Projet Prog 2 - Second rendu

Grégoire DHIMOÏLA, Léo JUGUET, Tristan PARCOLLET

25 mars 2023

1 Introduction

Ce rapport contient les avancées dans notre jeu vidéo, ainsi que les plans pour l'avenir. Pour rappel, le but du projet est de réaliser un jeu de type RTS fortement inspiré du jeu Faster Than Light et adapté aux besoins d'un RTS.

2 Le jeu final

2.1 Vue générale

Le jeu consistera en plusieurs clans qui s'affronteront dans l'espace avec pour but la destruction de la base adverse. Le joueur contrôlera l'un des clans. Chaque clan aura à sa disposition une base, quelques vaisseaux-mères, et une flotte de drones/petits vaisseaux. Le monde contiendra diverses ceintures d'astéroides, parmis lesquels certains seront minables.

2.2 Les différentes unités

Les vaisseaux-mères pourront être aménagés via un menu adapté et ce sera au joueur de les spécialiser ou non. Ils pourront installer des modules parmis les suivant :

- Les modules de combat consisteront simplement en différentes armes et upgrades.
- Les modules de soutien serviront à la maintenance des drones et des vaisseaux-mères, et potentiellement réhabiliter les épaves.
- Les modules constructeurs serviront à la construction des drones et éventuellement des systèmes de défense. Ils pourront construire des drones avec diverses caractéristiques.
- Les modules de minage permettront d'être plus efficace dans le raffinement des matériaux et ainsi d'en extraire plus par astéroide. Il pourront également servir à scrapper les épaves.

Les drones seront extrêmement basiques et pourront être contrôlés individuellement ou en groupe via une sélection apropriée. Ils pourront attaquer un vaisseau ou un groupe de vaisseaux ennemis, miner des astéroides, et transférer des ressources entre eux, la base, et les vaisseaux-mères.

2.3 La gestion de la base et des ressources

La base consistera en un astéroïde plus gros que les autres et aménagé pour servir de QG à l'état-major des armées en jeu. Elle servira de point de ralliement pour tous les vaisseaux, et permettra de stocker les ressources récoltées. Ce sera éventuellement elle qui permettra de construire les vaisseaux-mères.

La déstruction d'un vaisseau génèrera une épave, minable également, et servant à récupérer principalement du scrap. Elle pourra aussi être réhabilitée par des modules de réparation pour un coût très réduit.

Les ressources seront utiles pour faire fonctionner les vaisseaux-mères, en particulier l'uranium et l'etherum. Les autres ressources serviront à la construction ou à la maintenance.

3 Ce qui est actuellement implémenté

Durant cette période, nous nous sommes majoritairement concentrés sur l'élaboration des outils et comportements qui permettront d'avancer beaucoup plus vite et efficacement lors de la suite du projet. Nous ne nous sommes cependant pas attardés sur l'utilisation de ces outils pour la "démo", qui reste très basique par rapport à tout ce qui est actuellement implémenté. En particulier en ce qui concerne les controllers du joueur et de l'IA qui ont des comportements très basiques.

Actuellement, toutes les actions et tous les comportements sont implémentés mais tous ne sont pas utilisés pour la démo. De même, tous les outils qui serviront à finaliser le jeu sont développés. Les drones peuvent attaquer, miner, transférer des ressources. Les vaisseaux-mères peuvent simplement se déplacer, c'est à leurs modules de faire des actions. Les actions des modules sont fonctionnelles, mais pas encore gérées par le controller.

Tous les types de boutons et de menus qui seront utiles au joueur sont déjà implémentés. En particulier, le menu de construction de modules et d'interaction avec eux, pour ce qui concerne les vaisseaux-mères, et les boutons qui serviront notamment à la navigation entre les différents menus et à la sélection de certaines actions.

La gestion des événements, que ce soit les événements d'input classiques du joueur, ou bien des événements personnalisés, est en place. Typiquement, des acteurs peuvent s'abonner à des événements personnalisés, qui appelleront automatiquement une certaine fonction lors de leur réalisation.

La génération d'aléatoire cohérente qui servira à la génération et au placement des astéroides, des unités et de tous les objets à l'initialisation de la partie est implémentée.

Les outils utiles au contrôle et au bon déroulement du jeu sont implémentés. Il y a le gamestate, la caméra, les controllers, le manager de textures, la GUI, le générateur d'aléatoire et le manager d'événements.

4 Ce qui a été ajouté depuis le dernier rendu

4.1 Le système de caméra

Celui-ci permet de fixer la vue sur un acteur particulier, ou de la fixer sur un point quelconque et de le déplacer manuellement. Elle peut également zoomer ou dézoomer.

