Groupe Alternant BUT3
Année : 2025 - 2026

INSTITUT UNIVERSITAIRE DE TECHNOLOGIE
AMIENS
Université de Picardie Jules Verne

R5.8 : Qualité de développement

Projet : Space Zinzins

Documentation Technique

Julian Cazin - Clément Duberseuil - Arthur Jacquot - Corentin Dupuy

Table des Matières :

I. Introduction

A. Présentation du jeu

Space Zinzins est un jeu vidéo de type Shoot ‘Em Up développé avec PyGames. Le joueur contrôle un vaisseau spatial qui tire sur des vagues d’ennemis lui arrivant dessus. Il doit survivre le plus longtemps possible en tuant tous les ennemis de chaque vague. Afin de l’aider, plusieurs bonus lui facilitant la tâche apparaîtront de manière aléatoire au cours de chaque vague.

B. Objectifs de la documentation technique

Cette documentation a pour but de :

- Décrire l’architecture technique du projet.
- Détaillez les choix technologiques (moteur, bibliothèques, langages).
- Fournir un support aux développeurs pour la maintenance et l’évolution du jeu.
- Servir de référence lors du déploiement et des mises à jour.

II. Architecture du jeu

Le jeu est programmé en **Python 3.10** avec le package **PyGames 2.6.0.**

A. Système de rendu

La résolution de base est de 800x600 pixels. C’est un rendu 2D basé sur SDL via Pygame. Pour la gestion du rafraîchissement, on utilise `pygame.display.flip()` (qui correspond à 60 FPS).

B. Gestion des ressources

Les sprites sont au format PNG. Les sons sont au format WAV/MP3. Le chargement est contrôlé via `pygame.image.load()` et `pygame.mixer.Sound()`.

III. Conception du gameplay

A. Mécaniques de jeu

Le joueur se déplace horizontalement à gauche ou à droite lorsqu'il appuie sur la flèche directionnelle gauche ou droite. Il tire des projectiles vers le haut lorsque le joueur appuie sur la barre espace. Il possède un délai de 1 seconde entre chaque tir.

Les aliens apparaissent en vague d'ennemis. Les vagues d'aliens se déplacent horizontalement puis descendent d'un cran lorsqu'elles touchent un bord de l'écran. Les aliens possèdent un nombre de vie. De base, il est à un. Lorsque ce nombre tombe à 0, l'alien disparaît et augmente le score du joueur. Certains aliens peuvent tirer des projectiles qui blessent le joueur.

Le joueur possède un score qui augmente à chaque ennemi vaincu. Il possède trois vies au début de la partie. Après chaque vague, il récupère une vie jusqu'à un maximum de cinq. Lorsqu'il n'a plus de vie, le joueur perd et la modale de fin de partie apparaît.

B. Systèmes de contrôle

Le déplacement s'effectue de manière horizontale à l'aide des touches du clavier, pour cela, nous nous servirons des touches directionnelles que sont les flèches. L'action de tir s'effectuera à l'aide du bouton espace, tandis que l'activation du bonus se fera par la touche Alt. Le clic gauche de la souris servira quant à lui au moment de cliquer sur les boutons permettant de relancer une partie, de quitter le jeu, ou de lancer une partie.

C. Interactions joueur-environnement

Les interactions entre les différents éléments du jeu s'effectuera comme ceci :

- Collision de l'alien et de la balle du joueur → l'alien est détruit
- Collision entre l'alien et le joueur → le joueur a perdu
- Collision entre la balle de l'alien et le joueur → explosion le joueur perd une vie et la balle est détruite

IV. Graphismes et Animation

A. Modélisation 2D

Les designs qui représenteront les éléments du terrain seront dans un style pixel-art afin d'accentuer l'effet rétro du jeu. Le jeu ne possède pas de shaders (rendu 2D simple). Nous utilisons des couleurs fixes et des images bitmap. Nous avons repris des designs sur internet car le but du jeu n'est pas d'être commercialisé.

B. Animation

Les animations des aliens s'effectuent par l'alternance de sprites (clignotement, déplacement, …). Les explosions sont animées par des sprites sheets. Les tirs agissent de la même façon.

V. Son et Musique

A. Effets sonores

Des effets sonores se produisent lorsque :

- Le joueur se fait toucher
- Un alien se fait toucher
- Un boss apparaît
- Le joueur meurt
- Le joueur tir

B. Musique d'ambiance

La musique de fond sera une boucle 8-Bit, cela permettra encore une fois d'accentuer le côté rétro du jeu.

L'intégration des différents effets sonores se fera à l'aide de pygame.mixer et seront déclenchés par des événements (tir, collision, …).

VI. Optimisation et Performance

A. Gestion des ressources système

Les images ainsi que le fond d'écran et les effets sonores sont préchargés au démarrage du jeu afin d'assurer les meilleurs performances possibles en partie, un système de réutilisation des objets à égalements était mis en place au travers de liste (ex : projectile)

B. Optimisation du code

Les boucles de jeu sont optimisées grâce à la librairie Delta time (pygame.time.Clock()). Nous utilisons également rect.colliderct() pour gérer les collisions entre les différents éléments du jeu.

C. Tests de performance

Les tests des performances doivent atteindre au minimum 60 fps constant lorsque moins de 100 objets se trouvent affichés.

VII. Documentation du Code

A. Organisation du code

Le code est organisé sous forme de “namespace”, on va donc pouvoir retrouver tout ce qui va être relatif au gameplay, à l’audio, à l’écran, etc … Les répertoires sont également séparés entre les scripts, les préfabriqués, etc… Au sein du code, on organise les différentes parties en classe afin de séparer la logique pour une meilleure compréhension et isolation (screen, player, etc…), les classes sont séparées.

B. Commentaires et documentation interne

Les commentaires doivent respecter du mieux possible un style PEP8 afin d’assurer une cohérence des commentaires dans le projet pour une meilleure compréhension. Une docstring sera également appliquée à chaque classe et méthode développée.

VIII. Déploiement et Maintenance

A. Configuration requise

La configuration minimale requise pour pouvoir développer le jeu est un Python supérieur la version 3.10 ainsi qu’un Pygame en version 2.6.0. Il est également nécessaire d'être sous un système d’exploitation Windows 10 minimum.

B. Procédure d’installation

La procédure d’installation et de lancement se fait par ligne de commandes au sein du répertoir du jeu :

Installation des dépendances : `pip install -r requirements.txt`

Lancement du jeu : `python main.py`

C. Gestion des mises à jour

La gestion des différentes versions de l'application et du jeu se fait via Github, les différents commits apportés à la branche principale "main" apportent de nouvelles fonctionnalités, tests ou correctifs de qualités. Les correctifs complets sont quant à eux publiés sous forme de tags/releases.

IX. Procédures de test technique

A. Tests unitaires

Le code doit contenir des tests unitaires afin d'assurer le bon fonctionnement du code. Ces tests permettent de vérifier le côté technique des classes et des fonctions développées. Pour tester, nous utiliserons la bibliothèque pytest dans sa version 8.4.1. Chaque fichier aura son propre fichier de test associé.

B. Tests d'utilisation

En plus des tests unitaires, nous avons un plan de test qui permet de vérifier le bon fonctionnement du jeu dans ses scénarios d'utilisation. Ceux-ci assurent le bon fonctionnement du code d'un point de vue utilisateur. Ils permettent de vérifier que l'ensemble de l'application fonctionne comme souhaité.