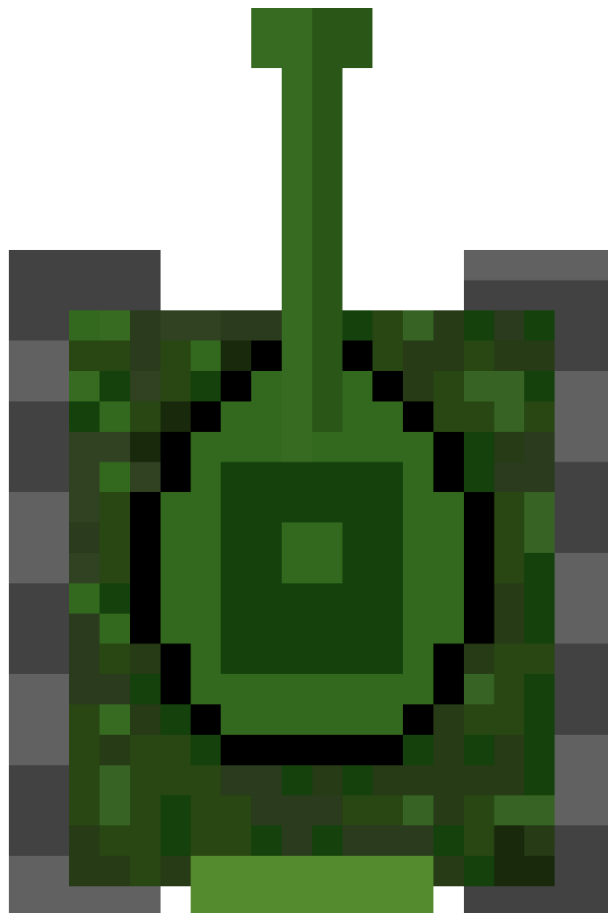
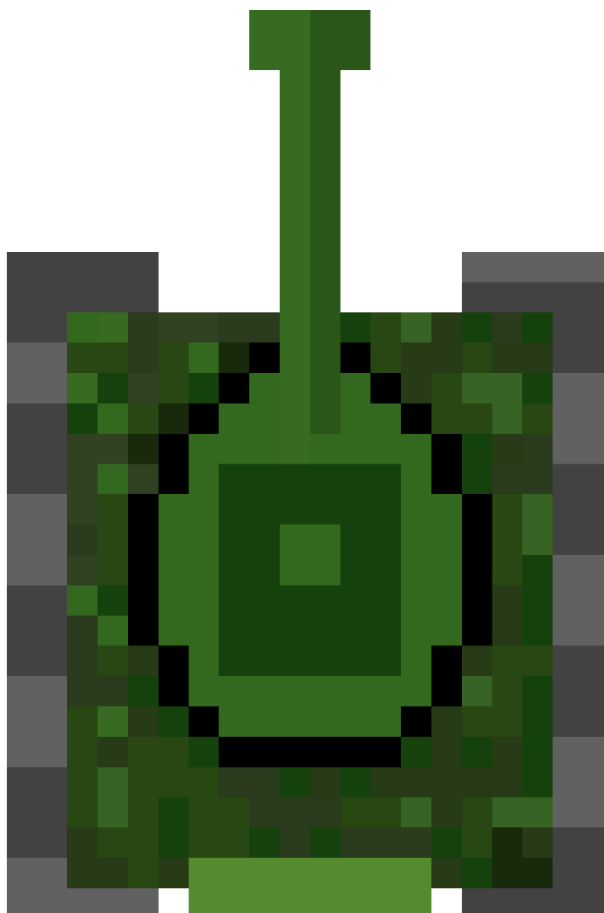


TankBattle





Mathis Olaya – MIN2A
Lausanne
2024

Table des matières

1	Analyse préliminaire	4
1.1	Introduction	4
1.2	Objectifs	4
1.3	Gestion de projet	5
2	Analyse / Conception.....	5
2.1	Gameplay.....	5
2.1.1	Le joueur :	5
2.1.2	Les ennemis :	5
2.1.3	Les protections	5
2.1.4	Les points de vie.....	5

2.1.5	Les niveaux.....	6
2.2	Concept.....	6
2.3	Analyse fonctionnelle	6
2.3.1	Sprites.....	6
2.3.2	Niveaux.....	6
2.3.3	Obstacles.....	6
2.3.4	Highscores.....	7
2.3.5	Ennemi.....	7
2.3.6	Joueur.....	7
2.3.7	Lancement.....	8
2.3.8	Collisions	8
2.3.9	Score.....	9
2.4	Stratégie de test	9
3	UX	9
3.1	Analyse de l'UX	9
3.1.1	Personas.....	9
3.1.2	Palette graphique	11
3.1.3	Eco-conception et accessibilité.....	11
3.2	Conception de l'UX.....	12
3.2.1	Définitions des Wireframes	12
3.2.2	Définition de l'High fidelity	14
3.3	Choix effectués.....	15
4	Réalisation	15
4.1	Points de design spécifiques.....	15
4.1.1	Placement de protections spécifique.....	15
4.1.2	Faire supprimer les munitions lorsqu'elles sortent de l'écran	16
4.1.3	Menu pause.....	17
4.2	Déroulement	18
4.3	Mise en place de l'environnement de travail	18
4.3.1	Installation des logiciels	18
4.3.2	Environnement	19
4.3.3	Liste des fichiers.....	20
4.4	Description des tests effectués	21
4.4.1	Sprint 1.....	21
4.5	Erreurs restantes	24
5	Conclusions	24
5.1	Objectifs	24
5.2	Positifs / négatifs	24
5.3	Difficulté du projet.....	25
5.4	Améliorations possibles.....	25
6	Annexes.....	26
6.1	Manuel de référence.....	26
6.2	Journal de travail	26

1 Analyse préliminaire

1.1 Introduction

TankBattle est un jeu de combat type Shoot me up. Celui-ci est réalisé dans le cadre de plusieurs projets à l'ETML sur les modules suivants : C106 (base de données), 320 (Programmation orientées objets), 322 (Expérience utilisateur). Ce projet rallie différents modules et permet de réunir ces différentes compétences en un seul projet et permet de m'apporter une meilleure expérience technique sur ces modules.

1.2 Objectifs

L'objectif de ce projet est de rallier mes différentes connaissances en un seul jeu. Pour cela, le jeu devra contenir différents critères selon le module, les voici :

UX : Maquettes du menu principal, de l'écran de jeu, et du meilleur score.

Programmation : Différents niveaux contenant un joueur et des ennemis avec un certain nombre de points de vie qui peuvent se déplacer et tirer. Il devra également y avoir des obstacles permettant au joueur d'esquiver les balles ennemies. L'objectif est donc de connaître la programmation orientée objets.

Ce projet sera réalisé selon un thème orienté militaire choisi au préalable.

1.3 Gestion de projet

Pour la gestion de ce projet, nous avons recours à différents outils afin de pouvoir s'organiser convenablement. En premier temps nous utilisons GitHub qui nous permet de sauvegarder notre travail au fur et à mesure et d'avoir un suivi de tout le projet. Nous utilisons également IceScrum qui nous permet de planifier des tâches et qui sert également à avoir un journal de travail. Concernant la documentation, tout se trouvera dans ce rapport.

