nawiedzonego drzewa.

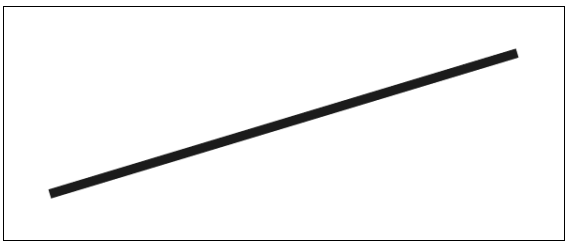
Wprowadzenie

Celem tego rozdziału jest przedstawienie najważniejszych możliwości elementów `canvas` wprowadzonych w języku HTML5 (nazywanych także „płótnem"). Możliwości te zostaną zademonstrowane w serii coraz bardziej złożonych przykładów. API — interfejs programowania aplikacji — elementów `canvas` dostarcza podstawowych narzędzi niezbędnych do rysowania i określania

wyglądu (stylu) różnych typów podścieżek, takich jak linie, łuki, krzywe kwadratowe, krzywe Béziera, oraz do łączenia ich w celu tworzenia ścieżek. Interfejs ten zapewnia także pełne wsparcie dla rysowania tekstów, udostępniając przy tym kilka właściwości pozwalających na określanie ich stylu. Zaczynajmy!

Rysowanie linii

Podczas rozpoczynania nauki korzystania z płócien HTML5 większość osób jest zainteresowana rysowaniem podstawowych elementów. Ta receptura pokazuje, jak narysować linię.



Jak to zrobić

Aby narysować fragment pochyłej linii prostej, należy wykonać następujące czynności:

1. Zdefiniować kontekst `2d` (dwuwymiarowy) płótna i określić styl linii:

```
window.onload = function(){
  // pobranie elementu canvas na podstawie jego identyfikatora
  var canvas = document.getElementById("myCanvas");
  // deklaracja kontekstu 2D przy użyciu metody getContext() obiektu canvas
  var context = canvas.getContext("2d");
  // ustawienie szerokości linii na 10 pikseli
  context.lineWidth = 10;
  // ustawienie koloru linii na niebieski
  context.strokeStyle = "blue";
```

2. Umiejscowić kontekst płótna i narysować linię:

```
  // ustawienie położenia kursora
  context.moveTo(50, canvas.height - 50);
  // rysowanie linii
  context.lineTo(canvas.width - 50, 50);
  // wyświetlenie linii przy wykorzystaniu wybranego wcześniej koloru
  context.stroke();
};
```

3. Umieścić element canvas w treści dokumentu HTML:

```
<canvas id="myCanvas" width="600" height="250" style="border:1px
solid black;">
</canvas>
```

Pobieranie kodów przykładów

Zarówno kody przykładów, jak i inne zasoby można pobrać z serwera FTP wydawnictwa Helion — *ftp://ftp.helion.pl/przyklady/ht5cre.zip*.

Jak to działa

Jak widać na powyższym przykładzie, zanim będzie można odwołać się do elementu canvas na podstawie jego identyfikatora, należy poczekać na wczytanie całej strony. Można to zrobić, używając inicjalizatora window.onload. Po zakończeniu wczytywania strony można już odwołać się do elementu canvas w drzewie DOM strony, używając w tym celu metody document.getElementById(), a następnie zdefiniować dwuwymiarowy (2d) kontekst płótna, przekazując parametr "2d" w wywołaniu metody getContext() obiektu canvas. W ostatnich dwóch rozdziałach książki pokazano, że można tworzyć także konteksty trójwymiarowe. W tym przypadku w wywołaniu metody getContext() należy przekazać parametr "webgl", "experimental-webgl" bądź jeszcze inny.

Podczas rysowania konkretnego elementu, takiego jak ścieżka, podścieżka lub kształt, koniecznie należy pamiętać o tym, że jego styl można zmienić w każdej chwili — zarówno przed rozpoczęciem, jak i po zakończeniu rysowania — aby styl jednak został uwzględniony, trzeba go zastosować bezpośrednio po narysowaniu elementu. Grubość linii można określić przy użyciu właściwości lineWidth, a jej kolor przy zastosowaniu właściwości strokeStyle. Można uznać, że te czynności przypominają rysowanie na kartce papieru — zanim zaczniemy rysować, wybieramy kredkę (strokeStyle) o określonej grubości (lineWidth).

Kiedy już weźmiemy do ręki kredkę, możemy umieścić ją w dowolnym miejscu (punkcie rozpoczęcia rysowania). Do tego celu służy metoda moveTo():

```
context.moveTo(x,y);
```

Kontekst płótna można sobie wyobrazić jako kursor służący do rysowania. Metoda moveTo() tworzy nową podścieżkę w określonym punkcie. Lewy górny wierzchołek płótna ma współrzędne (0,0), natomiast współrzędne prawego dolnego wierzchołka to szerokość i wysokość elementu canvas.