4.2 Manager de textures

Il permet la gestion dynamique des textures utiles, permettant de ne pas charger deux fois la même texture, et de décharger une texture non utilisée. (note : cette dernière fonctionnalité n'est pas encore implémentée).

4.3 Event

Nous avons travaillé sur un système de gestion des événements classiques d'input du joueur, et d'événements plus personnalisés. Ce système permet de gérer les entrées du joueur et les évènements qui se déroulent en jeu, et de les associer à des fonctions. Ainsi, il est facile d'executer une fonction dès qu'un évènement est déclenché. Nous avons également une structure de dictionnaire qui maintient l'état courant du clavier.

L'idee est d'avoir pour chaque action un event, typiquement pour deplacer la camera, si on utilise ZQSD, on ne veut pas que la caméra regarde ZQSD à chaque frame pour décider ou non de mettre à jour son état, mais que les événements ZQSD s'occupent eux même d'appeler la mise à jour de la caméra dès qu'ils sont appelés.

Cela permet d'optimiser fortement la complexité du jeu puisque tous les tests de tous les acteurs et de tous les outils sur tous les événements pour savoir quoi faire (écoute active) sont remplacés par une simple écoute passive.

4.4 Controller

Nous avons mis en place un système de contrôleur. Le contrôleur du joueur s'occupe d'écouter les ordres de ce dernier et affecte aux unités qu'il contrôle les actions associées. Le contrôleur de l'IA pour l'instant s'occupe d'assigner des actions aléatoires aux unités qu'il contrôle, mais il pourra être plus sophistiqué à l'avenir.

4.5 GUI

Elle a été complétée par rapport à la dernière fois, et permet maintenant de gérer n'importe quel type de bouton ou d'interface qui sera utile pour la suite du projet.

Un ajout majeur concernant ce point par rapport à la dernière fois est le menu de construction, qui permettra de gérer les modules des vaisseaux-mères.

Pour compléter ce dernier menu un module permettant de scroll sera rajouté prochainement.

Des barres de progression ont également implémentées et serviront principalement à illustrer les barres de vie des unités

4.6 Map

La génération d'aléatoire par bruit de Perlin a permis de créer des cartes aux propriétés intéressantes. Elles sont périodiques, donc si la carte s'avère être trop petite, nous pourrons toujours l'agrandir artificiellement de manière continue. Elles sont également très jolies et font penser à des nébuleuses, permettant la future immersion du joueur. Elles permettront également de servir de base de génération d'autres placements aléatoires, comme le placement des champs d'astéroïdes par exemple.

4.7 Base, ressources et comportements liés

Nous avons implémenté la base et une plus grande diversité de ressources. Le système de minage a également été mis à jour, donnant une capacité de transport maximale aux vaisseaux, qui, dès qu'elle est atteinte, enclenche automatiquement un comportement de transfert des ressources vers la base.

4.8 Modules

Les modules des vaisseaux-mères sont en cours d'implémentation. Leur structure est prête et fonctionnelle mais ils ne sont pas encore utilisés. On peut cependant les construire grâce au menu qui apparaît lorsque l'on clique sur un vaisseau-mère.

4.9 Collisions

Nous avons également introduit un système de collision entre les acteurs. Pour l'instant, il s'agit du simple algorithme naïf en $O(n^2)$, mais il pourra être optimisé dans la suite si jamais les collisions sont trop lourdes, soit en ne considérant que les acteurs affichés, soit en implémentant des quadtree.

Les collisions se font avec rebond pour que ce soit visuellement plus réaliste que le simple arrêt. Cela permet également de limiter les blocages.

5 Ce qu'il reste à faire

Il reste à faire les comportements particuliers des vaisseaux-mères en leur intégrant les modules et les inputs associés via le controller. Il faut également faire des sprites pour que le jeu soit effectivement jouable, et non pas seulement théoriquement. En ce qui concerne les sprites, on peut également faire de simples animations, typiquement sur les réacteurs des vaisseaux.

Actuellement, les actions ne dépendent pas du temps que la frame a mis à se dérouler. Ceci est amené à changer de sorte que les actions soient constantes par rapport au temps du joueur (en secondes) et non pas au temps du jeu (en frame).

Il faut intégrer la nouvelle gestion d'événements, ainsi qu'un UIComponent qui permet le scroll afin d'améliorer le menu de construction des modules à la GUI.

De plus si l'on a le temps, on envisage d'ajouter un système de contrainte à la GUI afin de plus facilement poser les différents éléments. On aimerait également pouvoir ajouter des modules a l'aide de fichier texte pour facilement en rajouter avec différents paramètres.