|  |
| --- |
| TankBattle |

Pour projet Shoot Me Up avec XCL

Table des matières

1 Analyse préliminaire 3

1.1 Introduction 3

1.2 Objectifs 3

1.3 Gestion de projet 4

2 Analyse / Conception 4

2.1 Gameplay 4

2.2 Analyse de l’UX 5

2.3 Conception de l’UX 6

2.4 Concept 6

2.5 Analyse fonctionnelle 6

2.6 Stratégie de test 6

3 Réalisation 7

3.1 Points de design spécifiques 7

*3.1.1* *…* 7

*3.1.2* *…* 8

*3.1.3* *…* 8

3.2 Déroulement 8

3.3 Mise en place de l’environnement de travail 8

3.4 Description des tests effectués 8

3.5 Erreurs restantes 8

4 Conclusions 8

5 Annexes 9

5.1 Manuel de référence 9

5.2 Journal de travail 9

*NOTE L’INTENTION DES UTILISATEURS DE CE CANEVAS :*  
*Toutes les parties en italiques sont là pour aider à comprendre ce qu’il faut mettre dans cette partie du document. Elles n’ont donc aucune raison d’être dans le document final.*

*De plus, en fonction du type de projet, il est tout à fait possible que certains chapitres ou paragraphes n’aient aucun sens. Dans ce cas il est recommandé de les retirer du document pour éviter de l’alourdir inutilement.*

**Analyse préliminaire**

**Introduction**

TankBattle est un jeu de combat type Shoot em 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.

*Ce chapitre décrit brièvement le projet, le cadre dans lequel il est réalisé, les raisons de ce choix et ce qu'il peut apporter à l'élève ou à l'école. Il n'est pas nécessaire de rentrer dans les détails (ceux-ci seront abordés plus loin) mais cela doit être aussi clair et complet que possible (idées de solutions). Ce chapitre contient également l'inventaire et la description des travaux qui auraient déjà été effectués pour ce projet.*

*Ces éléments peuvent être repris des spécifications de départ.*

**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.

Base de données : Gestion des scores.

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

*Ce chapitre énumère les objectifs du projet. L'atteinte ou non de ceux-ci devra pouvoir être contrôlée à la fin du projet. Les objectifs pourront éventuellement être revus après l'analyse.*

*Ces éléments peuvent être repris des spécifications de départ.*

**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.

***Ce chapitre décrit la méthode de gestion de projet utilisée, ainsi que les éventuelles particularités requises par le contexte et/ou le chef de projet***

**Analyse / Conception**

**Gameplay**

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 500 millisecondes avant de pouvoir tirer à nouveau.

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.

Les protections

Sur la carte se trouveront des protections permettant au joueur de se cacher des missiles ennemis. Ceux-ci seront automatiquement placés sur la carte au lancement de la partie. Ils auront une taille fixe, ainsi que des points de vie. Ils pourront donc être détruits par le joueur ou un ennemi.

Les points de vie

Chaque entité aura des points de vie, le joueur ainsi que les ennemis auront 3 points de vie, à chaque fois qu’ils recevront un missile, ils en perdront un. Les protections auront elles 5 points de vies, rendant donc les structures plus solides. Lorsqu’un ennemi atteint 0 point de vie, il meurt ou disparaît si c’est une protection. Si c’est le joueur, la partie se termine.

Les niveaux

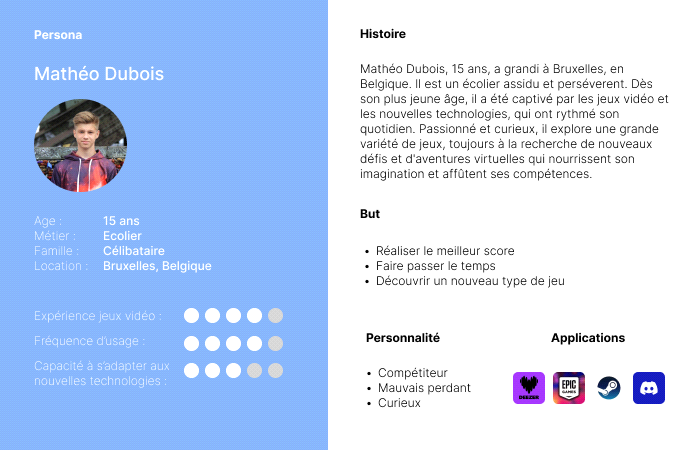
Le jeu comportera différents niveaux, pour que le joueur atteigne le second niveau, il devra réussir le premier, et ainsi de suite.