Po umieszczeniu kursora rysującego w określonym punkcie możemy narysować linię, używając metody lineTo(), w której wywołaniu przekazywane są współrzędne punktu końcowego:

```
context.lineTo(x,y);
```

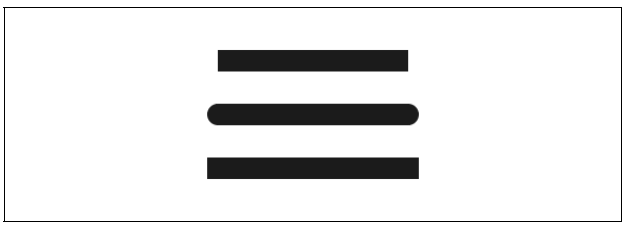
I w końcu, aby linia stała się widoczna, trzeba wywołać metodę `stroke()`. Jeśli przed narysowaniem linii nie wybierzemy innego koloru, to domyślnie będzie ona miała kolor czarny.

Poniżej przedstawiono podstawową procedurę, jaką należy wykonać w celu narysowania linii przy użyciu API elementów `canvas`:

1. Określenie stylu linii (odpowiadające wybraniu kredki o danej grubości).
2. Umieszczenie kontekstu płótna w wybranym miejscu przy użyciu metody `moveTo()` (odpowiadające umieszczeniu kredki w danym miejscu kartki papieru).
3. Narysowanie linii poprzez wywołanie metody `lineTo()`.
4. Wyświetlenie linii za pośrednictwem wywołania metody `stroke()`.

Dodatkowe informacje

Rysowane linie mogą mieć trzy różne rodzaje zakończeń: `butt` (prostokątne), `round` (okrągłe) oraz `square` (kwadratowe). Styl zakończeń można określać przy użyciu właściwości `lineCap` kontekstu płótna. Domyślnie stosowane jest zakończenie `butt`. Wszystkie trzy style zakończeń zostały przedstawione na poniższym rysunku. Na samej górze widoczny jest domyślny styl zakończeń — `butt`; środkową linię narysowano przy wykorzystaniu stylu zakończeń `round`, a najniższą — przy zastosowaniu stylu `square`.



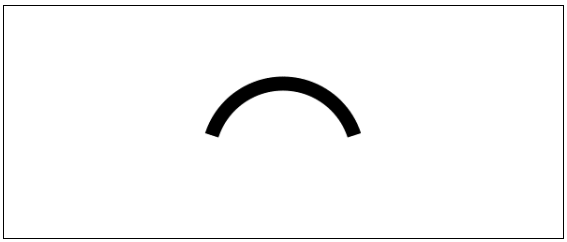
Należy zwrócić uwagę, że środkowa i dolna linia są nieco dłuższe od górnej, choć ich długości są takie same. Dzieje się tak dlatego, że w przypadku zastosowania stylu `round` oraz `square` zakończenia linii powiększają jej długość o wartość równą jej szerokości. Jeśli rysujemy na przykład linię o długości 200 pikseli i szerokości 10 pikseli i zastosujemy przy tym styl zakończeń `round`, to ostateczna długość linii wyniesie 210 pikseli, gdyż każde z jej zakończeń powiększy jej długość o 5 pikseli.

Patrz także

- „Rysowanie zygzaków”.
- „Połączenie wszystkich wiadomości — rysowanie odrzutowca” w rozdziale 2.

Rysowanie łuku

Czasami może się pojawić konieczność narysowania idealnego łuku. Ta receptura może się przydać, jeśli chcemy narysować radosną tęczę, uśmiechniętą buźkę lub jakieś diagramy.



Jak to zrobić

Aby narysować łuk, należy wykonać następujące czynności:

1. Zdefiniować kontekst `2d` płótna i określić styl łuku:

```
window.onload = function(){
    var canvas = document.getElementById("myCanvas");
    var context = canvas.getContext("2d");
    context.lineWidth = 15;
    context.strokeStyle = "black"; // kolor linii
```

2. Narysować łuk:

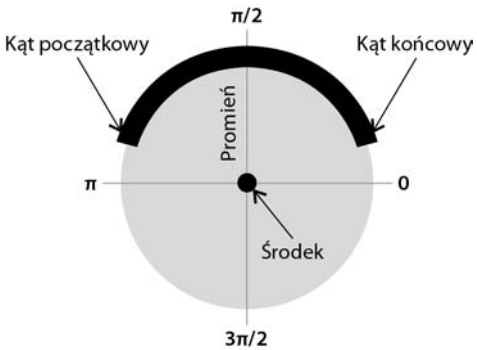
```
context.arc(canvas.width / 2, canvas.height / 2 + 40, 80, 1.1 * Math.PI,
↪1.9 * Math.PI, false);
context.stroke();
};
```

3. Umieścić element `canvas` w treści dokumentu HTML:

```
<canvas id="myCanvas" width="600" height="250" style="border:1px solid black;">
</canvas>
```

Jak to działa

Do narysowania łuku, zdefiniowanego jako fragment umownego okręgu, służy metoda `arc()`. Przyjrzyjmy się następującemu diagramowi:



Umowny okrąg jest definiowany przy użyciu punktu stanowiącego jego środek i promienia. Z kolei rysowany fragment okręgu definiujemy, podając parę kątów — początkowy i końcowy — oraz informację, czy łuk ma być rysowany zgodnie z kierunkiem ruchu wskazówek zegara, czy w kierunku przeciwnym.

```
context.arc( srodekX, srodekY, promien, katPoczatkowy,
     katKoncowy, przeciwnieDoRuchuWskazowekZegara);
```

Trzeba pamiętać, że kąty zaczynają się od wartości 0π z prawej strony okręgu i zmieniają się w kierunku zgodnym z kierunkiem ruchu wskazówek zegara, poprzez wartości 3π/2, π, π/2, z powrotem do 0. W tym przykładzie zastosowaliśmy kąt początkowy o wartości 1,1π oraz kąt końcowy o wartości 1,9π. Oznacza to, że kąt początkowy znajduje się nieco powyżej środka, z lewej strony umownego okręgu, a kąt końcowy — nieco powyżej środka, z prawej strony okręgu.

Dodatkowe informacje

Wartości kąta początkowego i końcowego nie muszą wcale zawierać się w zakresie od 0π do 2π — mogą to być dowolne liczby rzeczywiste, gdyż nic nie stoi na przeszkodzie, by kąty zachodziły na siebie.

Załóżmy, że kąt początkowy wynosi 3π. Odpowiada to jednemu pełnemu obrotowi wokół okręgu (2π) oraz połowie kolejnego obrotu (1π). Innymi słowy, wartość 3π jest odpowiednikiem wartości 1π. I jeszcze jeden przykład — wartość –3π także jest odpowiednikiem wartości 1π, gdyż w tym przypadku kąt zatoczy jeden pełny obrót i dodatkowe pół obrotu wokół okręgu, w kierunku przeciwnym do kierunku ruchu wskazówek zegara, i ostatecznie znajdzie się w położeniu odpowiadającym wartości kątowi 1π.

Innym sposobem rysowania łuków w elemencie `canvas` jest użycie metody `arcTo()`. W tym przypadku łuk definiowany jest na podstawie punktu kontekstu, punktu kontrolnego oraz promienia.

```
context.arcTo(punktKontrolnyX1, punktKontrolnyY1, punktKoncowyX, punktKoncowyY,
↪promien);
```

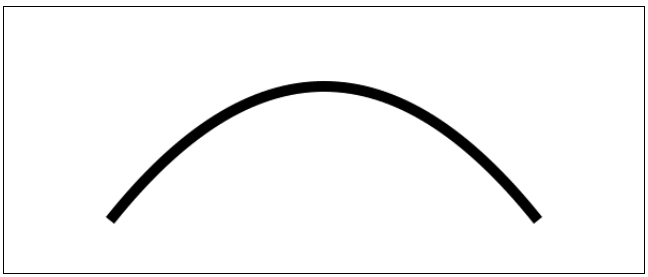
W odróżnieniu od metody `arc()`, która rysuje okrąg wokół określonego środka, metoda `arcTo()` zależy od punktu kontekstu, przez co jest nieco podobna do metody `lineTo()`. Metoda `arcTo()` jest najczęściej używana do tworzenia zaokrąglonych wierzchołków podczas rysowania ścieżek i kształtów.

Patrz także

- „Rysowanie okręgu" w rozdziale 2.
- „Animowane koła zębate" w rozdziale 5.
- „Animowany zegar" w rozdziale 5.

Rysowanie krzywej kwadratowej

W tej recepturze wyjaśniono, jak rysować krzywe kwadratowe. Krzywe tego typu zapewniają znacznie większą elastyczność i pozwalają na rysowanie krzywizn bardziej naturalnych od swych kuzynów łuków i doskonale nadają się do tworzenia dowolnych kształtów.



Jak to zrobić

Aby narysować krzywą kwadratową, należy wykonać następujące czynności:

1. Zdefiniować kontekst `2d` płótna:

```
window.onload = function(){
    var canvas = document.getElementById("myCanvas");
    var context = canvas.getContext("2d");

    context.lineWidth = 10;
    context.strokeStyle = "black"; // kolor linii
```

2. Odpowiednio umieścić punkt kontekstu i narysować krzywą kwadratową:

```
    context.moveTo(100, canvas.height - 50);
    context.quadraticCurveTo(canvas.width / 2, -50, canvas.width
```

```
            - 100, canvas.height - 50);
        context.stroke();
};
```

3. Umieścić element `canvas` w treści dokumentu HTML:

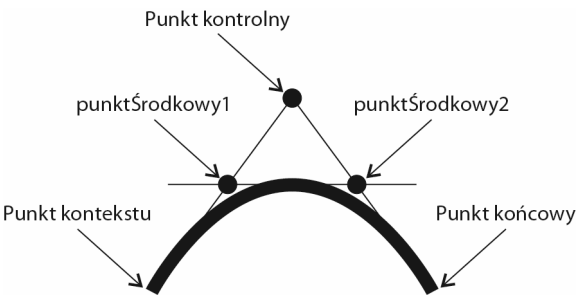
```
<canvas id="myCanvas" width="600" height="250" style="border:1px solid black;">
</canvas>
```

