Coder un Casse-Briques en JavaScript

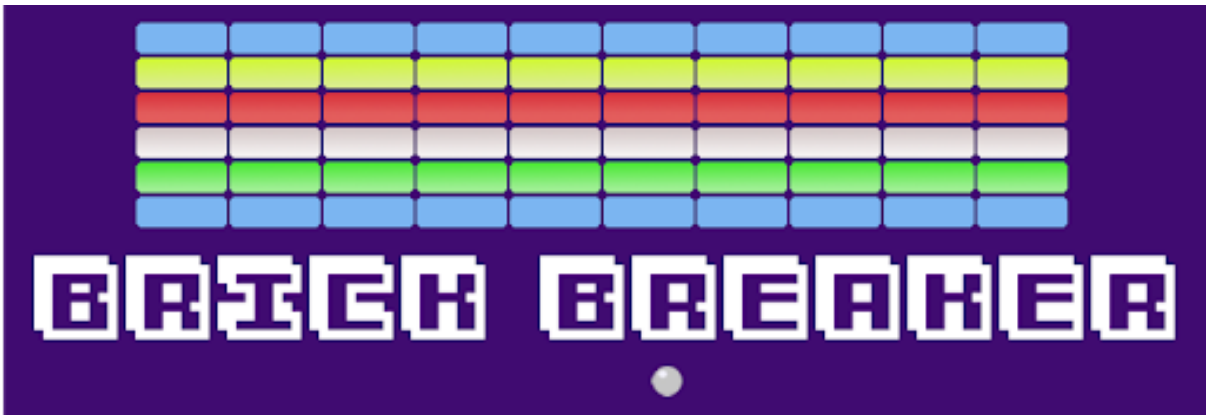
Niveau Débutant • 13-16 ans • 1h30

1. Présentation du jeu

Casse-Briques est un jeu d'arcade apparu en 1975 où le joueur contrôle une raquette et doit faire rebondir une balle pour casser le plus de briques possible, sans toutefois faire tomber la balle.

Certains éléments du code de **Casse-Briques** ont été perdus… À vous de compléter le code du jeu pour qu'il fonctionne de nouveau !

Dans ce sujet, vous apprendrez à programmer des boucles, des conditions en **JavaScript** et le fonctionnement global d'un jeu web.



2. Outils & Ressources

2.1 Ressources

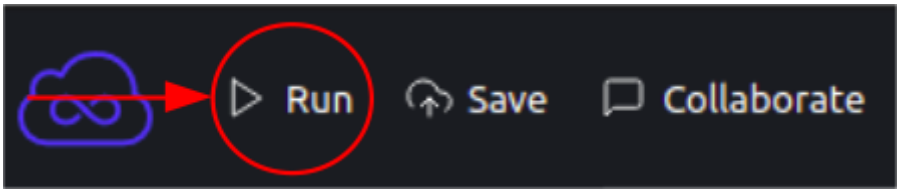
Pour commencer, rendez-vous sur **https://jsfiddle.net/org_emma/cdeua1vo/8/**

L'écran peut être disposé de différentes manières, il te suffit d'aller dans "Settings" en haut à droite de ton écran. Pour la session d'aujourd'hui tu vas sélectionner : "Tabs (columns)". Ton écran est maintenant en deux parties : à gauche les fichiers, c'est là que tu vas programmer le jeu. Et à droite, tu as un écran où tu peux voir ton jeu.

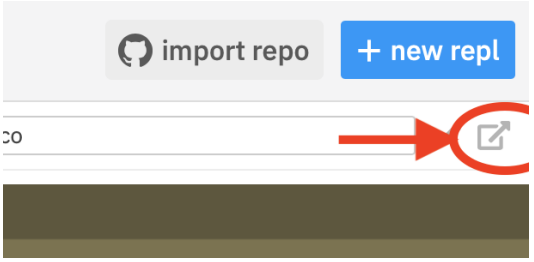
2.2 Prise en main de la plateforme

Sur la partie en haut à gauche de l'écran, tu peux voir plusieurs langages.
Le langage HTML, **JavaScript**, et le langage CSS.
Aujourd'hui, tu n'auras besoin de modifier que le fichier se trouvant dans le langage **JavaScript**. Tu peux toutefois aller voir ce qu'il se passe dans les autres langages si tu le souhaites.

Pour lancer ton code appuie sur la flèche verte "**RUN**" en haut de l'écran.



Si ton écran est trop petit pour afficher le jeu en entier, tu peux l'ouvrir dans un nouvel onglet en cliquant sur le bouton suivant :



2.3 JavaScript

Plusieurs fonctions sont présentes dans le fichier JavaScript. Ton rôle va être de les compléter.

```
function drawBricks()
```

ℹ La fonction **drawBricks** est appelée à chaque tour de boucle. C'est ici que nous coderons l'affichage des briques en état "non-cassé".

```
function updateBallPosition(dx, dy)
```

ℹ La fonction **updateBallPosition** va permettre de modifier la position de la balle selon une vitesse prédéfinie.

```
function didCollideWithBrick(brick)
```

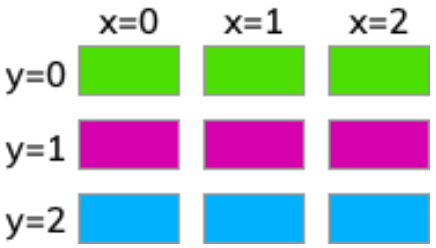
ℹ La fonction **didCollideWithBrick** est la fonction du jeu qui permet de faire disparaître les briques et d'augmenter le compteur de points. Cette fonction est appelée à chaque fois qu'une collision a lieu entre la balle et une brique.