2 Analyse / Conception

2.1 Gameplay

2.1.1 Le joueur :

Le joueur aura la capacité de se déplacer sur l'axe horizontal en utilisant les touches 'A' et 'D'. Ceci lui permettra d'esquiver les différentes balles ennemies. De plus le joueur pourra tirer en appuyant sur le clic gauche de la souris. Une munition partira de son arme sur l'axe vertical avec une trajectoire linéaire. Cependant, le joueur devra attendre un délai de 200 millisecondes avant de pouvoir tirer à nouveau.

2.1.2 Les ennemis :

Les ennemis apparaîtront automatiquement au début de la partie, ils se situeront dans la partie supérieure de l'écran. Les ennemis seront sous forme de tank et leur objectif sera d'éliminer le joueur. Pour se faire, le tank pourra envoyer des missiles. A la différence du joueur, le tank ne pourra pas se déplacer, cependant il aura la capacité de nous viser et donc de tirer en diagonal.

2.1.3 Les protections

Le joueur aura la capacité de se protéger des missiles ennemis en plaçant des protections. Pour ce faire, il lui suffira d'effectuer un clic droit, et une protection se placera là où son curseur se trouve. Il pourra en poser deux dès le lancement de la partie, puis une toutes les 20 secondes.

2.1.4 Les points de vie

Chaque entité aura des points de vie, le joueur aura 3 vies, et les ennemis 2, à chaque fois qu'ils recevront un missile, ils en perdront une. Les protections auront elles 5 points de vies, rendant donc les structures plus solides. Lorsqu'une entité atteint 0 point de vie, elle disparaît, si c'est le joueur, la partie se termine.

2.1.5 Les niveaux

Le jeu comportera différents niveaux, le niveau 1 comportera deux tanks, dès que le joueur les auras éliminer, le niveau suivant commencera avec cette fois plus de tank. A partir de 8 tanks, les ennemis réapparaissent infiniment.

2.2 Concept

Un diagramme UML est un schéma visuel permettant de visualiser la structure et le comportement du logiciel. Mon diagramme UML, étant trop grand, ne rentrait pas dans le rapport. Il peut être visualisé en cliquant [ici](#) ou dans le dossier P_ShootMeUp/POO/docs/diagrammeUML.pdf

2.3 Analyse fonctionnelle

2.3.1 Sprites

(Auteur: Mathis Olaya)

En tant que joueur, je souhaite avoir différents sprites, afin d'embellir le jeu.	
Tests d'acceptance:	
Joueur	Durant la partie, quand je regarde mon joueur, c'est un militaire avec une arme.
Background	Durant la partie, quand je regarde le fond, c'est une map (rue)
Protections	Depuis la partie, quand je regarde les protections ce sont des sacs de sable avec des barbelés.
Points de vie du joueur	Depuis la partie, quand je regarde les points du vie du joueur, ce sont des casques militaires pour chaque HP.
Point de vie du tank	Depuis la partie, quand je regarde le tank je vois au dessus, qu'il a un sprite qui montre ses vies sous forme de LifeBar
Chargeur	Depuis le jeu, quand je regarde les munitions restantes, il y a une icone de munitions à coté du nombre de balles restantes.

2.3.2 Niveaux

(Auteur: Mathis Olaya)

En tant que joueur je souhaite pouvoir jouer a différents niveaux, afin de pouvoir éviter une répétition.	
Tests d'acceptance:	
Niveau suivant	Depuis la partie, quand je fini le niveau actuel, je passe automatiquement au niveau suivants
Affichage	Depuis la partie, quand je regarde en haut a gauche, je vois le niveau actuel
Difficulté	Dans le jeu, le premier niveau contient 2 tanks, puis a chaque niveau + 2 tanks. Des qu'il y a 8 tanks, le jeu passe en mode infini avec un tank qui répareit a chaque mort

2.3.3 Obstacles

(Auteur: Mathis Olaya)

En tant que joueur je souhaite pouvoir avoir des obstacles, afin de me protéger des missiles ennemis.	
Tests d'acceptance:	

Position	Pendant la partie, quand je clic droit je peux poser des protections ou je clique dans une limite ou les tanks ni le joueur ne peuvent entrer. (ex: impossible de poser une protection derrière le joueur)
Vie	Depuis la partie, quand je regarde les obstacles ils peuvent recevoir 5 balles avant de disparaître
Protection	Depuis la partie, quand un obstacle reçoit une munition, la munition se supprime et l'obstacle perd un point de vie
Limite	Dans la partie, je peux poser au une protections toutes les 20 secondes
Superposition	Dans la partie, quand j'essaye de superposer deux protections, la nouvelle ne se place pas

2.3.4 Highscores

(Auteur: Mathis Olaya)

En tant que joueur quand je fais ma meilleure partie je peux le revoir afin de sauvegarder mes performances

Tests d'acceptance:

Record Quand je fini une partie, mon score est sauvegardé dans un fichier texte
Mort Quand je meurs, le HiScore s'affiche avec le score actuelle

2.3.5 Ennemi

(Auteur: Mathis Olaya)

En tant que joueur je souhaite combattre contre des ennemis afin d'ajouter de la difficulté au jeu

Tests d'acceptance:

Lancement	Au lancement du jeu, un nombre X d'ennemi apparaissent en dehors de l'écran et avance jusqu'à une position fixe à une certaine position X, Y. X et choisi semi-aléatoirement selon une liste prédéfinie et Y est 175.
Déplacement	Durant la partie, quand je regarde les ennemis, ils ne peuvent pas se déplacer une fois leur position finale atteinte.
Tire	Depuis le jeu, quand je regarde les ennemis ils peuvent me tirer dessus en visant ma position.
Délai tire	Depuis le jeu quand je regarde les ennemis, ils doivent attendre un délai de 2.2s avant de pouvoir retirer
Sprites	Durant la partie, quand je regarde les ennemis, ce sont des tanks.
Points de vie	Dans le jeu, quand je regarde au-dessus des ennemis, ils ont une barre de HP (ils résistent à 2 tirs).
Dégats	Dans le jeu, quand je touche un ennemi avec une balle, il perd des points de vie.
Mort	Dans le jeu quand un ennemi n'a plus de point de vie, il meurt et disparaît

2.3.6 Joueur

(Auteur: Mathis Olaya)

En tant que joueur, je souhaite avoir un personnage a déplacer, pour interagir avec le jeu

Tests d'acceptance:

Déplacement	Depuis le jeu, en appuyant sur certaines touches, mon joueur se déplace sur l'axe X.
Touches de déplacement	Dans le jeu, quand je clique sur A,D mon personnage se déplace sur un côté.
Limite	Depuis le jeu quand je me déplace je ne peux pas sortir de la bordure de l'écran avec une marge de 50.
Tir	Depuis le jeu, quand je clique sur le bouton gauche de ma souris, le joueur tire une munition qui part tout droit (verticalement)
Délai	Dans le jeu, après avoir tiré, je ne peux plus tirer pendant 200 millisecondes.
Munitions	Dans le jeu, quand je regarde en bas, j'ai une limite de 15 munitions.
Chargement de l'arme	Depuis le jeu, quand le joueur appuie sur 'R' l'arme se recharge durant 1.5 seconde et le joueur ne peut plus tirer.
Calcul de trajectoire de la munition	Depuis le jeu quand une balle sort de l'écran elle se supprime.
Points de vie	Depuis la partie mon joueur a 3 de points de vie et ceux-ci peuvent être vérifié en regardant en bas à gauche de l'écran
Dégâts personnels	Dans le jeu, quand je me fais toucher par une balle, je perd des points de vie.
Mort	Dans le jeu quand je n'ai plus de points de vie je meurt et la partie se finit

2.3.7 Lancement

(Auteur: Mathis Olaya)

En tant que joueur je souhaite pouvoir lancer le jeu afin de pouvoir y jouer	
Tests d'acceptance:	
Menu	Depuis le bureau, quand je lance le jeu j'arrive sur un écran de menu avec un Bouton start et exit
Bouton	Depuis le menu quand je clique sur le bouton start le niveau se lance avec le joueur, ennemi, et la possibilité de poser des protections
Tank	Dans le jeu quand les ennemis apparaissent ils ne se chevauchent pas.
Relancer	A la fin de la partie, quand j'appuie sur restart, je peux rejouer une partie
Interface	Au lancement du jeu, il y a une interface fonctionnel avec les boutons jouer, et quitter
Pause	Dans la partie, quand le joueur appuie sur ESC, un menu de pause s'ouvre

2.3.8 Collisions

(Auteur: Mathis Olaya)

En tant que joueur je souhaite avoir des collisions afin de pouvoir gagner ou perdre la partie	
Tests d'acceptance:	
Tank	Dans le jeu quand une munitions rentre dans un tank le tank perd une vie
Protections	Dans le jeu, quand une munitions allié ou adverse rentre dans une protection elle perd une vie.
Munitions	Dans le jeu quand une munitions rentre en contact avec un tank, un joueur ou une protection, elle se supprime

Joueur	Dans le jeu, quand le joueur se fait toucher par un missile ennemi, il perd une vie
--------	---

2.3.9 Score

(Auteur: Mathis Olaya)

En tant que joueur, je souhaite avoir des scores, pour voir mon niveau	
Tests d'acceptance:	
Position	Dans le jeu, quand je regarde en bas à droite je vois le score
Calcul	Dans le jeu quand je vois le score, je remarque qu'il s'incrémente uniquement de 5 pts lorsque je tue un tank.
Fin	Dans le jeu, quand je meurs, je vois en gros et rouge mon score.

2.4 Stratégie de test

Pour vérifier le bon fonctionnement de mon jeu vidéo, je vais effectuer diverses séries de test qui me permettront d'évaluer les améliorations à effectuer. En premier temps je vais vérifier les tests d'acceptances IceScrum. Je vais les reprendre un par un et vérifier que le travail effectué respectent bien les tests créés au préalable.

Ensuite, je vais créer une solution de test (MSTest) sur mon projet, et y créer différents tests sur des parties spécifique du jeu. Les tests s'exécuteront par la suite automatiquement.

Et finalement, je vais faire tester mon jeu à plusieurs camarades, pour avoir leur ressenti, et vérifier qu'il n'y a pas de bugs.

Toute cette stratégie de test me permet de vérifier différents aspects du jeu, le contenu est respecté, le jeu est fonctionnel, etc... Cependant il est tout de même très compliqué d'avoir une couverture de test complète (dans MSTest) pour quelques raisons. Déjà, effectuer des tests pour chaque partie du code prendrait beaucoup de temps. De plus certaines parties sont très compliquées à tester. Exemple :

- Imaginons que nous souhaitons tester le fait que quand le joueur appuie sur A, le joueur se déplace sur la gauche. Et bien, cela est en fait compliqué (dans mon cas), car je serais obligé de simuler l'appui de la touche 'A', ce qui est plus ou moins compliqué.

3 UX


3.1 Analyse de l'UX

3.1.1 Personas

Ce chapitre concerne l'analyse de l'expérience utilisateur. En premier temps, il a fallu concevoir deux Personas. Ceux-ci sont des personnages fictifs créés afin de représenter différents types d'utilisateurs qui pourraient utiliser notre produit. Ils aident à mieux comprendre les besoins, les comportements et les frustrations des utilisateurs cibles. Les voici :

Persona

Mathéo Dubois



Age : 15 ans
Métier : Ecolier
Famille : Célibataire
Location : Bruxelles, Belgique

Expérience jeux vidéo : ● ● ● ● ●
Fréquence d'usage : ● ● ● ● ●
Capacité à s'adapter aux nouvelles technologies : ● ● ● ● ●

Histoire

Mathéo Dubois, 15 ans, a grandi à Bruxelles, en Belgique. Il est un écolier assidu et persévérant. Dès son plus jeune âge, il a été captivé par les jeux vidéo et les nouvelles technologies, qui ont rythmé son quotidien. Passionné et curieux, il explore une grande variété de jeux, toujours à la recherche de nouveaux défis et d'aventures virtuelles qui nourrissent son imagination et affûtent ses compétences.


But

- Réaliser le meilleur score
- Faire passer le temps
- Découvrir un nouveau type de jeu

Personnalité

- Compétiteur
- Mauvais perdant
- Curieux


Applications



Ce premier persona est orienté en tant qu'un joueur aguerri, il a de l'expérience dans le domaine des jeux-vidéos et y joue régulièrement. Ses objectifs sont précis, devenir le meilleur.

Persona

Sophie Mallet



Age : 26 ans
Métier : Assistante dentaire
Famille : Divorcée
Location : Bordeaux, France

Expérience jeux vidéo : ● ● ● ● ●
Fréquence d'usage : ● ● ● ● ●
Capacité à s'adapter aux nouvelles technologies : ● ● ● ● ●

Histoire

Sophie Mallet, 26 ans, est assistante dentaire à Bordeaux. Bien qu'elle ne soit pas une grande passionnée des jeux vidéo, elle s'y aventure de temps en temps afin de s'y divertir depuis quelques années. Ce sont ses neveux qui lui ont donné envie de jouer aux jeux vidéos.


