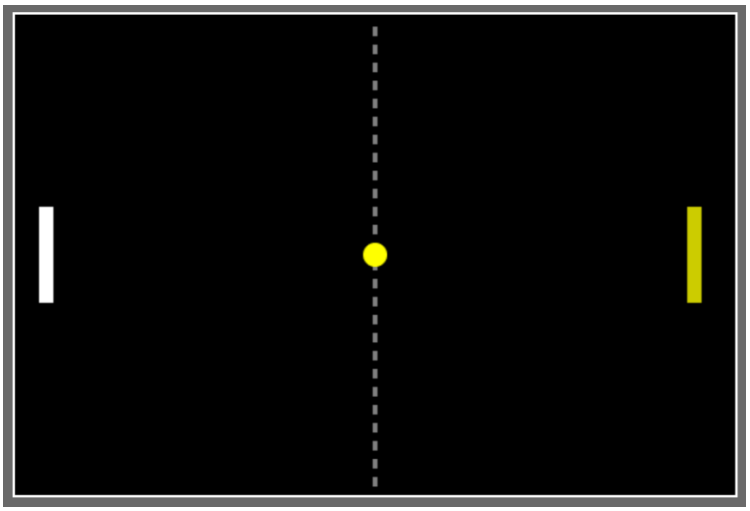
Ćwiczenie 3 – Gra w ping-ponga



To ćwiczenie polega na zbudowaniu gry w ping-ponga. Przeciwnikiem gracza (człowiek) jest komputer.

Ćwiczenie podzielono na dwa etapy (dwa kolejne tygodnie):

Cz. I:

1. **Przygotowanie dokumentu HTML**
2. **Przygotowanie grafiki gry**
3. **Podstawowa animacja piłki i gracza 1**
4. **Wykrycie kolizji piłki z krawędziami pola gry**

Cz. II:

1. **Przyspieszanie ruchu piłki**
2. **Animacja gracza 2 (komputer)**
3. **Wykrycie kolizji z paletkami**
4. **Wyświetlanie komunikatów**

Część I.

Przygotuj dokument HTML

Przykładowo:

- w dokumencie html w sekcji head tworzymy style dla dokumentu i elementu canvas.

```
<style>
        body {
            margin: 0;
```

```
            padding: 0;
            height: 100vh;
            display: flex;
            align-items: center;
            justify-content: center;
            background-color: dimgray;
            }
        canvas {
           border: 2px solid white;
           }
    </style>
```

- w body dokumentu umieszczamy element canvas, a poniżej umieszczamy skrypt do obsługi gry (lub w dowolny sposób – np. w head):

```
<canvas></canvas>
<script> code here </script>
```

Tradycyjnie wewnątrz skryptu powołujemy obiekt do obsługi canvas i pobieramy obiekt rysowania w kontekście 2d:

```
const canvas = document.querySelector('canvas');
const ctx = canvas.getContext('2d');
```

(querySelector – służy do wskazania/wyszukania pojedynczego elementu, pamiętamy że elementowi canvas można nadać także identyfikator id i użyć metody getElementById).

Przygotuj kod do rysowania obiektów: tło, piłka, 2 rakietki

Każdy element może być rysowany w odrębnej funkcji, które następnie zostaną wywołane wewnątrz „głównej" funkcji, np.:

Tło:

```
function table() {
        // funkcja rysująca stół
        ctx.fillStyle = 'black';
        ctx.fillRect(0, 0, cw, ch);    //omówienie wartości cw i ch poniżej)
        // tu można dodać także kod rysowania linii środkowej także omówiony
        poniżej;
    }
```

Piłka:

```
function ball() {
       // code here;
       // funkcja rysująca piłkę
}
```

Rakietki:

```
function player1() {
	// code here;
	// funkcja rysująca rakietkę lewą sterowaną za pomocą ruchu myszy
}


function player2() {
	// code here;
	// funkcja rysująca rakietkę prawą poruszaną automatycznie
}


function game(){
	table();
	ball();
	palyer1();
	player2();
}
```

cw i ch w użyte w funkcji table to stałe zadeklarowane globalnie i odnoszące się do szerokości i wysokości canvas:

```
canvas.width = 600;  //lub dowolna inna wartość
canvas.height = 400;  //lub dowolna inna wartość

const cw = canvas.width;
const ch = canvas.height;
```

W ten sam sposób, czyli jako stałe w przestrzeni globalnej, należy zadeklarować także stałe dla:

- **promienia piłki,**
- **wysokości i szerokości rakietek,**
- **współrzędnych x rakietek (osobno dla każdej z nich) – ta współrzędna nie zmieni się w trakcie gry - uwzględniając, że paletki mogą być lekko odsunięte od krawędzi pola gry (lewej lub prawej) o 20-40 px.**

Potrzebne zmienne globalne:

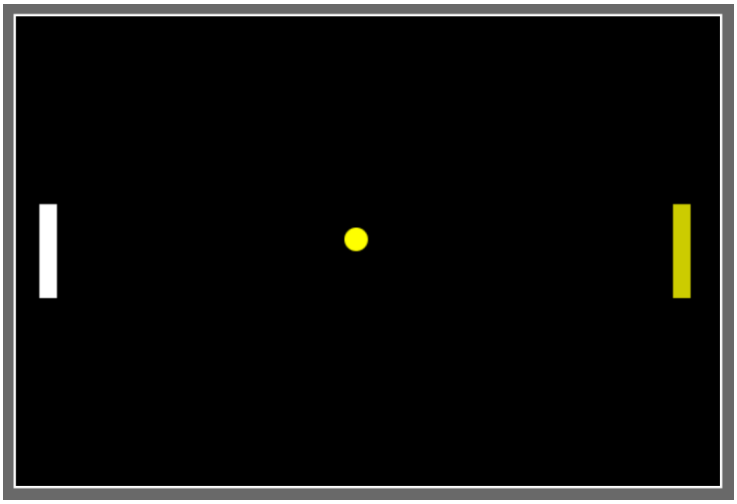
- **- współrzędne piłki**
- **- współrzędne y paletek**

```
const ballSize = 10; //promień piłki;
let ballX = cw / 2;  //początkowe położenie x piłki to połowa szerokości canvas
let ballY = ch / 2;  //początkowe położenie y piłki to połowa wysokości canvas

// dodać 2 stałe dla określenie szerokości i wysokości paletek
// dodać 2 stałe dla określenie współrzędnej x obu paletek
// dodać 2 zmienne dla określenia współrzędnej y obu paletek
```

Współrzędne piłki i współrzędna y paletek będą ulegały zmianie w trakcie gry i te zmienne deklarujemy w przestrzeni globalnej poprzez let lub var. Przy czym, wartości współrzędnych położenia piłki i rakietek można powiązać odpowiednio z właściwością width i height obiektu canvas.

Nadaj zmiennym początkowe wartości uwzględniając położenie obiektów jak na rysunku (kolory obiektów są dowolne):



Dodanie linii dzielącej pole gry na połowę:

Można zamieścić odpowiedni kod w funkcji table(), przy czym jeśli chcemy uzyskać linię przerywaną składającą się z osobnych prostokątów rysujemy je za pomocą pętli, np.:

```
function table() {
    // w funkcji jest już kod rysujący prostokąt tła
    for (let linePosition = 10; linePosition < ch; linePosition += 15) {
            // kod rysowania prostokąta, uwzględniający wartość linePosition jako
            współrzędną y położenia prostokąta; współrzędna x jest wartością
            stałą.
        }
}
```

Animacja piłki

Zadeklaruj zmienne globalne dla prędkości poruszania się piłki:

```
let ballSpeedX = 1; //wartość przykładowa, później może być nieco większa
let ballSpeedY = 1; //wartość przykładowa, j.w.
```

i w funkcji rysującej piłkę dodaj linie kodu poruszające piłkę:

```
ballX += ballSpeedX;
ballY += ballSpeedY;
```

Poza funkcjami dodaj metodę setInterval do wywoływania rysowania wszystkich elementów (animacji) w określonych odstępach czasu:

```
setInterval(game, 1000/60);   //później metodę można przypisać do zmiennej, aby
można było zatrzymać animację.
```

Wykrycie kolizji z krawędziami pola gry

Wewnątrz funkcji rysującej piłkę dodaj dwie instrukcje warunkowe, które wykrywając położenie piłki tuż przy krawędzi będą zmieniać wartość położenia piłki (ballSpeedX , ballSpeedY) i w ten sposób odbijać piłkę od krawędzi, np. dla zetknięcia z lewą i prawą krawędzią canvas:

```
if (ballX <= 0 || ballX >= cw) {
            ballSpeedX = -ballSpeedX;
        }
// dodać kolejny if, który zmieni wartość ballSpeedY po zetknięciu z górną i
dolną krawędzią canvas
```

Warunki muszą zostać odpowiednio skorygowane, tak aby piłka nie przekraczała krawędzi pola gry tylko zbliżała się do niego maksymalnie na długość swojego promienia.

Poruszanie rakietki lewej (gracz)

Użyj metody addEventListener do nasłuchiwania ruchu myszy w obrębie elementu canvas:

```
canvas.addEventListener('mousemove', playerPosition);
```

Funkcja, którą należy teraz dodać (w powyższym przykładzie nazwana playerPosition), zostanie wywołana w momencie wykrycia zdarzenia. Będzie pobierać informacje o zdarzeniu (event), czyli w tym wypadku ruchu myszy. Informacje te to między innymi dane o położeniu kursora myszy w oknie przeglądarki (odpowiadają za to właściwości clientX i clientY).

Ponieważ wartość położenia kursora myszy w obrębie okna przeglądarki nie pokrywa się ze współrzędnymi w obrębie obiektu canvas, do skorygowania położenia myszy względem canvas należy użyć zmiennej określającej położenie górnej krawędzi canvas względem okna przeglądarki:

```
var topCanvas = canvas.offsetTop;

function playerPosition(evt){
         playerY = evt.clientY - topCanvas ;

          //playerY to współrzędna y lewej paletki, która będzie zmieniać się
          wraz z położeniem kursora myszy, powyższy kod należy skorygować tak,
          aby kursor myszy pokrywał się z połową wysokości paletki
}
```

Kod w powyższej funkcji należy uzupełnić jeszcze w ten sposób, by paletka nie wysuwała się poza górną i dolną krawędź canvas, np.:

```
if (playerY <= 0){
         playerY = 0;
}
// dodać kolejny if, który ograniczy ruch paletki do dolnej krawędzi canvas
```