* *Le joueur*
* *Les ennemis*
* *Les déplacements*
* *Les niveaux*
* *Le tir*

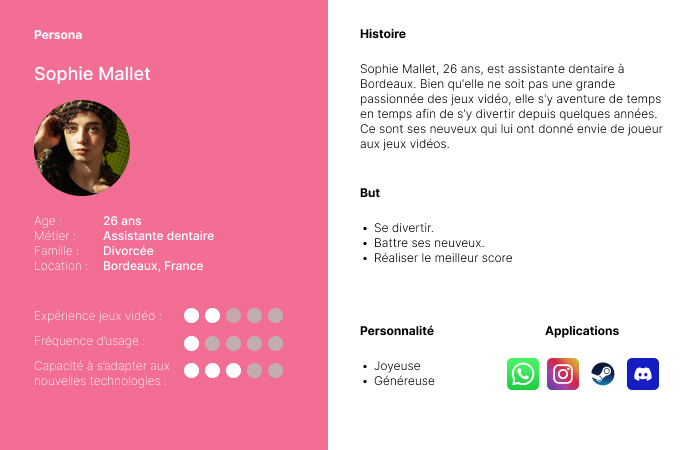
***Analyse de l’UX***

**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 :



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.

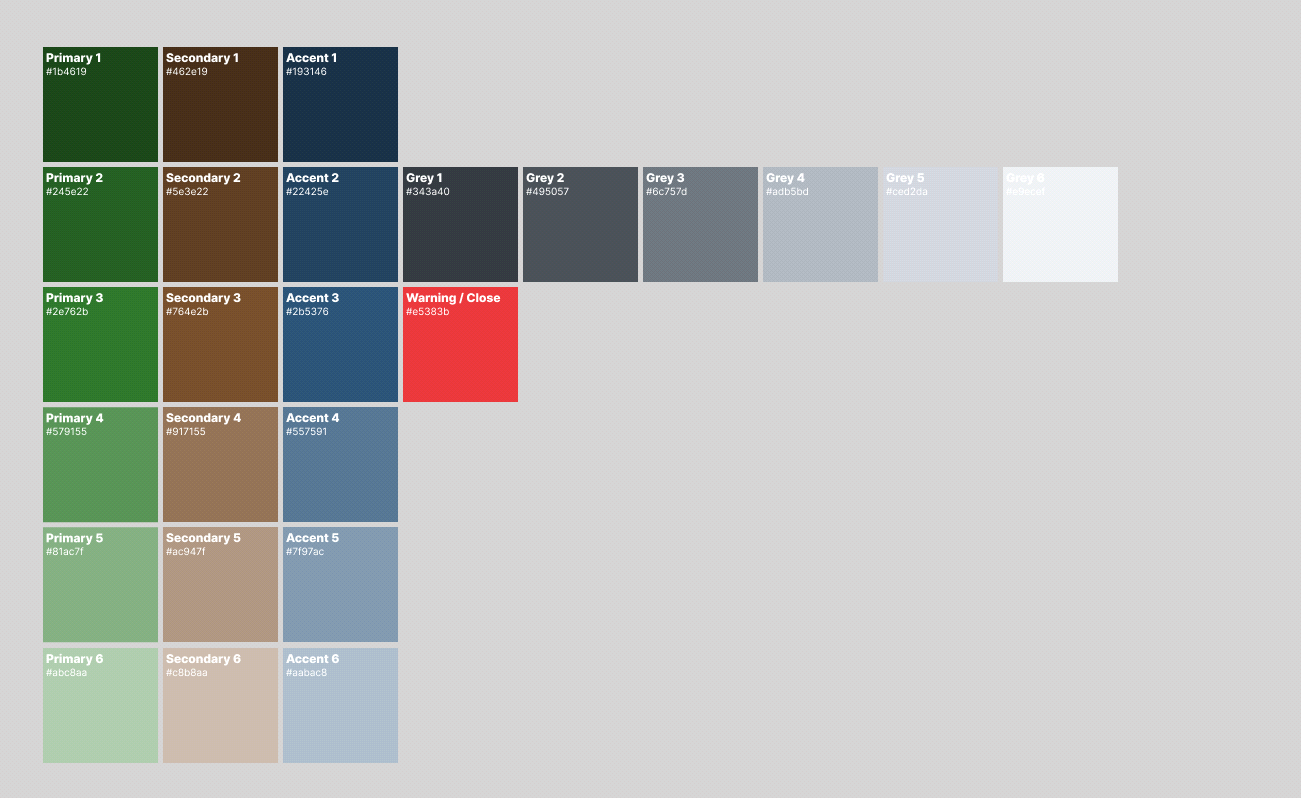


Ce second persona est bien différent du premier, Sophie est une amatrice des jeux vidéo et n’y joue que de temps à autres. Ses objectifs sont tout à fait différents car elle souhaite uniquement se divertir, et non pas de jouer en compétition.