But

- Se divertir.
- Battre ses neveux.
- Réaliser le meilleur score

Personnalité

- Joyeuse
- Généreuse

Applications



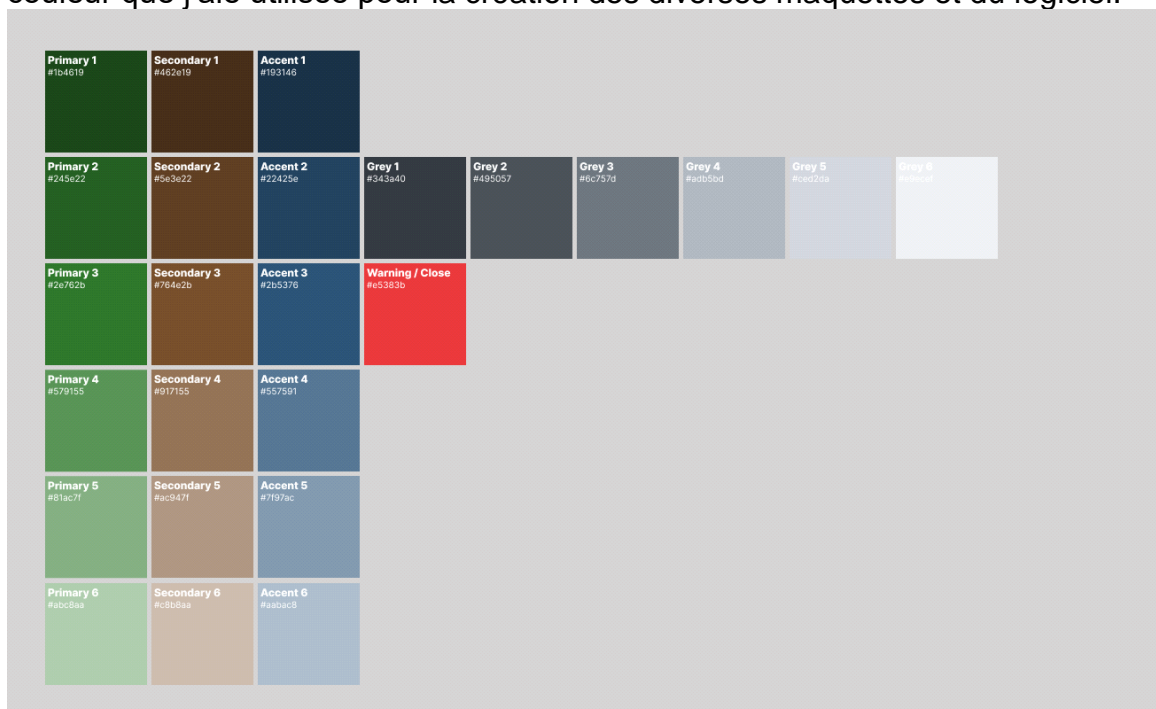
Ce second persona est bien différent du premier, Sophie est une amatrice des jeux vidéo et n'y joue que de temps à autres. Ses objectifs sont tout à fait

différents car elle souhaite uniquement se divertir, et non pas de jouer en compétition.

Ces différents critères permettent donc d'adapter le jeu en fonction de leurs objectifs et personnalités.

3.1.2 Palette graphique

Pour la conception graphique de ce jeu, j'ai dû choisir une palette de couleur qui représentait correctement le thème de TankBattle. Il est également important de choisir des couleurs ayant bien ensemble. Voici la palette de couleur que j'ai utilisée pour la création des diverses maquettes et du logiciel.



On y retrouve différentes variétés de couleurs. La principale est la couleur verte, elle est la couleur qui sera le plus souvent utilisée. Elle représente totalement le thème militaire du jeu. La couleur secondaire est utilisée moins fréquemment, principalement pour les couleurs de fonds. Elle se marie bien avec le vert. Et finalement la couleur d'accent est bleue. Elle permet donc d'accentuer divers éléments sur l'interface et est utilisée avec parcimonie. On y retrouve ensuite une variété de gris, celle-ci se trouve par défaut sur les palettes et est, pour mon cas, principalement utilisées pour la couleur des textes. Et en dernier, on trouve la couleur rouge qui est unique, elle est simplement utilisée pour les boutons « Fermer » ou les messages d'alertes.

3.1.3 Eco-conception et accessibilité

L'éco-conception est le fait de rendre son interface la plus économe possible, pour ce faire, une des pratiques courantes est d'utiliser des couleurs sombres afin de réduire la luminosité et donc de réduire la consommation d'énergie. J'ai

également créé une interface sobre et épurée en utilisant la typographie de base et en limitant les éléments inutiles.

L'accessibilité, elle, est le fait d'avoir une interface simple, compréhensible et utilisable par tous types de personnes (avec ou sans handicap). Pour ce faire, j'ai différencié chaque objet avec un contraste afin de permettre à l'utilisateur de correctement différencier chaque objet.

J'ai donc mélangé l'éco-conception et l'accessibilité afin de produire une interface économe tout en restant accessible par n'importe qui.

3.2 Conception de l'UX

Ce chapitre résume la conception des maquettes des différents écrans du jeu. Il se résume en 3 chapitres, les Wireframes, l'éditeur de niveau en haute-fidélité et les choix effectués.

3.2.1 Définitions des Wireframes

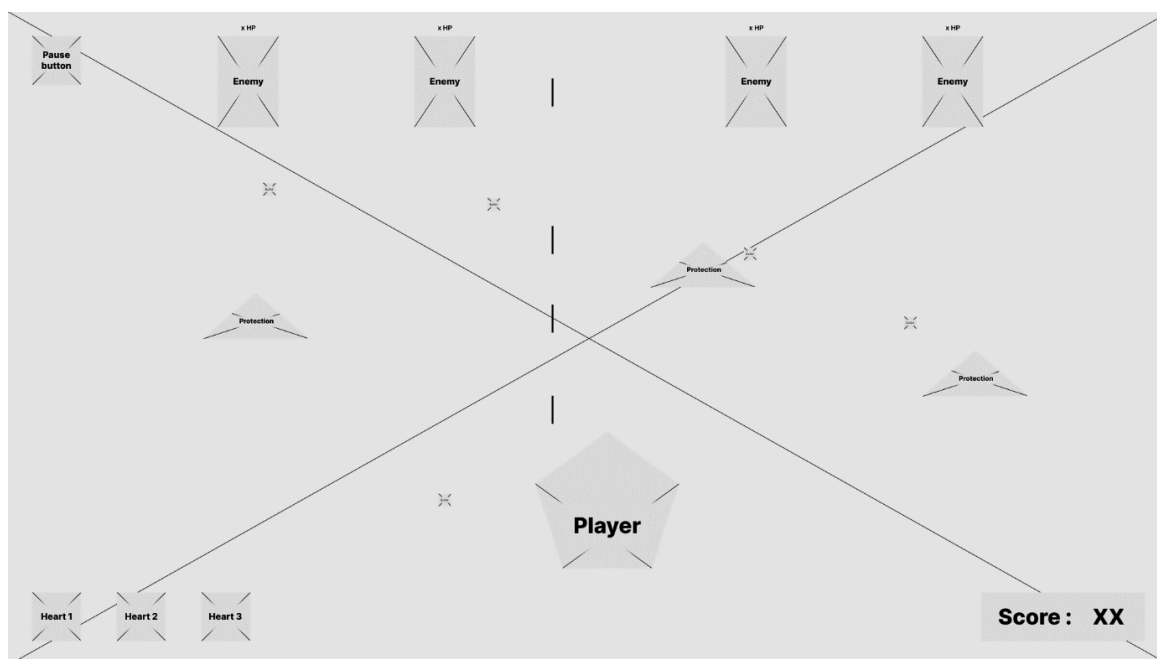
Un wireframe est une représentation simplifiée d'une interface en utilisant uniquement des lignes pour décrire la structure de l'interface. Il ne comporte aucune couleur et aucune « données ». Il sert souvent à visualiser la disposition d'un design. Ci-dessous se trouvera donc tous les Wireframes demandés :