Jak to działa

Krzywe kwadratowe rysowane w elemencie `canvas` są definiowane przez punkt kontekstu, punkt kontrolny oraz punkt końcowy:

```
context.quadraticCurveTo(kontrolnyX,kontrolnyY,punktKocowyX,punktKoncowyY);
```

Przeanalizujmy następujący diagram:



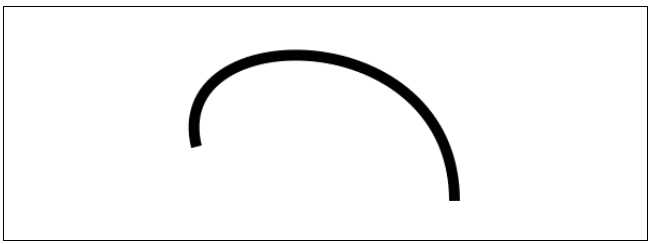
Krzywizna krzywej tego typu jest styczna do trzech stycznych charakterystycznych. W swojej pierwszej części krzywa kwadratowa jest styczna do umownej prostej przechodzącej przez punkt kontekstu i punkt kontrolny. W szczytowym miejscu wygięcia krzywa jest styczna do umownej prostej przechodzącej przez dwa punkty środkowe — punktŚrodkowy1 oraz punktŚrodkowy2. W swojej końcowej części krzywa jest styczna do umownej prostej przechodzącej przez punkty kontrolny i końcowy.

Patrz także

- „Połączenie wszystkich wiadomości — rysowanie odrzutowca” w rozdziale 2.
- „Wyzwolenie potęgi fraktali — rysowanie nawiedzonego drzewa”.

Rysowanie krzywej Béziera

Jeśli krzywa kwadratowa nie zaspokaja naszych potrzeb, to być może zrobi to krzywa Béziera. Krzywe Béziera, nazywane także krzywymi sześciennymi, są najbardziej zaawansowanym rodzajem krzywych dostępnych w API elementach `canvas`.



Jak to zrobić

Aby narysować krzywą Béziera, należy wykonać następujące czynności:

1. Zdefiniować kontekst `2d` płótna:

```
window.onload = function(){
    var canvas = document.getElementById("myCanvas");
    var context = canvas.getContext("2d");

    context.lineWidth = 10;
    context.strokeStyle = "black"; // kolor linii
    context.moveTo(180, 130);
```

2. Odpowiednio umieścić punkt kontekstu i narysować krzywą kwadratową:

```
    context.bezierCurveTo(150, 10, 420, 10, 420, 180);
    context.stroke();
};
```

3. Umieścić element `canvas` w treści dokumentu HTML:

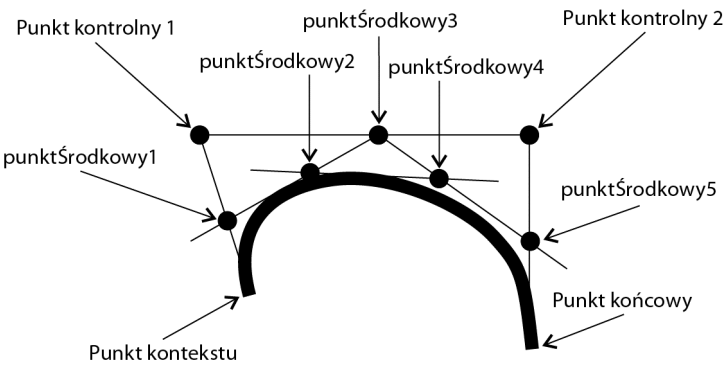
```
<canvas id="myCanvas" width="600" height="250" style="border:1px solid black;">
</canvas>
```

Jak to działa

Krzywe Béziera rysowane w elementach `canvas` są definiowane przez punkt kontekstu, dwa punkty kontrolne oraz punkt końcowy. Dodatkowy punkt kontrolny zapewnia znacznie większą kontrolę nad postacią krzywizny, niż było to możliwe w przypadku krzywych kwadratowych:

```
context.bezierCurveTo(punktKontrolnyX1, punktKontrolnyY1, punktKontrolnyX2,
↪punktKontrolnyY2, punktKoncowyX, punktKoncowyY);
```

Przeanalizujmy następujący diagram:



W odróżnieniu od krzywych kwadratowych, definiowanych przez trzy styczne charakterystyczne, krzywe Béziera są definiowane przez pięć stycznych. Pierwsza, początkowa część krzywej jest styczna do umownej prostej przechodzącej przez punkt kontekstu i pierwszy punkt kontrolny. Kolejna część krzywej jest styczna do umownej prostej przechodzącej przez punktŚrodkowy1 oraz punktŚrodkowy3. Wierzchołek krzywizny jest styczny do umownej prostej przechodzącej przez punktŚrodkowy2 i punktŚrodkowy4. Czwarta część krzywej jest styczna do umownej prostej przechodzącej przez punktŚrodkowy3 oraz punktŚrodkowy5. Ostatnia część krzywej jest styczna do umownej prostej przechodzącej przez drugi punkt kontrolny i punkt końcowy.

