Jack la Trouille

version 1.1.1

DIVERSITY
BY EPITECH

I. Introduction

Le moment est venu, Gimgim va devoir affronter Jack La Trouille. Halloween commence dans quelques heures seulement, il faut vite finir les derniers préparatifs avant que l'ennemi n'envoie ses hordes de zombitrouilles à l'assaut ! Le gardien va avoir besoin de courage et de pièges efficaces pour sauver les habitants du village !

Ils arrivent !!!

II. Consignes

- Reprends le projet que tu as réalisé la dernière fois.
- Lis tout avant de commencer !
- Tu peux reprendre votre dépôt Github du jour précédent au nom : “**cc_jack_la_trouille**“.
- N’oublie pas de push régulièrement ton projet.
- Lors de ce projet, ce lien pourrait t’être grandement utile.
- Internet est un outil formidable pour découvrir le fonctionnement des choses, utilise-le régulièrement !
- Si tu bloques, n’hésite pas à demander de l’aide à tes camarades ou à un Cobra.

⚠ Attention

Le code des exemples est incomplet, tu devras rajouter/modifier quelques éléments pour que cela fonctionne.

Un `...` dans le code signifie que tu dois compléter le code par toi-même en utilisant les informations du sujet.

Un `//` est un commentaire pour t’aider à comprendre. Ce qui se trouve sur la même ligne est ignoré par le programme.

Information

Tu trouveras dans les ressources le fichier “Zombitrouille.pde“ que tu utiliseras à la place de ton “Orb.pde”.

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, il faut faire un compteur de zombitrouilles, qui affichera le nombre d'ennemis vaincus dans l'arène.

Tu vas donc créer un score dans le coin supérieur gauche de ta fenêtre. Pour cela :

* Récupère le fichier "Zombitrouille.pde" dans les resources du jour.
* Affiche une image de fond sur laquelle sera affiché le score "score_rect.png". Place l'image en haut à gauche de ta fenêtre.
* Affiche du texte représentant le score par-dessus l'image de fond, en utilisant la police de caractères "halloween_font.ttf" 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.

Au début du fichier "JackLaTrouille.pde", prépare l'affichage du score :

```
PImage score_background;
PFont score_font;
int score;
```

Dans la fonction **setup()**, charge l'image puis la police et donne lui une taille de **50**. Par défaut, le score sera à **0**.

```
score_background = loadImage("...");
score_font = createFont("...", ...);

textFont(score_font); // il faut utiliser la même police d'écriture pour tout le jeu
score = ...; // le score est par défaut à zéro
```

Dans la fonction **draw()**, il faut appeler les fonctions pour afficher l'arrière-plan du score et son texte. N'oublie pas d'ajuster la position pour obtenir l'effet attendu.

```
image(score_background, ..., ...);
text(score, ..., ...); // affiche le score en x et y
```

