Spicy Invaders

ETML – Lausanne

LG – JS – AB – TR

Tables des matières

[1. Introduction 3](#_Toc42262576)

[2. Planification et journal de travail 3](#_Toc42262577)

[3. Object à atteindre 4](#_Toc42262578)

[4. Schéma UML 5](#_Toc42262579)

[5. Document technique 6](#_Toc42262580)

[6. Mode d’emploi 7](#_Toc42262581)

[7. Conclusion 8](#_Toc42262582)

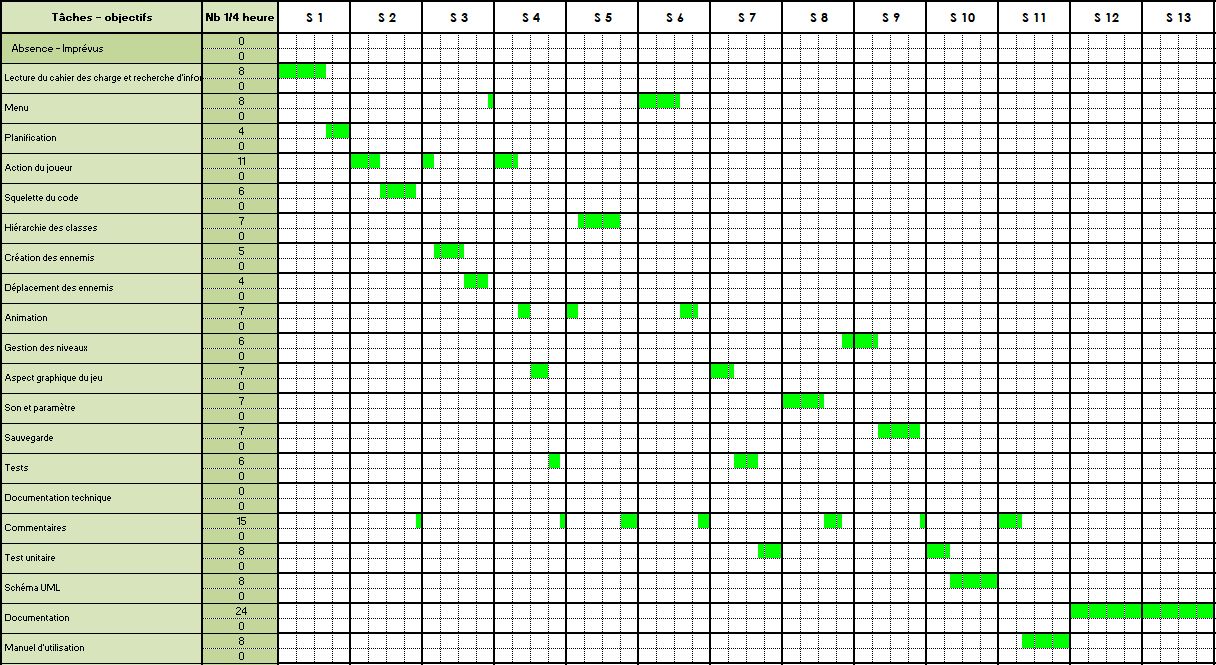
# Introduction

Ce document a pour but d’expliquer notre projet Spicy invader inspiré d’un célèbre shoot 'em up sur borne d’arcade.

Notre objectif était de faire une nouvelle version du jeu en C# sur console. Pour ce faire, nous avons travaillé en groupe, notamment avec l’aide de gitKraken pour gérer les avancées et les versions de chacun.

Lien vers le git : <https://github.com/BetterBlood/SpicyInvaders>

# Planification et journal de travail

Nous avons décidé de montrer uniquement la planification car en raison du corna-virus et du projet pluridisciplinaire, le journal de travail n’a aucun sens par rapport à la planification initial.