- Menu principal
- Ecran de jeu
- Editeur de niveau
- High scores

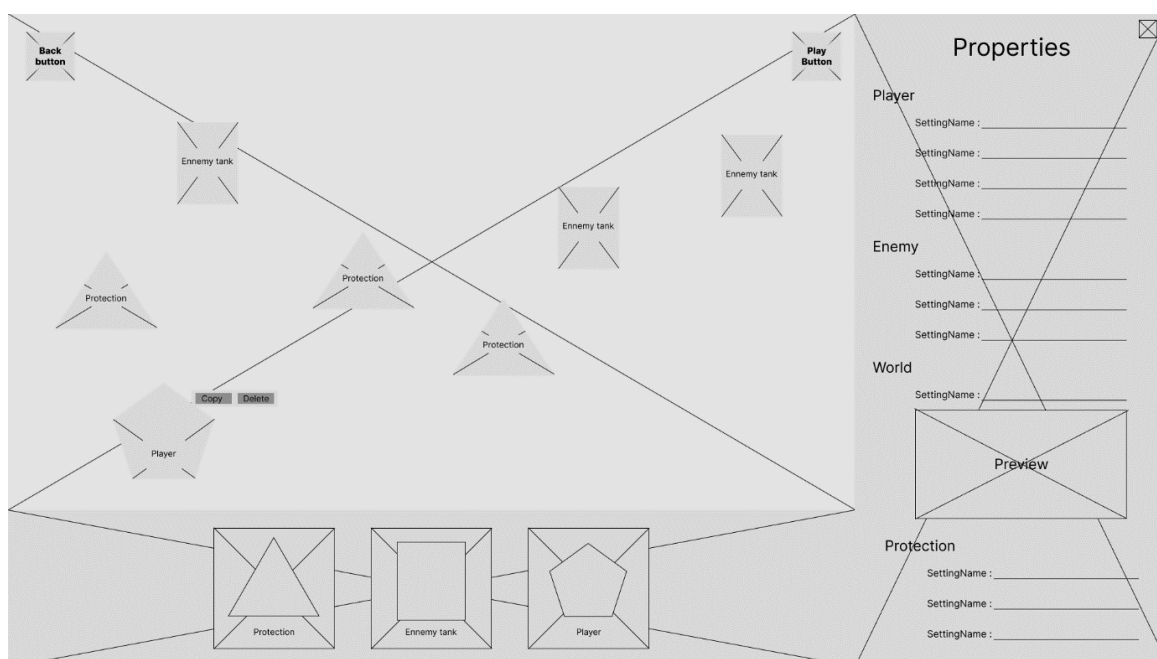
Les voici :



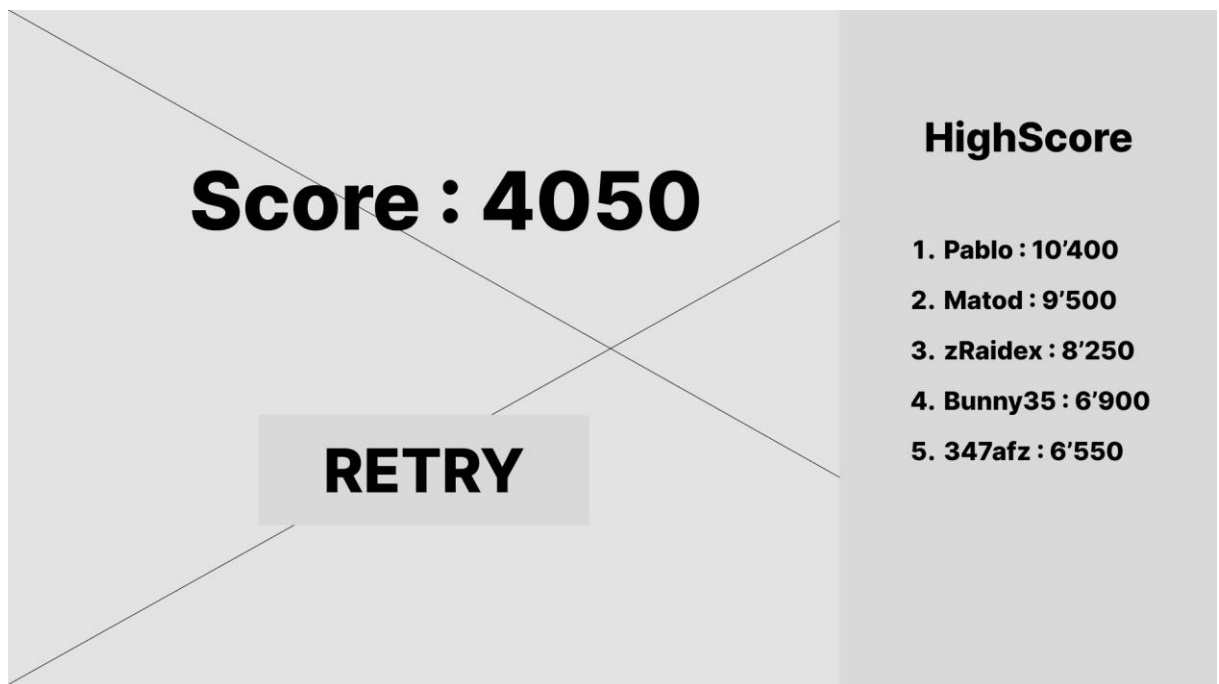
Maquette 1: Menu affiché lorsque le jeu se lance



Maquette 2: Affiche durant une partie



Maquette 3: Affichage de l'éditeur de niveau



Maquette 4: Scène lorsque le joueur est mort

3.2.2 Définition de l'High fidelity

Une maquette hautefidélité est une maquette avancée du Wireframe, il comporte des couleurs, des polices et ressemble au rendu final de l'interface. Cependant l'interface n'est pas cliquable et/ou utilisable. Cependant, pour cette interface, il nous était demandé uniquement l'interface de l'éditeur de niveau.

Pour le choix de la map, l'utilisateur peut dérouler un drop down qui affiche une liste de map prédéfinies. L'utilisateur peut en choisir une, la map actuelle est écrite sur le drop down, et un aperçu est affiché juste en dessous.



3.3 Choix effectués

Pour ces différentes maquettes, j'ai donc décidé de partir sur un design sobre, simple et efficace. Tout en rendant l'interface lisible et compréhensible pour tout type de personne en mélangeant correctement les couleurs préalablement choisies sur la palette. J'ai également gardé la police d'écriture de base car elle est très facilement lisible. La palette a été choisie afin de respecter le thème militaire du jeu.

