Cahier des charges
Projet Licence 3 - Space Wars

Tony Deborguere
Baris Tekeli
Malik Touat
Qi Zhiwen

Date de rédaction: 22 juin 2016

Table des matières

Chapitre 1

Présentation du projet

Fiche d'identité du projet		
Nom du projet :	Space Wars	
Objet :	Création d'un clone de Space Invaders avec une partie Player Versus Player en réseaux.	
Membres du projet :	DEBORGUERE TONY	(t.deborguere@gmail.com)
	TEKELI BARIS	(tekelibaris@gmail.com)
	TOUAT MALIK	(mal.touat@gmail.com)
	QI ZHIWEN	(531940615@qq.com)
Commanditaire :	JULIEN DEHOS	
Date de début :	06 Juin 2016	
Date de fin :	22 Juin 2016	

1.1 Règle du jeu

— Le jeu commence avec un vaisseau qui a 3 points de vie et une vague de $5 * 11$, soit 55 aliens.
— Le joueur, qui incarne le vaisseau, a la possibilité de se déplacer uniquement à l'horizontal et de tirer des missiles très rapide.
— Un seul missile du vaisseau peu être présent sur la carte à la fois, ce qui veut dire que pour tirer un missile, le précédent doit être détruit.
— La vague d'ennemis se déplace horizontalement et descend petit à petit.
— Si la vague descend au point de toucher le vaisseau, alors celui-ci perd 1 point de vie et la vague remonte de 4 lignes.
— Si le joueur atteint 0 point de vie, la partie est terminée et le score est enregistré (uniquement dans le cas où le score actuelle dépasse le meilleur score enregistré).
— Entre la vague et le vaisseau, il y a 3 boucliers.

— Les boucliers seront détruits petit à petit après chaque impact avec un missile (adverse ou allié).
— Les missiles du vaisseau vont uniquement de bas en haut, et ceux de l'ennemi de haut en bas. Les missiles progressent uniquement verticalement et ne peuvent être déviés.
— Un alien touché par un missile se voit détruit avec celui-ci.
— Chaque alien détruit donne un malus à l'adversaire (uniquement en coup par coup) et rapporte des points de score, selon le type d'alien.
— La vague est constituée de 4 types d'aliens :
  — 1er et 2e rang : type 1 → 10 points.
  — 3e et 4e rang : type 2 → 20 points.
  — 5e rang : type 3 → 50 points.
— Si le vaisseau est touché par un missile, le vaisseau perd 1 point de vie, le jeu se met en pause pendant 1 seconde et la partie reprend tout de suite. Quant au missile, il sera détruit.
— Si un alien est derrière un autre alien, seul celui qui est devant peut tirer.
— A la fin de la partie, le perdant aura un malus pour la partie d'après. (uniquement en coup par coup).

1.1.1 Conditions de victoire

— Détruire tous les aliens sur la carte.

1.2 Utilisation du logiciel

1.2.1 Installation

Dans le répertoire */Space_Wars-master*, on retrouve le *README* où on peut lire les règles du jeu, les instructions pour l'installation et les contacts.

1.2.2 Lancement du logiciel

Pour lancer le jeu, ouvrez un terminal dans le répertoire */Space_Wars-master* et lancer le *Makefile* en entrant la ligne de commande suivante :
— make
Puis entrer la commande qui suit pour lancer le logiciel :
— ./bin/main.out

1.2.3 Ressource du logiciel

Dans le répertoire */Space_Wars-master* se trouve un autre répertoire */src* qui contient les divers fichiers C++ et H que le logiciel utilise. Ces fichiers ne doivent en aucun cas être modifiés par une personne inexpérimentée dans le langage C++ sous peine d'avoir un logiciel qui ne fonctionnera pas.

Toujours dans le même répertoire */Space_Wars-master* se trouve les répertoires */ressource*, avec les images, les fonts et les sons du jeu, et */Documents* qui contient le cahier des charges.

