Rapport de projet de refonte d'un jeu vidéo

Jean-Charles KAING 3808150

Shiyao CHEN 28707756

Avril 2024

Contents

1 Introduction

1.1 Objectif du projet

L'équipe de développement du projet est composé de Shiyao Chen et de Jean-Charles Kaing.

L'objectif du projet est de développer un jeu en 2D de type Space Shooter en utilisant React. Dans ce jeu, les joueurs prendront le contrôle d'un vaisseau spatial et affronteront des vagues d'ennemis dans l'espace.

1.2 Analyse fonctionnelle

Dans le cadre de notre processus de développement, nous avons effectué une analyse fonctionnelle du jeu. Cela inclut l'identification des différentes actions que les joueurs peuvent effectuer ainsi que leurs attentes vis-à-vis du jeu. Nous avons utilisé des use cases pour décrire les interactions entre les utilisateurs et le système.

1.2.1 Usercase

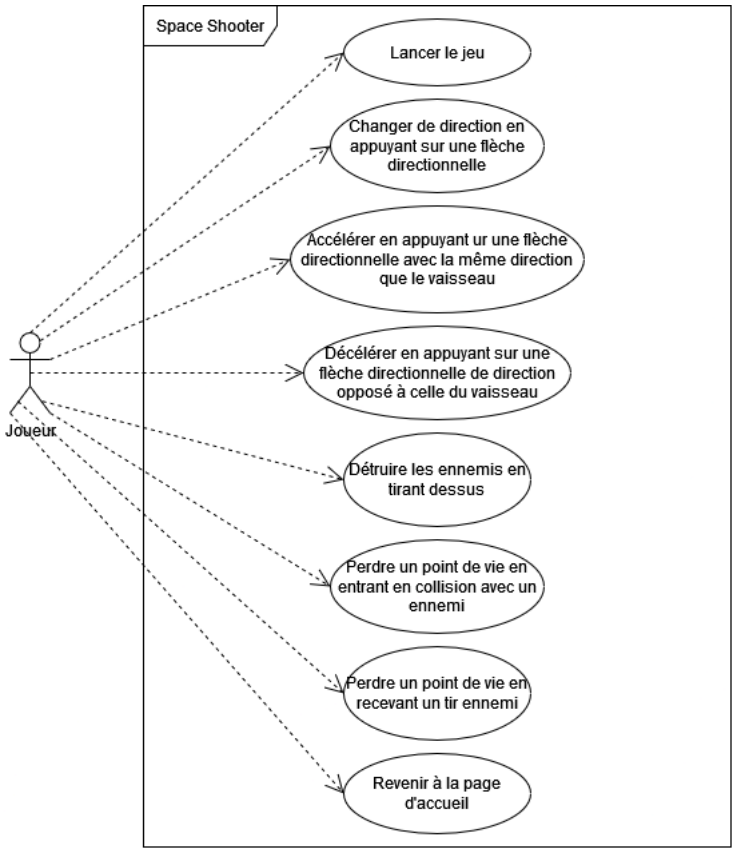

Figure 1: Usercase du jeu Space Shooter

1.2.2 Userstory

Le joueur lance l'application et clique sur "Start Game". Le jeu commence, avec le score en haut à gauche de l'écran et cinq points de vie en haut à droite de l'écran.

Le joueur déplace le vaisseau spatial à l'aide des touches directionnelles, et accélère ou décélére. Les vaisseaux ennemis viennent du haut de l'écran et se dirigent vers le bas et peuvent tirer.

Le vaisseau du joueur lance des salves de tirs quasiment en continu. Lorsque un tir du vaisseau du joueur touche un vaisseau ennemi, il gagne des points et le vaisseau ennemi est détruit.

Lorsque le vaisseau du joueur est touché par un tir ou qu'il rentre en collision avec un autre vaisseau, le joueur perd un point de vie. Lorsque le joueur perd tout ses point de vie, il a finit la partie, et une notification lui indique le nombre de points qu'il a gagné. Il clique sur "OK" et revient à la page principale.

1.3 Illustration avec un Wireframe

Pour visualiser et planifier l'interface utilisateur ainsi que les différentes scènes du jeu, nous avons dessiné des wireframes. Ils nous ont permis de définir la disposition des éléments de jeu, les interactions utilisateur et l'architecture globale de l'interface utilisateur.