4 Réalisation

4.1 Points de design spécifiques

4.1.1 Placement de protections spécifique.

Lorsque le joueur essaye de placer une protection, il y a une multitude de conditions qui sont vérifiées :

- La protection se trouve dans la zone de placement ?
- Le joueur dispose de protections ?
- La protection ne superpose pas une autre déjà existante ?

Pour ce faire voici l'ordre de l'exécution du code.

- a. Vérifier que le joueur appuie sur le clic droit tout en vérifiant qu'il ait relâché le clic précédent. Cela permet d'éviter qu'en un clic, il pose plusieurs protections. Et il faut également vérifier que la protection se trouve dans les limites (entre les tanks et le joueur). Pour calculer cette zone, voici le calcul :
Créer un rectangle, sa coordonnée X est 0 car il peut poser de gauche à droite. Puis calculer sa position Y de départ. Nous savons que la limite doit se situer plus loin que le tank. Donc, on prend la position (sa position se trouve au centre de la texture) limite du tank, et on y ajoute la moitié de la hauteur de la texture du tank. Grâce à cela, la limite commencera au bout du canon du tank ! Ensuite, la largeur est tout l'écran (donc 1920). Et finalement la hauteur est un calcul complexe. Nous savons que le rectangle de la limite acceptée se finit devant le bout du fusil du joueur, donc la hauteur de l'écran (1080) - la zone supprimée en haut (car sinon le rectangle est décalé étant donné qu'on l'a abaissé.) - la hauteur du joueur. Tout cela nous donne une zone délimitant au pixel près la limite où les protections peuvent être placées.
- b. Créer une hitbox pour le rectangle dont la position est centrée sur le curseur de la souris, avec une hauteur et largeur définie par la texture.

Pour calculer la position, il faut utiliser : Position X du curseur - (Largeur de la protection / 2 * échelle de la texture), et faire de même pour Y.

- c. Après cela, il faut vérifier le mode de constructions. Au lancement le joueur peut poser 2 protections sans devoir attendre, puis après cela, une toutes les 20 secondes. Donc le programme vérifie s'il a déjà posé 2 constructions (la limite), si non, le programme place une protection (la logique de placement est expliquée plus loin). Puis incrémente d'un le nombre de protections posées. S'il à poser 2 protections, on passe le mode construction limitant le positionnement de protections à toutes les 20 secondes.
- d. La partie précédente concernait donc le mode placement de deux protections. Mais dans le cas inverse, si le joueur a déjà posé ces deux protections, on vérifie si le joueur a attendu assez de temps pour en placer une nouvelle (20 secondes). Si oui, alors on place une construction. A noter que dans la méthode update, on vérifie si le mode est en mode placement unique, et si oui, on augmente le timer.
- e. Pour placer les protections, une autre logique est mise en place, il faut vérifier que la nouvelle protection ne se superpose pas avec les autres protections. Le programme détient une liste de toutes les protections actuellement en vie. En premier temps on vérifie si dans toute la liste, notre protection intersecte avec une autre protection existante. Si oui, alors on arrête le tout. Une fois que toute la liste a été vérifiée, et que celle-ci n'a pas été interrompue, alors cela signifie que la protection ne se superposera pas avec une existante. Nous plaçons donc la protection.

4.1.2 Faire supprimer les munitions lorsqu'elles sortent de l'écran

Lorsque les munitions sont tirées, il se peut qu'elles sortent de l'écran. Il est primordial de les supprimer afin d'éviter que le programme les calcule sans cesse. Cela peut causer des problèmes techniques s'il y a beaucoup de munitions. Pour chaque munition, la méthode Update() s'effectue à chaque frames. Le programme vérifie donc à chaque tic s'il la balle est en dehors de l'écran.

- a) La méthode CheckLimitPosition() s'effectue
- b) Cette méthode contient :

```
if(isOutOfBounds)  
    EntityManager.Remove(this);
```

- c) isOutOfBounds est une propriété de type bool qui retourne true si $\text{Position.Y} < 0 \parallel > \text{Position.Y} > \text{LaHauteurDeLEcran}$. Le '||' est là car les munitions vont dans les sens (haut et bas), car il y a les munitions alliées et ennemies.
- d) Si le résultat est true, alors la méthode Remove s'effectue ayant pour objet la munition. Elle est ensuite retirée de la liste.

4.1.3 Menu pause

Le Menu pause est unique est entièrement créer par moi-même. Tout simplement car sur MonoGame, il n'est pas possible de créer des boutons. J'ai donc dû les créer moi-même.

- a) Créer un bouton, pour ce faire j'ai créé une hitbox qui définit la zone et la surface du bouton. Dans le menu, chaque bouton est ajouté à un dictionnaire ayant pour valeur un booléen. Celui-ci est vrai si l'utilisateur survole le bouton sans forcément y cliquer.
- b) Lorsqu'un menu est créé, il faut fournir différents paramètres permettant le bon fonctionnement. Le premier est une liste d'action, il permet de fournir au menu toutes les actions qu'il devra contenir. Les actions sont déclarées comme ceci : (*commentaire volontairement supprimé ici pour alléger l'affichage)

```
public enum Action
{
    Resume,
    Start,
    Exit,
    Restart,
}
```

Le second paramètre est une spriteFont qui définira la police d'écriture. A noter que des calculs seront automatiquement effectués pour centrer toute l'interface en fonction du nombre de bouton fourni en paramètre.

- c) Une fois le menu créé, le programme va regarder si le curseur du joueur se trouve au-dessus d'un bouton, si oui on change le booléen en true. Sinon on le met en faux. Et lors du dessin, le programme changera la couleur de fond en fonction du isHover. Ensuite si l'utilisateur effectue un clic gauche. On effectue la méthode OnClick ayant comme paramètre l'action du bouton chargée au préalable. A noter que la liste des boutons[0] est égale à la liste d'action[0]. Dans ce sens, nous pouvons retrouver l'action par rapport à quel bouton cliqué.

```
for (int i = 0; i < Buttons.Count; i++)
{
    //Regarder si le curseur est sur le bouton
    if(Buttons.Keys.ElementAt(i).Contains(GlobalHelpers.Input.GetMousePosition()))
    {
        //Si le curseur se trouve par dessus, alors changer la valeur en true pour le bouton correspondant.
        Buttons[Buttons.Keys.ElementAt(i)] = true;

        //Si en plus il clique, effectuer l'action.
        if (GlobalHelpers.Input.isLeftClicking())
        {
            OnClick(ButtonActionTitle[i]);
        }
    }
    else
    {
        //Sinon elle est fausse
        Buttons[Buttons.Keys.ElementAt(i)] = false;
    }
}
```

- d) Si le joueur clique effectivement sur la surface, on effectue un switch(action), contenant des 'case Action.XXX' avec chaque action créer dans l'enum. De cette façon, chaque Action possède une

partie de code, et nous pouvons effectuer le code affecter au bouton !

```
Switch(action)
{
    Case Action.Exit :
        Environment.Exit(0);
}
```

Ceci est un exemple du switch.