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

**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’aie 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.

**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.

***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-fidelité et les choix effectués.

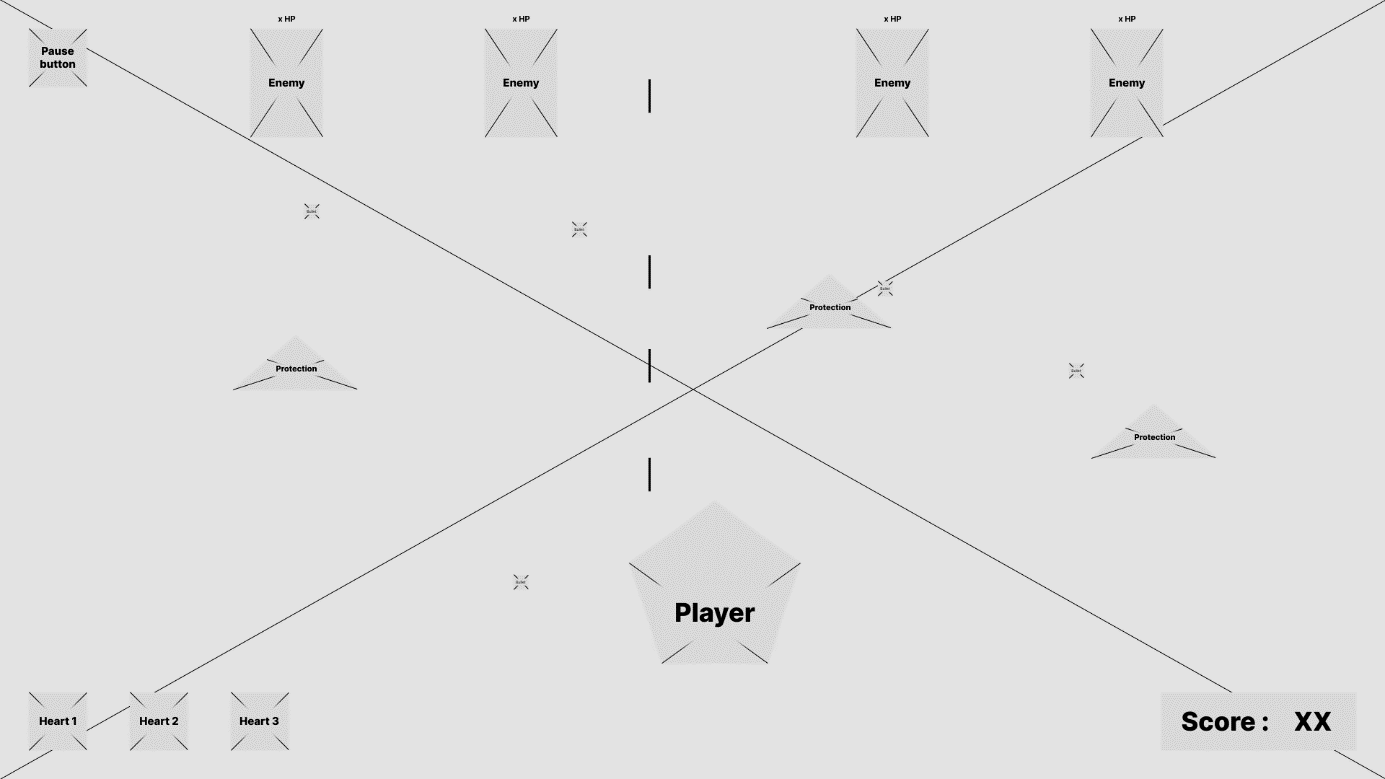