Cependant pour garder les informations importantes concernant les tâches, nous avons choisi de présenter la répartition de toutes ses tâches sous la forme d’un tableau récapitulatif.

| Titre | Description | Auteurs |
| --- | --- | --- |
| Recherche d’information | Recherche d’information global sur le projet et lecture du CDC | Adrian, Laetitia et Jeremiah |
| Planification | Planification globale du projet | Toine, Adrian, Laetitia et Jeremiah |
| Menu | Création du menu du jeu | Adrian |
| Action du joueur | Le joueur tire et se déplace | Adrian, Laetitia et Jeremiah |
| Squelette du code |  | Jeremiah |
| Hiérarchie des classes |  | Jeremiah |
| Création des ennemis |  | Jeremiah |
| Déplacement des ennemis | Déplacement en groupe des ennemis | Laetitia |
| Animation | Animation d’introduction et de Game Over | Adrian et Jeremiah |
| Gestion des niveaux et des difficultés | Création de divers niveaux et choix des difficultés | Jeremiah |
| Aspect graphique du jeu | Dessin des personnages et design du menu | Toine, Adrian et Jeremiah |
| Son et paramètre | Musique et bruitage | Toine et Jeremiah |
| Sauvegarde et gestion du scores | Sauvegarde des scores | Jeremiah |
| Tests | Vérification du fonctionnement du jeu | Adrian, Laetitia et Jeremiah |
| Documentation technique | Doxygen | Laetitia |
| Commentaire | Tout type de commentaires dans le code | Laetitia et Jeremiah |
| Manuel d’utilisation |  | Adrian, Laetitia et Jeremiah |
| Test Unitaire |  | Jeremiah |
| Schéma UML | UML fait avec Visual Studio Entreprise | Jeremiah |
| Documentation |  | Toine, Adrian, Laetitia et Jeremiah |