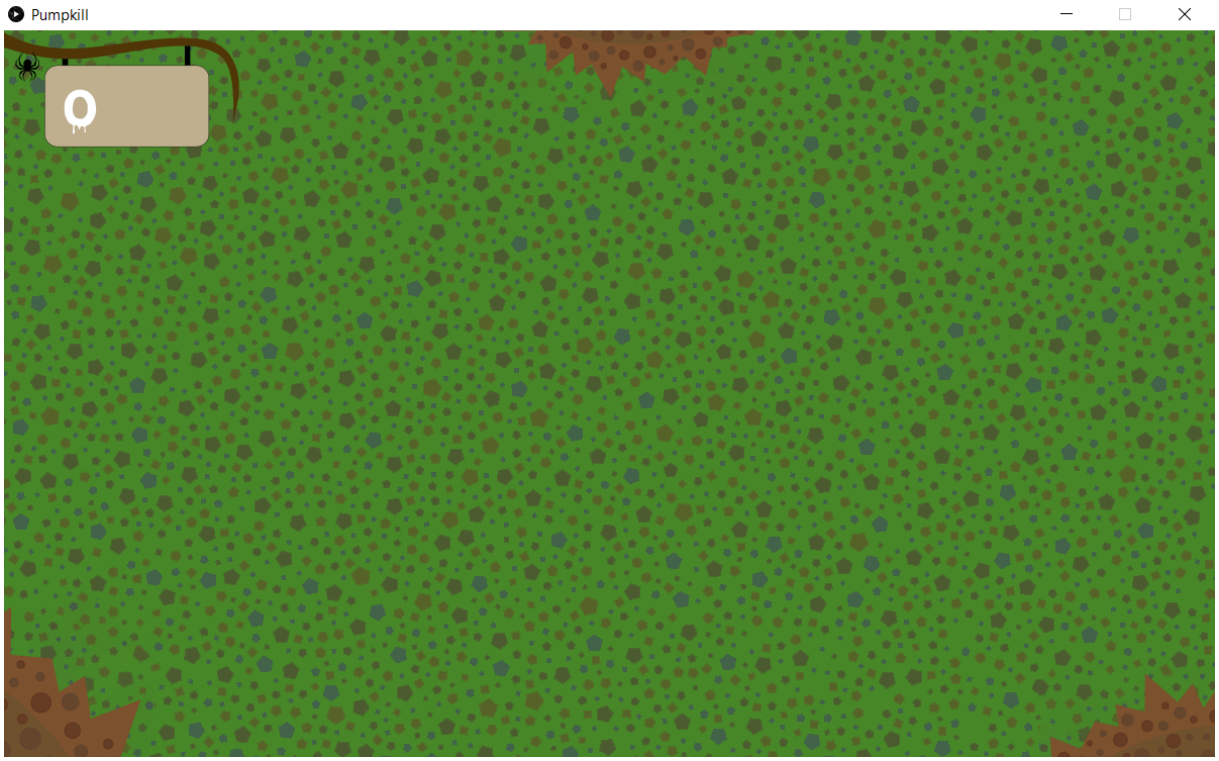
Et hop, un compteur, un !

IV. L’entraînement est fini, prépare-toi 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 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, en y plaçant des maisons pleines des bonbons dont ils raffolent ! Pour cela, tu peux afficher **5** maisons tout à gauche de la fenêtre.

À partir d'ici, on va considérer que notre fenêtre est un quadrillage de cases chacune de taille **120**x**120** pixels. Lors du premier jour, pour faire apparaître les orbes, tu prenais en compte seulement la position verticale, ici ce seront les positions horizontale et verticale.

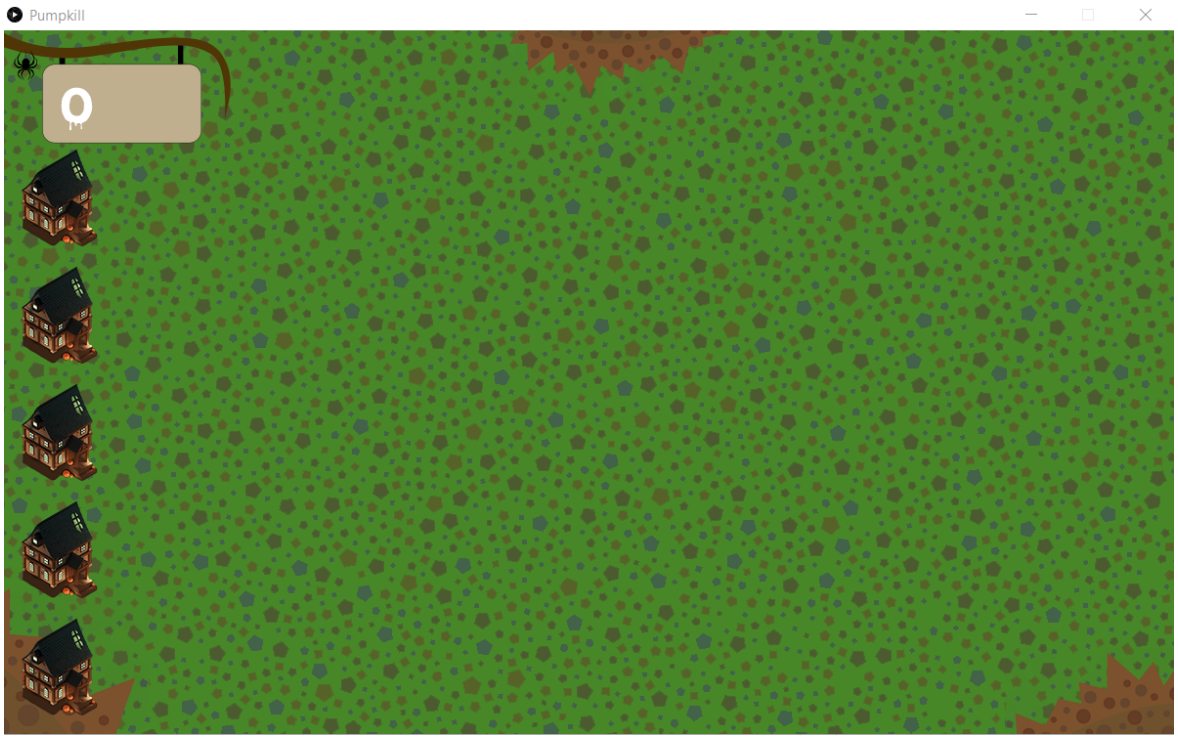
Tu l'auras donc deviné, il faut afficher une maison par ligne dans le quadrillage en respectant ces consignes :