- e) Conclusion :** Tout ce code me permet maintenant d'ajouter des boutons et des actions très rapidement. Pour créer une nouvelle interface il suffirait simplement de faire :

```
menu = new Menu(new List<Action>{Action.Resume, Action.Restart, Action.Exit}, spriteFont);
```

Et / ou pour créer des actions il me suffirait d'ajouter le nom dans l'enum. Et d'y ajouter le code correspondant dans le switch !

4.2 Déroulement

Le déroulement des story s'est globalement bien passé, cependant j'ai eu quelques problèmes, notamment dans la réalisation de l'apparition des tanks.

Le problème était que certains tanks se superposaient lors de l'apparition. Au départ, j'ai essayé de comparer la position de tous les tanks entre-deux, et d'en générer un ne croisant pas avec les autres tanks déjà existants. Mais cela posait un problème, car le programme essayait de générer une position aléatoire valable au minimum 100 fois avant de comprendre qu'aucune position n'étaient disponibles. Donc à partir d'un certain nombre de tank, cela faisait crasher le jeu et/ou pouvait provoquer des longs délais d'attente. Pour régler ce problème, j'ai créé un dictionnaire contenant comme clés des positions, et comme valeurs des booléens disant si la position est occupée ou non. Au lancement, le programme regarde chaque position disponible grâce au booléen, et choisi aléatoirement une position parmi celle-ci.

Sinon, mise à part celle-ci, toutes les autres story se sont déroulées sans accros.

4.3 Mise en place de l'environnement de travail

4.3.1 Installation des logiciels

1) Visual Studio

Se rendre sur le site de [Visual Studio](#), et installer la version Community, qui ne demande pas de licence.

2) Monogame

Il est important d'installer le moteur de jeu, pour cela lancer Visual Studio, dans l'onglet en haut de l'écran, sélectionner extensions, puis gérer les extensions. Et dans la barre de recherche, taper « Monogame ». Installez-le et redémarrez VS.

3) GitHub Desktop

GitHub desktop s'avère être très utile, et permet de récupérer le travail à jour en quelque clic. Pour cela, il faut se rendre sur le site de Github, et d'y

installer la version Desktop. Connectez-vous à votre compte, et cloner le repos contenant la solution.

4.3.2 Environnement

Pour accéder au code source du projet, il suffit de se rendre sur [GitHub](#), de cloner le projet, puis d'ouvrir la solution à : « POO/TankBattleV2/TankBattleV2.sln ». A partir de là vous aurez accès à toutes les classes et donc tout le code.

La version de mon système d'exploitation est Windows 10 22H2 19045.4894. J'ai également utilisé Visual Studio, dont la version est 17.9.4.

Pour le matériel, un simple PC de bureautique avec un Intel® Core™ i7-11700 de 2.50Ghz, 32 GO de RAM, 512 GO de stockage, et un intel® UHD Graphics 750 a été utilisé.

4.3.3 Liste des fichiers

Ci-dessous se trouve la liste complète des fichiers du jeu. En premier temps se trouve les classes, puis les autres fichiers.

Classe :

Bullet.cs : Configuration et calcul des tirs ennemis et alliés.

Player.cs : Logique de déplacement, placement de protection, tir, du joueur.

Protection.cs : Configuration et dessin de la protection

Tank.cs : Logique de tir automatique, déplacement automatique

Entity.cs : Modèle pour toutes les entités avec des propriétés définies (Tout objet ayant une position, point de vie etc..)

IEntity.cs : Défini toutes les méthodes qu'une entité doit inclure

IMovable.cs : Défini toutes les propriétés qu'une entité ayant la capacité de bouger doit inclure

IShootable.cs : Défini toutes les propriétés qu'une entité ayant la capacité de tirer doit inclure

EntityManager.cs : S'occupe de gérer toutes les entités (Update, Draw, Suppression, Collision, etc...)

Level.cs : S'occupe de lancer la partie, et d'initier les entités. Avec un nombre de tank prédéfinis.

Config.cs : Gère la configuration de l'écran et les paramètres de la partie.

EntityConfig.cs : Défini toutes les caractéristiques de chaque entité (ex : vitesse, cooldown de tir, etc...)

GlobalHelpers.cs : Gère les entrées de l'utilisateur et la création de nombre aléatoire.

Visuals.cs : S'occupe de charger tout les aspects visuels du jeu au lancement.

Buttons.cs : Permet la création de bouton.

Menu.cs : Permet la création d'une interface en utilisant des boutons.

Program.cs : S'occupe de créer GameRoot.

GameRoot.cs : Fichier par défaut, gérant le jeu.

Autres :

*dossier Content : Stock tous les sprites du jeu.

app.manifest : Fichier de configuration de l'application.

TankBattleV2.sln : Solution du projet.

4.4 Description des tests effectués

4.4.1 Sprint 1

4.4.1.1 Collisions

Tank	Dans le jeu quand une munitions rentre dans un tank le tank perd une vie	OK 30 Oct
Protections	Dans le jeu, quand une munitions allié ou adverse rentre dans une protection elle perd une vie.	OK 30 Oct
Munitions	Dans le jeu quand une munitions rentre en contact avec un tank, un joueur ou une protection, elle se supprime	OK 30 Oct
Joueur	Dans le jeu, quand le joueur se fait toucher par un missile ennemi, il perd une vie	OK 30 Oct

4.4.1.2 Lancement

Menu	Depuis le bureau, quand je lance le jeu j'arrive sur un écran de menu avec un Bouton start et exit	OK 30 Oct
Bouton	Depuis le menu quand je clique sur le bouton start le niveau se lance avec le joueur, ennemi, et la possibilité de poser des protections	OK 30 Oct
Tank	Dans le jeu quand les ennemis apparaissent ils ne se chevauchent pas.	OK 30 Oct
Relancer	A la fin de la partie, quand j'appuie sur restart, je peux rejouer une partie	OK 30 Oct
Interface	Au lancement du jeu, il y a une interface fonctionnel avec les boutons jouer, et quitter	OK 30 Oct
Pause	Dans la partie, quand le joueur appuie sur ESC, un menu de pause s'ouvre	OK 30 Oct

4.4.1.3 Joueur

Déplacement	Depuis le jeu, en appuyant sur certaines touches, mon joueur se déplace sur l'axe X.	OK 18 Sep
-------------	--	-----------------

Touches de déplacement	Dans le jeu, quand je clique sur A,D mon personnage se déplace sur un côté.	OK 18 Sep
Limite	Depuis le jeu quand je me déplace je ne peux pas sortir de la bordure de l'écran avec une marge de 50.	OK 18 Sep
Tir	Depuis le jeu, quand je clique sur le bouton gauche de ma souris, le joueur tire une munition qui part tout droit (verticalement)	OK 18 Sep
Délai	Dans le jeu, après avoir tiré, je ne peux plus tirer pendant 200 millisecondes.	OK 18 Sep
Munitions	Dans le jeu, quand je regarde en bas, j'ai une limite de 15 munitions.	OK 30 Oct
Chargement de l'arme	Depuis le jeu, quand le joueur appuie sur 'R' l'arme se recharge durant 1.5 seconde et le joueur ne peut plus tirer.	OK 30 Oct
Calcul de trajectoire de la munition	Depuis le jeu quand une balle sort de l'écran elle se supprime.	OK 18 Sep
Points de vie	Depuis la partie mon joueur a 3 de points de vie et ceux-ci peuvent être vérifié en regardant en bas à gauche de l'écran	OK 2 Oct
Dégâts personnels	Dans le jeu, quand je me fais toucher par une balle, je perd des points de vie.	OK 2 Oct
Mort	Dans le jeu quand je n'ai plus de points de vie je meurt et la partie se finit	OK 2 Oct

