*Dossier réalisé par BERTIN Thomas*

# ***Dossier-Projet ISN :***
# ***Casse-Brique***

*Équipe du projet composé de :*
*FABRIGOULE Théo*
*MENGUY-TEMPLIER Loïc*
*BERTIN Thomas*

# ***Sommaire***

# ***Introduction***

## ➔ *Introduction au sujet*

Nous avons choisi comme projet d'ISN de créer un jeu et plus particulièrement un Casse-Brique. En effet, nous avons eu l'occasion pendant ce début d'année scolaire de nous initier au début de programmation d'un Casse-Brique. Cette expérience nous a particulièrement plu et nous avons donc choisi ce sujet comme projet.
De plus, apprendre comment est programmé un jeu m'a toujours intrigué contrairement à la création d'un site par exemple. Ce fut donc une aubaine pour moi.

# ***Cahier des charges***

## ➔ *Le jeu*

**Nom du jeu :** Casse-Brique

**Jouabilité :** 1 joueur, se joue à l'aide de la souris

**But du Jeu :** Réussir à atteindre le score maximal en cassant toutes les briques du niveau

**Principe du jeu :** Notre projet se comporte comme tout autre Casse-Brique.
Le joueur contrôle une barre, appelée « raquette », en bas de l'écran lui permettant de renvoyer une balle en déplaçant cette dernière horizontalement,vers la droite ou vers la gauche. Cette balle peut rebondir sur tous les bords de la fenêtre du jeu à l'exception de celle du bas puisqu'en effet c'est au joueur de la renvoyer grâce à la barre. La balle peut aussi rebondir sur les briques, ainsi ces dernières disparaissent à chaque contact avec la balle et laissent donc le passage à celle-ci.

## ➔ *Fonctionnalités*

- Système de niveaux qui permettrait qu'à chaque fois que la partie est gagnée, de pouvoir la recommencer avec un niveau supérieur où les briques seraient positionnées différemment et où la difficulté serait accrue avec une accélération de la vitesse de la balle.
- Système de score où à chaque fois qu'une brique se trouve être cassée par contact avec la balle ferait monter le score de un point.

- Système afin de pouvoir gagner le jeu qui apparaît si le score total est atteint , soit, si toutes les briques parvinrent à être cassées.
- Système de vie qui permettrait à l'utilisateur d'avoir en sa possession trois chances. Il perd une vie lorsqu'il n'arrive pas à renvoyer la balle à l'aide de la barre et donc que la balle passerait au travers du bord bas de la fenêtre de jeu. Si le joueur perd ses trois vies, il perd la partie et doit tout recommencer.
- Système afin de recommencer le niveau une fois que celui-ci est perdu ou gagné sans avoir à quitter complètement le jeu puis à le rouvrir.

# ***Conception***

## ➔ *Langage de programmation*

Afin de créer notre projet, nous avons du choisir un langage de programmation et avons décidé d'utiliser le langage Java à l'aide de Processing.
En effet, Processing est une bibliothèque Java et un environnement de développement libre nous permettant de créer divers programmes notamment des jeux directement en langage Java. Ainsi nous avons aussi la possibilité avec Processing de pouvoir lire nos programmes créés sous forme d'applications indépendantes pour Windows par exemple.
De même, nous pouvons vendre notre jeu mais dans ce cas présent il nous faut alors utiliser des images ou bien des sons libre de droit. Malheureusement, notre jeu ne pourrait donc pas être commercialisé à cause de différents « copyright » notamment avec celui présent sur notre image de fond.

## ➔ *Algorithmes principaux*

- ***Création des briques***

Pour commencer, il nous a fallu set up les briques comme si elles représentaient des lignes et des colonnes grâce à une variable liste transformé en tableau

```
int[][] briques = new int [5][15] ; // Liste de brique [Nombre de lignes][Nombre de colonnes]
```

Ainsi, nous avons pu set up les briques ,dans le void approprié ,comme existantes mais ne sont toujours pas affichées sur le jeu et ne possèdent pas encore de « hitbox » qui représenterait la zone de touche d'une brique.

```
void setup() {
    for (int i = 0; i < 5; i = i + 1) {  //  Set up des lignes de briques
      for (int j = 0; j < 15 ; j = j + 1) { // Set up des colonnes de briques
          briques[i][j] = 1 ; // Set up des briques comme existantes mais non placées
```

Puis nous allons réitérer la même sorte de boucle for afin de modéliser les briques dans le void draw. Ici nous avons décidé de créer 3 lignes de briques composées de 14 briques chacune. Ainsi, nous avons choisi les valeurs par lesquelles seraient multipliées i et j, correspondant respectivement aux numéro de ligne et de colonne, afin que les briques soient correctement placées et nous avons aussi choisi quelles lignes apparaîtraient dans la fenêtre de jeu en choisissant la valeur 2 à i. En effet, ici les 3 lignes de briques qui s'affichent sont les lignes numéro 2 , 3 et 4 correspondant aux 3 dernières du tableau.

```
for (int i = 2; i < 5; i = i + 1) { // Création de lignes de briques (numérotées de 2 à 4)
    for (int j = 1; j < 15; j = j + 1) { // Création de colonnes de briques (numérotées de 1 à 14)
        fill (1000,1000,1000) ; // Couleur des briques (blanc)
        if (briques[i][j] == 1) {
          rect (j*50, i*60, 45, 30) ; // Modélisation des briques selon les coordonnées j et i
```

- ***Démolition des briques***

Le but du jeu étant de détruire la totalité des briques présentes sur la fenêtre de jeu, il nous a fallu nous pencher sur la question. Pour ce faire, nous avons choisi de procéder par colonnes

```
      if (x>=50 && x<=95){
        if (briques[4][1] == 1){     // Si il y a une brique, sur la 3éme ligne et sur la 1ére colonne
        if (y+25<=300 && y+25>=260){
          briques[4][1] = 0; // Disparition de la brique
          speedy = -speedy; // La balle rebondit
          score = score + 1; // Augmentation du score
   }
       }

       if (briques[3][1] == 1){
   if (y+25<=230 && y+25>=200){
     briques[3][1] = 0;
     speedy = -speedy;
     score = score + 1 ;
   }
       }
        if (briques[2][1] == 1){
   if (y+25<=170 && y+25>=140){
     briques[2][1] = 0;
     speedy = -speedy;
     score = score + 1 ;
   }
 }
       }

// On répète ensuite l'opération précédente mais sur une nouvelle colonne
                    ellipse (x, y, 25, 25) ; // Création de la balle
```

En effet x et y ,correspondant respectivement à l'abscisse et à l'ordonnée de la balle, l'algorithme vérifie si les coordonnées de cette dernière correspond à une brique existante dans une des colonnes.Si c'est le cas, alors la brique en question se casse et la balle rebondit, ajoutant un point au score total.

- ***Perdre dans le jeu***

Tout d'abord, il nous a fallu créer une variable vie perdant un point à chaque fois que les conditions de défaite sont atteintes.

```
if (y>=575) { // Si la balle dépasse 575 (en abscisse)
  lifes = lifes - 1 ; // Le nombre de vies diminue
  x = int(random(25,775)); // Position réapparition balle sur X
  y=350 ; // Position réapparition balle sur Y
```

Ainsi, la balle réapparaît, de la même manière qu'elle apparaît, c'est-à-dire, de façon aléatoire sur l'axe des abscisses mais reste la même sur l'axe des ordonnées.
Une fois toutes les vies perdues, un écran de défaite apparaît enlevant alors toutes les briques encore présentes ,de même que la balle, et affichant « Game Over » et la chance de rejouer en cliquant sur une image.

```
if (lifes==0) { // Quand on a perdu toutes les vies
   text ("GAME OVER", 340, 220) ; // Afficher "GAME OVER"
   lifes = 0 ; // Reset des vies
   x=-1000 ; // La balle disparaît de l'écran
   y=-1000 ; // IDEM
   score = 0; // Reset du score
   restart = loadImage("restart21.jpg");
   for (int i = 2; i < 5; i = i + 1) { // Création de lignes de briques (numérotées de 2 à 4)
          for (int j = 1; j < 15; j = j + 1) { // Création de colonnes de briques (numérotées de 1 à 14)
            briques[i][j] = 0; // Reset des briques
          }
   }
   image(restart,315, 320);
```

- ***Gagner dans le jeu***

La variable score a été crée et augmente de un point à chaque briques touchées comme vu précédemment. Le score de 42 représente le score une fois que toutes les briques ont été cassées

```
if (score==42){ // Si on a cassé toutes les briques
       text ("Congratulations, YOU WIN", 285, 220) ; // Afficher "Congratulations, YOU WIN"
     x=-1000 ; // La balle disparaît de l'écran
     y=-1000 ; // IDEM
     nextlvl = loadImage("nextlvl.png");
     nextlvl.resize(100, 125) ; // Redimensionnement de l'image
     image(nextlvl,360, 320); // Bouton pour passer au niveau suivant
```

Un écran de victoire apparaît alors lorsque le score atteint 42 faisant disparaître la balle et affichant une phrase de victoire ainsi que l'opportunité de passer au niveau supérieur en cliquant sur une image.

- ***Boutons recommencer/prochain niveau***

Il s'est posé un autre problème lors de notre programmation. En effet, lorsque l'on perdait ou lorsque l'on gagnait, nous avons voulu créer un système de bouton où il suffirait de cliquer afin de recommencer ou de passer au niveau supérieur.
Ainsi, il nous a fallu créé un nouveau void dans lequel nous avons définit une zone où le clic de la souris serait effectif.

```
void mousePressed(){

    if ((mouseX>330) && (mouseX<507) && (mouseY>320) && (mouseY<400)) // Zone dans laquelle le clic est effectif
      if(clic==false) {
        clic=true;
      }
  }
```

Le clic de la souris étant définit comme faux de base, alors la condition ci-dessus peut fonctionner. `boolean clic=false;`
Puis, il nous a suffi d'incrémenter les images ,servant de bouton, aux coordonnées où le clic de la souris est effectif lorsque nous sommes amenés à perdre ou à gagner. Ainsi, il nous a fallu redimensionner une image afin qu'elle rentre dans la zone de clic.

Comme par exemple :

```
nextlvl = loadImage("nextlvl.png");
nextlvl.resize(100, 125) ; // Redimensionnement de l'image
image(nextlvl,360, 320); // Bouton pour passer au niveau suivant
```

Par conséquent, si nous ajoutons la condition suivante à la suite de l'algorithme qui a pour but d'afficher l'écran de défaite alors, si l'utilisateur clique sur le bouton « restart », les briques réapparaissent de la même manière qu'auparavant et la balle réapparaît encore de façon aléatoire sur l'axe des abscisses. Les vies sont alors remises à 3.

```
if (clic==true){
    for (int i = 2; i < 5; i = i + 1) { // Création de lignes de briques (numérotées de 2 à 4)
      for (int j = 1; j < 15; j = j + 1) { // Création de colonnes de briques (numérotées de 1 à 14)
        fill (1000,1000,1000) ; // Couleur des briques (blanc)
        briques[i][j] = 1; // Reset des briques
    if (briques[i][j] == 1) {
      rect (j*50, i*60, 45, 30) ; // Modélisation des briques
    }
 }
 }
  x = int(random(25,775)); // Position réapparition balle sur X
  y=350 ; // Position réapparition balle sur Y
  lifes = lifes+3 ; // Remise des vies
  clic = false;
 }
```

De même, si nous ajoutons la condition suivante à la suite de l'algorithme qui a pour but d'afficher l'écran de victoire alors, si l'utilisateur clique sur le bouton « nextlvl »,les briques réapparaissent de la même manière encore, pareil en ce qui concerne la balle et le score est remis à 0. La variable n représentant les niveaux augmente alors de 1.

```
if (clic==true){ // Si on clique sur le bouton
n = n + 1 ;
    for (int i = 2; i < 5; i = i + 1) { // Création de lignes de briques (numérotées de 2 à 4)
      for (int j = 1; j < 15; j = j + 1) { // Création de colonnes de briques (numérotées de 1 à 14)
        fill (1000,1000,1000) ; // Couleur des briques (blanc)
        briques[i][j] = 1; // Reset des briques
    if (briques[i][j] == 1) {
      rect (j*50, i*60, 45, 30) ; // Modélisation des briques
    }
 }
}
  x = int(random(25,775)); // Position réapparition balle sur X
  y=350 ; // Position réapparition balle sur Y
  score = 0; // Reset du score
  clic = false;
}
```

## ➔ *Données secondaires*

- ***Mouvement de la balle***

Afin de rendre mobile notre balle, il nous a fallu ajouter une vitesse à cette dernière

```
int speedx = 5 ; // Vitesse initiale sur l'axe x
int speedy = 5 ; // Vitesse initiale sur l'axe y


 x = x+speedx ; // Mouvement de la balle sur l'axe x
 y = y+speedy ; // Mouvement de la balle sur l'axe y
```

En effet, le chiffre associé à la variable speedx et speedy correspond à la vitesse de la balle selon l'axe des abscisses et l'axe des ordonnées puisqu'on associe l'addition de la variable speedx avec x à la variable x et on associe l'addition speedy avec y à la variable y. Tous deux représentant les coordonnées de la balle alors cette dernière change tout le temps de coordonnées et est donc mobile.

- ***Apparition de la balle***

Nous avons voulu rendre l'apparition de la balle de sorte à ce qu'elle soit aléatoire.

```
int x = int(random(25,775)); // Position initiale de la balle sur l'axe x
int y = 350 ; // Position initiale de la balle sur l'axe y
```

Nous avons donc modifié la variable x de sorte à ce qu'elle soit aléatoire et puisse surprendre le joueur.

- ***Les niveaux***

Premièrement, nous avons créé la possibilité de passer au niveau supérieur et nous avons alors réfléchi sur ce que cela pourrait apporter à l'expérience de jeu. Nous avons donc cherché un moyen d'augmenter la vitesse de la balle à chaque fois que l'on passe au niveau supérieur. Deux listes ont alors été créées.

```
int[] niveau = {1,2,3,4,5,6,7,8,9,10}; // Liste des niveaux
int[] vitesse = {5,7,9,10,11,12,13,14,15,16}; // Liste des différentes vitesses
```

Ces listes nous permettent donc, grâce à l'algorithme ci-dessous, de faire augmenter la vitesse de la balle à chaque montée de niveau puisque la variable n augmente de 1 à chaque fois que le joueur presse le bouton « nextlvl ».

```
if (niveau[n] == n){
   speedx = vitesse[n];
   speedy = vitesse[n] ;
 }
```

```
int n = 1 ; // Niveau initial
```

- ***Affichage***

Ici, nous avons créé des indications sur nos vies, notre score et notre niveau actuel et sont placées dans la fenêtre de jeu selon des coordonnées précises. La taille du texte est préalablement choisi

```
textSize (20); // On définit la taille du texte
text ("LEVEL " + n, 360, 30) ; // Affichage du niveau (en fonction de la variable "n")
text ("lifes = " + lifes, 20, 70) ; // Affichage du nombre de vies
text ("score = " + score, 20, 30) ; // Affichage du score (en fonction de la variable "score")
```

- ***Déplacement de la barre***

Dans le but de pouvoir jouer, l'utilisateur doit pouvoir prendre le contrôle de la barre afin de pouvoir renvoyer la balle

```
rect(mouseX, 575, 150, 20) ; // Le déplacement de la barre dépend de celui de la souris
```

Pour se faire, nous avons conçu un rectangle dirigeable seulement sur l'axe des abscisses par le biais de la souris. Le rectangle n'étant pas conçu pour se déplacer sur l'axe des ordonnées, sa coordonnée est alors fixe.

```
if (y>=560) {
if (x>=mouseX && x<=mouseX+150) { // Si la balle heurte la barre
  speedy = -speedy ; // Alors la balle change de direction
}
```

Ainsi, si la balle touche la barre dirigée par le joueur alors cette dernière se voit rebondir.

- ***Bords de la fenêtre***

Dans un Casse-Brique, la balle doit rebondir sur les bords de la fenêtre de jeu à l'exception de celle du bas. Nous avons donc mis en place l'algorithme ci-dessous afin que cette dernière change de direction lorsque les coordonnées de la balle se trouve être aux extrémités de l'écran de jeu. `size (800, 600) ; // Taille de la fenêtre`

```
 if (x>=800) { // Si la balle heurte le côté droit
   speedx = -speedx ; // Alors la balle change de direction
 }
 if (y<=0) { // Si la balle heurte le haut
   speedy = -speedy ; // Alors la balle change de direction
 }
if (x<=0) {  // Si la balle heurte le côté gauche
   speedx = -speedx ; // Alors la balle change de direction
 }
```

# ***Conclusion***

## ➔ *Bilan*

Au final, notre cahier des charges est quasiment respecté. En effet, toutes les fonctionnalités du cahier des charges sont présentes dans le jeu à l'exception du système qui fait en sorte que lorsque l'utilisateur passe au niveau suivant, les briques se repositionnent de façon aléatoire ou du moins repositionnées autrement qu'auparavant.

## ➔ *Améliorations possibles*

Notre Casse-Brique est fini mais il reste tout de même quelques améliorations envisageables et donc des rectifications probables. En effet quelques ajouts me parcourent l'esprit.
Tout d'abord, la présence d'un « timer » ou bien en français d'un chrono. Il permettrait d'avoir un temps imparti pour finir le niveau ce qui aurait pour but d'ajouter de la difficulté.
Ensuite, la possibilité d'obtenir des bonus aléatoire lorsque le joueur parvint à casser un certains nombre de briques. Comme par exemple, un bonus qui ferait agrandir provisoirement la taille de la barre que contrôle le joueur ou bien un bonus qui ferait ralentir le temps et donc la balle.
A contrario, la présence de malus, ayant une probabilité plus faible de se produire, pourrait aussi être ajouté. Comme par exemple, un malus qui ferait donc accélérer le temps ou bien qui modifierait la taille de la barre en la rendant plus petite.