Jack la Trouille

version 1.0.1

DIVERSITY by EPITECH

I. Introduction

Halloween commence dans quelques heures seulement, il faut vite finir les derniers préparatifs avant que **Jack La Trouille** n'envoie ses hordes de zombitrouilles ! Gimgim va avoir besoin de bonbons explosifs et de barrières pour sauver les habitants du village (et les bonbons) !

Ils arrivent !!!

Le moment est venu, Gimgim va devoir affronter **Jack La Trouille**.

II. Consignes

- Reprenez le projet que vous avez réalisé la dernière fois.
- Lisez tout avant de commencer !
- Lors de ce projet, ce lien pourrait t'être grandement utile.
- Si tu bloques, rappelle-toi que tu es accompagné(e) ! Demande de l'aide à tes camarades ou à un Cobra (ceux-là ne mordent pas).
- Internet est un outil formidable pour découvrir le fonctionnement des choses, sers-t'en régulièrement !

Explication du code :

- Un ... dans le code signifie que tu dois le compléter par toi-même en utilisant les informations du sujet.
- Un // est un commentaire pour t'aider à comprendre. Ce qui se trouve après sur la ligne est ignoré par le programme.

III. 1, 2, 3, citrouille !

La grande bataille arrive, et le nombre de ressources du village est limité. Pour aider à gérer les stocks de bonbons explosifs et de barils, il faut faire un compteur de zombitrouilles, qui comptabilisera le nombre d'ennemis que Gimgim vaincra dans son arène.

Tu vas donc, dans le coin supérieur gauche de ta fenêtre, afficher un score. Il te suffit d'augmenter le score à chaque fois que Gimgim terrasse un zombitrouille. Pour l'affichage de celui-ci, il faut faire cela :

- Renomme ta classe `Orb` en `Zombitrouille`. Tu peux aussi modifier les autres noms de variables si tu penses que c'est utile. (Il faut le faire à travers tout ton code !)
- Afficher une image de fond sur laquelle sera affiché le score (`score_rect.png`). Tu peux placer l'image en haut à gauche de ta fenêtre.
- Afficher du texte représentant le score par-dessus l'image de fond, en utilisant la police de caractères fournie avec le sujet. La taille de la police doit être de `50`. Pour cela, il faut utiliser la fonction `createFont("...", ...)`.
- Augmenter le score de 1 pour chaque zombitrouille terrassé.

À ajouter au début du fichier `JackLaTrouille.pde` :

```
PImage score_background; //On prépare l'image de fond pour le score
PFont score_font; //On prépare la police d'écriture du score
int score;
```

À ajouter dans la fonction `setup()` :

```
score_background = loadImage("...");

//On charge la police du score et on lui donne une taille de 50
score_font = createFont("...", ...);

//On utilise notre police d'écriture pour tout notre jeu
textFont(score_font);
score = ...; //On met notre score par défaut à zéro
```

À ajouter dans la fonction `draw()` :