**Définitions des Wireframes**

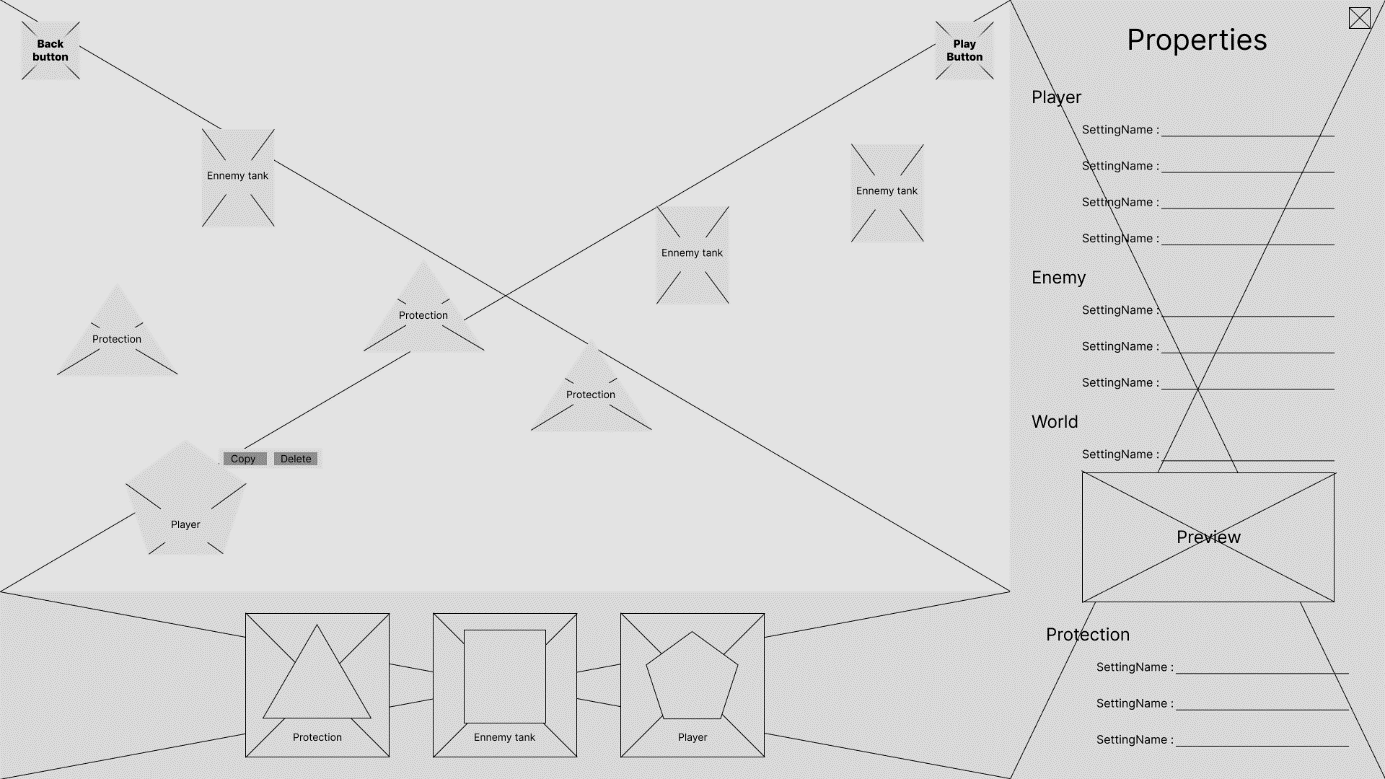
Un wireframe est une représentation simplifiée d'une interface en utilisant uniquement des lignes pour décrire la structure de l’interface. Il ne comporte aucune couleur et aucune « données ». Il sert souvent à visualiser la disposition d'un design. Ci-dessous se trouvera donc tous les Wireframes demandés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