4.4.1.4 Ennemi

Lancement	Au lancement du jeu, un nombre X d'ennemi apparaissent en dehors de l'écran et avance jusqu'à une position fixe à une certaine position X, Y. X et choisi semi-aléatoirement selon une liste prédéfinie et Y est 175.	OK 30 Oct
Déplacement	Durant la partie, quand je regarde les ennemis, ils ne peuvent pas se déplacer une fois leur position finale atteinte.	OK 30 Oct
Tire	Depuis le jeu, quand je regarde les ennemis ils peuvent me tirer dessus en visant ma position.	OK 2 Oct
Délai tire	Depuis le jeu quand je regarde les ennemis, ils doivent attendre un délai de 2.2s avant de pouvoir retirer	OK 30 Oct
Sprites	Durant la partie, quand je regarde les ennemis, ce sont des tanks.	OK 2 Oct

Points de vie	Dans le jeu, quand je regarde au-dessus des ennemis, ils ont une barre de HP (ils résistent à 2 tirs).	OK 2 Oct
Dégats	Dans le jeu, quand je touche un ennemi avec une balle, il perd des points de vie.	OK 2 Oct
Mort	Dans le jeu quand un ennemi n'a plus de point de vie, il meurt et disparaît	OK 2 Oct

4.4.1.5 Score

Position	Dans le jeu, quand je regarde en bas à droite je vois le score	OK 30 Oct
Calcul	Dans le jeu quand je vois le score, je remarque qu'il s'incrémente uniquement de 5 pts lorsque je tue un tank.	OK 30 Oct
Fin	Dans le jeu, quand je meurs, je vois en gros et rouge mon score.	OK 30 Oct

4.4.1.6 Obstacles

Position	Pendant la partie, quand je clic droit je peux poser des protections ou je clique dans une limite ou les tanks ni le joueur ne peuvent entrer. (ex: impossible de poser une protection derrière le joueur)	OK 30 Oct
Vie	Depuis la partie, quand je regarde les obstacles ils peuvent recevoir 5 balles avant de disparaître	OK 30 Oct
Protection	Depuis la partie, quand un obstacle reçoit une munition, la munition se supprime et l'obstacle perd un point de vie	OK 30 Oct
Limite	Dans la partie, je peux poser au une protections toutes les 20 secondes	OK 30 Oct
Superposition	Dans la partie, quand j'essaye de superposer deux protections, la nouvelle ne se place pas	OK 30 Oct

4.4.1.7 Niveaux

Niveau suivant	Depuis la partie, quand je fini le niveau actuel, je passe automatiquement au niveau suivants	OK 30 Oct
Affichage	Depuis la partie, quand je regarde en haut a gauche, je vois le niveau actuel	OK 30 Oct
Difficulté	Dans le jeu, le premier niveau contient 2 tanks, puis a chaque niveau + 2 tanks. Des qu'il y a 8 tanks, le jeu passe en mode infini avec un tank qui répareit a chaque mort	OK 30 Oct

4.4.1.8 Sprites

Joueur	Durant la partie, quand je regarde mon joueur, c'est un militaire avec une arme.	OK
--------	--	----

		30 Oct
Background	Durant la partie, quand je regarde le fond, c'est une map (rue)	OK 30 Oct
Protections	Depuis la partie, quand je regarde les protections ce sont des sacs de sable avec des barbelés.	OK 30 Oct
Points de vie du joueur	Depuis la partie, quand je regarde les points de vie du joueur, ce sont des casques militaires pour chaque HP.	OK 30 Oct
Point de vie du tank	Depuis la partie, quand je regarde le tank je vois au dessus, qu'il a un sprite qui montre ses vies sous forme de LifeBar	OK 30 Oct
Chargeur	Depuis le jeu, quand je regarde les munitions restantes, il y a une icone de munitions à coté du nombre de balles restantes.	OK 30 Oct

4.5 Erreurs restantes

Durant toute la réalisation du projet, et également lors des tests, je n'ai pu constater aucunes erreurs. Cependant, avec plus d'heure de jeu, de test, d'analyse, il est possible de trouver de très petites erreurs.

5 Conclusions

Pour conclure, je vais tout résumer en 4 points pertinents :

1. Les objectifs ayant été atteints ou non
2. Les points positifs et négatifs du projet
3. Les différentes difficultés du projet
4. Les améliorations possibles sur le projet

5.1 Objectifs

Je suis satisfais de mon travail, car j'ai réussi à créer un jeu fonctionnel, avec des déplacements, tirs, protections. Les objectifs principaux que je m'étais fixés ont été réalisés avec succès, et tout est fonctionnel. Cependant, avec plus de temps, j'aurai pu ajouter certaines fonctionnalités au jeu, tel qu'un boss final.

5.2 Positifs / négatifs

Ce projet est selon moi positif car il introduit correctement l'utilisation des classes et des interfaces. Et de plus, la réalisation d'un jeu rend le projet plus amusant.

Je n'ai pas particulièrement apprécié l'utilisation d'IceScrum, car créer une structure efficace y prend trop de temps, un temps qui pourrait être mieux investi

dans le développement du code. Cela est particulièrement vrai lorsqu'on utilise une méthode agile en étant seul sur le projet.

5.3 Difficulté du projet

Le projet a été complexe au départ, car je ne connaissais pas le moteur de jeu MonoGame. J'ai donc dû apprendre à l'utiliser et à l'adapter à mon projet. Tout en utilisant les classes et les interfaces, qui étaient des concepts encore inconnus lors de la création de ce projet.

5.4 Améliorations possibles

Si je devais améliorer mon jeu, j'ajouterais certaines fonctionnalités, tels qu'un boss final, ou encore des habilités que le joueur pourrait utiliser.

6 Annexes

6.1 Manuel de référence

Le manuel de référence est accessible depuis le chemin :

« P_ShootMeUp/POO/docs/ManuelDeRéférence.xml ».

6.2 Journal de travail

Le journal de travail est accessible depuis le chemin :

« P_ShootMeUp/POO/docs/ T-POO-MOA-Final.pdf ».

6.3 Utilisation de l'IA

Durant ce projet, j'ai parfois eu recours à l'IA lorsque le problème dépassait mes connaissances, en utilisant principalement ChatGPT. J'ai principalement fait appel à l'IA pour la création de dictionnaires, un concept qui m'était inconnu avant ce projet. Grâce à l'IA, j'ai pu apprendre et comprendre comment les utiliser. À noter que je n'ai jamais utilisé le code proposé par l'intelligence artificielle sans en comprendre la logique.