1.3 Définition des besoins

1.3.1 Contexte général

Dans le cadre de notre fin d'année, nous devons développer un nouveau jeu qui dispose d'une interface graphique.
Nous avons choisi le jeu nommé *Space Invaders* comme base, et l'avons cloné pour obtenir *Space Wars*. Space Wars est un jeu d'arcade pouvant être joué seul, ou à deux en réseaux.

1.3.2 Besoins et priorités

Les deux besoins évoqués par le client sont :

Besoin 1 Possibilité de jouer en réseaux.
- Soit en tour par tour.
- Soit en temps réel.

Besoin 2 Une interface graphique (avec la bibliothèque SFML).

L'interface se doit d'être clair et simple d'utilisation pour l'utilisateur, et devra faire l'objet d'un développement soigné pour ne pas nuire au jeu. L'aspect le plus important est bien évidement le fait que le jeu soit fonctionnel avant la date de fin du projet, même si cela implique de sacrifier certaines fonctionnalités.

1.4 Etat du produit au début du projet

Nous sommes parties de rien, nous avons tout créé de A à Z.

1.5 Spécifications

- Logiciel fonctionnant sous les distributions linux (Debian & Basée sur Debian).
- Création et Gestion d'un Menu avec une interface graphique.

  Avant la création du jeu, nous voulions commencer l'interface menu sur laquelle on peut lancer l'un des modes de jeu prévu.
  Il nous a suffit d'ajouter une image et une musique de fond, des boutons et un curseur pour naviguer sur l'interface et de gérer les différents événements. L'utilisateur peut se déplacer dans le menu avec les flèches directionnelles du clavier et sélectionner ce qu'il souhaite faire avec la touche entrer.
- Fonctionnalités de choix du type de jeu :
  - Fonctionnalités d'un jeu solo.

Nous avons commencé par créer la fenêtre du jeu et modéliser petit à petit l'interface graphique de celui-ci en ajoutant les sprites du vaisseau, les différents aliens et les boucliers

Puis nous avons intégré au code les divers événements à gérer : le déplacement du vaisseau, des ennemis, création et déplacement du tir du vaisseau et des ennemies et l'impact entre tout ces objets.

Enfin, après avoir inclus les sons (la mort d'un alien, du vaisseau, ou le tir de ce dernier), le score et le nombre de vie du joueur, nous avons défini les conditions de fin de partie.

Le joueur joue une partie seul, obtient un score et peut recommencer s'il le souhaite après la fin de la partie.

Le score est enregistré après chaque fin de partie dans le cas où il bat le record. (fichier "record.txt")

— Fonctionnalités d'un mode de jeu joueur contre joueur, équivalent français du player versus player.

Le joueur joue une partie contre un autre joueur en réseaux. La partie se termine quand les deux joueurs meurent ou éliminent tous les aliens.

Nous avons repris l'interface du jeu pour y ajouter le score de l'adversaire afin de pouvoir jouer contre un autre joueur.

Ce mode de jeu en réseau est en temps réel.

Le score détermine le vainqueur.

Nous voulions faire un autre mode de jeu en réseau en le tour par tour, afin de faire s'affronter deux personnes selon leur score en fin de partie. Le gagnant de la partie donnerait des malus à l'adversaire pour la partie d'après. Mais faute de temps, nous avons du abandonner ce mode de jeu.

— Fonctionnalités d'un mode de jeu joueur contre ordinateur.

Le joueur joue une partie contre l'intelligence artificielle. La partie se termine quand les deux joueurs meurent ou éliminent tous les aliens.

Nous voulions faire en sorte de pouvoir donner un second mode de jeu pour jouer seul, en dehors du mode solo. Ce mode nous aurait permit de faire des duels ou de s'entraîner avec une intelligence artificielle dont le niveau de difficulté aurait été modifiable.

Par manque de temps, nous étions dans l'obligation d'abandonner la réalisation de ce mode de jeu.

— Fonctionnalités de gestion d'une partie :

— Définir les conditions de victoire :

Mode solo Éliminer tous les aliens présents sur la carte

Mode joueur contre joueur Avoir un score plus élevé que son adversaire.

— Fonctionnalités de déplacement avec le clavier :

Flèche de droite Se déplacer à droite.

