Compte rendu Jeu vidéo python : “Space shooter”

Ce document a pour but de présenter notre jeu Space Shooter. L'objectif est de piloter un vaisseau spatial et de détruire les ennemis qui arrivent, plus on tue d'ennemis plus notre score grimpe, la partie dure un temps infini)

Concept du jeu

Le joueur contrôle un vaisseau qui se déplace dans l'espace (x). Des vaisseaux ennemis apparaissent en haut de l'écran et descendent vers le bas. Le joueur doit les éliminer avant qu'ils n'atteignent le bas de l'écran.

Si un ennemi réussit à passer, la partie se termine. Le joueur perd aussi s'il n'a plus de points de vie (à cause des collisions avec les ennemis ou les tirs du boss).

A la fin de chaque vague, un boss apparaît. Ce boss est plus grand, possède beaucoup plus de vie, et tire des lasers rouges sur le joueur.

Architecture du code

Le code est séparé en plusieurs fichiers Python , nous avons également mis une docstring dans le code pour le rendre plus lisible et claire :

1. main.py
   Fichier principal qui contient la boucle de jeu, le menu, la gestion des explosions et l'affichage de tous les éléments à l'écran.

2. player.py
   Classe Joueur qui gère le vaisseau du joueur. Elle contient les méthodes pour le déplacement, le tir, la prise de dégats et l'amélioration.

3. enemy.py
   Classe Ennemi pour les vaisseaux ennemis basiques. Ils descendent vers le bas avec une vitesse qui dépend de la difficulté.

4. boss.py
   Classe Boss pour le vaisseau ennemi de fin de vague. Il se déplace de gauche à droite et tire des lasers vers le joueur.

5. laser.py
   Classe laser utilisée pour les projectiles. Elle fonctionne dans les deux directions : vers le haut (joueur) ou vers le bas (boss).

Commandes

- Flèches directionnelles ou z/q/s/d : déplacement du vaisseau
- Le tir est automatique (pas besoin d'appuyer sur une touche)
- Espace: démarrer une partie depuis le menu
- Echap : quitter le jeu ou retourner au menu

Fonctionnalités techniques :

Défilement du fond :
Le fond spatial défile de haut en bas pour donner une impression de mouvement. Deux copies de l'image sont utilisées et se replacent automatiquement quand elles sortent de l'écran.

Explosions :
Quand un ennemi est détruit, une explosion apparaît. Cette explosion est un cercle qui grandit progressivement tout en devenant transparent.

Difficulté progressive :
La difficulté augmente selon le niveau et la vague. La formule utilisée est : difficulté = niveau + (vague - 1) * 0.2. (On peut changer ces valeurs pour modifier l'équilibrage du jeu).

Amélioration du vaisseau :
Chaque fois que le joueur bat un boss, son vaisseau monte d'un niveau (maximum niveau 6). Les améliorations sont : vitesse de déplacement, cadence de tir, et dégâts des lasers. L'image du vaisseau change aussi.

Système de progression

Le jeu est divisé en niveaux. Chaque niveau contient 5 vagues.

Pour chaque vague :
- Le joueur doit éliminer un certain nombre d'ennemis (10 au niveau 1)
- Une fois l'objectif atteint, le boss apparait
- La victoire contre le boss permet de passer à la vague suivante

Après 5 vagues, le joueur passe au niveau suivant. Le nombre d'ennemis à éliminer augmente avec les niveaux.

Dossier resources

Le dossier "asssets" contient les ressources du jeu :
- BG Music.mp3 : musique de fond
- Backgound/2D space BG.png : image du fond spatial
- Enemies/ : images des vaisseaux ennemis (4 dossiers pour les vagues)
- Space ships/ : images du vaisseau joueur (Level 1 à Level 6)

Gestion des collisions

Les collisions sont détectées avec des rectangles (hitbox). Chaque objet possède une méthode get_rect() qui retourne sa position et sa taille.

Collisions vérifiées :
- Laser du joueur contre ennemi : l'ennemi est détruit (+10 points)
- Laser du joueur contre boss : le boss perd de la vie
- Joueur contre ennemi : le joueur perd 20 PV, l'ennemi est détruit
- Joueur contre boss : le joueur perd 30 PV
- Laser du boss contre joueur : le joueur perd 15 PV

Barre de vies

Le joueur possède 100 points de vie. Une barre est affichée en haut à droite de l'écran. Le calcul du remplissage utilise le ratio : vie / vie_max.

Le boss possède également une barre de vie, positionnée au-dessus de lui.

Conclusion :

Nous avons implémenté un système de jeu complet
avec un menu, des ennemis, un boss, et un système de progression.

Evolution future :

Pour la matière “ia” nous allons prochainement intégrer un algorithme d' apprentissage renforcé (utilisation du perceptron) ainsi que un algorithme de recherche (A* ou Dijkstra).