Patrz także

- „Stosowanie wartości losowych we właściwościach kształtów — rysowanie pola kwiatów" w rozdziale 2.
- „Połączenie wszystkich wiadomości — rysowanie odrzutowca" w rozdziale 2.

Rysowanie zygzaków

Ta receptura przedstawia sposób rysowania ścieżki, która powstanie z połączenia podścieżek i utworzy łamaną — zygzak.

Jak to zrobić

Aby narysować łamaną, należy wykonać następujące operacje:

1. Zdefiniować kontekst `2d` płótna:

```
window.onload = function(){
    var canvas = document.getElementById("myCanvas");
    var context = canvas.getContext("2d");

    var startX = 85;
    var startY = 70;
    var zigzagSpacing = 60;
```

2. Określić styl łamanej i rozpocząć rysowanie ścieżki:

```
    context.lineWidth = 10;
    context.strokeStyle = "#0096FF"; // kolor niebieskawy
    context.beginPath();
    context.moveTo(startX, startY);
```

3. Narysować siedem łączących się odcinków i wyświetlić ścieżkę, wywołując metodę `stroke()`:

```
    // rysowanie siedmiu linii prostych
    for (var n = 0; n < 7; n++) {
        var x = startX + ((n + 1) * zigzagSpacing);
        var y;
        if (n % 2 == 0) { // jeśli n jest parzyste
            y = startY + 100;
        }
        else { // jeśli n jest nieparzyste
            y = startY;
        }
        context.lineTo(x, y);
    }

    context.stroke();
}
```

4. Umieścić element `canvas` w treści dokumentu HTML:

```
<canvas id="myCanvas" width="600" height="250" style="border:1px solid black;">
</canvas>
```

Jak to działa

Aby narysować zygzak, należy połączyć ze sobą kilka ukośnych linii, tworząc w ten sposób ścieżkę. Można to zrobić przy użyciu pętli, w której w nieparzystych iteracjach będzie rysowana linia ukośna skierowana w dół i w prawo, a w iteracjach parzystych — linia ukośna skierowana w górę i w prawo.

Najważniejszym aspektem jest tu metoda `beginPath()`. Jej wywołanie deklaruje początek rysowania ścieżki. Dzięki temu koniec każdej linii — podścieżki — będzie definiował początek kolejnej podścieżki. W razie pominięcia wywołania tej metody trzeba by przy użyciu metody `moveTo()` mozolnie umieszczać punkt kontekstu płótna przed rozpoczęciem rysowania każdej linii, by zapewnić, że początek każdej kolejnej rysowanej linii będzie się pokrywał z końcem poprzedniej. Wywołanie metody `beginPath()` jest także konieczne w przypadku tworzenia kształtów, o czym mowa w następnym rozdziale.

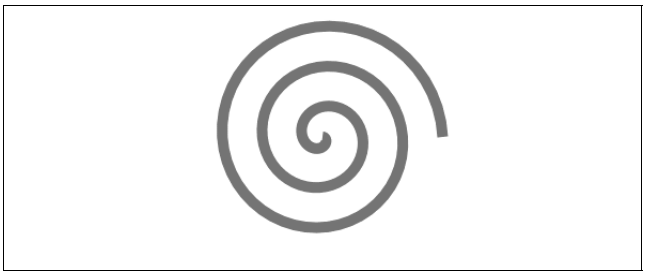
Style połączeń linii

Warto zwrócić uwagę na to, że połączenia pomiędzy kolejnymi segmentami rysowanego zygzaka są spiczaste. Wynika to z tego, że domyślnym stylem połączeń linii w płótnach HTML5 jest `miter`. Korzystając z właściwości `lineJoin` kontekstu płótna, można także zmienić styl połączeń na `round` (połączenia zaokrąglone) lub `bevel` (połączenia ukośne).

Jeśli łączone segmenty linii są stosunkowo cienkie i nie łączą się pod ostrymi kątami, to zauważenie jakichkolwiek różnic pomiędzy poszczególnymi stylami połączeń może być trudne. Zazwyczaj różnice pomiędzy nimi stają się wyraźne przy liniach, których grubość przekracza 5 pikseli, a kąty pomiędzy nimi są stosunkowo niewielkie.

Rysowanie spirali

Uwaga — ta receptura może działać hipnotycznie. W tym przykładzie narysujemy spiralę — ścieżkę składającą się z sekwencji krótkich linii.