Flèche de gauche Se déplacer à gauche.

— Fonctionnalités de tir avec le clavier :

Barre d'espace Tirer un missile.

— Fonctionnalités d'impact :

Quand il y a un impact, le missile qui est l'élément A est détruit, et des dégâts sont causés à l'élément B :

Le missile sera détruit à l'impact.

L'alien sera détruit à l'impact.

Le vaisseau perd un point de vie à l'impact ou est détruit s'il ne lui restait qu'un seul point de vie.

— Fonctionnalités de tir de l'alien :

Une erreur concernant les tirs de l'alien a été détectée : Les aliens peuvent tirer même s'ils sont mort. Nous n'avons pas eu le temps de gérer ce problème.

— Fonctionnalités Réseau :
  — Connexion Client / Serveur.
    — Le client se connecte au serveur.
    — Le serveur attend la connexion d'un deuxième client sur le même port pour lancer une partie en réseau.
    — Lors d'une partie en tour par tour :
      — A la fin de la partie en cours, le score du premier joueur est enregistré, et le second doit battre le score du premier. (non réalisé)
    — Lors d'une partie en temps réel :
      — La partie se termine lorsque les deux joueurs élimine tous les aliens sur la carte ou meurent.
      — Celui qui a le meilleur score après la fin de la partie gagne.

— Fonctionnalité d'intelligence artificiel. (Optionnel)

IA Une intelligence artificielle qui permet de jouer à Space Wars contre l'ordinateur. (non réalisé)

— Interface utilisateur :

Affichage simple respectant l'aspect graphique de Space Invaders.

Chapitre 2

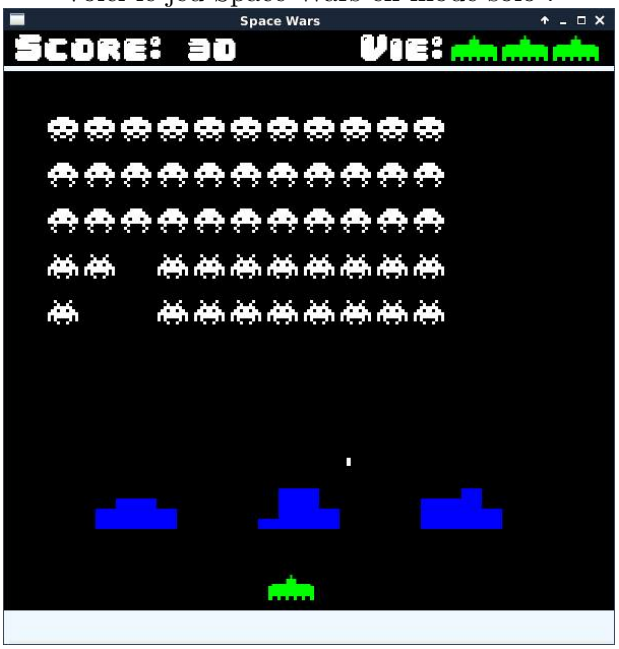
Réalisation

2.1 Présentation du logiciel réalisé

Le jeu Space Wars est utilisable via les distributions Linux. Lorsque nous lançons le jeu, nous avons :

- Une petite introduction.
- Un menu qui nous permet de :
  - Jouer une partie seul.
  - Jouer une partie contre un autre joueur en réseau :
    - Créer une partie. (si non-créée)
    - Rejoindre une partie. (si déjà créée)
    - Retourner au menu.

Voici le jeu Space Wars en mode solo :



Le jeu Space Wars en mode joueur contre joueur en réseaux aura la même interface graphique, à l'unique différence qu'il y aura le score du joueur 2 affiché à côté du score du joueur 1.