* Charge l'image une seule fois

* Affiche l'image chargée à 5 emplacements différents sur ta fenêtre. Inspire-toi de la partie III pour afficher les maisons !

Information

On peut se repérer dans un quadrillage avec une coordonnée (ligne, colonne). Par exemple, la coordonnée (0,0) se situe en dans le coin le plus en haut à gauche de ta fenêtre !



Mais qu'elles sont belles, ces maisons !

c. Bienvenue à Zombiville !

Bien, les zombitrouilles commencent à apparaître ! Il va falloir leur donner une apparence digne d'Halloween.

> ⚠ Attention
>
> Tu devras peut-être modifier la taille de l'image et sa position !

Voici les étapes que tu peux appliquer dans "Zombitrouille.pde" :

* Trouver les variables qui servent à afficher la couleur orange d'un orbe et les effacer.
* Créer une variable qui contient l'image de zombitrouille, et y charger le fichier "zombitrouille.png".
* Remplacer le contenu de la méthode **display()** de 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 fait pour afficher les maisons, c'est très similaire.

> ⚠ Attention
>
> 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 que tu mettras au début du fichier « JackLaTrouille.pde » :

```
Boolean[] candy_is_here; // tu prépares ta variable pour savoir si les 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;
}
```

Ils ont l'air appétissant ces bonbons !

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

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

b. … Destruction !

Les bonbons que tu viens de placer sont des pièges ! Si un zombitrouille marche dessus, une explosion emportera tous les autres présents sur le terrain. Pour déclencher le mécanisme, il faut que tu détectes la collision entre un bonbon et un ennemi, puis que tu fasses disparaître la friandise et détruises tous les monstres ! Il va donc falloir te rendre dans la classe **Zombitrouille** pour :

* Créer une méthode **int get_line_number()** qui renvoie la variable **line_number**.
* Ajouter la condition ci-dessous dans la fonction **is_alive()**. Elle doit vérifier si le zombietrouille est à portée de piège.

Il est temps de compléter cette section :

```
if (position_x < ...)
{
 return 3;
}
```

Dans la fonction **draw()** du fichier "JackLaTrouille.pde", ajoute la condition **else if** suivante. Puis, complète la de manière à ce que si **is_alive()** renvoie la valeur **3**, tu vérifies s'il y a un piège présent sur la ligne du zombitrouille.

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

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

Gimgim commence à voir double, il y a de plus en plus de zombitrouilles qui arrivent dans l'arène ! Par ailleurs, Jack La Trouille a trouvé un moyen de rendre ses sbires bien plus agiles qu'au début. Plus le temps passe, plus la vitesse des monstres augmente.

Voici comment procéder pour intégrer cette fonctionnalité au jeu :

- Augmente le nombre d'ennemis apparaissant en un certain nombre de secondes et ce, à chaque minute qui s'écoule.
- Modifie la fonction **spawn()** pour augmenter la vitesse d'apparition des monstres !
- Modifie la vitesse des sbires de Jack la Trouille dans la classe **Zombitrouille**.