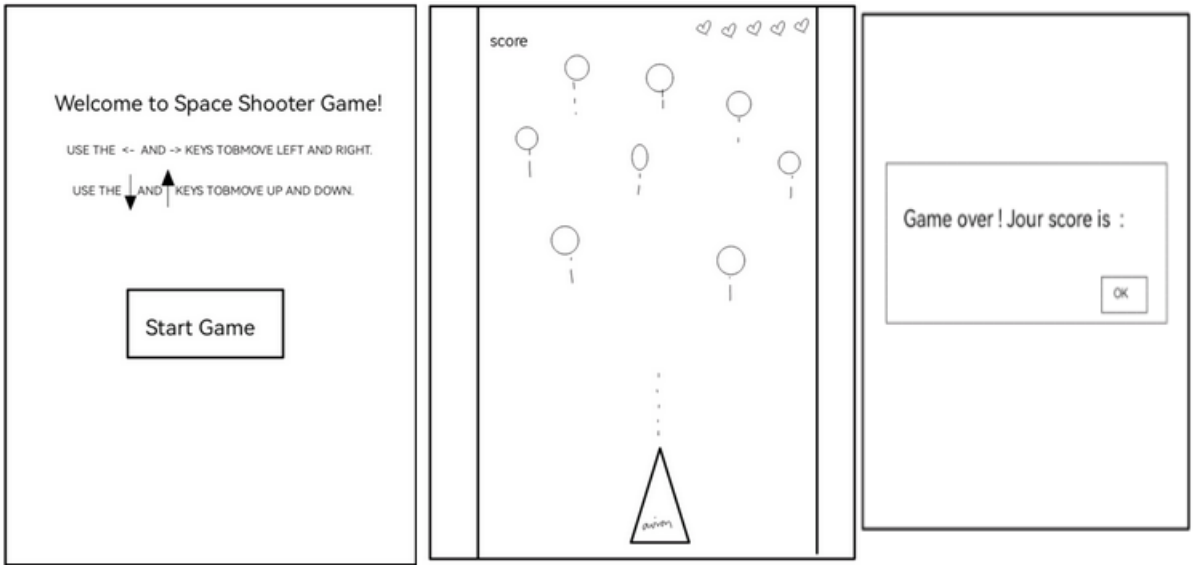

Figure 2: Les wireframes du jeu Space Shooter

1.4 Technologies utilisées

Pour le développement de notre jeu Space Shooter, nous avons choisi d'utiliser React en tant que framework principal. Voici un tableau comparatif des technologies les plus fréquemment utilisées sur le marché, justifiant notre choix de React :

Technologie	Avantages	Inconvénients
React	• Approche déclarative pour la création d'interfaces utilisateur • Composants réutilisables • Virtual DOM pour des performances optimisées • Grande communauté et écosystème • Performant	• Complexité pour les débutants • Nécessite des outils supplémentaires pour la gestion de l'état global et du routage
Phaser.js	• Spécialisé dans les jeux HTML5 • Grande documentation et communauté active	• Moins de flexibilité que React • Nécessite une certaine connaissance du développement de jeux pour une utilisation efficace
Pixi.js	• Hautes performances pour le rendu graphique • Supporte le WebGL pour un rendu accéléré sur les navigateurs modernes	• Moins d'abstraction que React • Nécessite une certaine expertise en graphisme pour une utilisation efficace
Unity	• Puissant moteur de jeu pour les jeux 2D et 3D • Large gamme de fonctionnalités intégrées • Prise en charge de multiples plateformes • Grande communauté et ressources disponibles	• Courbe d'apprentissage initiale plus raide

Table 1: Tableau comparatif des technologies les plus fréquemment

Nous avons choisi React pour sa popularité, sa grande communauté de développeurs et sa flexibilité, qui nous permettra de créer une interface utilisateur interactive et réactive pour notre jeu.

2 Technique

Après avoir défini les membres de notre équipe et l'objectif de notre projet, passons maintenant à la partie technique. Nous explorerons l'architecture de notre application, mettant en lumière l'utilisation de React pour gérer l'interface utilisateur, l'état du jeu et la logique de rendu. Ces éléments sont essentiels pour assurer une expérience de jeu fluide et engageante pour nos utilisateurs.

2.1 Architecture générale de l'application

Dans le cadre de notre projet Space Shooter, nous avons adopté une architecture basée sur le modèle MVC (Modèle-Vue-Contrôleur), qui divise l'application en trois composants principaux :

1. Modèle :
- 'state.ts' correspond à cette couche en tant que gestionnaire de l'état global de l'application, y compris le stockage et la récupération des données. Il est responsable de la logique de jeu (par exemple, les collisions), de la gestion des entités telles que les vaisseaux spatiaux, les ennemis, les projectiles, ainsi que des interactions entre ces entités.

2. Vue :
- 'index.tsx' est le point d'entrée de l'application côté client et peut être considéré comme faisant partie de la couche vue. Il est responsable de l'interface utilisateur et de l'affichage graphique du jeu.
- 'renderer.ts' est également une partie de la couche vue, responsable du rendu des composants de l'interface utilisateur, y compris la représentation visuelle des entités du jeu, des menus et des écrans de jeu.