Jak to zrobić

Aby narysować spiralę o określonym punkcie centralnym, należy wykonać następujące czynności:

1. Zdefiniować kontekst `2d` płótna i zainicjować parametry spirali:

```
window.onload = function(){
    var canvas = document.getElementById("myCanvas");
```

```
var context = canvas.getContext("2d");

var radius = 0;
var angle = 0;
```

2. Określić styl rysowanej spirali:

```
context.lineWidth = 10;
context.strokeStyle = "#0096FF"; // kolor niebieskawy
context.beginPath();
context.moveTo(canvas.width / 2, canvas.height / 2);
```

3. Zatoczyć trzy pełne obroty wokół punktu centralnego (przy czym na każdy obrót będzie przypadać po 50 segmentów linii). Każdy segment będzie rysowany przy użyciu metody `lineTo()`, od końca poprzedniego segmentu do aktualnie wyznaczonego punktu, przy czym za każdym razem promień będzie zwiększany o `0.75`. Po zakończeniu trzech pełnych obrotów spirala zostanie wyświetlona poprzez wywołanie metody `stroke()`:

```
for (var n = 0; n < 150; n++) {
    radius += 0.75;
    // pełny obrót będzie się składał z 50 iteracji
    angle += (Math.PI * 2) / 50;
    var x = canvas.width / 2 + radius * Math.cos(angle);
    var y = canvas.height / 2 + radius * Math.sin(angle);
    context.lineTo(x, y);
}
context.stroke();
};
```

4. Umieścić element `canvas` w treści dokumentu HTML:

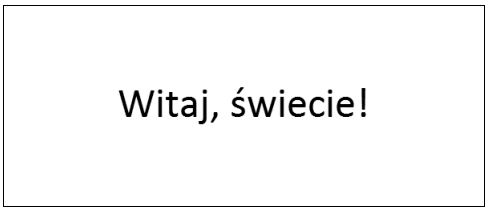
```
<canvas id="myCanvas" width="600" height="250" style="border:1px solid black;">
</canvas>
```

Jak to działa

Aby narysować w elemencie `canvas` spiralę, możemy umieścić kursor w jego środku, a następnie narysować sekwencję bardzo krótkich linii, zwiększając przy tym kąt i odległość od punktu centralnego, przy czym każda kolejna linia będzie się zaczynać w punkcie zakończenia poprzedniej. Analizując ten algorytm rysowania, można wyobrazić sobie, że jesteśmy dzieckiem, które stoi na chodniku z kawałkiem kredy w ręce. Pochylamy się, przykładamy kredę do chodnika i zaczynamy kręcić się w koło (nie za szybko, chyba że chcemy, by się nam zakręciło w głowie). Podczas kręcenia się odsuwamy kredę coraz dalej od siebie. Po kilku obrotach okaże się, że narysowaliśmy śliczną, małą spiralę.

Praca z tekstem

Niemal wszystkie aplikacje korzystają z wyświetlania tekstów, by efektywnie przekazywać informacje użytkownikom. Ta receptura pokazuje, jak wyświetlić w elemencie `canvas` optymistyczne powitanie.



Jak to zrobić

Aby wyświetlić na płótnie tekst, należy wykonać następujące czynności:

1. Zdefiniować kontekst `2d` płótna i określić styl prezentowanego tekstu:

```
window.onload = function(){
    var canvas = document.getElementById("myCanvas");
    var context = canvas.getContext("2d");

    context.font = "40pt Calibri";
    context.fillStyle = "black";
```

2. Wyrównać tekst w poziomie i w pionie, a następnie go wyświetlić:

```
// wyśrodkowanie tekstu w poziomie
    context.textAlign = "center";
    // wyśrodkowanie tekstu w pionie
    context.textBaseline = "middle";
    context.fillText("Witaj, świecie!", canvas.width / 2, 120);
};
```

3. Umieścić element `canvas` w treści dokumentu HTML:

```
<canvas id="myCanvas" width="600" height="250" style="border:1px solid black;">
</canvas>
```

Jak to działa

Gdy wyświetlamy tekst w elemencie `canvas`, możemy zdefiniować styl oraz wielkość czcionki (właściwość `font`), kolor czcionki (właściwość `fillStyle`), wyrównanie tekstu w poziomie (właściwość `textAlign`) i w pionie (właściwość `textBaseline`). Właściwości `textAlign` można przypisać wartości `left`, `center` bądź `right`, a właściwości `textBaseline` wartości `top`, `hanging`, `middle`, `alphabetic`, `ideographic` lub `bottom`. Domyślną wartością właściwości `textAlign` jest `left`, natomiast właściwości `textBaseline` — `alphabetic`.

Dodatkowe informacje

Oprócz metody `fillText()` API elementów `canvas` udostępnia metodę `strokeText()`:

```
context.strokeText("Witaj, świecie!", x, y);
```

Metoda ta nie wyświetla samego tekstu, lecz wypełnia zajmowany przez niego obszar określonym kolorem. Aby wyświetlić tekst i jednocześnie wypełnić kolorem zajmowany przez niego obszar, należy wywołać zarówno metodę `fillText()`, jak i `strokeText()`. Aby tekst miał odpowiednią wielkość, w pierwszej kolejności należy wywołać metodę `fillText()`, a dopiero po niej metodę `strokeText()`.

Patrz także

- „Rysowanie trójwymiarowego tekstu z cieniem".
- „Tworzenie odbicia lustrzanego" w rozdziale 4.
- „Rysowanie prostego logo i losowe określanie jego położenia, obrotu oraz skali" w rozdziale 4.

