Joshua LEMOINE & Berachem MARKRIA - *E3FI, groupe 2I*

PROJET

SPACE INVADERS

rapport de projet

Problématique du projet

Ce projet en C# consiste à développer le célèbre jeu Space Invaders, un classique de l'arcade où le joueur contrôle un vaisseau spatial afin de défendre la Terre contre une invasion extraterrestre.

L'objectif principal du projet est de créer une application interactive offrant une expérience de jeu fluide et divertissante. Le défi réside dans la mise en œuvre des éléments essentiels du jeu, tels que la gestion des mouvements du vaisseau spatial, le tir de projectiles, la génération des ennemis et la gestion des collisions.

Le code doit être structuré de manière à garantir une bonne facilité de maintenance et de lecture tout en suivant les principes de programmation orientée objet. De plus, le rendu visuel du jeu doit être soigné, avec une interface utilisateur attrayante et des graphismes reflétant l'esthétique emblématique du jeu Space Invaders. Ce projet offre une opportunité d'appliquer les concepts fondamentaux de la programmation en C# tout en créant une expérience ludique pour les utilisateurs.

Stack technique du projet

Architecture

```
-- SpaceInvaders
|   ├── GameObjects
|   |   ├── Bonus
|   |   |   ├── Bonus.cs
|   |   |   ├── HealingBonus.cs
|   |   |   ├── ScoreBonus.cs
|   |   |   ├── ShieldBonus.cs
|   |   ├── Decorations
|   |   |   ├── Background.cs
|   |   |   ├── Menu.cs
|   |   |   ├── MenuButton.cs
|   |   ├── Enemies
|   |   |   ├── Boss.cs
|   |   |   ├── EnemyBlock.cs
|   |   ├── Missiles
|   |   |   ├── Missile.cs
|   |   |   ├── MonoShooting.cs
|   |   |   ├── MultiShooting.cs
|   |   |   ├── ShootingSystem.cs
|   |   ├── Utils
|   |   |   ├── Controls.cs
|   |   |   ├── HighScoreManager.cs
|   |   |   ├── SerializerFileManager.cs
|   |   |   ├── SoundManager.cs
|   |   |   ├── Vecteur2D.cs
|   |   ├── Bunker.cs
|   |   ├── GameObject.cs
|   |   ├── GeneralObject.cs
|   |   ├── PlayerSpaceship.cs
|   |   ├── SimpleObject.cs
|   |   ├── SpaceShip.cs
|
├── LICENSE
├── README.md
├── SpaceInvaders.sln
├── app.config
├── ClassDiagram1.cd
├── Form1.cs
├── Form1.Designer.cs
├── Form1.resx
├── Game.cs
├── Program.cs
└── SpaceInvaders.csproj
```

Le dossier GameObjects regroupe toutes les classes que nous avons ajoutées au projet initial. Parmis ces classes trois sont intéressante à définir:

- ***GameObject***: tous les objets faisant partie intégrante du jeu
- ***GeneralObject***: tous les objets ayant besoin d'être dessinés et actualisé mais qui ne font pas partie du jeux en lui même (les décorations par exemple)
- ***SimpleObject***: regroupe tous les objets qui devront gérer des actions basiques (gestion de la vie par exemple).

On peut ensuite voir les différents dossier :

- **Bonus**: regroupe tous les bonus qui héritent de Bonus.cs, cela permet de n'avoir qu'à redéfinir la méthode d'effet.
- **Decorations**: toutes les décorations du jeux (ce qui hérite de GeneralObject)
- **Ennemies**: regroupe les classes qui représentent des ennemis. Ne comprend pas SpaceShips puisqu'il n'est pas forcément un ennemi.

- **Missiles**: regroupe les classes qui s'occupent de la gestion des tirs. Ce système à été mis en place à la création du boss. La classe la plus intéressante est *ShootingSystem* qui simplifie la gestion des tirs malgré leurs différences (tir simple, salve…)
- **Utils:** dossier qui regroupe toutes les classes utilitaires, la gestion du son, des contrôles, du leaderboard…

Problèmes rencontrés

1. Gestion des Collisions :

- **Problème :** La gestion des collisions nous a donné du fil à retendre, surtout en ce qui concerne les interactions avec les bunkers et la coordination des tirs entre vaisseaux alliés et ennemis.
- **Solution :** Une longue relecture de nos calculs de dimensions à été faite et nous avons fini par corriger ce bug. Une des corrections a été de constamment préciser la taille de chaque image dans les fonctions d'affichage, même si nous utilisions leur taille par défaut.

3. Wrap des Ennemis :

- **Problème :** L'implémentation du "wrap" des ennemis, permettant leur déplacement d'un bord de l'écran à l'autre, présentait des difficultés. Le calcul des dimensions de l'EnemyBlock après la perte d'une colonne posait des problèmes.
- **Solution :** Ce problème venait de la récupération des ennemis les plus à gauche/droite/haut/bas, ce qui provoque une mauvaise utilisation du wrap.

5. Différenciation des Systèmes de Tirs (Création du Boss) :

- **Problème :** La gestion des différents systèmes de tirs, en particulier lors de la création du boss, nécessitait une attention particulière. La transition du stockage direct des missiles dans le vaisseau spatial vers l'utilisation d'une classe Shooting System a été complexe.
- **Solution :** Nous avons mis en place la gestion des tirs grâce à la composition. Avec la création d'une classe ShootingSystem ajoutée aux vaisseaux pour gérer de manière plus modulaire le style de tir.

7. Ajout de General Object pour Éviter le Code Inutile des Objets Décorations :

- **Problème :** La création des menus a introduit du code potentiellement inutile. La présence de beaucoup de lignes de code inutiles dans les décorations nécessitait une optimisation.
- **Solution** : L'introduction de la classe General Object a permis d'éviter la redondance de code en prenant en charge les fonctionnalités les plus basiques.

Ces ajustements ont contribué à résoudre efficacement les problèmes rencontrés, améliorant la fonctionnalité, la lisibilité et l'esthétique globale du jeu Space Invaders.

> *“On a beaucoup appris grâce au projet.*
>
> *Nous sommes heureux de ce que nous avons réalisé ! “*
>
> – Joshua LEMOINE et Berachem MARKRIA