Slower Than Light

Grégoire DHIMOÏLA, Léo JUGUET, Tristan PARCOLLET.

23 février 2023

Résumé

Dans ce rapport, nous présentons la construction de notre jeu vidéo, fortement inspiré de Faster Than Light. Actuellement, seule la première partie du projet est implémentée.

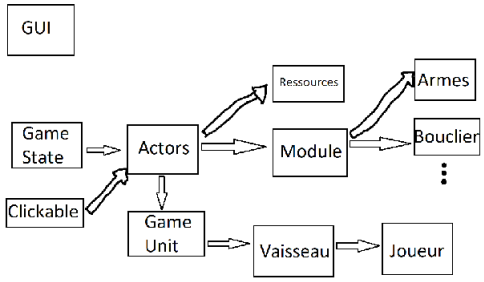
1 Introduction

Le but du projet est d'implémenter un jeu de type Real Time Strategy (RTS). Pour cela, nous avons décidé de nous inspirer de FTL et de l'adapter aux besoins d'un RTS, en ajoutant par exemple une mécanique de minage pour récupérer des ressources, et des déplacements continus plutôt que discrets comme dans le jeu d'origine.

2 Le Backend

2.1 L'architecture globale

(work in progress)



Un game state, s'occupe d'afficher le jeu et de maintenir les actors qui sont en jeu.
Les actors, sont les entités du jeu qui peuvent etre affiche.
Game Unit est une class qui implémente des fonctions de bases pour les unités du jeu.

Le joueur sera a terme, sans doute géré par un Controller, qui s'occupera d'envoyer les instructions au vaisseau du joueur.

2.2 Le Gameplay

(work in progress)

Le joueur doit pouvoir se déplacer sur la carte. Il doit y avoir des collisions entre les astéroides et les vaisseaux. Les astéroides doivent bouger de manière à ce que le joueur puisse les éviter assez facilement. Le joueur doit pouvoir sélectionner une cible (astéroide pour miner, ennemi pour attaquer, allié pour). Il doit aussi pouvoir sélectionner les modules de son vaisseau pour réassigner la répartition de la puissance. Les dégats sont d'abord appliqués au bouclier, qui se régénère au cours du temps, et lorsqu'il est tombé, ils s'appliquent au vaisseau et réduisent la puissance disponible. Les ennemis, les alliés, les marchands, ... doivent avoir une IA dépendante de leur caste. Le joueur peut améliorer son vaisseau n'importe quand, il peut acheter et vendre des modules chez un marchand. Le joueur peut gérer la priorité de l'assignation de sa puissance, pour que lorsqu'il est endommagé, certains modules soient impactés après les autres.

2.3 Gameplay actuel

Le joueur peut se déplacer, Il peut sélectionner des ennemis, et éventuellement les tuer. Les ennemis se déplacent aléatoirement. Si le joueur est assez proche, les ennemis suivent le joueur.

2.4 Aspects de base actuellement implémentés

2.5 GUI

Une gui (game user interface) basique est proche d'avoir une implémentation ancrée sur la branch main. Elle comprends des boutons, des textBox (text éditable), des verticalBox (container affichant ces éléments en colonne) et des comboBox (Un bouton de choix multiples) Tous les éléments de la gui héritent d'une classe UIComponent qui comporte les fonctions et variable nécessaire pour tous les gui. Certains élément ont des évènement Bindable, cela signifie que l'on peut assigner une fonction a lancer quand une certaine action se déroule. Ainsi on peut facilement exécuter une fonction lorsqu'un bouton est clické ou quand un text est entrée dans la textBox.

2.6 Map

La génération de bruit de Perlin à n dimensions, qui sera centarl dans presque tout ce qui sera aléatoire (génération d'astéroides, de vaisseaux ennemis et alliés, déplacements, ...)

les vaisseaux les joueurs les armes (même si pas encore utilisées, elles sont là) la boucle de jeu principale

3 Les Graphismes

3.1 Le fond pas si diffus cosmologique

Pour générer cette image, on commence par remarquer que le bruit de perlin, c'est quand même très joli. Ensuite, on essaye de faire en sorte que le bruit de perlin code pour un angle, on met un paquet de particules dedans, et on remarque que l'image formée par les trajectoires des particules est très jolie. Cependant les trajectoires convergent. On essaye donc de bidouiller, et on rajoute un autre bruit qui code pour la norme des vitesses. Ca ne regle pas le problème de convergence. On rajoute une troisième dimention au bruit, afin qu'il évolue au cours du temps. On obtient alors des images très jolie qui ressemblent à des nébuleuses et qui conviendront.

Maintenant, on ne peut pas charger une si grande image en mémoire. On va donc la découper en chunks que l'on chargera dynamiquement en fonction de là où se trouve le joueur. Cela permet également d'exploiter l'aspect périodique du bruit et de l'image générée, afin de faire une map virtuellement infinie.

4 Les problèmes actuelles

Actuellement, il y a de forte chance pour qu'il y ait une fuite de mémoire (une implémentassions d'un ressource manager devrait aider a régler ces problèmes (tel que le chargement en double de texture ou le non déchargement de ces dernières). De plus le jeu se ferme lorsqu'un ennemis est tué. Enfin, un fix sur la souris doit être effectué.