3. Découverte du jeu

3.1 Afficher les briques

Si tu lances le jeu, tu peux t'apercevoir qu'aucune brique ne s'affiche et que la balle ne peut pas bouger ! Pas de panique, la première étape va être d'afficher les briques à l'écran.

Pour cela, il faut compléter la fonction **drawBricks**. Nous allons vérifier l'état de chaque brique, et si l'état (la variable **status**) est égal à **1**, alors on peut l'afficher. Nous allons créer *2 boucles* qui vont nous permettre de parcourir toutes les briques, comme un *tableau à 2 dimensions*. **Une boucle est une instruction qui se répète jusqu'à ce qu'une condition spécifiée soit atteinte**.



Commençons par parcourir les lignes. Ajoute la ligne de code suivante :

```
for (var yPos = 0; yPos < brickRowCount; yPos += 1) {

}
```

Nous allons à présent parcourir chaque colonne de chaque ligne, de manière à pouvoir vérifier chaque brique. Pour cela, ajoute une nouvelle boucle **à l'intérieur de la boucle précédente** (Entre les accolades **{}**).

```
for (var yPos = 0; yPos < brickRowCount; yPos += 1) {
   for (var xPos = 0; xPos < brickColumnCount; xPos += 1) {

   }
}
```

Cette ligne est une condition "**If**" qui signifie “**Si**” en français. Une condition est une action qui se réalise en fonction d’un résultat. Par exemple : “Si tu n’as pas fait tes devoirs, tu ne peux pas sortir”. **Les devoirs** sont le résultat attendu et s’ils ne sont pas faits tu ne peux pas sortir, **sortir** est l’action.

Si la valeur de la brique à la position X et Y est égale à 1 alors on fait une action.

```
if (bricks[xPos][yPos].status == 1) {
    //Une action
}
```

Nous créons maintenant l’action qui suit la condition. Si la condition est vraie alors nous faisons apparaître une brique à la position X et Y avec la fonction **displayBrick**.

```
if (bricks[xPos][yPos].status == 1) {
    displayBrick(xPos, yPos);
}
```

Lance le jeu pour voir ce qu’il se passe !

3.2 Mettre à jour la position de la balle

Maintenant que les briques de **Casse-Briques** s'affichent, il est temps de permettre à la balle de bouger. C'est ce que nous allons programmer dans cette partie.

Tout d'abord, la position de la balle est définie par ses coordonnées x et y, stockées dans les 2 variables suivantes : **x**, **y**. La vitesse de la balle, quant à elle, est définie dans les variables **dx**, **dy**. A chaque tour de boucle, *la vitesse est additionnée à la position de la balle,* ce qui lui permet d'avancer !

A l'intérieur de la fonction **updateBallPosition**, ajoute les 2 lignes suivantes. Comme tu peux le remarquer, la vitesse est donnée avec les paramètres **dx**, **dy** de la fonction.

```
x += dx;
y += dy;
```

3.3 Prise en charge de la collision

Dans cette partie, nous allons effectuer différentes actions qui ont lieu lors d'une collision entre une brique et la balle, comme modifier le statut des briques qui ont été en collision par exemple.

A l'intérieur de la fonction **didCollideWithBrick**, nous allons tout d'abord modifier le statut de la brique qui a été touchée. Lorsque l'état (la variable **status**) est égal à **0**, cela signifie que la brique est cassée. Ajoute la ligne de code suivante :

```
brick.status = 0;
```

Puis nous allons mettre à jour la trajectoire de la balle pour la faire rebondir, ainsi qu'ajouter **1** au compteur de points.
Pour modifier la trajectoire de la balle, nous allons multiplier sa vitesse **dy** par **-1**, de cette manière, la balle ira dans le sens opposé. Ajoute le code suivant à la suite :

```
dy = dy * -1;
score += 1;
```

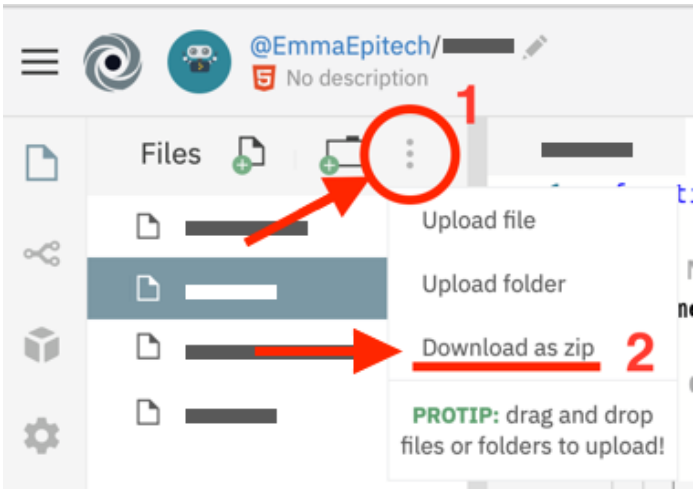
Désormais dès qu'une collision aura lieu, la balle changera de direction et ira dans le sens opposé.

*Appuie sur **RUN** et essaye de faire rebondir la balle sur une brique !*

Bonne chance à toi !

5. Sauvegarde ton programme

Tu peux enregistrer ton jeu sur ton ordinateur pour le modifier ou y jouer plus tard. Pour cela, clique sur “**Download as zip**”.



6. Quelques liens utiles

Pour apprendre le JavaScript :

➔ https://www.w3schools.com/js

Pour refaire l’exercice :

➔ https://jsfiddle.net/org_emma/cdeua1vo/8/

Pour voir nos autres exercices :

➔ https://repl.it/@EmmaEpitech

Pour plus d’informations sur nos activités :

➔ https://www.e-mma.org
➔ https://www.e-mma.org/codezchezvous