Jeu du Casse Brique en javascript partie 2

Déplacement du paddle :

Dans la fonction **draw(),** on va tester si c'est la gauche ou la droite qui est utilisé à chaque rafraichissement d'image.

```
if(rightPressed)
 {
    paddleX += 7;
}
else if(leftPressed)
{
    paddleX -= 7;
}
```

Si on appuie sur la gauche le paddle se déplace de 7 pixels à gauche et si on appuie sur la droite de 7 pixels à droite. Ça fonctionne mais le paddle sort du canevas, on va donc modifier les conditions pour que le paddle s'arrêtes au bord du canevas :

```
if(rightPressed && paddleX < canvas.width-paddleWidth)
 {
    paddleX += 7;
}
else if(leftPressed && paddleX > 0)
{
    paddleX -= 7;
}
```

La position sur l'axe x **paddleX** du paddle bouge de **0** à gauche jusqu'à **canvas.width-paddleWidth** à droite(les coordonnées du paddle sont le x et le y du coin gauche en haut comme pour le canvas).

Ajouter le game over :

Pour ajouter le game over on va modifier le code qui nous permet de faire rebondir la balle sur les murs :

```
if(posX + dx > canvas.width-ballRadius || posX + dx < ballRadius )
{
dx = -dx;
}


if(posY + dy > canvas.height-ballRadius || posY + dy < ballRadius )
{
dy = -dy;
}
```

On ne va plus permettre à la balle de rebondir sur les 4 murs mais que sur 3(gauche, haut et droit). Toucher le mur du bas mettra fin à la partie.

On va donc modifier le second **if** (l'axe vertical y) en y ajoutant un **else if** qui déclenchera un Game Over. Pour l'instant on va juste afficher un message et restart le jeu en rechargeant la page.

```
if(posY + dy < ballRadius )
{
dy = -dy;
}
else if(posY + dy > canvas.height-ballRadius)
{
    alert("GAME OVER !!!");
    document.location.reload();
}
```

Faire rebondir la balle sur le paddle :

Il reste encore une chose à faire, il faut créer une détection de collision entre le paddle et la balle. Le plus simple à faire est de vérifier si le centre de la balle se trouve entre les bords droit et gauche du paddle :

```
if(posY + dy < ballRadius )
{
dy = -dy;
}
else if(posY + dy > canvas.height-ballRadius)
{
    if(posX > paddleX && posX < paddleX + paddleWidth)
    {
        dy = -dy;
    }
    else
    {
        alert("GAME OVER !!!");
        document.location.reload();
    }
}
```

Mettre en place les variables « briques »

Pour créer les briques on va avoir besoins de plusieurs choses :

- Une boucle imbriquée qui va parcourir un tableau à 2 dimensions
- Quelques variables pour définir les informations des briques tel que la largeur, la hauteur, les colonnes, les lignes ….

Ajoutons tout d'abords ces quelques variables :

```
var brickRowCount = 3;
var brickColumnCount = 5;
var brickWidth = 75;
var brickHeight = 20;
var brickPadding = 10;
var brickOffsetTop = 30;
var brickOffsetLeft = 30;
```

Dans l'ordre nous avons défini :

- Le nombre de lignes : 3
- Le nombre de colonnes : 5
- Une largeur : 75
- Une hauteur : 20

- Un padding entre les briques pour qu'elles ne se touche pas entre elles et qu'elles ne commencent pas a être tracé sur le bord

On va ensuite stocker nos briques dans un tableau à 2 dimensions. Il contiendra les colonnes (c), qui contiendrons les lignes(r) qui chacune contiendrons un objet défini par une position x et y :

```
var bricks = [];
for(var c=0; c<brickColumnCount; c++)
{
    bricks[c] = [];
    for(var r=0; r<brickRowCount; r++)
    {
        bricks[c][r] = { x: 0, y: 0 };
    }
}
```

Dessiner les briques :

On va maintenant créer une fonction pour dessiner les briques :

```
function drawBricks()
{
    for(var c=0; c<brickColumnCount; c++)
    {
        for(var r=0; r<brickRowCount; r++)
        {
            bricks[c][r].x = 0;
            bricks[c][r].y = 0;
            ctx.beginPath();
            ctx.rect(0, 0, brickWidth, brickHeight);
            ctx.fillStyle = "#ed151b";
            ctx.fill();
            ctx.closePath();
        }
    }
}
```

On parcoure les colonnes et les lignes pour attribuer une position x et y à chaque brique, on les dessine sur le canvas à chaque itération de la boucle. Le problème est qu'elle s'affiche toutes au même endroit, aux coordonnées (0,0). Pour résoudre ce problème on va ajouter un petit calcul qui va définir la position x et y de chaque brique à chaque itération :

```
var brickX = (c*(brickWidth+brickPadding))+brickOffsetLeft;
var brickY = (r*(brickHeight+brickPadding))+brickOffsetTop;
```

Puis dans la fonction on assigne les valeurs de **brickX** et **brickY** comme coordonnées, a chaque fois, voici la version finale de la fonction **drawBricks()** :

```
function drawBricks()
 {
    for(var c=0; c<brickColumnCount; c++)
     {
        for(var r=0; r<brickRowCount; r++)
        {
            var brickX = (c*(brickWidth+brickPadding))+brickOffsetLeft;
            var brickY = (r*(brickHeight+brickPadding))+brickOffsetTop;

            bricks[c][r].x = brickX;
            bricks[c][r].y = brickY;
            ctx.beginPath();
            ctx.rect(brickX, brickY, brickWidth, brickHeight);
            ctx.fillStyle = "#ed151b";
            ctx.fill();
            ctx.closePath();
         }
    }
}
```

Détecter les collisions :

On va faire une fonction de détection de collisions qui va parcourir toutes les briques et comparer la position de chaque brique avec les coordonnées de la balle lorsque chaque image est dessinée. On va définir une variable **b** pour stocker l'objet brique dans chaque boucle de la détection de collisions.

```
function collisionDetection()
{
    for(var c=0; c<brickColumnCount; c++)
     {
        for(var r=0; r<brickRowCount; r++)
        {
            var b = bricks[c][r];
             // calculs
         }
    }
}
```

Si le centre de la balle se trouve à l'intérieur des coordonnées d'une de nos briques, nous changerons la direction de la balle. Pour que le centre de la balle soit à l'intérieur de la brique, les quatre affirmations suivantes doivent être vraies :

- La position x de la balle est supérieure à la position x de la brique.
- La position x de la balle est inférieure à la position x de la brique plus sa largeur.
- La position y de la balle est supérieure à la position y de la brique.
- La position y de la balle est inférieure à la position y de la brique plus sa hauteur.

```
function collisionDetection()
{
    for(var c=0; c<brickColumnCount; c++)
     {
        for(var r=0; r<brickRowCount; r++)
        {
            var b = bricks[c][r];
            if(posX > b.x && posX < b.x + brickWidth && posY > b.y && posY < b.y + brickHeight)
                dy = -dy;
         }
    }
}
```

Fin de la deuxième partie

Exercice : Faites en sorte que quand la balle touche une brique celle-ci disparaisse