Information

Tu peux utiliser la fonction **millis()** pour t'aider à faire apparaître les zombitrouilles !

Ils ne sont pas venus au village pour se tourner les tiges !

VII. Bientôt la fin de la bataille

Si un zombitrouille atteint une maison, et que le maléfique Jack La trouille réussit à s'emparer de tous les bonbons, le village est alors défait. Il faut alors l'annoncer avec un écran de Game Over digne de ce nom !

Pour çela, tu dois d'abord importer une nouvelle image dans ton code. Pour l'afficher, il va falloir rajouter plusieurs éléments :

- Une variable de type **PImage** nommée **game_over_screen** en haut du fichier.
- L''image "game_over.png" doit être ajoutée à la variable dans la fonction **setup()**.
- Une autre variable **Boolean** au nom de **game_over** au début du fichier. Tu lui donneras la valeur « false » dans **setup()**.
- Une condition **if** tout autour du contenu de la fonction **draw()**, comme ceci :

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

- Un **else if** supplémentaire dans **draw()**, en dessous de celui que tu avais ajouté dans la partie VI :

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

- La condition suivante sera à placer dans la fonction **mousePressed()** :

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

Et voilà, le jeu a une vraie fin maintenant !

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

Et voilà, désormais, Gimgim a une nouvelle chance de se défendre de Jack La Trouille. Si tu veux aller plus loin, voici quelques idées d'améliorations :

* Des super-zombitrouilles, avec plusieurs points de vie : il faudrait plusieurs clics pour les vaincre, par exemple.

* Tu peux tenter d'animer les zombitrouilles ! Une image, que tu pourras utiliser pour animer les zombitrouilles, t'est fournie dans le dossier de ressources. Tu peux aussi te renseigner sur l'animation de *spritesheet*.

* Tu peux aussi continuer avec le bonus mystère page suivante...

Joyeux Halloween !

IX. Bonus : Levez les boucliers !

a. Remettre des pièges devant les maisons

Gimgim souhaite renforcer ses défenses face aux vagues d'ennemis. La première chose que tu peux faire, c'est remettre des bonbons explosifs devant les maisons lorsqu'il n'y en a plus.

Les consignes suivantes te permettront de coder la fonction **void buy_candy**() qui servira à créer les bonbons sur le plateau de jeu. Pour ce faire, essaie de détecter si le joueur utilise le clic droit ou le clic gauche dans ton code.

Dans le cas où le joueur appuie sur le clic droit et que la souris est sur l'une des dernières cases :

* Tu dois pouvoir faire réapparaître un bonbon explosif s'il n'y en a pas déjà un.
* Cette action consomme des points du score pour pouvoir replacer des bonbons sur la carte de façon automatique, ils coûtent **10** points et tu ne dois pas pouvoir arriver à un score négatif.

> **Information**
>
> Pour détecter les clics, tu auras besoin d'utiliser la variable intégrée dans Processing, **mouseButton**.

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 sbires du malveillant Jack La Trouille : les barils explosifs. Ils permettront de bloquer jusqu'à **5** zombitrouilles, c'est super solide !

> **Information**
>
> Le baril coûte **5** points !

Il faut donc pouvoir placer les barils avec la souris. Ensuite, il faut les afficher. Enfin, tu dois faire en sorte que lorsqu'un zombitrouille rencontre le baril, il disparaisse et 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** est la classe qui va regrouper ces fonctions :
  - **Barrel(int x, int y)** qui définit le constructeur de la classe.
  - **void display()** qui va afficher l'image du baril.
  - **int is_broken()** qui sert à vérifier si le baril est toujours en place.
  - **int touch(int x, int y)** qui vérifie si un zombitrouille a touché le baril.

- **void check_barrel()** sert à vérifier si les barils ont été touchés en comparant leur position avec celle de la horde de monstres.

- **void draw_barrel()** va afficher les barils et récupère leur image via la fonction display de la classe **Barrel**.

- **void buy_a_barrel()** va placer un baril sur la carte et retirer son coût au score. Comme pour les bonbons explosifs, tu ne peux pas avoir un nombre négatif de points.

Avec ça, les zombitrouilles auront du mal à passer !