Conception d’un jeu vidéo simple avec OpenGL

Traitement et Synthèse d’image

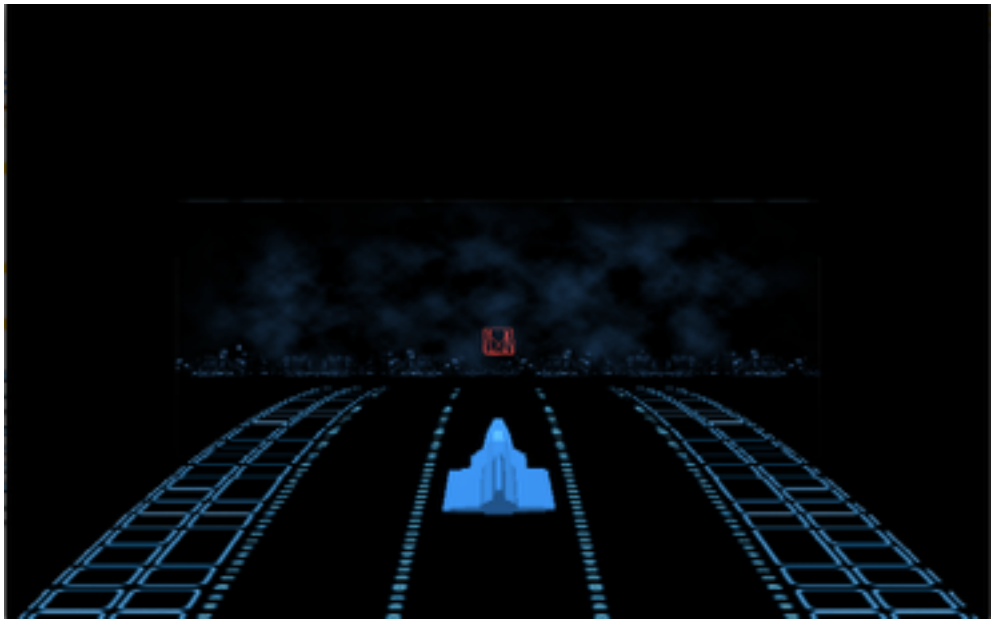
Sohier Charles-Antoine
Veleine Mickaël

20 janvier 2014 – 4ETI – Groupe D

I. Introduction

Dans ce projet, nous mettons à profit nos connaissances de programmation et de l'API OpenGL, dans la conception d'un jeu vidéo rudimentaire. Nous avons choisi de développer un jeu typé arcade dont les mécanismes sont très simples et très intuitifs.

II. Principe du jeu



Notre jeu est un jeu de course. L'utilisateur contrôle un véhicule bleu qui se déplace sur une route infinie. Cette route est composée de trois voies. L'utilisateur peut à l'aide du clavier déplacer son véhicule sur les trois voies. Un cube apparait à intervalle régulier sur l'une des voies.

Le but du jeu consiste a éviter que le cube et le véhicule se rencontrent. À terme, la difficulté du jeu vient de la vitesse d'apparition du cube et du véhicule qui augmentent. Cependant cette fonctionnalité n'a pas encore été implémentée dans le jeu.

III. Mécanismes du jeu

a. La route infinie

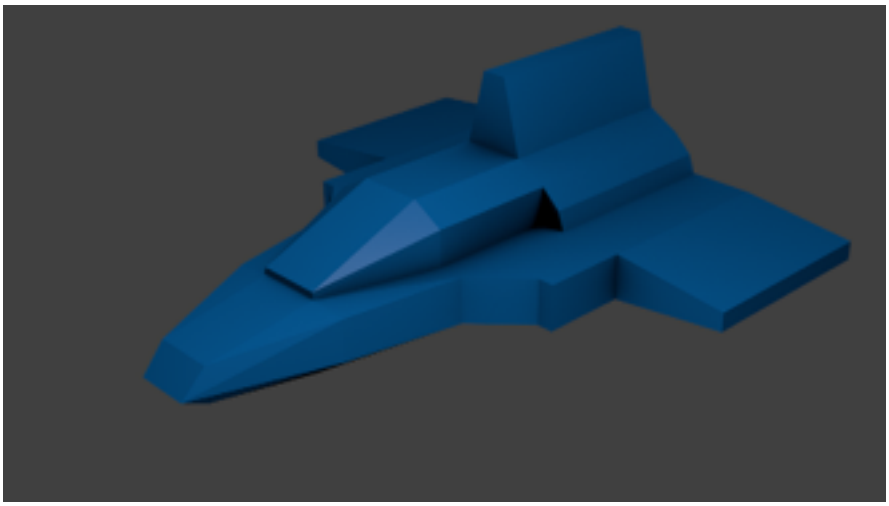
La modélisation de la route a été la première étape dans la réalisation du projet. De ce fait, la génération de cette route a orienté nos choix de solutions pour les mécanismes du jeu. Cette route est en réalité un cylindre en rotation sur l'axe de sa longueur. On simule ainsi la vitesse en faisant tourner plus ou moins vite le cylindre.

b. Le cube

Sur la route gravite un cube qui lui aussi suit la rotation comme la route. Afin que la rotation soit plus facile a implémentée le cube a ses points positionnés loin de l'origine de l'objet. Le cube change de position a chaque révolution lorsque il se retrouve en dehors du champ de la caméra. À ce moment, le cube prend une des trois voies aléatoirement.

c. Le véhicule

Au premier plan de l'affiche on trouve le véhicule ou vaisseau qu'incarne l'utilisateur. Cet objet ne subit pas de rotation dans le jeu. Ca position peut varier entre les trois voies de la route. Comme il s'agit de l'objet susceptible de bouger le plus, celui-ci possède des phases de transition dans ces déplacements.



d. Condition de perte

Le jeu ne possède pas de condition de victoire mais des conditions de perte. On perd le jeu dès lors que le vaisseau et le cube sont entrés trois fois en collision. le test de collision se fait une fois par tour, dès que le cube et le vaisseau sont au même niveau.

IV. Implémentation

a. La caméra

La caméra est « fixe » dans l'espace. Nous avons cependant modifier les paramètres de distance focale. Afin d'agrandir la FOV (en anglais Field of View) on a augmenté le paramètre de fovy de la fonction gluPerspective.

b. La route infinie

Dans la fonction generate_cylinder() on génère les vertex qui composent la route. On a choisi de définir ces vertex à l'aide de boucle et de donc de mettre en équation les points du cylindre. Cela nous permet par exemple, de choisir le nombre de vertex qui composent le cylindre et de modifier assez facilement ce paramètre. Plus la route possède de points plus elle apparait lisse à l'écran.

c. Cube et vaisseau

Les objets cube et vaisseau sont importés depuis des fichiers .OBJ. Le vaisseau et le cube ont été modélisés sur un logiciel de CAO : Blender. On importe ces objets avec la bibliothèque glm. Nous n'avons cependant pas pu importer de texture avec cette méthode.

d. Texture

Afin de simplifier l'implémentation des textures nous avons pris l'option de générer ces derniers comme des « sprites » , c'est-à-dire une seule image sur laquelle se trouve les différentes textures. Sur les abscisses allant de 0 à 0.5 on retrouve ainsi la texture de la route et sur les abscisses de 0.5 à 1 on retrouve la texture du fond.