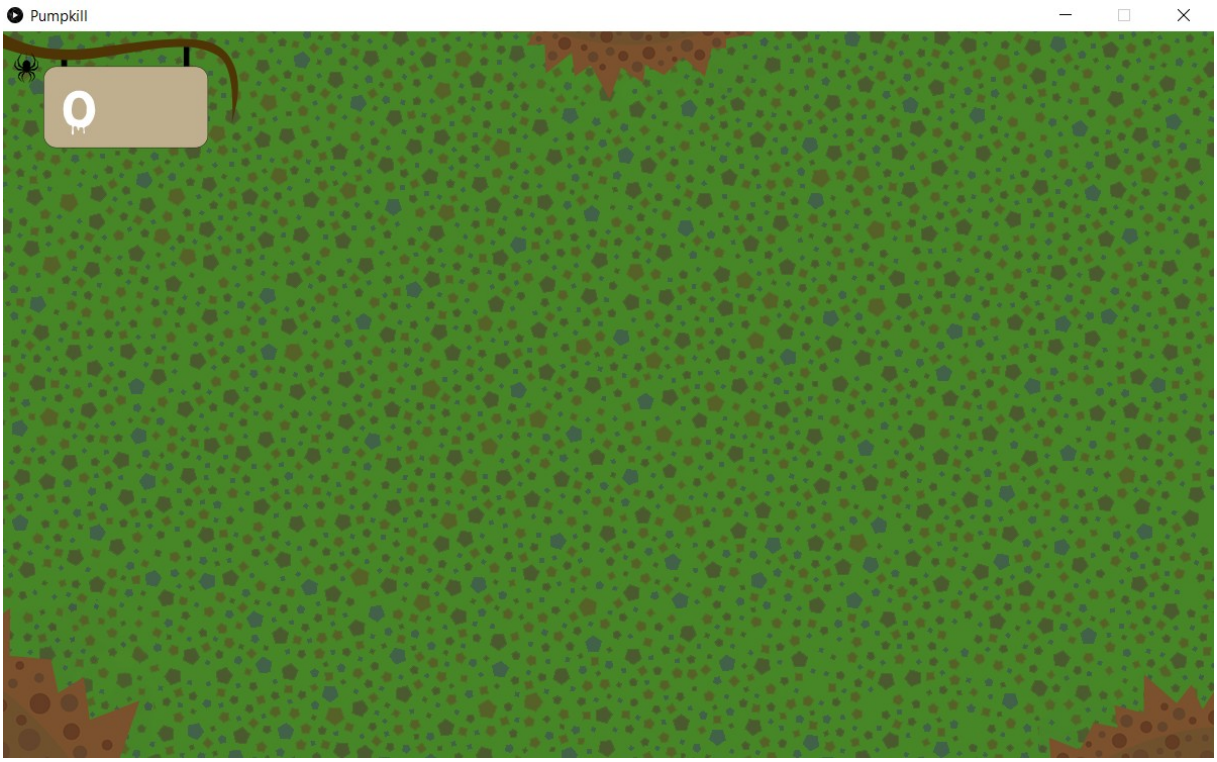
```
image(score_background, ..., ...);
//On affiche un texte a la position x = 60 et y = 95,
//correspondant à notre score
text(score, ..., ...);
```

IV. L’entraînement c’est fini, on se prépare pour la bataille !

Gimgim s’est assez entrainé dans l’arène avec des orbes la veille. À présent, l’heure du combat est venue. Il va maintenant combattre de vrais zombitrouilles du vilain Jack La Trouille, en chair et en graine !

a. À la recherche du terrain idéal

Tout d’abord, il faut préparer le sol de l’arène pour accueillir la bataille. Remplace `training_background.png` par `battle_background.png` dans ton code !



