TP ImgNum 3_2 : Création d'un Jeu Vidéo Space Invaders avec Processing

Clément Aubeuf – clement.aubeuf@univ-fcomte.fr

Objectif :
Ce TP a pour objectif de vous faire réaliser un jeu 2D de type **Space Invaders** en utilisant Processing. Vous allez apprendre à manipuler des objets, créer des ennemis, gérer les collisions, et animer des éléments du jeu. À la fin du TP, vous aurez un jeu où le joueur contrôle un vaisseau spatial qui doit détruire des vagues d'ennemis.

---

1. Prise en main : Écran de jeu et configuration

Tâche :
Initialisez une fenêtre de jeu de 800x600 pixels et affichez le vaisseau contrôlé par le joueur au bas de l'écran.

Étapes détaillées :

- Créez la fenêtre de jeu avec `size(800, 600);`.
- Créez une classe `Vaisseau` pour gérer le vaisseau du joueur. Ce vaisseau sera représenté par un triangle ou un rectangle de 40x20 pixels, positionné en bas de l'écran.
- Dessinez le vaisseau dans la méthode `draw()` avec `triangle()` ou `rect()` pour représenter le joueur.
- Positionnez initialement le vaisseau au centre de l'écran, en bas (par exemple, `x = width / 2` et `y = height - 50`).

Précisions :

- Vous pouvez rendre le design du vaisseau plus intéressant en ajustant la forme ou en utilisant une image importée avec `loadImage()` et `image()`.

---

2. Déplacement du vaisseau

Tâche :
Permettez au joueur de déplacer son vaisseau horizontalement à l'aide des touches gauche et droite du clavier.

Étapes détaillées :

- Utilisez la fonction `keyPressed()` pour gérer les déplacements avec les touches fléchées. Par exemple :
  ```
  if (keyCode == LEFT) { x -= vitesse; }
  if (keyCode == RIGHT) { x += vitesse; }
  ```
- Définissez une variable `vitesse = 5` pour ajuster la vitesse de déplacement.
- Limitez le déplacement du vaisseau aux bords de l'écran avec des conditions :
  ```
  if (x < 0) { x = 0; }
  if (x > width - 40) { x = width - 40; }.
  ```

Précisions :

- Vous pouvez implémenter une accélération ou un freinage pour rendre le contrôle plus fluide, bien que cela ne soit pas obligatoire dans ce type de jeu.

---

3. Tir de projectiles

Tâche :
Le vaisseau doit pouvoir tirer des projectiles vers le haut lorsqu'une touche (comme la barre d'espace) est pressée.

Étapes détaillées :

- Créez une classe `Projectile` pour gérer les tirs. Chaque projectile sera représenté par un petit rectangle ou un cercle.
- Dans `keyPressed()`, vérifiez si la barre d'espace est pressée (`key == ' '`) pour créer un nouveau projectile et l'ajouter à une liste de projectiles actifs.
- Dans la méthode `draw()`, mettez à jour la position des projectiles (en les déplaçant vers le haut) et dessinez-les à l'écran avec `rect()` ou `ellipse()`.
- Supprimez les projectiles qui sortent de l'écran pour éviter d'accumuler des objets inutiles.

Précisions :

- Assurez-vous que les projectiles partent du centre du vaisseau (`x + 20`, au sommet du vaisseau).
- Vous pouvez ajuster la fréquence des tirs pour éviter que le joueur ne tire trop rapidement (par exemple, en imposant un délai entre deux tirs).

---

4. Création des ennemis

Tâche :
Créez plusieurs ennemis disposés en formation dans le haut de l'écran, qui se déplacent horizontalement et changent de direction lorsqu'ils touchent un bord de l'écran.

Étapes détaillées :

- Créez une classe `Ennemi` pour gérer chaque ennemi. Ils peuvent être représentés par des rectangles ou des images importées.
- Générez un tableau 2D d'ennemis (par exemple, 5 rangées de 10 ennemis).
- Faites en sorte que les ennemis se déplacent horizontalement. Lorsque le premier ou le dernier ennemi atteint le bord de l'écran, tous les ennemis doivent descendre d'un niveau et changer de direction.
- Dans la méthode `draw()`, dessinez chaque ennemi et mettez à jour leur position.

Précisions :

- Vous pouvez ajuster la vitesse des ennemis pour rendre le jeu plus ou moins difficile.
- Lorsque les ennemis descendent d'un niveau, vous pouvez augmenter leur vitesse progressivement pour accroître la difficulté.

