Temat: Animacja z użyciem HTML 5

Przygotował: mgr inż. Tomasz Michno

1 Wstęp

Wraz z HTML5 został wprowadzony nowy znacznik – canvas, który jest przeznaczony do tworzenia grafiki rastrowej z użyciem języka JavaScript. Element canvas powinien być tworzony w następujący sposób:

```
<canvas id=”nazwa” width=”szerokość” height=”wysokość”>
        treść alternatywna
</canvas>
```

Znacznik zamykający jest wymagany, natomiast w środku powinna znajdować się treść alternatywna – wyświetlana w przypadku braku obsługi znacznika canvas w przeglądarce (np. informacja o niewspieranej przeglądarce lub tradycyjny rysunek wczytany za pomocą znacznika img).

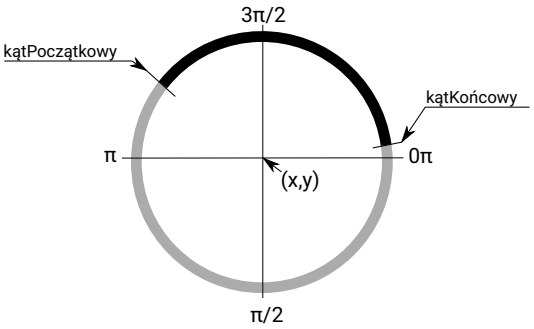
Do rysowania po elemencie canvas używa się języka JavaScript. W tym celu należy najpierw pobrać wskaźnik na wybrany element używając metody getElementById(), a następnie pobrać kontekst metodą getContext:

```
var canvas = document.getElementById('nazwa');
var context = canvas.getContext('2d');
```

Następnie w celu narysowania figury należy użyć odpowiedniej metody obiektu canvas:

- fillStyle = kolor - atrybut przechowujący kolor używany do wypełnienia, kolor podawany jest w taki sam sposób jak w HTML'u lub w CSS'ie:
  nazwaAngielska - np. blue
  #0000FF - zapis szesnastkowy
  rgb(0,0,255) - użycie funkcji rgb
- strokeStyle = kolor - atrybut przechowujący kolor linii
- lineWidth= liczba - grubość linii w pikselach
- globalAlpha = wartość - ustawienie przezroczystości, wartość od 0 do 1
- strokeRect(x, y, w, h) - rysuje wypełniony prostokąt o lewym górnym rogu w punkcie x,y oraz szerokości w i wysokości h
- fillRect(x, y, w, h) - rysuje wypełniony prostokąt o lewym górnym rogu w punkcie x,y oraz szerokości w i wysokości h
- rysowanie łamanych, wielokątów, łuków okręgów:
  - beginPath() - rozpoczęcie rysowania
  - moveTo(x, y) - przesunięcie punktu początkowego do (x,y)
  - lineTo(x, y) - narysowanie linii od bieżącego punktu do podanego w parametrach
  - arc(x, y, promień, kątPoczątkowy, kątKońcowy, kierunek) - rysuje łuk (lub okręg), x,y - środek, kątPoczątkowy - kąt początkowy łuku, kątKońcowy - kąt końcowy łuku, kierunek - kierunek zgodny z ruchem wskazówek zegara, true lub false

- fill() - właściwe wypełnienie figury (na końcu rysowania)
- stroke() - właściwe narysowanie figury



Przykład:
narysowanie łuku o czerwonej krawędzi:

```
<!DOCTYPE html>
<html lang="pl">
 <head>
  <title>HTML5 canvas</title>
  <script type="text/javascript">
   function draw(){
    var canvas = document.getElementById('myCanvas');
    var context = canvas.getContext('2d');
    context.beginPath();
    context.strokeStyle='rgb(255,0,0)';
    context.lineWidth = 5;
    context.arc(120, 100, 20, 3*Math.PI/2, Math.PI/2, true);
    context.stroke();
   }
  </script>
 </head>
 <body onload="draw();">
  <canvas id="myCanvas" width="640" height="480"></canvas>
 </body>
</html>
```

Używanie obrazków:

Należy utworzyć obiekt Image():

```
var obrazek = new Image();
```

ustawić ścieżkę dostępu do pliku:

```
obrazek.src = "obrazek.jpg"
```

narysować obrazek po jego wczytaniu:

```
obrazek.onload = function () { context.drawImage(obrazek, 0, 0); };
```

Powyżej została użyta metoda drawImage(), która jako parametry przyjmuje obiekt obrazka oraz współrzędne, gdzie ma on zostać narysowany.
Istnieją jeszcze dwie odmiany tej funkcji:

- drawImage(obrazek, x, y, w, h) – gdzie dodatkowo podajemy szerokość (w) oraz wysokość obrazka (h) – przydatne np. przy skalowaniu
- drawImage(obrazek, sx, sy, sw, sh, dx, dy, dw, dh) – rysowanie wycinka:
  obrazek - obiekt obrazka
  sx, sy, sw, sh – współrzędne lewego górnego rogu prostokąta obcinającego oraz jego szerokość i wysokość
  dx, dy, dw, dh – położenie rysowanego wycinka oraz jego szerokość i wysokość

Transformacje – metody kontekstu:

- translate(x, y) - przesunięcie o wektor x,y
- scale(sx, sy) - skalowanie
- rotate(kąt) - obrót, kąt podany w radianach

Dwie przydatne funkcje przy przekształceniach:

- save() - zapis stanu kontekstu (aktualnie ustawionych transformacji, stylów itp., rozwiązanie podobne do glPushMatrix lub glPushAttrib z OpenGL)
- restore() - przywrócenie zapisanego stanu kontekstu

Tekst

- fillText("tekst", x, y) – narysowanie wypełnionego tekstu, zaczynając od pozycji (x, y)
- strokeText("tekst", x, y) – narysowanie tekstu bez wypełnienia
- font = "Rozmiar Czcionka" – zmiana rozmiaru i czcionki użytej dla tekstu
- measureText("tekst") – pomiar wielkości tekstu przy aktualnie wybranej czcionce (posiada m.in. atrybut width, który określa długość tekstu)
- `textAlign = wartość – wyrównanie tekstu (podobnie jak w HTML'u i CSS'ie – np. left, right, center itp.)`

przykład:

```
context.font = "12pt Arial Black";
context.strokeStyle = "blue";
context.fillText("Fizyka i animacja", 20, 120);
```

Animacja

W JavaScript istnieją dwie funkcje służące do wykonywania funkcji co pewien ustalony czas:

- setInterval(funkcja, czas)
- setTimeout(funkcja, czas)

W obu przypadkach jako pierwszy parametr podajemy nazwę funkcji, która ma być uruchamiana cyklicznie po upływie czasu podanego w milisekundach jako drugi parametr. W przeciwieństwie do setTimeout(), setInterval() wywołuje funkcję, a następnie zaczyna odliczanie od początku bez czekania na zakończenie jej wykonania.

Metodami oraz atrybutami canvas przydatnymi podczas animacji są:

- `width, height – szerokość i wysokość płótna`
- `clearRect(x, y, szerokość, wysokość) – wyczyszczenie prostokąta o podanych rozmiarach`

Bardzo dobrym pomysłem (zwłaszcza w symulacjach) jest oddzielenie animacji obiektów od rysowania sceny, np. animacja przesuwającej się po ekranie litery A może wyglądać następująco:

```
<!DOCTYPE html>
<html lang="pl">
 <head>
  <title>HTML5 canvas</title>
  <script type="text/javascript">

   var x=20;  // położenie początkowe
   var y=120; // położenie początkowe
   var dx=1; // krok przesunięcia po osi x

   function draw(){ // funkcja rysująca
    var canvas = document.getElementById('myCanvas');
    var context = canvas.getContext('2d');
    context.clearRect(0, 0, canvas.width, canvas.height);
    context.font="32pt Arial Black";
    context.strokeStyle="blue";
    context.fillText("A", x, y);
   }

   function animate(){ // funkcja "animująca"
     if(x<0 || x > 320) dx*=-1;  // "odbiajanie od krawędzi"
     x+=dx;
   }
```

```
    function init(){
        setInterval(draw, 67); // około 15 fps
        setInterval(animate, 10); // przesunięcie co 10 ms
    }

  </script>

            <style type="text/css">
                canvas { border: 1px solid black; }
            </style>

 </head>
 <body onload="init()">
  <canvas id="myCanvas" width="320" height="240"></canvas>
 </body>
</html>
```

Specyfikacja znacznika canvas:

http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#the-canvas-element