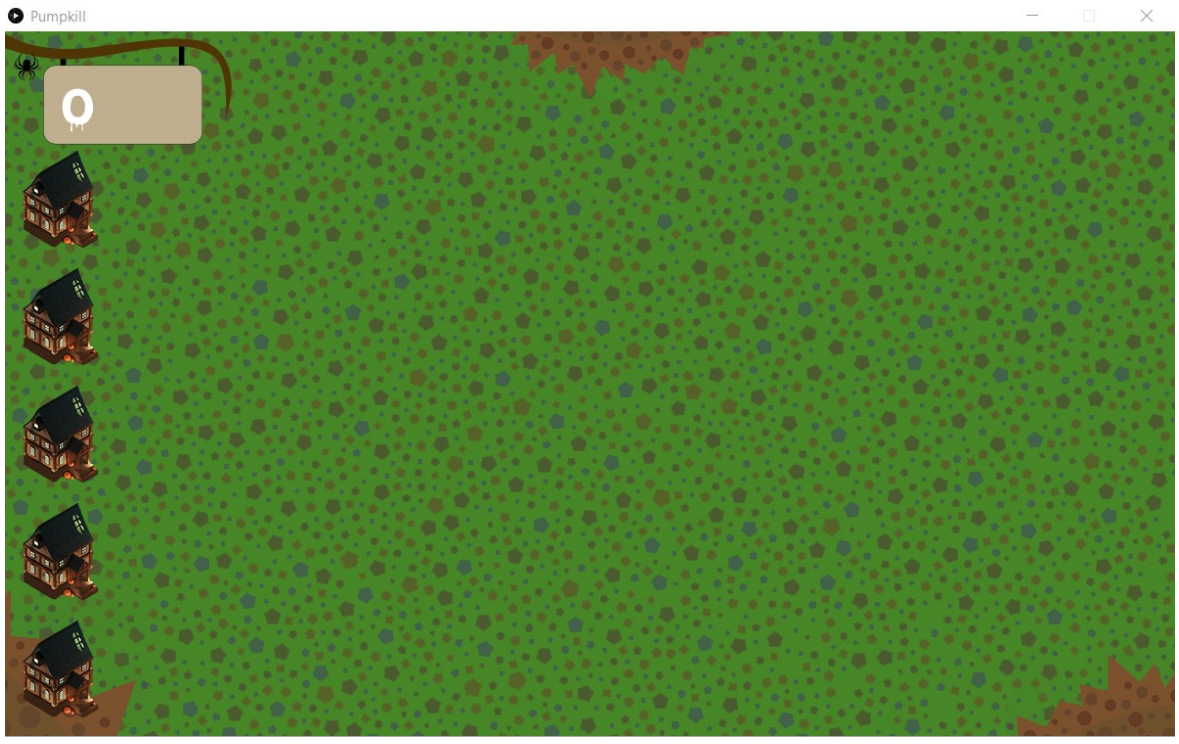
Et voilà, un sol tout neuf !

b. À la rescousse du village

Les zombitrouilles sont en chemin pour attaquer les habitations des villageois. Il faut donc les attirer avec un leurre vers l'arène, c'est-à-dire en y plaçant des maisons pleines des bonbons dont ils raffolent ! Pour cela, tu peux afficher cinq maisons tout à gauche de la fenêtre.

À partir d'ici, on va considérer que notre fenêtre est composée de cases carrées, faisant 120x120 pixels. Lors du premier jour, il y avait 5 « lignes » où faire apparaître les orbes, et bien ici ces lignes deviennent découpées en cases, comme dans un quadrillage.

Tu l'auras donc deviné, il faut afficher une maison par ligne dans le quadrillage. Pour ce faire, tu dois charger l'image qu'une seule fois et afficher celle-ci à 5 emplacements différents. Inspire-toi de la partie III. pour afficher les maisons !



Mais qu'elles sont belles, ces maisons !

c. Bienvenue à Zombitrouille

Bien, les zombitrouilles ont l'air de venir ! Remplace les cercles dans la classe `Zombitrouille` par une image de zombitrouille.

Attention : il faudra peut-être modifier la taille de l'image et la position !

Voici les étapes que tu peux suivre :

- Trouver les variables qui servent à afficher un cercle, et effacer celles concernant la couleur (en effet, elle ne servira plus)
- Créer une variable pour contenir l'image de zombitrouille, et y charger le fichier zombitrouille.png (inspire-toi de ce que tu as pu faire pour les parties précédentes)
- Remplace le contenue de la méthode `display()` dans ta classe `Zombitrouille` par l'affichage de ton image

Il ne reste plus qu'à les détruire !

V. Les bonbons : une solution temporaire...

a. Apparition et...

Maintenant qu'il y a des maisons à protéger et de véritables ennemis qui vont se jeter sur Gimgim, il va devoir créer des bonbons « spéciaux ». En effet, ces bonbons seront placés devant les maisons.

Ainsi, tu vas devoir afficher un bonbon sur chaque ligne, dans la « case » à droite des maisons ! Inspire-toi de ce que tu as fais pour afficher les maisons, c'est très similaire.

Rappelle-toi, on fait comme s'il y avait des cases de 120x120 pixels, donc si les maisons sont placées en X = 0, alors une case plus loin, X est égal à ?