---

5. Collisions entre projectiles et ennemis

Tâche :
Lorsque les projectiles du joueur touchent un ennemi, celui-ci doit être détruit.

Étapes détaillées :

- Pour chaque projectile, vérifiez s'il entre en collision avec un ennemi en utilisant une condition sur leurs positions (`if (projectile.x > ennemi.x && projectile.x < ennemi.x + width)`).
- Si une collision est détectée, supprimez l'ennemi de la liste d'ennemis et le projectile de la liste des projectiles.
- Ajoutez un score à chaque ennemi détruit (par exemple, 100 points).

Précisions :

- Vous pouvez utiliser la fonction `dist()` pour simplifier la détection de collision si les projectiles et les ennemis sont circulaires.
- Pour améliorer la lisibilité du code, regroupez la logique de suppression des objets (ennemis et projectiles) dans des fonctions séparées.

---

6. Les tirs ennemis

Tâche :
Implémentez un système où certains ennemis tirent des projectiles vers le joueur à intervalles réguliers.

Étapes détaillées :

- Créez une méthode pour que des ennemis choisis aléatoirement puissent tirer des projectiles vers le bas.
- Les projectiles ennemis doivent se déplacer vers le bas, et vous devez vérifier s'ils touchent le joueur (en utilisant un système de collision similaire à celui des projectiles du joueur).
- Si un projectile ennemi touche le vaisseau, le joueur perd une vie.

Précisions :

- Vous pouvez limiter la fréquence des tirs ennemis avec un système de temps (`millis()`).
- Ajoutez une animation ou un effet sonore lorsque le vaisseau est touché.

---

7. Gestion des vies et du score

Tâche :
Ajoutez un système de vies pour le joueur et un score qui augmente à chaque ennemi détruit.

Étapes détaillées :

- Créez une variable `vies` qui commence avec 3 vies. À chaque fois qu'un projectile ennemi touche le joueur, décrémentez cette variable.
- Lorsque le joueur perd toutes ses vies, affichez un écran "Game Over" et arrêtez le jeu.
- Ajoutez une variable `score` qui augmente lorsque le joueur détruit un ennemi.
- Affichez le score et le nombre de vies restantes en haut de l'écran en utilisant la fonction `text()`.

Précisions :

- Vous pouvez faire apparaître des vies supplémentaires ou des bonus à intervalles réguliers pour aider le joueur.

---

8. Niveau et difficulté croissante

Tâche :
Lorsque tous les ennemis sont détruits, générez une nouvelle vague d'ennemis plus rapide et plus difficile.

Étapes détaillées :

- Après que tous les ennemis d'un niveau sont détruits, augmentez la vitesse des ennemis pour le niveau suivant.
- Ajoutez des comportements supplémentaires pour rendre les ennemis plus difficiles (par exemple, ennemis qui tirent plus souvent ou se déplacent plus rapidement).
- Relancez le placement des ennemis, mais en ajoutant un ou deux rangs supplémentaires à chaque niveau pour rendre la progression plus complexe.

Précisions :

- La difficulté doit augmenter graduellement pour maintenir l'intérêt du joueur sans rendre le jeu trop difficile dès le départ.
- Vous pouvez ajouter une indication du niveau actuel en haut de l'écran avec `text()`.

Pour aller plus loin :

1. **Graphismes et animations** : Remplacez les formes géométriques simples par des sprites animés pour le vaisseau, les ennemis et les projectiles. Utilisez `loadImage()` pour importer des images et `image()` pour les afficher.
2. **Musique et effets sonores** : Ajoutez de la musique de fond et des effets sonores pour les tirs, les explosions et les collisions à l'aide de la librairie `Sound` de Processing.
3. **Vagues d'ennemis aléatoires** : Créez des formations d'ennemis générées de manière aléatoire, avec des ennemis de tailles, de vitesses et de comportements différents.
4. **Power-ups** : Ajoutez des bonus que le joueur peut collecter, comme des tirs multiples, des boucliers protecteurs, ou une augmentation de la vitesse de tir.
5. **Écrans de pause et de menu** : Ajoutez un menu principal permettant de commencer une nouvelle partie, de reprendre une partie, ou de consulter les scores. Implémentez également un écran de pause.
6. **Multijoueur local** : Implémentez un mode multijoueur où deux joueurs peuvent coopérer pour détruire les vagues d'ennemis, en utilisant des contrôles séparés (par exemple, un joueur avec les touches fléchées et l'autre avec `WASD`).
7. **Sauvegarde des scores** : Ajoutez un tableau des meilleurs scores pour que les joueurs puissent comparer leurs performances.