2.2 Présentation technique

Diagramme de classe

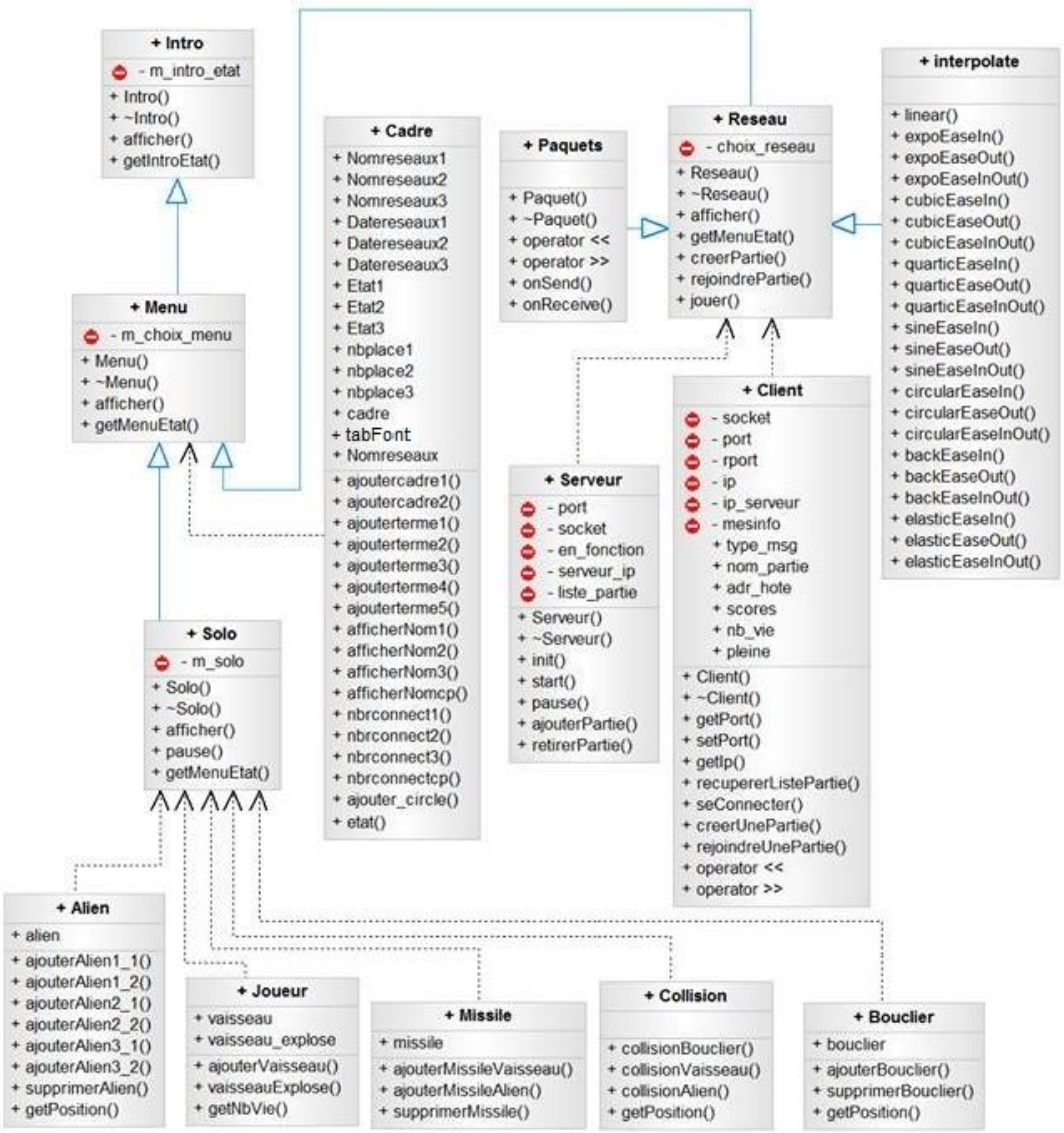
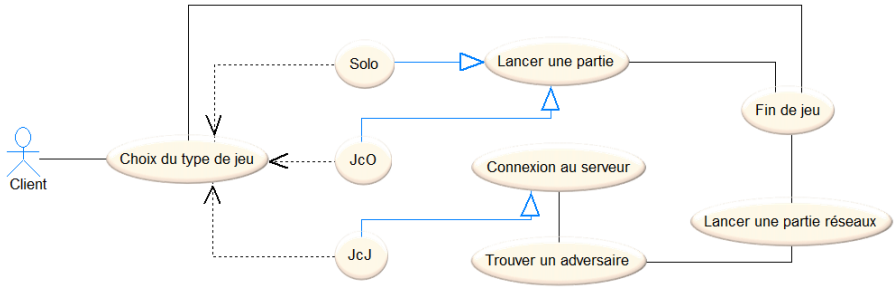