Il va également te falloir un tableau de booléens (si tu ne sais pas ce que c'est, demande au Cobra le plus proche), que tu mettras au début du fichier :

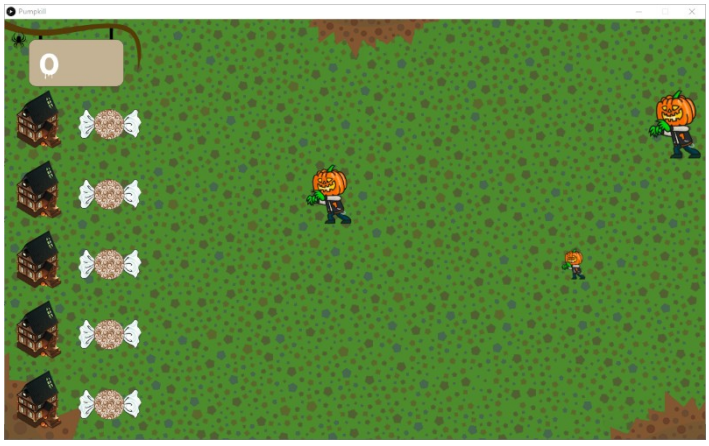
```
Boolean[] candy_is_here; //On prépare un tableau de boolean pour savoir si nos bonbons ont été mangés
```

Il te servira à savoir s'il faut afficher chaque bonbon ou non. Au départ, toutes les cases du tableau seront à true. Pour cela, ajoute ces lignes dans setup() :

```
for(int y = 0; y < 5; y++)
{
  candy_is_here[y] = true;
}
```

Enfin, il va falloir créer une nouvelle fonction pour gérer l'affichage des bonbons, et que tu appelleras dans draw() :

```
void drawCandy()
{
  for (int y = 0; y < 5; y++) {
    if (candy_is_here[y] == true) {
      image(candy_image, ..., (... + 1) * 120);
    }
  }
}
```



Ils ont l'air appétissant ces bonbons !

b. Destruction !

Les bonbons que tu viens de placer ne sont pas là juste pour décorer ! Ils ont un pouvoir assez intéressant et utile : si un zombitrouille marche sur un bonbon, alors *BOUM*, le bonbon explosera et emportera tous les zombitrouilles présents sur le terrain !

Il va falloir que tu détectes la collision entre un bonbon et un zombitrouille, puis si la collision a lieu, que tu fasses disparaître le bonbon et détruises tous les zombitrouilles !

Pour ce faire, tu dois d'abord ajouter une méthode dans ta classe Zombitrouille :

- `int getLineNumber()` qui renvoie tout simplement la variable `lineNumber`.

Maintenant que tu peux récupérer la ligne de chaque zombitrouille, ajoute une condition dans la fonction `is_alive()`, où tu vérifies si le zombitrouille atteint la position (en X) où un bonbon pourrait se trouver (pense aux valeurs des cases de notre quadrillage imaginaire). Encore une fois, inspire-toi du code déjà présent. Lorsque la condition est remplie, il faudra renvoyer `3`.

À présent, il faut vérifier s'il y a bien une citrouille à cet endroit-là ! Tu vas alors aller dans la fonction `draw()`, dans le fichier principal, et y ajouter une condition `else if` qui vérifiera si `is_alive()` a renvoyé la valeur `3`. Si c'est le cas, alors tu vérifies s'il y a bien un bonbon présent sur la ligne de la citrouille:

```
} //l'accolade fermant le if déjà présent avant
else if (hord.get(i).is_alive() == ...) {
  if (candy_is_here[hord.get(i).getLineNumber()] == true) {
    //on arrête d'afficher le bonbon
    candy_is_here[hord.get(i).getLineNumber()] = ...;
        hord.clear(); //on efface tous les zombitrouilles
  }
  else {
    hord.get(i).move();
    hord.get(i).display();
  }
} else //la else qui était déjà présent avant
```

Attention : Ici, `hord` désigne notre liste de zombitrouilles (« hord » en anglais signifie « horde » en français, on a ici une horde de zombitrouilles). Tu l'as sûrement nommée différemment, pense à modifier le code fourni pour qu'il fonctionne chez toi !

VI. Jack La Trouille entre en scène, les citrouilles se déchaînent !

Pendant tout ce temps, Jack La Trouille préparait des plans pour augmenter le nombre de ses zombitrouilles. Malheureusement pour Gimgim, il a décidé de mettre son plan en action et il va lui donner du fil à retordre !

a. De plus en plus de zombitrouilles...

Au fur et à mesure que le temps passe, il y a de plus en plus de zombitrouilles qui arrivent dans l'arène !

Tu peux donc augmenter le nombre de zombitrouilles apparaissant en X secondes, et ce à chaque minute qui s'écoule. Rappelle-toi de ce que tu avais fait avec la fonction `millis()` ! Tu peux modifier la fonction `spawn()` pour augmenter la vitesse d' apparition des zombitrouilles !

b. ... et qui vont de plus en plus vite !

Jack La Trouille a trouvé un moyen de rendre ses zombitrouilles de plus en plus agiles ! Plus le temps passe, plus la vitesse des zombitrouilles augmente.

Modifie la vitesse des zombitrouille dans la classe. Tu peux t'inspirer de la fonction `spawn()`, du fichier principal, et reproduire son fonctionnement mais en modifiant la vitesse des zombitrouilles.

Attention : La vitesse, ça augmente plus vite que ce qu'on a tendance à croire !

VII. La fin de la bataille

Si un zombitrouille atteint une maison, c'est que Gimgim est tombé au combat. Quel triste sort pour le village ! Le Maléfique Jack La trouille a réussi à s'emparer de tous les bonbons, et le village est dévasté.

Ajoute un écran de Game Over. Pour ce faire, il faut d'abord créer une nouvelle image. Tu connais la chanson, maintenant : tu crées une PImage nommée game_over_screen en haut du fichier, tu y charges l'image game_over.png dans la fonction setup().

Pour l'afficher, il va falloir faire quelque chose d'un peu particulier : crée Boolean game_over, donne-lui la valeur false dans setup(). Ensuite, place un if tout autour du contenu de la fonction draw(), comme ceci:

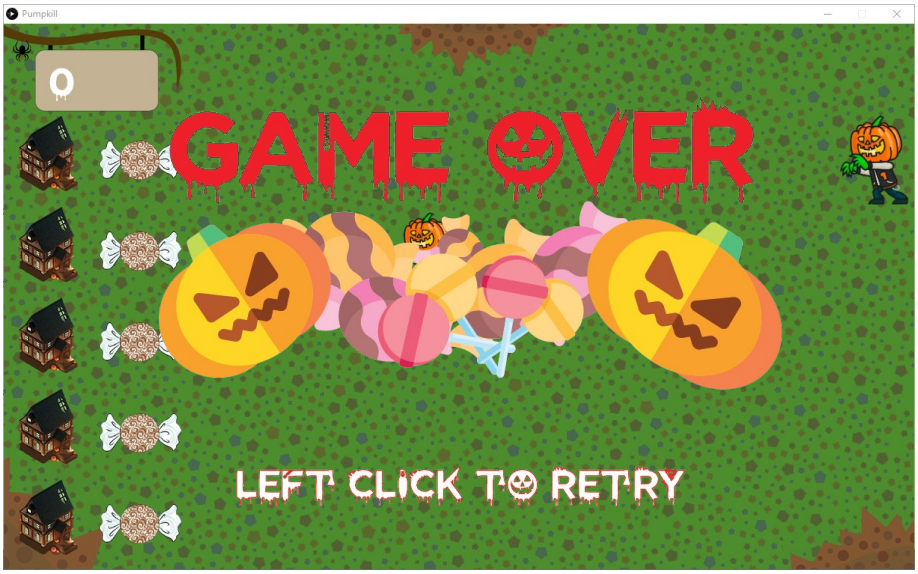
```
if (game_over == false) {
  //tout le code déjà contenu dans draw()
}
```

On va aussi ajouter un autre else if dans draw(), en dessous de celui qu'on a ajouté juste avant :

```
else if (hord.get(i).is_alive() == 1){
      game_over = true;
      image(game_over_screen, 0, 0); //on affiche le game over
}
```

Enfin, on ajoute dans la fonction mousePressed() :

```
if (game_over == true){
    setup(); //on réinitialise le jeu
}
```



Oh non, c'est perdu !

VIII. Pour aller plus loin (partie optionnelle) : Levez les boucliers !

a. Remettre des bonbons explosifs devant les maisons, simple mais efficace

La première chose que l'on peut faire, c'est remettre des bonbons devant les maisons lorsqu'il n'y en a plus. C'est simple, dit comme ça, mais les zombitrouilles ne sont pas très malins : ils s'arrêtent au premier obstacle qu'ils rencontrent !

Pour ce faire, essaie de détecter si le joueur utilise le clic droit ou le clic gauche. A partir de là, tu peux dire que si le joueur appuie sur le clic droit et que la souris est sur l'une des dernières cases, cela rajoute un bonbon explosif s'il n'y en a pas déjà un. Utilise les points du score pour pouvoir replacer des bonbons sur la carte de façon automatique, il coute 10 points et attention tu ne peux pas arriver à un score négatif.

Tu auras besoin d'utiliser la variable intégrée dans Processing, `mouseButton`, et tu devras implémenter les fonctions suivantes :

- `void buyCandy()` cette fonction servira à faire créer les bonbons sur le plateau de jeux

b. Des barils anti-cucurbitacées-mort-vivants de qualité !

Il ne manque plus qu'un élément pour que le système de défense puisse faire battre en retraite les zombitrouilles du malveillant Jack La Trouille : les barils explosifs. **Ils permettent de bloquer cinq zombitrouilles**, c'est super solide !

Pour information, le baril coute 5 points de score et les bonbons coutent 10 points !

Il faut donc pouvoir placer les barils, avec la souris. Ensuite, il faut les afficher. Enfin, il faut que lorsqu'un zombitrouille rencontre le baril, il meure et qu'un point de vie soit retiré au baril. Si le baril arrive à 0 point de vie, il est détruit.

Pour créer les barils tu auras besoin d'implémenter la classe ainsi que les fonctions suivantes :

- `class Barrel` cette classe vas regrouper les fonctions suivantes :
  - `Barrel(int x, int y)` cette fonction défini le constructeur de la class
  - `void display()` cette fonction vas devoir afficher l'image du baril
  - `int died()` cette fonction sert à vérifier si le baril n'est pas mort
  - `int touch(int x, int y)` cette fonction sert à vérifier si une zombietrouille a touché le baril

- `void checkBarrel()` cette fonction sert à vérifier si les barils ont été touchés en comparant leur position avec celle de la horde
- `void drawBarrel()` cette fonction sert à afficher le baril , elle récupère l'image via la fonction display de la class
- `void buyBarrel()` cette fonction sert à placer un baril ainsi qu' à gérer le cout de ce dernier

Avec ça les zombitrouilles auront du mal à passer

IX. Toujours plus loin, toujours plus haut, toujours plus fort !

Si tu es arrivé(e) ici, c'est que tu as réussi toutes les étapes du sujet. Tu peux en être fier(ère), ce n'était pas une mince affaire ! S'il reste du temps avant la fin de la session, voici des fonctionnalités que tu pourrais ajouter à ton programme. Si tu as une envie différente, fais-toi plaisir : ton programme, ton choix !

- Une fin heureuse ! (Choisis comment tu veux faire, plutôt temps écoulé ou nombre d'ennemis vaincus ?)
- Des super-zombitrouilles, avec plusieurs points de vie : il faudrait plusieurs clics pour les tuer, et les barils leur résisteraient moins bien.
- Ajouter des sons pour chaque action ou de la musique. Tu peux utiliser le format mp3 pour tes fichiers audios.
- Les zombitrouilles font tomber des objets en disparaissant, comme par exemple un multiplicateur de score sur une durée limitée, une réapparition instantanée de tous les bonbons...
- Tu peux tenter d'animer les zombitrouilles ! Une image, que tu pourras utiliser pour animer les zombitrouilles, t'est fournie. Tu peux te renseigner sur l'animation de *spritesheet*.