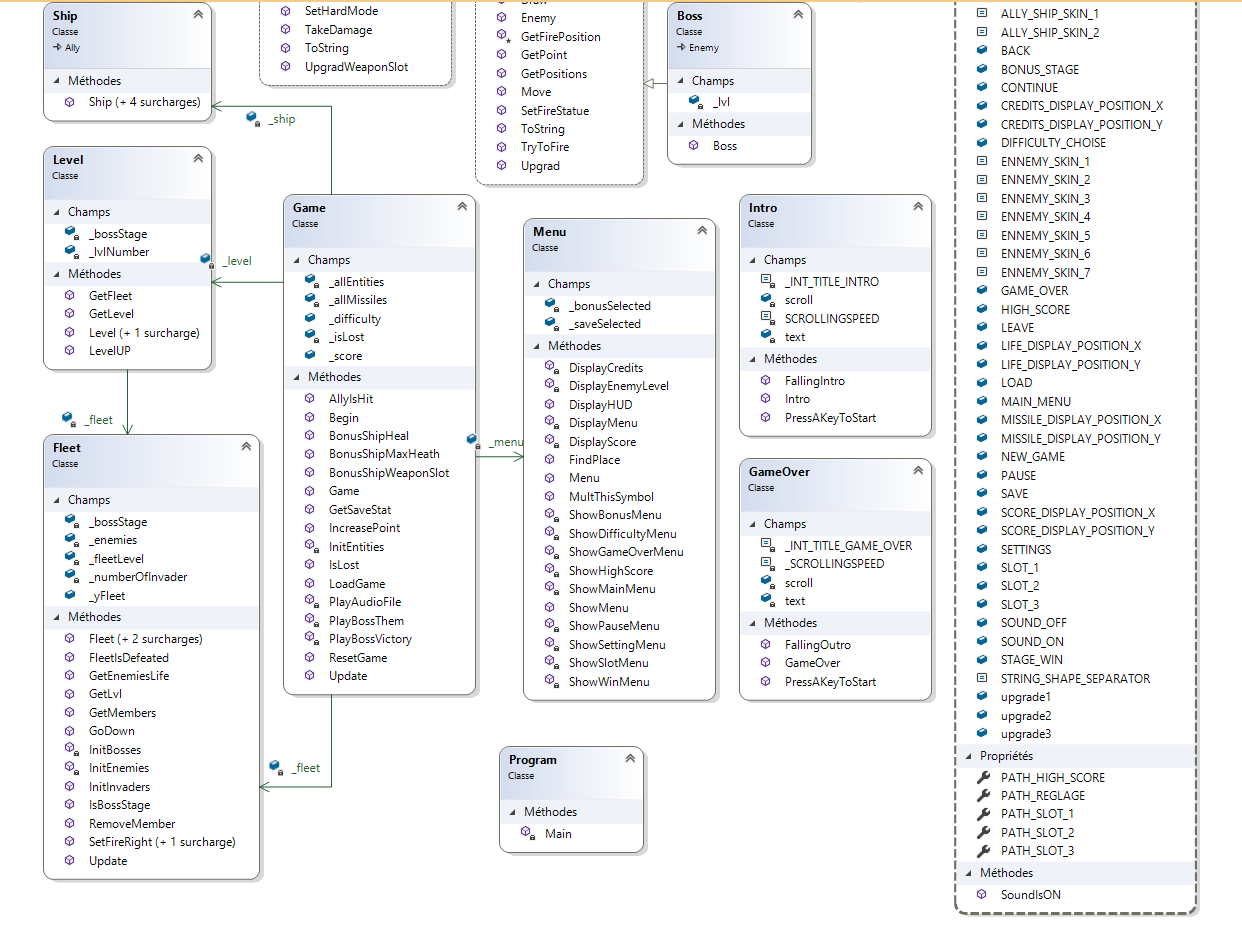
# Object à atteindre

Pour ce projet, nous avions une liste de fonctionnalités à atteindre :

* Jeu fonctionnel (début-fin)
* Utilisation des flèches directionnelle
* Des ennemis
* Son et réglage
* Différents niveaux de difficulté
* Highscore
* Crédits

# Schéma UML

La première partie du schéma contient les objets avec lesquelles le joueur interagit et pourrait se traduire par des objets réels. Par exemple, il s’agit des ennemis et du vaisseau du joueur, ce sont les entités présentes dans le jeu.



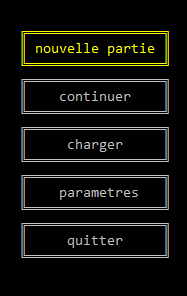
La seconde partie contient ce qui gère le fonctionnement du programme. Il s’agit des niveaux et du menu. Les objets de la première partie sont gérés par cette partie. La classe Game est celle qui gère principalement le fonctionnement du jeu. A droite, il y a la classe UseFull qui contient tous les attributs utilitaires, typiquement les constantes.

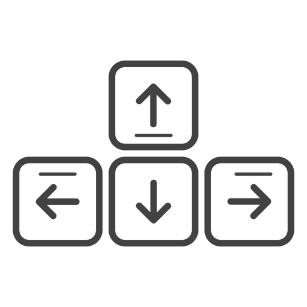
# Document technique

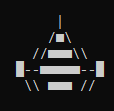
La documentation technique a été faite avec Doxygen, elle nous a permis de corriger divers commentaires. Lien vers la documentation : [Doxygen Spicy Invaders](../Documentation/Doxygen/index.html).

(Attention le lien hypertext peut ne pas être disponible en format PDF)

# Mode d’emploi

Après la page d’intro, le jeu s’ouvre sur le menu.

Pour se déplacer dans le menu, il faut utiliser les flèches directionnelles haut et bas puis sélectionner avec la touche espace.

Une fois le jeu lancé, pour contrôler votre vaisseau utilisez les flèches directionnelles droite et gauche. La touche espace vous permet de tirer. Attention, vous ne pouvez pas tirer à l’infini, une fois votre chargeur vide vous devez attendre un petit temps de rechargement. Vous pouvez mettre le jeu en pause avec la touche « P ».

Votre vaisseau

Vous ennemis vous tireront aussi dessus, alors ne tombez pas à court de vie, si vous voulez éviter le Game Over. Quand vous battez un boss, vous pouvez obtenir une amélioration de votre vaisseau, mais vous avez un nombre maximum de missiles.

Votre but est de vaincre un maximum d’ennemi sans obtenir un Game over !

Votre avez perdu !

Vos missiles disponibles

Vos vies

Un ennemi

Un boss

# Conclusion

Finalement, nous avons réussi à faire toutes les fonctionnalités voulues.

Cependant, nous avons mis en place des fonctionnalités supplémentaires :

* La possibilité d’améliorer son vaisseau
* Des boss
* Une augmentation de la difficulté en fonction de la progression des niveaux
* Des animations de début et fin de jeu

Dans ce projet, nous aurions pu encore améliorer les points suivants :

* Il y a certaines parties du code qui ne sont pas optimisées (visible avec les TODO)
* Possibilités d’avoir différents types de missiles et modifier la puissance de tires
* Des améliorations aléatoires du vaisseau

Ce projet nous a permis d’approfondir nos connaissances en programmation C#. Nous avons également pu apprendre à utiliser Github ainsi que Doxygen que nous n’avions jamais utilisé auparavant.