Transformacje

Dla zawartości elementu canvas możemy używać kilku funkcji transformacji. Canvas jako powierzchnia to obszar, na który możemy nałożyć "macierz transformacji", która następnie go zmienia. Co to oznacza? Jeżeli przykładowo nałożymy na canvas efekt rotate(), rysowane następnie figury będą obrócone - mimo tego, że rysujemy je normalnie.

Mamy kilka funkcji, które możemy wykorzystywać do transformacji:

- ✓ `setTransform(a, b, c, d, e, f)` ustawia macierz transformacji, kolejne parametry to: a - pozioma skala, b - pionowe pochylenie (skew), c - poziome pochylenie, d - pionowa skala, e - poziome przemieszczenie, f - pionowe przemieszczenie
- ✓ `rotate()` obraca płótno, co oznacza, że rysowane elementy będą pojawiać się w innej pozycji, płótno obracane jest względem punkty 0,0
- ✓ `translate(x, y)` obraca płótno, co oznacza, że rysowane figury będą pojawiać się obrócone
- ✓ `scale(x, y)` skaluje płótno, co oznacza, że rysowane figury będą pojawiać się przeskalowane, płótno skalowane jest względem punktu 0,0

Na początek `setTransform()`. Analizując poszczególne parametry możemy powiedzieć, że rysunek zachowa się „normalnie" wtedy, gdy parametry a (pozioma skala) oraz d (pionowa skala) będą równe 1, a reszta będzie równa 0. Sprawdźmy:

```
<canvas width="400" height="400" id="canvas"></canvas>

<script>
    const canvas = document.querySelector("canvas");
    const ctx = canvas.getContext("2d");

    const gradient = ctx.createLinearGradient(0, 0, canvas.width, canvas.height)
    gradient.addColorStop(0, "red")
    gradient.addColorStop(0.5, "purple")
    gradient.addColorStop(1, "blue")
    ctx.fillStyle = gradient

    ctx.setTransform(1, 0, 0, 1, 0, 0)

    ctx.fillRect(50, 50, 250, 200)
</script>
```

Sprawdźmy pionowe pochylenie:

```
<canvas width="400" height="400" id="canvas"></canvas>

<script>
    const canvas = document.querySelector("canvas");
    const ctx = canvas.getContext("2d");

    const gradient = ctx.createLinearGradient(0, 0, canvas.width, canvas.height)
    gradient.addColorStop(0, "red")
    gradient.addColorStop(0.5, "purple")
    gradient.addColorStop(1, "blue")
    ctx.fillStyle = gradient

    ctx.setTransform(1, 0.2, 0, 1, 0, 0)

    ctx.fillRect(50, 50, 250, 200)
</script>
```

Jak widać, nawet niewielka zmiana parametru b (tylko o 0.2) spowodowała zmiany. Podobna sytuacja występuje w przypadku zmiany parametru c:

```
<canvas width="400" height="400" id="canvas"></canvas>

<script>
    const canvas = document.querySelector("canvas");
    const ctx = canvas.getContext("2d");

    const gradient = ctx.createLinearGradient(0, 0, canvas.width, canvas.height)
    gradient.addColorStop(0, "red")
    gradient.addColorStop(0.5, "purple")
    gradient.addColorStop(1, "blue")
    ctx.fillStyle = gradient

    ctx.setTransform(1, 0, 0.2, 1, 0, 0)

    ctx.fillRect(50, 50, 250, 200)
</script>
```

Spróbujmy także zmienić położenie w pionie i poziomie naszego elementu. Wykorzystajmy w tym celu parametry e oraz f:

```
<canvas width="600" height="400" id="canvas"></canvas>

<script>
    const canvas = document.querySelector("canvas");
    const ctx = canvas.getContext("2d");

    const gradient = ctx.createLinearGradient(0, 0, canvas.width, canvas.height)
    gradient.addColorStop(0, "red")
    gradient.addColorStop(0.5, "purple")
    gradient.addColorStop(1, "blue")
    ctx.fillStyle = gradient

    ctx.fillRect(50, 50, 250, 200)
    ctx.setTransform(1, 0, 0, 1, 50, 50)
    ctx.fillRect(50, 50, 250, 200)
</script>
```

Jak widać, nastąpiło przesunięcie o 50 jednostek w pionie i poziomie względem początkowej figury.

Teraz sprawdźmy, jak działa `rotate()`. Niebieski kwadrat to kwadrat narysowany przed obrotem canvas. Kwadrat żółty ma identyczne współrzędne i wymiary jak niebieski, ale został on narysowany już po obrocie canvasu o 45 stopni.