3. Contrôleur :
- Bien qu'il n'y ait pas de fichier spécifiquement dédié au contrôleur, la logique de contrôle peut être répartie entre les fichiers existants. Le contrôleur est responsable de la gestion des interactions utilisateur, telles que les entrées au clavier, et de la transmission de ces commandes au modèle pour modifier l'état du jeu.

4. Configuration Globale :
- 'conf.ts' est un fichier de configuration globale contenant des valeurs utilisées à travers toute l'application. Il stocke des constantes et des paramètres nécessaires dans plusieurs parties de l'application.

Cette répartition en couches et composants correspond bien à l'architecture MVC et facilite la compréhension et la maintenance de votre projet Space Shooter.

2.2 Présentation de la couche client

La couche client est composé de index.tsx et de renderer.ts qui sont responsables du rendu des composants de l'interface utilisateur.

Le rendu graphique obtenu est conforme à la maquette de départ, c'est-à-dire, lorsque l'on lance l'application, on arrive sur la page d'accueil, puis on clique sur start pour commencer le jeu. Une notification apparait et nous demande si on est sûr de commencer le jeu. On clique sur OK et la partie démarre.

Au cours de la partie, le score est indiquée en haut à gauche de l'écran, et les points de vie sont en haut à droite. Le joueur prend les commandes du vaisseau spatial à l'aide des flèches directionnelles et tire des salves de lasers afin de détruire les ennemis et gagner des points. Les ennemis apparaissent en haut de l'écran et se dirige vers le bas de l'écran à des vitesses différentes. Les ennemis peuvent tirer au laser.

Lorsque le vaisseau spatial rentre en collision avec un ennemi ou lorsqu'il est touché par un tir de laser ennemi, il perd un point. Le vaisseau est invincible pendant un court temps après avoir perdu une vie. Le joueur perd la partie lorsqu'il a perdu ses cinq points de vie.

Lorsque le joueur perd la partie, une notification apparait avec son score obtenu. Il clique sur OK et revient à la page principale.

2.3 Présentation de la couche data

La couche data est composé de state.ts qui est responsable de la gestion de l'état global de l'application.

Les données sont définies par différents types :

- Coord qui est composé des coordonnées x y et de la vitesse dx dy.
- Ball correspond à la représentation de lasers. Le type prend des coordonnées coord de type Coord, des points de vie life, et le temps d'invincibilité invincible.
- Size correspond aux dimensions de l'écran de jeu avec height et weidth.
- Polygone_plane représente le vaisseau spatial dirigé par le joueur. Le type prend un tableau de coordonnées points, une cordonnées centre, des points de vie life, des points précédent de vie d'invincibilité prevLife, des points d'invincibilité invincible, un compteur de clignotement blinkCounter, et un compteur de tir shootCounter.

- Polygone_ennemis représente un vaisseau ennemi. Le type prend un tableau de coordonnées points, une cordonnées centre, des points de vie life, des points d'invincibilité invincible, et un compteur de tir shootCounter.

- State, correspond à l'état global. Le type prend un vaisseau spatial plane de type Polygone_plane, un Array de tirs planeshot émanent du vaisseau du joueur, un Array d'ennemis ennemis, un Array de tirs ennemis ennemishots, la taille de l'espace de jeu size, le score et le boolean endGame.

2.4 Présentation de la couche serveur

Le serveur est responsable de la gestion de la logique de jeu. Il effectue la détection des collisions entre les projectiles du joueur, le vaisseau spatial et les ennemis. Si l'avion du joueur entre en collision avec le vaisseau spatial ou ses balles, la vie du joueur sera réduite. Si cela se produit cinq fois, le jeu sera terminé. Lorsque les projectiles du joueur entrent en collision avec le vaisseau spatial, celui-ci est détruit et des points sont gagnés.

En raison du fait que les vaisseaux du joueur et les vaisseaux spatiaux sont tous deux des objets irréguliers, nous utilisons l'algorithme de séparation d'axes (SAT) pour détecter les collisions. Cet algorithme implique de projeter la forme des objets (généralement des polygones convexes) sur différents axes et de vérifier s'ils se chevauchent le long de ces axes. Si aucune séparation n'est détectée sur tous les axes, alors les objets se chevauchent et une collision est détectée. L'algorithme SAT est une méthode courante de détection de collisions, particulièrement adaptée à la gestion des collisions impliquant des formes irrégulières.

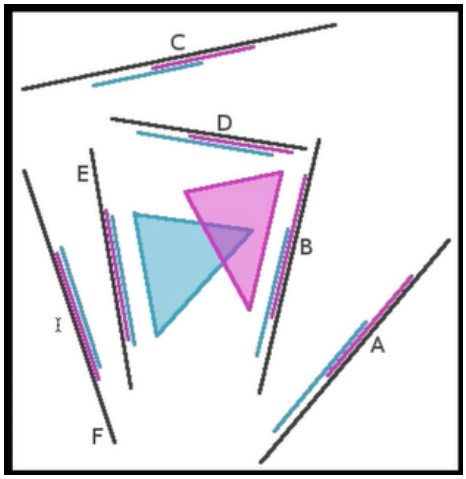

Figure 3: Collision entre deux objet convexe

Tout d'abord, nous traitons l'objet à tester comme une coque convexe, puis utilisons l'algorithme SAT pour détecter les collisions.