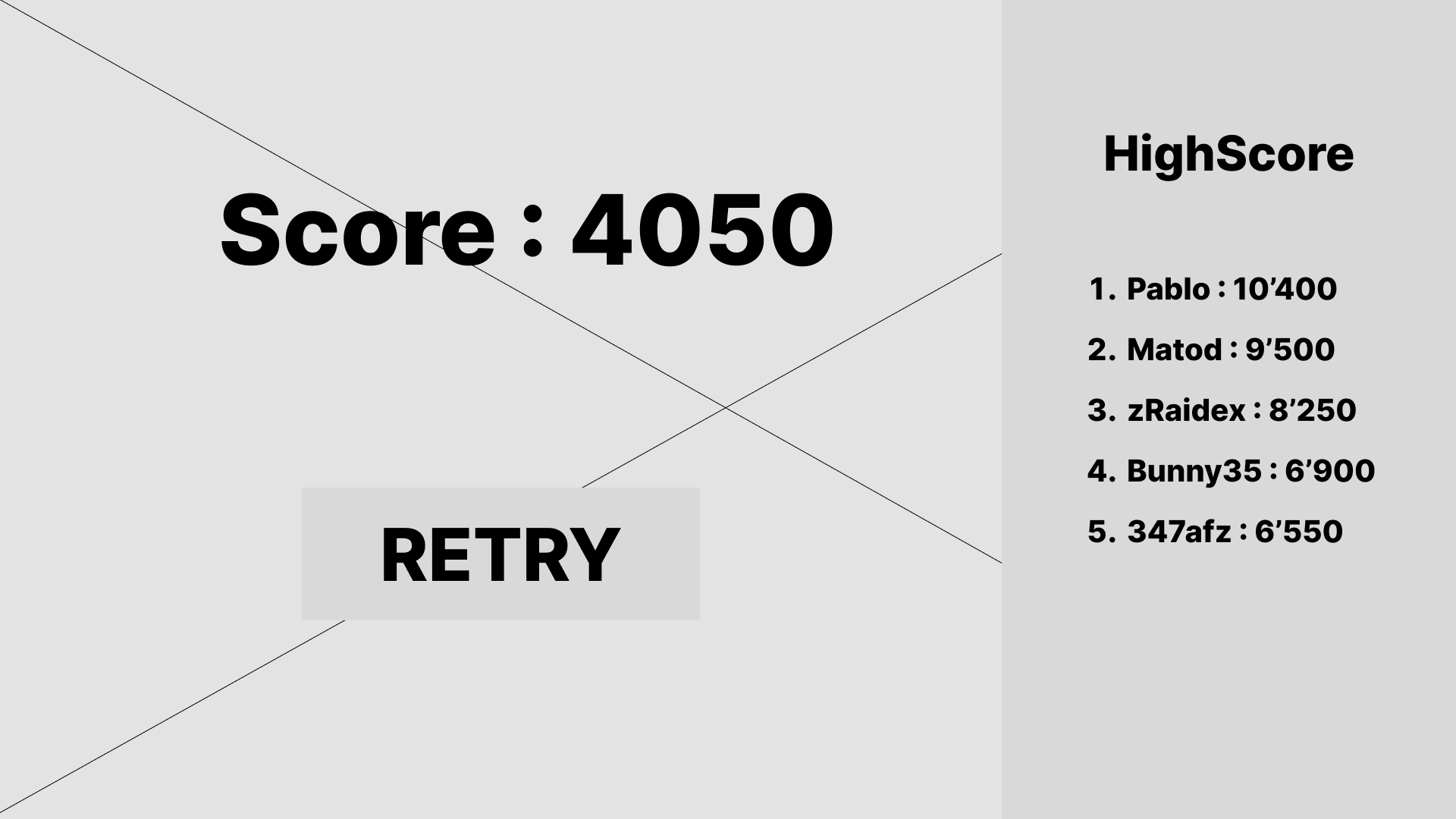
* Menu principal
* Ecran de jeu
* Editeur de niveau
* High scores

Les voici :









* **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.



