BONNEVILLE Geoffrey

Ada Stars

16 février 2016

Contexte

Ce rapport synthétise le travail effectué pour réaliser le “Jeu de Star Wars” en ADA. Il accompagne les sources du projet.

Le projet est au moment de la soumission publiquement accessible sur gitlab et github :

https://github.com/G-Ray/ada-stars

https://gitlab.com/G-Ray/ada-stars

Contenu

Architecture du projet

Le projet est décomposé en 9 modules :

Config : Il s'agit des valeurs paramétrables pour modifier le comportement du jeu. On peut ainsi spécifier plus d'astéroides, une vitesse plus rapide, une meilleure résolution, une plus longue distance d'affichage etc...

Main : On initialise SDL dans le main. Le jeu est initiliasé, puis est lancé dans une boucle infinie. C'est dans cette boucle que l'on réagit à l'appui de la touche "espace" pour mettre en pause, ou reprendre le jeu.

Asteroid : Ce package initialise les propriétés des astéroïdes aléatoirement : position en x, y, z, couleur, taille.

Collision : Il s'agit d'une task qui se charge de détecter une collision de manière assez intelligente : Seuls les asteroid suffisamment proches sont analysés. On calcule ensuite sur ces astéroides la distance en 3D qui les sépare du vaisseau, en prenant en compte le rayon de chacun.

Move : On trouve dans Move une task qui déplace le vaisseau en fonction des entrées clavier du joueur. Afin d'obtenir un mouvement plus réaliste, move simule une inertie lorsque le joueur arrête d'accélérer dans un sens. (x y et z).

Scene : Scene contient tout ce qui permet d'afficher la scène complète. Il calcule également le temps nécessaire en millisecondes pour obtenir une image. On obtient ainsi un nombre d'images par seconde (FPS) qui est affiché dans la console chaque seconde.

Spacecraft : Il s'agit du vaisseau spatial que le joueur contrôle. Il dispose de 3 vecteurs vitesses en x,y et z, ainsi que ses positions, toujours en x, y et z. Le vaisseau est un objet protégé. On doit donc passer par des "Getters" et "Setters" pour modifier ou lire ses propriétés.

Timer : Timer est une task qui décompte le temps pour arriver à la fin de la partie. Par défaut dans la configuration, il s'agit de 30 secondes. Timer met à jour le compteur si le joueur décide de mettre le jeu en pause.

State : On trouve dans "state" une seule variable partagée : Terminated qui indique si le jeu doit être terminé. Terminated peut être mis à vrai lors d'une collision ou bien lorsque le timer est terminé (le joueur à gagné).

Utilisation

Pour une utilisation optimale, il est préférable d'ajuster la configuration selon les capacités de sa machine. Les paramètres par défaut permettent un affichage à 60fps sur une machine modeste. Les paramètres impactants le plus les performances sont : la résolution (taille de la fenêtre), le niveau de détail des astéroides, ainsi que la distance d'affichage des astéroides.

Fonctionnalités

Les fonctionnalités demandées sont toutes implémentées, hormis le fait que le jeu doit s'arrêter lorsque le vaisseau du joueur sort du champ d'astéroides. Cependant, l'univers devient progressivement rouge pour signaler au joueur qu'il doit revenir dans le champ d'astéroide. On pourrait ajouter un compteur lorsque le joueur s'éloigne et au bout de 3 secondes il perdrait.

Quelques précisions:

- Les astéroides sont des sphères de couleurs aléatoires
- La source de lumière choisie permet un meilleur éclairage de la scène.

Le programme a été optimisé de manière à être fluide (60 Images/s) sur une machine modeste, grâce à du clipping et à un calcul de collision adapté.

Bugs

La manière de sortir du programme n'est pas propre. Il y a des exceptions qui sont générées avec l'appel à SDL_SDL_h.SDL_Quit; Je n'ai pas réussi à corriger cela.git