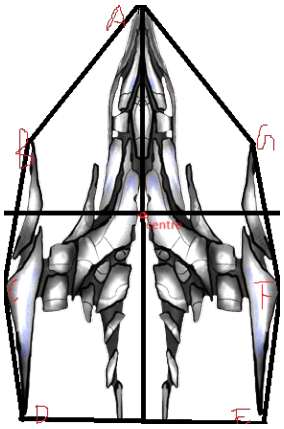

Figure 4: enveloppe convexe de player

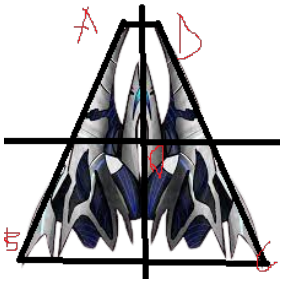

Figure 5: enveloppe convexe de l'ennemie

Voici le pseudo-code de l'algorithme SAT :

Algorithm 1 Algorithme SAT

```
1: procedure SHAPEOVERLAP_SAT(shape1, shape2)
2:     poly1 ← shape1
3:     poly2 ← shape2
4:     n ← 1
5:     while N ≤ 2 do
6:         if n = 2 then
7:             poly1 ← p2
8:             poly2 ← p1
9:         end if
10:        for i ← 0, length_poly1 do
11:            b ← (a + 1) mod length_poly1
12:            axisProj ← {x : −(poly1[b].y − poly1[a].y), y : poly1[b].x − poly1[a].x}
13:            d ← √(axisProj.x × axisProj.x + axisProj.y × axisProj.y)
14:            axisProj.x ← axisProj.x/d
15:            axisProj.y ← axisProj.y/d
16:            min_p1 ← +∞, max_p1 ← −∞
17:            for all p ∈ poly1 do
18:                q ← (p.x × axisProj.x + p.y × axisProj.y)
19:                min_p1 ← min(min_p1, q)
20:                max_p1 ← max(max_p1, q)
21:            end for
22:            min_p2 ← +∞, max_p2 ← −∞
23:            for all p ∈ poly2 do
24:                q ← (p.x × axisProj.x + p.y × axisProj.y)
25:                min_p2 ← min(min_p2, q)
26:                max_p2 ← max(max_p2, q)
27:            end for
28:            if ¬(max_p2 ≥ min_p1 && max_p1 ≥ min_p2) then
29:                return faux
30:            end if
31:        end for
32:    end while
33:    return vrai
34: end procedure
```

3 Conclusion et retour d'expérience

La conclusion et la rétrospective sur le projet Space Shooter fournissent une opportunité précieuse de réfléchir sur le travail accompli, d'apprendre de l'expérience et de planifier pour l'avenir.

Notre équipe a réalisé un travail considérable dans le développement du jeu Space Shooter. Nous avons réussi à implémenter les fonctionnalités essentielles du jeu, y compris la navigation dans l'espace, les projectiles, la destruction des ennemis et la gestion du score.

Et tout au long du projet, nous avons appris l'importance de la planification et de la communication dans un environnement de développement collaboratif. Nous avons également appris à résoudre efficacement les problèmes techniques et à rechercher des solutions innovantes face aux défis rencontrés.

Bien que nous soyons satisfaits des fonctionnalités actuelles du jeu, il reste des possibilités d'amélioration. Nous pourrions explorer des fonctionnalités multijoueurs pour ajouter une dimension sociale au jeu, ainsi que des améliorations graphiques pour une expérience visuelle plus immersive.

Pour l'avenir, nous pourrions envisager de continuer à développer le jeu en ajoutant de nouvelles fonctionnalités et en améliorant l'expérience utilisateur. Nous pourrions également explorer d'autres technologies ou plateformes pour élargir la portée du jeu et atteindre un public plus large.

En conclusion, le projet Space Shooter a été une expérience enrichissante qui nous a permis de mettre en pratique nos compétences en développement de jeux et de renforcer notre collaboration d'équipe. Nous sommes fiers du résultat obtenu et nous sommes impatients de poursuivre notre progression dans le domaine passionnant du développement de jeux vidéo.

4 Annexe

Bittle, William. (2010, January 1). SAT (Separating Axis Theorem) : www.dyn4j.org/2010/01/sat/

Szasz Janos. (2005, April 27) . The algorithmicx package : https://texdoc.org/serve/algorithmicx/0