|  |
| --- |
| Projet XYZ |

Pour projet Shoot Me Up avec XCL

Table des matières

[1 Analyse préliminaire 3](#_Toc499021832)

[1.1 Introduction 3](#_Toc499021833)

[1.2 Objectifs 3](#_Toc499021834)

[1.3 Planification initiale 4](#_Toc499021835)

[2 Analyse / Conception 4](#_Toc499021836)

[2.1 Concept 4](#_Toc499021837)

[2.2 Stratégie de test 4](#_Toc499021838)

[2.3 Risques techniques 4](#_Toc499021839)

[2.4 Planification 4](#_Toc499021840)

[2.5 Dossier de conception 5](#_Toc499021841)

[3 Réalisation 5](#_Toc499021842)

[3.1 Dossier de réalisation 5](#_Toc499021843)

[3.2 Description des tests effectués 6](#_Toc499021844)

[3.3 Erreurs restantes 6](#_Toc499021845)

[3.4 Liste des documents fournis 6](#_Toc499021846)

[4 Conclusions 6](#_Toc499021847)

[5 Annexes 7](#_Toc499021848)

[5.1 Résumé du rapport du TPI / version succincte de la documentation 7](#_Toc499021849)

[5.2 Sources – Bibliographie 7](#_Toc499021850)

[5.3 Journal de travail 7](#_Toc499021851)

[5.4 Manuel d'Installation 7](#_Toc499021852)

[5.5 Manuel d'Utilisation 7](#_Toc499021853)

[5.6 Archives du projet 7](#_Toc499021854)

*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

Ce projet a été réaliser dans le cadre d’une formation d’informaticien CFC à l’ETML. Le but est de réaliser individuellement un jeu en deux dimensions en se basant sur la programmation orienter objet, sujet enseigner parallèlement au projet pendant le module \*\*\*\*. Les apprentis on le choix sur le style et la forme du jeu à réaliser. Le jeu est fait en langage C# à l’aide de l’IDE Visual Studio 2022. La gestion du projet est faite avec IceScrum, et la gestion des fichiers avec GitHub. Ce projet permet a l’apprenti de d’apprendre ou de se perfectionner dans la programmation orienter objet, et d’appliquer la gestion de projet « agile » via IceScrum.

*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

Le but de ce projet est de réaliser un jeu en 2D avec Microsoft forms

*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 utilisé IceScrum, un outil qui permet de planifier et de gérer des projets de façon agile. Nous avons fait ce projet en utilisant un seul sprint sur toutes la durée du projet.

***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 se trouve dans une zone vers le bas de l’écran et dois empêcher les ennemis qui arrive du haut vers le bas d’entrer dans sa zone. Le joueur peut se déplacer comme bon lui semble dans la sa zone mais ne peux pas en sortir. Les ennemis tir devant eux pour attaquer le joueur. Le but du jeu est de survivre le plus longtemps possible afin de faire le score le plus haut. Le joueur a deux types d’attaque, un tir normal qui fera de légers dégâts aux ennemis, et un tir faisant des dégâts de zone, infligeant de lourds dégâts aux ennemis sur une grande zone. L’attaque de zone possède un cooldown de 10 secondes entre chaque nouvelle attaque. La partie se termine lorsque le joueur n’a plus de points de vie, ou si un ennemi est entré dans la zone du joueur.

Il y a des obstacles en mouvement devant la zone du joueur qui lui permettent de se protéger contre les tirs des bateau ennemis.

* *Le joueur*
* *Les ennemis*
* *Les déplacements*
* *Les niveaux*
* *Le tir*
* *La gestion des vies*
* *…*

## Concept

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

## Analyse fonctionnelle

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

## 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

Cooldown de l’attaque de zone :

Le cooldown utilise DateTime.Now pour déterminer si le

temps entre la dernière attaque et le temps présent est plus petit ou

plus grand que 10. Si le temps est plus petit, le temps restant avant la prochaine attaque est affiché, si il est plus grand ou égal a 10, l’attaque peut être exécutée.

## 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

Je n’ai pas eu de problème pour permettre au joueur de se déplacer, j’ai essayé de trouver comment faire pour que le joueur ne se déplace pas de façon saccadé (par accoup) mais je n’ai pas réussi.

Certaine story comme le fait de tirer un projectile au click mon pris beaucoup de temps à réaliser car je n’avais pas les connaissances requises pour réaliser cette fonction.

***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

Vous pouvez trouvez le code source de mon jeu dans mon repos github : [NicolaGolaz/P\_OceanInvader: db](https://github.com/NicolaGolaz/P_OceanInvader)

Vous y trouverez le dossier « OceanInvader », qui contient tous les fichiers qui compose mon jeu.

Le projet a été réaliser sur windows 11 et windows 10. La version de windows 11 est 23H2. L’IDE utilisé est Visual Studio 2022.

Les seuls matériel utilisé sont deux pc (clavier, souris,etc.).

* ***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/#:~:text=La%20dette%20technique%20survient%20quand,de%20plus%20en%20plus%20fr%C3%A9quents.) ***connue. S’appuyer sur la pratique des // TODO***

# Conclusions

Je n’ai pas pu atteindre la majeur de mes objectifs, du au fait que je manquais de connaissance pour réaliser les fonctions de mon jeu vidéo. J’ai réussi à afficher un joueur qui a la possibilité de tirer différent type de projectile et de se déplacer dans une zone définie.

*Développez en tous cas les points suivants :*

* *Objectifs atteints / non-atteints*
* *Points positifs / négatifs*
* *Difficultés particulières*
* *Suites possibles pour le projet (évolutions & améliorations)*

# Annexes

## Manuel de référence

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

## Journal de travail