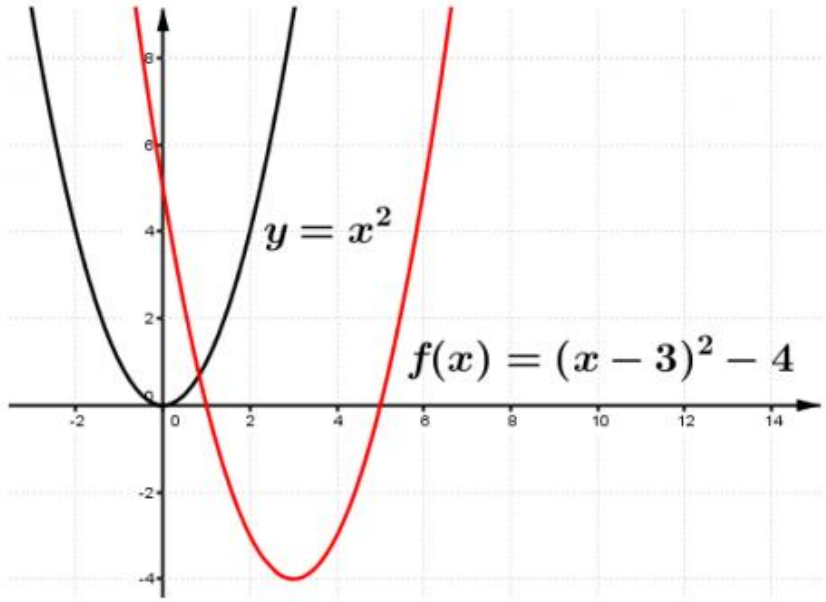
```
<canvas width="450" height="400" id="canvas"></canvas>

<script>
    const canvas = document.querySelector("canvas");
    const ctx = canvas.getContext("2d");
    ctx.fillStyle = "#00eeff"
    ctx.fillRect(150, 0, 50, 50)

    //obracamy canvas
    ctx.fillStyle = "#ffee00"
    ctx.rotate(0.25*Math.PI)
    ctx.fillRect(150, 0, 50, 50)
</script>
```

Transformacja `rotate` jest wykonywana względem punktu 0, 0. Niestety w canvas nie mamy właściwości transform-origin, która w css pozwala na zmianę takiego punktu. Żeby obracać dany element względem konkretnego punktu, musimy skorzystać z dodatkowych obliczeń.

Dla wyjaśnienia działania translate – przyjrzyjmy się rysunkowi poniżej:



Na początku narysowany został wykres funkcji $y=x^2$. Aby narysować wykres funkcji $f(x)=(x-3)2-4$ należałoby obliczyć nowe współrzędne poszczególnych punktów. Zajmuje to dość dużo czasu. Da się pominąć całą tą operację. Tak naprawdę możemy skorzystać z przesunięcia funkcji o wektor. Jeśli przesuniemy funkcję y o wektor $\vec{v}=[p,q]$ to otrzymamy funkcję $f(x)=f(x-p)+q$. Odczytajmy więc nasze współrzędne p oraz q – mają one wartość 3 i -4. Przesuwamy więc funkcję o wektor $\vec{v}=[3,-4]$. Możemy zamiast funkcji przesunąć układ współrzędnych i wtedy nasz początek układu znajdzie się w punkcie $[3,-4]$ - możemy wtedy w tym nowym układzie narysować wykres funkcji $y=x^2$.

Podobnie działa translate – przesuwamy canvas o daną wartość – jego początek znajduje się tak jakby w nowym punkcie, a co za tym idzie – punkt ten stanowi nowy punk (0,0) i względem niego możemy rozpocząć rysowanie.

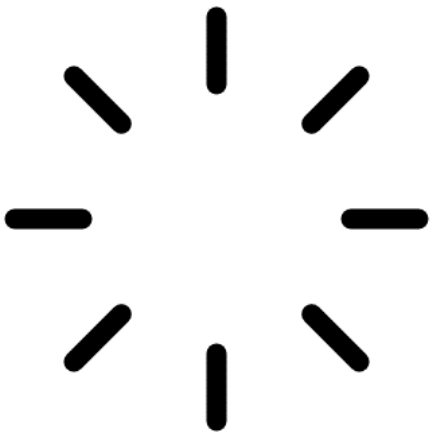
```
<canvas width="400" height="400" id="canvas"></canvas>

<script>
    const canvas = document.querySelector("canvas");
    const ctx = canvas.getContext("2d");

    const angle = 0.25*Math.PI

    function draw() {
        ctx.moveTo(100,0)
        ctx.lineTo(150,0)
        ctx.moveTo(-100,0)
        ctx.lineTo(-150,0)
    }

    ctx.lineWidth = 15
    ctx.lineCap = "round"
    ctx.translate(canvas.width/2, canvas.height/2); //przesuwamy układ
    ctx.beginPath();
    draw()
    ctx.rotate(angle)
    draw()
    ctx.rotate(angle)
    draw()
    ctx.rotate(angle)
    draw()
    ctx.stroke()
</script>
```



