Jeu du Casse Brique en javascript partie 3

Faire disparaitre les briques après que la balle les a touchées :

Pour faire disparaitre les briques qui ont été touché par la balle on va rajouter un paramètre status à chaque brique comme ceci :

```
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, status: 1 };
    }
}
```

Puis il faudra tester la valeur **status** de chaque brique dans la fonction **drawBricks()** avant de la dessiner. Si **status** vaut 1 alors on dessine, si il vaut 0 ça veux dire que la balle a touchée la brique donc on ne l'a dessine pas :

```
function drawBricks()
{
    for(var c=0; c<brickColumnCount; c++)
    {
        for(var r=0; r<brickRowCount; r++)
        {
            if(bricks[c][r].status == 1)
            {
                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();
            }
        }
    }
}
```

Suivi et mise à jour de l'état dans la fonction de détection de collision :

Il faut maintenant impliquer le status de brique dans la fonction **collisionDetection() :** si la brique est active son status vaut 1, on testera si la collision a eu lieu, si il y a eu une collision on définira le status sur 0 :

```
function collisionDetection()
{
    for(var c=0; c<brickColumnCount; c++)
    {
        for(var r=0; r<brickRowCount; r++)
        {
            var b = bricks[c][r];
            if(b.status == 1)
            {
                if(posX > b.x && posX < b.x + brickWidth && posY > b.y && posY < b.y + brickHeight)
                {
                    dy = -dy;
                    b.status = 0;
                }
            }
        }
    }
}
```

Calculer le score :

On va d'abord définir une variables scores initialisées à 0 :

```
var scores = 0;
```

Puis une fonction **drawScore()** pour créer et mettre à jour le score :

```
function drawScore()
{
    ctx.font = "16 px Arial";
    ctx.fillStyle = "#ed151b";
    ctx.fillText("Score: " + scores, 8, 20);
}
```

- font() : pour définir la taille et le type de la police
- fillStyle() : pour définir la couleur de la police
- fillText() : pour définir la position du texte sur le canvas, le 1er paramètre est le texte et les 2 autres paramètres sont les coordonnées où le texte est placés sur le canvas

Puis il faut incrémenter le score à chaque collision d'une brique, on ajoutes une ligne dans la fonction **collisionDetection() :**

```
function collisionDetection()
{
    for(var c=0; c<brickColumnCount; c++)
     {
        for(var r=0; r<brickRowCount; r++)
        {
            var b = bricks[c][r];
            if(b.status == 1)
            {
                if(posX > b.x && posX < b.x + brickWidth && posY > b.y && posY < b.y + brickHeight)
                {
                    dy = -dy;
                    b.status = 0;
                    scores++;
                }
            }
        }
    }
}
```

Ajouter un message de victoire lorsque toutes les briques ont été détruites :

On va ajouter un teste dans la fonction **collisionDetection() :**

```
function collisionDetection()
{
    for(var c=0; c<brickColumnCount; c++)
     {
        for(var r=0; r<brickRowCount; r++)
        {
            var b = bricks[c][r];
            if(b.status == 1)
            {
                if(posX > b.x && posX < b.x + brickWidth && posY > b.y && posY < b.y + brickHeight)
                {
                    dy = -dy;
                    b.status = 0;
                    scores++;
                    if(score == brickRowCount*brickColumnCount)
                    {
                        alert("C'est gagné, Bravo!");
                        document.location.reload();
                        clearInterval(interval);
                    }
                }
            }
        }
    }
}
```

Exercice : Ajouter plus de point par brique touchée et indiquez le nombre de points collectés dans la boite d'alerte de fin de partie

Essayer de changer la couleur de la balle à chaque fois qu'elle tape une brique ou un mur

Pour choisir aléatoirement une couleur voici une fonction :

```
function getRandomColor() {
     var letters = '0123456789ABCDEF';
     var color = '#';

        for (var i = 0; i < 6; i++)
        {
           color += letters[Math.floor(Math.random() * 16)];

        }
    return color;
    }
```

Cette fonction est composée :

- Une variable « letters » qui contient les chiffres de 0 à 9 et les lettres de A à F, ce sont tous les caractères qui composent un nombre hexadécimal
- Une variable « color » qui contient le caractère # qui est le premier caractère d'un code couleur en html/css
- Une boucle for qui va de 0 à 6 car un code couleur après le # est composé de 6 caractères parmi ceux contenu dans la variable letters
- A chaque tour de boucle ont choisi aléatoirement un caractère parmi les 16 de la variable letters et on le concatene avec la variable color :
    - 1er tour de boucle #e
    - 2éme tour #ed
    - 3éme #ed1
    - 4éme #ed15
    - 5éme #ed151
    - 6eme #ed151b

Pour en savoir plus sur l'objet Math et ses propriétés .floor et .random aller voir sur ce lien : https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Math

Faites-en sorte que la balle accélère quand elle touche le paddle

Essayer de changer le nombre de briques dans une colonnes ou dans une ligne ou bien leur position

Trouver le moyen d’ajouter le contrôle à la souris

Ajouter un nombre de vie