* **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.

**Diagramme UML**

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](../POO/docs/diagrammeUML.pdf)ou dans le dossier P\_ShootMeUp/POO/docs/diagrammeUML.pdf!

* *Diagramme de classe*
* *Diagramme(s) d’état*
* **Analyse fonctionnelle**

***Reprendre le contenu des User Stories d’IceScrum : Story + tests d’acceptance (avec IceTools)***

* **Stratégie de test**

***Décrire quels sont les MOYENS utilisés pour faire les tests, ne pas décrire les tests à effectuer !!!***

***Décrire l’environnement dans lequel se fait la sprint review***

*Décrire la stratégie globale de test :*

* *types de des tests et ordre dans lequel ils seront effectués.*
* *les moyens à mettre en œuvre.*
* *couverture des tests (tests exhaustifs ou non, si non, pourquoi ?).*
* *données de test à prévoir (données réelles ?)* ***et comment elles seront mises en place****.*
* *les testeurs extérieurs éventuels.*
* **Réalisation**
* ***Points de design spécifiques***

***Ce chapitre est constitué de plusieurs sous-chapitre.***

***Chaque sous-chapitre explique un point de design technique particulier, quelque chose que vous avez dû inventer pour répondre au besoin et qui ne peut pas s’expliquer par de simples commentaires dans le code.***

***Il s’agit d’explications techniques sur le fonctionnement du système. Les explications sont appuyées par des diagrammes, ou de très brefs éléments de code.***

***NE PAS mettre ici des pratiques usuelles que tout professionnel de la branche connaît déjà. Par exemple, n’EXPLIQUEZ PAS ICI CE QU’EST LE PATTERN MVC.***

***Exemple (simplifié à l’extrême) : Protection contre des formulaires mal intentionnés ou modifiés***

* ***Au moment de générer le formulaire, le script php :***
* ***Concatène les noms de tous les champs contenus dans le formulaire***
* ***Calcule un hash SHA256 de la chaîne obtenue***
* ***Ajoute un input nommé « CSRF » de type hidden dans le form***
* ***A la réception du POST du fromulaire***
* ***Concatène les noms des indices de $\_POST***
* ***Calcule un hash SHA256 de la chaîne obtenue***
* ***Vérifie que la valeur du champ CSRF correspond***
* ***…***
* ***…***
* ***…***
* **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 accrocs.

***Résumer comment s’est passé la réalisation de chaque story, ses difficultés, les alternatives envisagées mais rejetées, ses surprises, …***

* **Mise en place de l’environnement de travail**

Pour accéder au code source du projet, il suffit de se rendre sur [GitHub](https://github.com/MathisOlaya/P_ShootMeUp), 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 a été utilisé avec deux écrans.

* ***Comment accéder au code source***
* *la liste de tous les fichiers et une rapide description de leur contenu (des noms qui parlent !)*
* *les versions des systèmes d'exploitation et des outils logiciels*
* *la description exacte du matériel*

***Ce chapitre décrit précisément comment un employé qualifié peut recréer l’environnement dans lequel vous avez effectué ce travail***

* **Description des tests effectués**

***Reprendre les tests d’acceptance d’IceScrum au moyen de la feuille ad hoc d’IceTools***

* **Erreurs restantes**

*S'il reste encore des erreurs :*

* *Description détaillée*
* *Conséquences sur l'utilisation du produit*
* *Actions envisagées ou possibles*

***Reporter la*** [***dette technique***](https://www.premaccess.com/qu-est-ce-que-dette-technique-comment-la-maitriser/) ***connue. S’appuyer sur la pratique des // TODO***

* **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. 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.

1. 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.

1. 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 l'adapter à mon projet.

1. 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 utilisé.

* **Annexes**
* **Manuel de référence**

***Issu de la génération automatique à partir des commentaires***

* **Journal de travail**