Rysowanie łuków w ścieżkach – arc

Każdy zna podstawową metodę rysowania łuków – bierzmy do dłoni cyrkiel, ustawiamy go w wybranym punkcie, rozciągamy do wymaganego wymiaru i rysujemy potrzebne nam łuki. W przypadku canvasa do rysowania łuków służy funkcja `arc(x, y, radius, startAngle, endAngle, anticlockwise*)`, w której:

- ✓ `x, y` oznaczają miejsce postawienia „cyrkla”
- ✓ `radius` określa promień
- ✓ `startAngle` określa początkowy kąt rysowanego łuku (podajemy go w radianach)
- ✓ `endAngle` określa końcowy kąt rysowanego łuku (podajemy go w radianach)
- ✓ `anticlockwise` określa czy rysować z kierunkiem wskazówek zegara, czy odwrotnie

Krótkie przypomnienie, jak działa zamiana stopni na radiany:

$$360° = 2\pi \rightarrow n° = \frac{n\pi}{180} \; lub \; n° = \frac{2n\pi}{360}$$

Spróbujmy przeliczyć, jak wygląda to dla kąta 45 stopni:

$$45° = \frac{45\pi}{180} = \frac{\pi}{4}$$

Oraz dla kąta 270 stopni:

$$270° = \frac{370\pi}{180} = \frac{3\pi}{2}$$

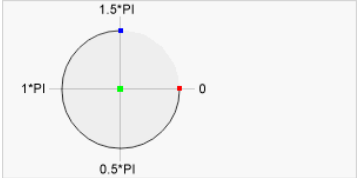
W JS możemy napisać funkcję, która pozwoli na przeliczanie stopni na radiany:

```
const angleToRadian = function(angle) {
    return Math.PI/180 * angle;
}
```

Możemy oczywiście wpisywać też „ręcznie” poszczególne kąty – pokażę oba przykłady. Spróbujmy narysować prosty łuk, którego środek znajduje się w połowie wysokości oraz szerokości naszego elementu canvas. Trzeci argument wskazuje na promień (100), natomiast argument czwarty to początek rysunku, piąty – koniec.

```
<canvas width="200" height="200" id="canvas">
</canvas>

<script>
    const angleToRadian = function(angle) {
        return Math.PI/180 * angle;
    }

    const canvas = document.querySelector("#canvas");
    const ctx = canvas.getContext("2d");

    ctx.beginPath();
    ctx.arc(canvas.width/2, canvas.height/2, 100, 0, angleToRadian(135));
    ctx.stroke();
</script>
```

Jak łatwo zauważyć, po prawej stronie znajduje się początek – nasze 0 (ułożenie poszczególnych kątów możesz sprawdzić na rysunku obok).



W tym przypadku skorzystaliśmy z funkcji. Poniżej przedstawię rozwiązanie, które nie wymaga wykorzystania funkcji, a jedynie przeliczenia odpowiedniego kąta i wpisania wartości do naszej funkcji.

Skorzystajmy ze znanego nam już wzoru:

$$n° = \frac{n\pi}{180} \rightarrow 135° = \frac{135\pi}{180} = \frac{3\pi}{4} = 0{,}75\pi$$

Teraz zapiszmy to w kodzie. W miejsce kąta wstawiamy obliczony przez nas wynik:

```
<canvas width="200" height="200" id="canvas">
</canvas>

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

    ctx.beginPath();
    ctx.arc(canvas.width/2, canvas.height/2, 100, 0, 0.75*Math.PI);
    ctx.stroke();
</script>
```

Spróbujmy teraz narysować półkole, ale tak, jakbyśmy wycinali tylko górną część okręgu (załóżmy, że promień ma 75 oraz, że zaczynami rysunek w środku elementu canvas). Obliczamy potrzebne nam kąty. Łuk powinien rozpoczynać się w kącie 180 stopni, a kończyć w kącie 360 stopni.

$$180° = \frac{180\pi}{180} = \pi \qquad 360° = \frac{360\pi}{180} = 2\pi$$

Podstawiamy dane do naszej funkcji i otrzymujemy wynik:

```
<canvas width="200" height="400" id="canvas">
</canvas>

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

    ctx.beginPath();
    ctx.arc(canvas.width/2, canvas.height/2, 75, Math.PI, 2*Math.PI);
    ctx.stroke();
</script>
```

Zgodnie z założeniem – narysowaliśmy interesującą nas figurę. Jeśli chcielibyśmy ją wypełnić, zamiast obrysować – wystarczy, że zmienimy funkcję `stroke()` na `fill()` i gotowe. Jednak nie zawsze działa to tak jak chcemy.

```
<canvas width="200" height="400" id="canvas">
</canvas>

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

    ctx.beginPath();
    ctx.arc(canvas.width/2, canvas.height/2, 75, Math.PI, 2*Math.PI);
    ctx.fill();
</script>
```

W ten sposób możemy tworzyć dowolne łuki oraz wypełnione figury wycięte z koła. Sprawdźmy, jak działa parametr `anticlockwise`. Domyślnie przyjmuje on wartość false – czyli rysunek podąża za wskazówkami. Spróbujmy do poprzedniej funkcji dodać parametr `anticlockwise` ustawiony na wartość `true`.

```
<canvas width="200" height="400" id="canvas">
</canvas>

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

    ctx.beginPath();
    ctx.arc(canvas.width/2, canvas.height/2, 75, Math.PI, 2*Math.PI, true);
    ctx.fill();
</script>
```

Jak widać, rysunek rozpoczął się w tym samym punkcie, ale rysowanie nastąpiło w przeciwnym kierunku.