Diagramme de cas d'utilisation



2.2.1 Points importants

Affichage

Nous avions comme contrainte d'utiliser la bibliothèque **SMFL** (**S**imple and **F**ast **M**ultimedia **L**ibrary) pour réaliser les interfaces graphiques du jeu. On a donc installé et utilisé les librairies via sfml-all. Nous avons utilisé des "Sprite" pour stocker nos images (alien, vaisseau, missile, bouclier), des textes avec différentes fonts pour afficher nos titres et nos informations (scores, vies, Nom du réseau, ...) et d'autres fonctionnalités de SFML pour créer les différentes interfaces graphiques.

Menu

Afin de gérer les différentes fonctionnalités du programme, un menu a donc été créé pour permettre à l'utilisateur de choisir la fonctionnalité du programme qui l'intéresse. Pour faire ce menu, on a donc créé au lancement de l'application une fenêtre de taille *800x600* avec une petite introduction qui affiche le logo de la librairie **SMFL**. Puis un menu apparaît suite à un effet de fondu. La classe Menu se base sur une variable qui passe d'un état à un autre. Les différents états sont définis dans un ENUM. La base de l'application est la classe Jeu qui gère les différents états du menu pour charger la classe approprié à l'état du Menu, en passant par référence la fenêtre pour l'affichage.

Jeu solo

Nous avons une classe solo qui utilise les classes bouclier, alien, vaisseau, joueur, collision et missile. Dans la classe solo, nous appelons toutes les classes nécessaire (celles ci-dessous) afin de gérer un jeu complet en solo. Dans la classe alien, nous gérons la création et l'élimination des aliens. Dans la classe bouclier, nous gérons l'ajout et la destruction de bouclier. Dans la classe joueur, nous gérons la vie, l'ajout et la destruction du joueur (vaisseau). Dans la classe collision, nous gérons toutes les collisions entre chaque entité. Dans la classe missile, nous gérons l'ajout et la suppression des missiles.

Réseau

Le jeu contient une partie jouable en réseau en temps réel. Nous connectons deux clients à un serveur qui sert de lien entre les deux clients. Chaque client communique avec le serveur via **UDP**. Le serveur tient une liste des clients et une liste des parties. Il est en charge de la transmission des informations à chacun des joueur d'une partie. Le serveur est donc une application extérieur au programme, qui est en charge de certaine fonctionnalités des parties en réseau. L'échange des données sur le réseau se fait via les paquets de la **SFML**.

État du produit à la fin du projet

Le produit (jeu) doit être fonctionnel à la fin du temps imparti, quitte à ne pas réaliser certaines fonctionnalités.

Portabilité

Le jeu doit être utilisable par toutes les distributions Linux à condition d'installer les packages suivants :