Rysowanie trójwymiarowego tekstu z cieniem

Osoby, które uważają, że zwyczajne, dwuwymiarowe teksty nie są szczególnie atrakcyjne, zainteresują się zapewne rysowaniem tekstów trójwymiarowych. Choć API elementów `canvas` nie zapewnia bezpośrednio możliwości rysowania takich tekstów, to jednak korzystając z tego API, stosunkowo łatwo można samodzielnie utworzyć funkcję `draw3dText()`.

Jak to zrobić

Aby narysować trójwymiarowy tekst, należy wykonać następujące czynności:

1. Zdefiniować kontekst `2d` płótna i określić styl prezentowanego tekstu:

```
window.onload = function(){
    var canvas = document.getElementById("myCanvas");
```

```
    var context = canvas.getContext("2d");

    context.font = "40pt Calibri";
    context.fillStyle = "black";
```

2. Określić wyrównanie i wyświetlić trójwymiarowy tekst:

```
// wyśrodkowanie tekstu w poziomie
    context.textAlign = "center";
    // wyśrodkowanie tekstu w pionie
    context.textBaseline = "middle";
    draw3dText(context, "Witaj, świecie 3D!", canvas.width / 2, 120, 5);
};
```

3. Zdefiniować funkcję `draw3dText()`, która będzie tworzyć kilka warstw tekstu i dodawać do niego cień:

```
function draw3dText(context, text, x, y, textDepth){
    var n;

    // rysowanie dolnych warstw
    for (n = 0; n < textDepth; n++) {
        context.fillText(text, x - n, y - n);
    }

    // rysowanie górnej warstwy z cieniem rzucanym na
    // warstwy niższe
    context.fillStyle = "#5E97FF";
    context.shadowColor = "black";
    context.shadowBlur = 10;
    context.shadowOffsetX = textDepth + 2;
    context.shadowOffsetY = textDepth + 2;
    context.fillText(text, x - n, y - n);
}
```

4. Umieścić element `canvas` w treści dokumentu HTML:

```
<canvas id="myCanvas" width="600" height="250" style="border:1px solid black;">
</canvas>
```

Jak to działa

Aby narysować trójwymiarowy tekst w elemencie `canvas`, można wyświetlić kilka nieznacznie przesuniętych względem siebie warstw zawierających ten sam tekst, tworząc w ten sposób imitację głębi. W tej recepturze głębokość tekstu wynosi 5, co oznacza, że nasza funkcja `draw3dText()` pięć razy wyświetli tekst `Witaj, świecie 3D!`, za każdym razem nieznacznie go przesuwając. Te dodatkowe warstwy tekstu zostaną wyświetlone na czarno, by zapewnić imitację głębi.

Następnie dodamy kolorową, wierzchnią warstwę. Ostatnią operacją jest dodanie rozmytego cienia poniżej tekstu, co można zrobić poprzez przypisanie odpowiednich wartości właściwościom `shadowColor`, `shadowBlur`, `shadowOffsetX` i `shadowOffsetY` kontekstu płótna. Właściwości te można stosować nie tylko przy wyświetlaniu tekstów, lecz także podścieżek, ścieżek i kształtów, o czym będzie jeszcze mowa w dalszej części książki.

Wyzwalanie potęgi fraktali — rysowanie nawiedzonego drzewa

Czym są fraktale? Można powiedzieć, że są one połączeniem matematyki ze sztuką. Znajdziemy je we wszelkiego rodzaju wzorcach występujących w naturze. Pod względem algorytmicznym fraktale są równaniami wykorzystującymi rekurencję. W tej recepturze narysujemy naturalnie wyglądające drzewo, zaczynające się od pnia rozdzielającego się na dwa konary, z których każdy następnie rozdziela się na dwie gałęzie. Po dwunastu takich powtórzeniach uzyskamy rozłożyste, pozornie chaotyczne drzewo z masą konarów i gałązek.

Jak to zrobić

Oto czynności, jakie należy wykonać, aby narysować drzewo, korzystając z fraktali:

1. Utworzyć rekurencyjną funkcję, która będzie rysować jedną gałąź, rozdzielającą się na dwie mniejsze gałęzie; następnie funkcja będzie wywoływać rekurencyjnie samą siebie, by narysować te dwie gałęzie, zaczynając od ich punktów końcowych:

```
function drawBranches(context, startX, startY, trunkWidth, level){
    if (level < 12) {
        var changeX = 100 / (level + 1);
        var changeY = 200 / (level + 1);

        var topRightX = startX + Math.random() * changeX;
        var topRightY = startY - Math.random() * changeY;

        var topLeftX = startX - Math.random() * changeX;
        var topLeftY = startY - Math.random() * changeY;

        // rysowanie prawej gałęzi
        context.beginPath();
        context.moveTo(startX + trunkWidth / 4, startY);
        context.quadraticCurveTo(startX + trunkWidth /
        ↪4, startY - trunkWidth, topRightX, topRightY);
        context.lineWidth = trunkWidth;
        context.lineCap = "round";
        context.stroke();

        // rysowanie lewej gałęzi
        context.beginPath();
        context.moveTo(startX - trunkWidth / 4, startY);
        context.quadraticCurveTo(startX - trunkWidth / 4, startY -
        trunkWidth, topLeftX, topLeftY);
        context.lineWidth = trunkWidth;
        context.lineCap = "round";
        context.stroke();

       drawBranches(context, topRightX, topRightY, trunkWidth * 0.7, level + 1);
       drawBranches(context, topLeftX, topLeftY, trunkWidth * 0.7, level + 1);
    }
}
```

2. Zainicjować kontekst płótna i rozpocząć rysowanie fraktala drzewa poprzez wywołanie funkcji `drawBranches()`:

```
window.onload = function(){
    canvas = document.getElementById("myCanvas");
    context = canvas.getContext("2d");

    drawBranches(context, canvas.width / 2, canvas.height, 50, 0);
};
```

3. Umieścić element `canvas` w treści dokumentu HTML:

```
<canvas id="myCanvas" width="600" height="250" style="border:1px solid black;">
</canvas>
```

Jak to działa

Aby narysować drzewo przy użyciu fraktala, należy przygotować rekurencyjną funkcję, która zdefiniuje matematyczną naturę drzewa. Jeśli poświęcimy chwilę na przestudiowanie wyglądu drzew (gdy to zrobimy, okaże się, że są naprawdę piękne), zauważymy, że wszystkie ich gałęzie rozdzielają się na mniejsze gałązki. Oznacza to, że nasza rekurencyjna funkcja powinna rysować jedną gałąź, która rozdziela się, dając początek dwóm kolejnym gałęziom, a następnie rekurencyjnie wywołać samą siebie, by narysować te dwie gałęzie wraz z kolejnymi, jeszcze mniejszymi gałęziami.

Teraz, kiedy już wiemy, jak ma działać nasz fraktal, możemy go zaimplementować, korzystając z API elementów `canvas`. Najprostszym sposobem utworzenia gałęzi rozdzielającej się na dwie kolejne jest narysowanie dwóch krzywych kwadratowych, wygiętych w przeciwnych kierunkach.

Gdybyśmy zastosowali dla każdej iteracji dokładnie tę samą procedurę, to narysowane drzewo byłoby idealnie symetryczne i mało interesujące. Aby wyglądało ono bardziej naturalnie, położenie punktów końcowych poszczególnych gałęzi będzie modyfikowane o wartości losowe.

Dodatkowe informacje

Najciekawszym aspektem tej receptury jest to, że każde narysowane drzewo będzie inne. Jeśli Czytelnik zaimplementuje powyższy przykład i będzie go wielokrotnie wyświetlał w przeglądarce, to każde z wygenerowanych drzew będzie unikalne. Można także spróbować zmodyfikować algorytm rysujący gałęzie, by tworzyć różne rodzaje drzew albo nawet dorysowywać liście na końcach najmniejszych gałązek.

Doskonałymi przykładami fraktali w naturze są muszle, płatki śniegu, pióra, rośliny, kryształy, góry, rzeki i błyskawice.

Skorowidz

D

E

F

G

H

K

L

M

O

P

R

S

T

V

W

X

HTML5 Canvas
Receptury

HTML5 szturmem zdobywa rynek stron (a w zasadzie już aplikacji) WWW. Co sprawia, że tak się dzieje? Ogrom nowych funkcji, które otwierają przed projektantami nowe możliwości. Usługi geolokalizacyjne, lokalne przechowywanie plików lub obsługa plików multimedialnych to tylko niektóre z nich. Wśród nowości jest również ta jedna wzbudzająca najwięcej emocji — *canvas* (płótno). Jest to element, który rewolucjonizuje grafikę i wizualizację na stronach WWW.

Dzięki tej książce błyskawicznie zaczniesz korzystać z potencjału elementu canvas. W trakcie lektury poznasz podstawowe możliwości API tego elementu, a wraz z kolejnymi stronami zaczniesz wykorzystywać coraz bardziej zaawansowane techniki tworzenia animacji, wykonywania operacji na obrazach i pisania gier. Rysowanie łuków, przekształcenia i obroty nie będą stanowiły dla Ciebie żadnego problemu. Ponadto w mgnieniu oka opanujesz zasady tworzenia wykresów oraz wizualizacji przestrzennych. Ta książka zasługuje na Twoją szczególną uwagę. Sięgnij po nią i twórz oszałamiające strony w sieci!

Gotowe przepisy do wykorzystania na Twojej stronie!

Element *canvas* to:

- nowość w języku HTML5
- nowe możliwości graficzne na Twojej stronie
- efektowna wizualizacja danych
- baza do tworzenia gier dla przeglądarek

helion.pl księgarnia internetowa

Nr katalogowy: 11931

Księgarnia internetowa: http://helion.pl

Zamówienia telefoniczne: 0 801 339900 0 601 339900

Helion

Sprawdź najnowsze promocje: http://helion.pl/promocje
Książki najchętniej czytane: http://helion.pl/bestsellery
Zamów informacje o nowościach: http://helion.pl/nowosci

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

Informatyka w najlepszym wydaniu

ISBN 978-83-246-5075-0

9 788324 650750

Cena: 59,00 zł