ĆWICZENIE 4 – rysowanie łuków – arc

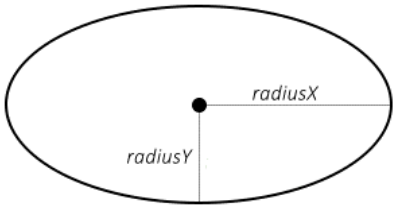
Utwórz plik o nazwie cw3_canvas.html. Utwórz prostą stronę zawierającą element canvas o dowolnych wymiarach. Narysuj:

1) koło (fill)
2) dowolny łuk oparty o funkcję arc

Wymiary obu figur dowolne (byle mieszczące się w elemencie canvas). Miejsce rozpoczęcia rysowania również dowolne. Gotowy plik zamieść na platformie moodle.

Rysowanie łuków w ścieżkach – ellipse

Podobną w działaniu do `arc()` jest metoda `ellipse(x, y, radiusX, radiusY, rotation, startAngle, endAngle, anticlockwise*)`, która służy do rysowania elips.

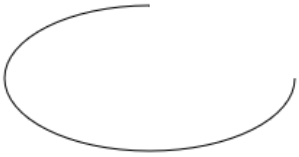


Różnicą między tymi dwoma funkcjami jest to, że zamiast jednego promienia podajemy 2: `radiusX` oraz `radiusY`. Dodatkowo możemy określić też kąt obrotu elipsy – `rotation`.

Spróbujmy narysować wycinek elipsy – łuk o zakresie od 0 do 270 stopni. Załóżmy, że środek znajduje się w połowie wysokości i szerokości canvas, promienie wynoszą odpowiednio 100 i 50, a kąt obrotu jest równy 0. Pamiętamy zależność z poprzednich przykładów i wiemy, jak obliczyć potrzebny kąt (otrzymamy wynik $1{,}5\pi$).

Podstawmy wszystko do funkcji:

```
<canvas width="250" height="300" id="canvas">
</canvas>

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

    ctx.beginPath();
    ctx.ellipse(canvas.width/2, canvas.height/2, 100, 50, 0, 0, 1.5*Math.PI);
    ctx.stroke();
</script>
```



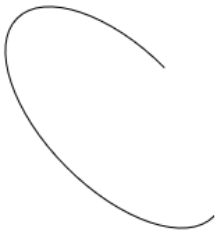
Jak widać, udało się narysować żądany łuk. Jeśli spróbujemy dodać argument `anticlockwise` ustawiony na wartość `true` otrzymamy następujący efekt:

```
<canvas width="250" height="300" id="canvas">
</canvas>

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

    ctx.beginPath();
    ctx.ellipse(canvas.width/2, canvas.height/2, 100, 50, 0, 0, 1.5*Math.PI, true);
    ctx.stroke();
</script>
```

Spróbujmy teraz obrócić figurę o kąt 45 stopni. Obliczamy jego wartość w radianach ($0{,}25\pi$).

Modyfikujemy kod – zmieniamy wartość kąta obrotu oraz usuwamy parametr `anticlockwise`:

```
<canvas width="250" height="300" id="canvas">
</canvas>

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

    ctx.beginPath();
    ctx.ellipse(canvas.width/2, canvas.height/2, 100, 50, 0.25*Math.PI, 0, 1.5*Math.PI);
    ctx.stroke();
</script>
```



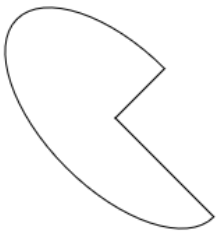
Zgodnie z przewidywaniami nasza figura obróciła się o 45 stopni.

Spróbujmy teraz wypełnić naszą figurę:

```
<canvas width="250" height="300" id="canvas">
</canvas>

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

    ctx.beginPath();
    ctx.ellipse(canvas.width/2, canvas.height/2, 100, 50, 0.25*Math.PI, 0, 1.5*Math.PI);
    ctx.fill();
</script>
```

Jak widać, powstało połączenie pierwszego punktu z ostatnim. Jeśli chcielibyśmy, żeby wypełniony został obszar wycięty przez naszą figurę i jej środek, to możemy narysować dodatkową linię prowadzącą do środka elipsy (`lineTo()`), a następnie zamknąć ścieżkę (`closePath()`):

```
<canvas width="250" height="300" id="canvas">
</canvas>

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

    ctx.beginPath();
    ctx.ellipse(canvas.width/2, canvas.height/2, 100, 50, 0.25*Math.PI, 0, 1.5*Math.PI);
    ctx.lineTo(canvas.width/2, canvas.height/2);
    ctx.closePath();
    ctx.stroke();
</script>
```



Wypełnijmy obszar:

```
<canvas width="250" height="300" id="canvas">
</canvas>

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

    ctx.beginPath();
    ctx.ellipse(canvas.width/2, canvas.height/2, 100, 50, 0.25*Math.PI, 0, 1.5*Math.PI);
    ctx.lineTo(canvas.width/2, canvas.height/2);
    ctx.closePath();
    ctx.fill();
</script>
```

Jak łatwo zauważyć, teraz wypełniony obszar bardziej odpowiada naszym wymaganiom.

ĆWICZENIE 5 – rysowanie łuków – ellipse

Utwórz plik o nazwie cw4_canvas.html. Utwórz prostą stronę zawierającą element canvas o dowolnych wymiarach. Narysuj:

1) dowolną elipsę obróconą o dowolny kąt (stroke)
2) dowolną wypełnioną figurę stanowiącą wycinek elipsy (fill)

Wymiary obu figur dowolne (byle mieszczące się w elemencie canvas). Miejsce rozpoczęcia rysowania również dowolne. Gotowy plik zamieść na platformie moodle.