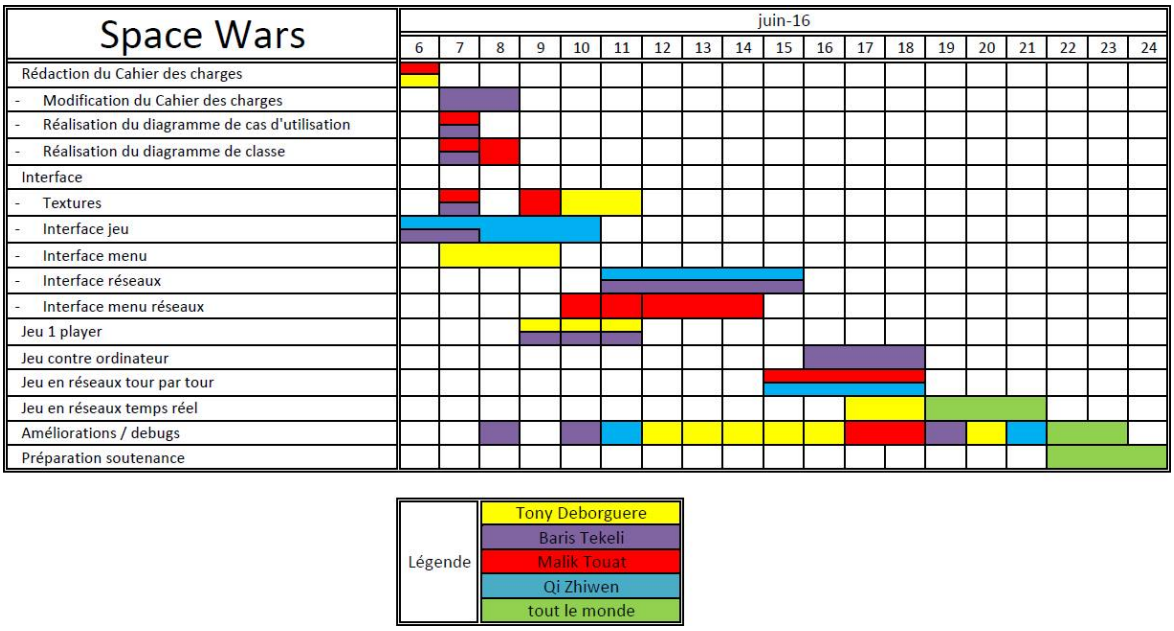
— libsfml-dev
— libboost-dev
— pkg-config

Chapitre 3

Bilan

3.1 Déroulement du projet

3.1.1 Prévisions



Comme indiqué sur le planning prévisionnel, nous avions prévu de finir le jeu solo assez tôt, de commencer l'IA, de faire le jeu en tour par tour et en temps réels en mettant le tout en réseau.

3.1.2 Problèmes rencontrés

Problème 1 Le temps.

Problème 2 Compétences pour le réseau.

Problème 3 La gestion des événements avec la souris.

Problème 4 La gestion des collisions automatiques.

3.1.3 Solutions

Problème 1 Nous avons décidé de ne pas faire tout ce que nous avions prévu :
- Nous avons fait un mode de jeu solo.
- Nous avons fait un mode de jeu en joueur contre joueur tour par tour

Problème 2 Nous avons fait en sorte de faire des recherches sur internet et de commencer la partie réseaux dès le début du projet afin de mettre toutes les chances de notre côté.

Problème 3 Pour gérer la souris, on finalement utiliser les fonctions de la SFML qui nous donne la taille d'un texte ou d'un sprite pour gérer ensuite la position de la souris par rapport à ces tailles

Problème 4 Nous avons dû faire une classe Collision avec une fonction pour chaque collision entre les différentes entités.

3.2 Conclusion (pour les projets futurs)

3.2.1 Gestion des tâches à accomplir

Nous avons attribué à chaque membre de notre groupe les tâches à accomplir dans lesquels ils étaient les **meilleurs** afin d'être le plus efficace possible. Ayant personne de doué en réseau, nous avons attendu avant de commencer cette partie, et nous avons encore perdu du temps à faire des recherches sur internet.

3.2.2 Ce que nous a apporté ce projet

Nous avons fait quelques erreurs, comme la gestion du temps. C'est pourquoi, pour les prochains projets, nous devrons avoir de meilleures **prévisions** afin d'être sûr de pouvoir finir le projet.

Nous savons mieux magner le langage C++, notamment avec la librairie SFML que nos avons utilisée lors de ces 3 dernières semaines.

3.3 Ressources

Vous pouvez récupérer le logiciel ainsi que les ressources de celui-ci sur ce lien :

https://github.com/l3info-MBT/Space_Wars

Vous trouverez sur ce lien des informations supplémentaires sur la bibliothèque SFML :

http://www.sfml-dev.org/index-fr.php

Enfin, vous trouverez sur ce dernier lien les images et les sons qui nous ont servi dans la réalisation du jeu :

http://www.classicgaming.cc/classics/space-invaders/