Jeśli chcielibyśmy np. obrócić tekst względem nowego punktu (0, 0), musimy posłużyć się wartościami wyrównującymi.

```
<canvas width="400" height="400" id="canvas"></canvas>

<script>
    const canvas = document.querySelector("canvas");
    const ctx = canvas.getContext("2d");

    const angle = 0.5*Math.PI

    function drawText() {
        ctx.textAlign = "center";
        ctx.textBaseline = "middle";
        ctx.font = `bold 60px Arial, sans-serif`;
        ctx.fillText("Canvas text", 0, 0)
    }

    ctx.translate(canvas.width/2, canvas.height/2); //przesuwamy układ
    ctx.beginPath();
    drawText()
    ctx.rotate(angle)
    drawText()
</script>
```

Teraz sprawdźmy działanie `scale()` w praktyce. Na początku przesuniemy nasz canvas. Następnie spróbujemy zastosować różne wartości argumentów x i y. Wyjściowy kod:

```
<canvas width="500" height="400" id="canvas"></canvas>

<script>
    const canvas = document.querySelector("canvas");
    const ctx = canvas.getContext("2d");

    function drawText() {
        ctx.textAlign = "center";
        ctx.textBaseline = "middle";
        ctx.font = `bold 60px Arial, sans-serif`;
        ctx.strokeText("Canvas", 0, 0)
    }

    ctx.translate(250, 200); //obracamy tekst w pozycji 250,200
    drawText()
</script>
```

Spróbujmy teraz zastosować naszą metodę `scale()`.

```
<canvas width="500" height="400" id="canvas"></canvas>

<script>
    const canvas = document.querySelector("canvas");
    const ctx = canvas.getContext("2d");

    function drawText() {
        ctx.textAlign = "center";
        ctx.textBaseline = "middle";
        ctx.font = `bold 60px Arial, sans-serif`;
        ctx.strokeText("Canvas", 0, 0)
    }

    ctx.translate(250, 200); //obracamy tekst w pozycji 250,200
    ctx.scale(1, 1)
    drawText()

</script>
```

Dla wartości (1, 1) nie widać różnicy. Oznacza to, że (1, 1) możemy potraktować jako wartości „wyjściowe”.

Spróbujmy teraz zastosować (-1, 1):

```
<canvas width="500" height="400" id="canvas"></canvas>

<script>
    const canvas = document.querySelector("canvas");
    const ctx = canvas.getContext("2d");

    function drawText() {
        ctx.textAlign = "center";
        ctx.textBaseline = "middle";
        ctx.font = `bold 60px Arial, sans-serif`;
        ctx.strokeText("Canvas", 0, 0)
    }

    ctx.translate(250, 200); //obracamy tekst w pozycji 250,200
    ctx.scale(-1, 1)
    drawText()

</script>
```

Jak widać – nastąpiła zmiana w poziomie. Analogicznie – jeśli zastosowalibyśmy wartości (1, -1) to wtedy tekst zostałby przerzucony w pionie.

Teraz wartość (-0,5, 2):

```
<canvas width="500" height="400" id="canvas"></canvas>

<script>
    const canvas = document.querySelector("canvas");
    const ctx = canvas.getContext("2d");

    function drawText() {
        ctx.textAlign = "center";
        ctx.textBaseline = "middle";
        ctx.font = `bold 60px Arial, sans-serif`;
        ctx.strokeText("Canvas", 0, 0)
    }

    ctx.translate(250, 200); //obracamy tekst w pozycji 250,200
    ctx.scale(-0.5, 2)
    drawText()

</script>
```

Tekst został przerzucony w poziomie i zwężony o połowę. Zmieniła się także jego wysokość.

I ostatnia próba – (2, -2):

```
<canvas width="500" height="400" id="canvas"></canvas>

<script>
    const canvas = document.querySelector("canvas");
    const ctx = canvas.getContext("2d");

    function drawText() {
        ctx.textAlign = "center";
        ctx.textBaseline = "middle";
        ctx.font = `bold 60px Arial, sans-serif`;
        ctx.strokeText("Canvas", 0, 0)
    }

    ctx.translate(250, 200); //obracamy tekst w pozycji 250,200
    ctx.scale(2, -2)
    drawText()

</script>
```

Tekst został przerzucony w pionie i powiększony dwukrotnie.

ĆWICZENIE 4 – transformacja, obrót skalowanie

Utwórz plik o nazwie imie_nazwisko_cw4.html. Utwórz prostą stronę zawierającą element canvas o dowolnych wymiarach. Korzystając ze znanych Ci wiadomości spróbuj utworzyć prostą tarczę zegara. Wykorzystaj poznane wiadomości. Gotowy plik zamieść na moodle.

Przykładowe rozwiązanie: