IOPETI Hugo
YVOZ Ludovic

Projet de méthodologie niveau avancé : Space Shooter

1. Présentation

Nous sommes 2 élèves en première année de licence mathématiques-informatique et nous avons suivis l'enseignement de méthodologie niveau avancée. Pour rappel, la tâche demandée était de réaliser les 4 niveaux (du niveau 0 au niveau 3) du projet initial puis d'ajouter quelques améliorations au jeu ainsi créé.
Ce rendu permet une compréhension plus aisée de notre projet.

1.1. Installation

Pour commencer, il vous faut récupérer le projet sur git (nous vous avons ajoutés à notre projet auparavant).

Une fois cette première étape réalisée, vous devez ouvrir un terminal Linux car le projet n'est fonctionnel que sur un environnement Linux.

Pour vérifier que vous possédez bien les bibliothèques nécessaires au bon fonctionnement de notre jeux, il vous suffit d'utiliser les commandes suivantes :

- Pour installer les bibliothèques SDL : *sudo apt-get install libsdl2-dev libsdl2-ttf-dev*.
- Si vous décidez de supprimer les fichiers exécutable créé lors de la compilation de notre code (ceux avec l'extension ".o" ainsi que l'exécutable nommé "spacebattle") : *make clean*.

1.2. Lancer le programme : les étapes à suivre

Premièrement, il faut compiler le code de notre projet, pour ce faire il suffit d'ouvrir le terminal d'une machine apte à compiler du C, puis de se rendre dans le dossier projet.
Ensuite, il faut utiliser la commande *make* qui permet de compiler automatiquement tous les fichiers nécessaire (le makefile est fournit avec notre projet).
Troisièmement, il faut appeler le fichier exécutable créé par le makefile : *./spacebattle*
Pour finir, il suffit de jouer au jeu en se déplaçant à l'aide des flèches directrices : "<-" "->" et en tirant avec la touche "espace".
Pour faire la documentation (latex, …) du projet il suffit de taper *make doc* dans le terminal et le fichier html et le fichier pdf se créeront automatiquement.

1.3. Règles du jeu

Vous contrôlez un vaisseau spatial situé en bas de l'écran, des vaisseaux ennemis apparaissent aléatoirement du haut de l'écran et descendent rapidement, il vous faut en laisser passer le moins possible. (Il est possible de modifier le nombre d'ennemis dans le code : il suffit de modifier la constante : NB_ENNEMIS)
Initialement, le vaisseau du joueur à 3 points de vie (eux aussi modifiables) et lorsqu'il entre en collision avec un vaisseau ou un missile ennemi, il perd 1 point de vie et si ceux-ci atteignent 0 la partie se termine.
Le joueur a également un score qui s'incrémente de 1 à chaque fois qu'il détruit un vaisseaux ennemi mais il faut faire attention car si il détruit un vaisseau infecté, il perd 1 de score et si le joueur meurt son score retombe à 0.
Un boss apparaît lorsque la moitié des ennemis est détruit (10 dans la version de base).
Si le joueur finit le jeu en ayant détruit tous les vaisseaux ennemis, son score est doublé.

2. Extensions réalisées

Nous avons suivis le thème imposé (l'espace) et avons donc réalisé des extensions en accord avec celui-ci.

Nous avons ajoutés :

- Le bonus "réacteur", celui-ci permet une fois récupéré par le vaisseau, d'augmenter sa vitesse, cela peut aider le vaisseau car il esquivera plus facilement les vaisseaux ennemis mais il sera aussi moins précis dans ses déplacements. Ce bonus dure jusqu'à la mort du vaisseau, donc jusqu'à la fin de la partie. Le bonus est très lent et donc assez simple à attraper. Il n'y a qu'un seul bonus réacteur par partie. On ne peut pas tirer sur ce bonus.

- Le bonus "nuke", celui-ci permet une fois récupéré par le joueur de détruire tous les vaisseaux présent sur le terrain, les ennemis détruits augmentent également de 1 point le score du joueur. Le bonus est très rapide et donc très dur à attraper. Il n'y a qu'un seul bonus nuke par partie. On ne peut pas tirer sur ce bonus.

- Le malus "vaisseau infecté" est un vaisseau qui agit comme les vaisseaux ennemis. Durant son court voyage, si il percute le vaisseau allié, celui-ci voit sa vitesse diminuée et si il se fait tirer dessus par le vaisseau, le joueur perd un point. Il y'a 5 fois moins de vaisseaux infectés que de vaisseaux ennemis.

- Le boss tire des missiles qui réduisent les point de vie du joueur. Il se déplace de droite à gauche et prend une direction aléatoire (+x ou -x) lorsqu'il est touché par le joueur. Le boss à 3 formes, la première (lorsqu'il est vert), le boss à un nombre de point de vie qui est dans l'intervalle "nombre de point de vie max et la moitié de ses points de vie", il tire à une vitesse raisonnable. La seconde forme (la forme verte et rouge),le boss à un nombre de point de vie qui est dans l'intervalle "la moitié de ses points de vie et le quart de ses points de vie", il tire plus rapidement. La dernière forme (la forme rouge), le boss à un nombre de point de vie inférieur ou égal au quart de ses points de vie et il tire encore plus rapidement. Le boss est seul sur le terrain et le portail apparaît lorsqu'il est vaincu.

- ➢ Les missiles du boss, ceux-ci sont destructibles par le vaisseau du joueur, ils faut donc les esquiver ou les détruire pour ne pas prendre de dégâts.

- ➢ Le portail, celui-ci mène à une nouvelle dimension lorsqu'il est pris par le joueur. Une fois le joueur dans la nouvelle dimension, la partie reprend et les vaisseaux et bonus restants reprennent leur comportement normal et arrivent progressivement sur l'écran. Si le joueur ne prends pas le portail, la partie se termine. On ne peut pas tirer sur ce sprite.

- ➢ Pour avoir une meilleur visibilitée sur le jeu et ce qui se passe à l'écran, nous avons créé des explosions qui diffèrent pour chaque sprite.

Nous tenons également à préciser que tous les sprites utilisés dans notre jeu, à l'exception du deuxième fond du jeu, ont été réalisés par nos soins et en aucun cas téléchargés sur internet. Ils ont été inspirés de films, jeux ou bien même de dessin animés ainsi que des modèles fournis par le sujet.

Nous avons réalisé un diagramme représentant les différents sprites créés par nous même pour la plupart et utilisés dans notre jeu en annexe.

3. Possibilitées d'extensions

Nous avions d'autres idées mais à cause de la contrainte de temps, nous n'avons pas pu les réaliser.

- ➢ Un menu permettant de "customiser", de personnaliser sa propre expérience de jeu : choisir des vaisseaux, fonds, différents missiles, différents ennemis...

- ➢ Des missiles plus performants (pouvoir tirer plusieurs missiles en même temps)

- ➢ Rajouter plusieurs boss ou bien d'autres types d'ennemis ce qui rendrai notre jeu plus complet.

- ➢ Peut-être un multi-joueur même si nous pensons que cela prendrait un temps considérable (par exemple, deux joueurs sur une même machine).

- ➢ Rajouter de la musique et des sons. (lors de la destruction d'un vaisseau par exemple ou juste une musique de fond)

- ➢ Une explosion pour le vaisseau du joueur, en effet, nous n'avons pas eu le temps de l'implémenter correctement car cette amélioration à amené un certain nombre de problèmes.

Le reste de nos idées ont été réalisées dans les temps impartis.

4. Problèmes rencontrés

Durant ce projet, nous avons rencontrés plusieurs problèmes récurrents tels que le découpage modulaire, nous avions beaucoup de mal à savoir si notre découpage était pertinent et efficace. Il nous a fallu plusieurs essais avant d'en trouver un qui nous paraissait cohérent. (car nous avons découpé notre code à la fin et non au fur-et-à mesure)
Il y un problème qui à été assez chronophage pour nous : le "buffer overflow".
En effet, ce problème n'apparaissait que sur l'une de nos deux machines et lorsque l'on a demandé à notre chargé de TP (Clément Beysson) de compiler notre code, il à eu cette même erreur.
Régler cette erreur est particulièrement difficile car elle n'apparaît sur aucune de nos machine actuellement ce qui nous empêche de travailler dessus.

Nous avons donc essayé de modifier des allocations de mémoire mais nous ne savons pas actuellement si le problème est réglé et nous espérons qu'il n'apparaîtra pas sur votre machine.
Un problème récurrent était bien évidemment l'optimisation du code.
Par ailleurs, la vidéo que l'on doit réaliser nous à permis de rencontrer un nouveau problème : les missiles, une fois sorti du terrain, était encore visible et donc, s'ils touchaient un ennemi, celui-ci était détruit, nous avons bien évidemment résolu ce problème.

5. Architecture du code

Nous avons réalisé un diagramme représentant l'arborescence de nos fichiers en annexe.

6. Compétences acquises

Ce projet nous a permis d'apprendre à découper notre code en module ce qui facilite grandement l'organisation de celui-ci. De plus notre code est beaucoup plus lisible et il est donc beaucoup plus facile de se repérer.
Par ailleurs, savoir travailler en groupe est une qualité indispensable, surtout dans le domaine de l'informatique, il est donc important de développer celle-ci.
De plus ce projet nous a appris à utiliser Git et les bibliothèques SDL.
Nous avons également découvert le concept de boucle de jeu.

7. Travail

Nous avons travaillé ensemble sur chaque étape de notre projet apportant des idées ensemble et contribuant ensemble à la bonne réalisation de celui-ci.

Pour travailler ensemble, nous avons utilisé principalement l'application "Discord" et sa fonction "partage d'écran".
Malheureusement l'écriture du code n'a été réalisée que sur une machine car l'un de nous deux ne pouvais pas coder en même temps de communiquer, mais nous étions toujours deux à réfléchir au code et à aucun moment une personne du groupe n'a travaillé tout seul. A l'heure actuelle, ce problème n'est toujours pas résolu malgré des recherches incessantes pour ce faire.
Nous avons également utilisé "Git" ce qui nous a permis de sauver le code écrit et de le tester sur nos deux machines.

Depuis le début des améliorations, nous nous retrouvions les après-midi de 14h à 17h voir 18h, week-end compris pour travailler et réfléchir sur le projet.
Les niveaux "basiques" du projet n'ont pas été d'une grande difficulté à l'exception du découpage du code, seules les améliorations nous ont réellement posé problème.

Par ailleurs, nous tenons à préciser que les vidéos sont saccadées mais ce n'est pas le cas du jeu en lui même, la capture vidéo en est responsable.

IOPETI Hugo
YVOZ Ludovic

8. Annexe

Diagramme 1

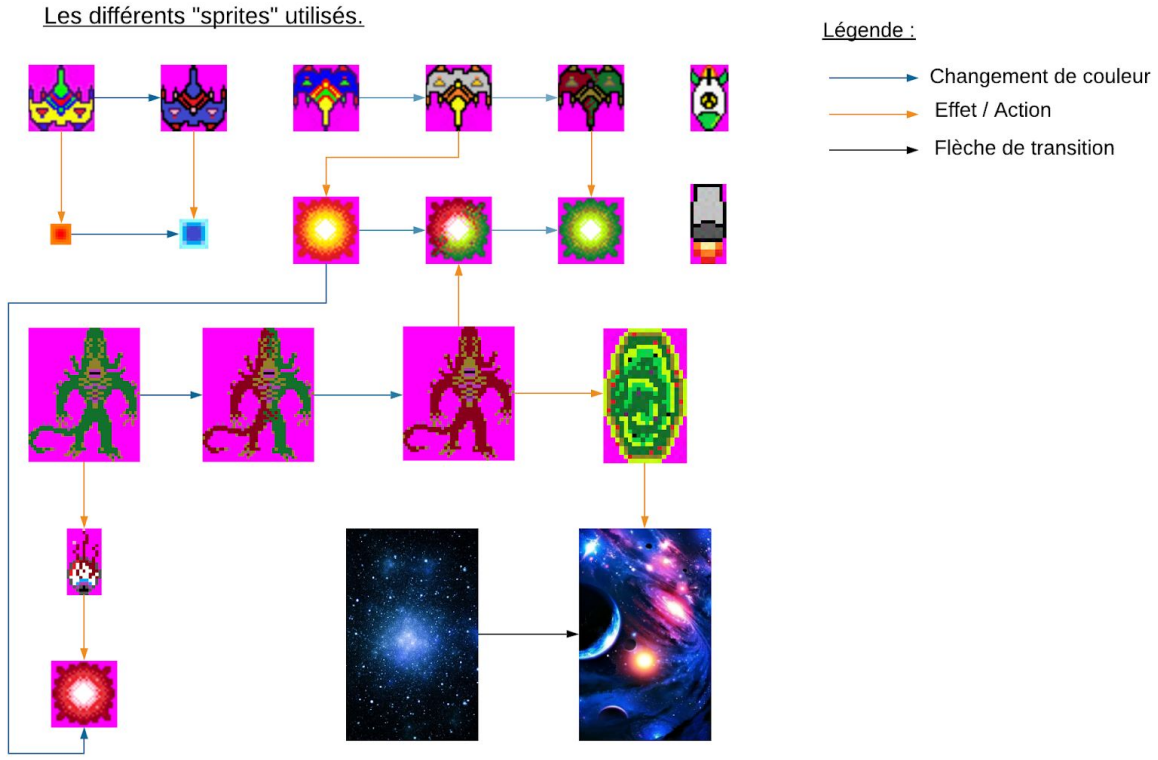

Diagramme 2