PROJET D'INFORMATIQUE 3D DE 2A

BR Space Program

Fin de 2A, 2024

Emre Ucar et Thomas Sauvage

ÉCOLE POLYTECHNIQUE
IP PARIS

TABLE DES MATIÈRES

INTRODUCTION

Dans notre scène 3D, un astronaute se balade librement dans son système solaire. C'est alors qu'il découvre un cube aux propriétés très particulières...

Il va alors pouvoir défier les lois de la physique euclidienne pour entamer un voyage interstellaire !

1
LE VAISSEAU SPATIAL

Le joueur se déplace dans le monde en vaisseau spatial. Le modèle du vaisseau a été trouvé sur internet, il est libre de droit.

1.1 Contrôles

Le vaisseau spatial peut être contrôlé avec les touches suivantes :

— z : Avant
— s : Arrière
— q : Gauche
— d : Droite
— a : Rotation gauche
— e : Rotation droite
— f : Haut
— v : Bas

— Maj + f : Plein écran
— Dans les paramètres GUI, il est possible d'activer la caméra libre

1.2 Animation des flammes

Le modèle des flammes du vaisseau spatial ont été trouvés sur internet et sont libre de droits. Nous affichons plusieurs fois ce modèle dans des couleurs différentes pour créer des flammes plus esthétiques.

Nous avons aussi ajouté un effet de glow sur les flammes, comme décrit dans la partie 4.

Les flammes sont animées. Premièrement, la flamme de chaque propulseur ne s'affiche que lorsque ce dernier est allumé par le joueur (par exemple, les propulseurs de gauche s'activent quand l'utilisateur va à droite).

Ensuite, les flammes ont un léger mouvement de crépitement pour des raisons esthétique. Ce dernier est réalisé en appliquant un modificateur de scale sur le modèle, il varie autour de 1 selon une somme de sinusoïdales, affin de créer un mouvement pseudo erratique.

2
GÉNÉRATION PROCÉDURALE DE PLANÈTES

Les surface des planètes sont générées de la manière suivante : Premièrement, on créé une sphère avec un rayon donné. Ensuite, on modifie l'altitude de chaque point selon un bruit de perlin.

Les paramètres de ce bruit peuvent être modifiés pour générer différentes planètes. Il est aussi possible de faire varier la couleur et l'atmosphère (voir la partie 4).

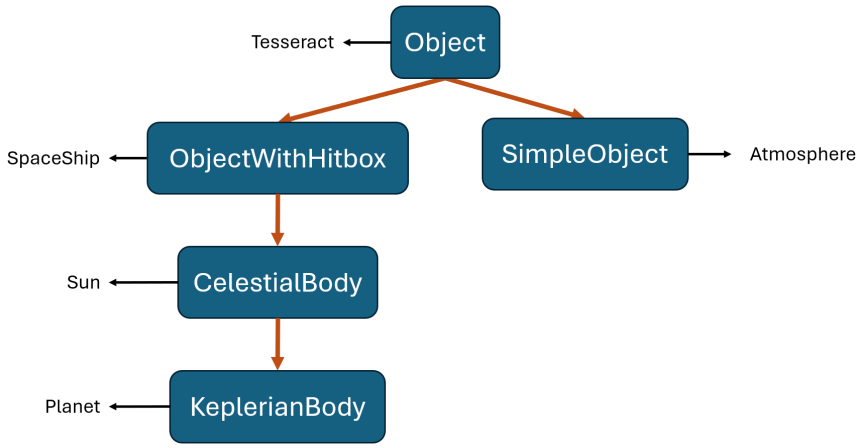

Figure 1 – Les différentes classes utilisées pour représenter les objets de nos scènes

3 SIMULATION PHYSIQUE

Nous avons mit en place une simulation de la gravité Newtonienne à laquelle le vaisseau spacial et les astres du monde du problème à trois corps sont soumis.

3.1 Les objets massique (CelestialBody)

Les objets possédant une masse non négligeable dérivent tous de la classe CelestialBody qui défini une masse et une position. Ils sont stockés dans une liste. Les objets soumis à la gravité n'ont qu'à parcourir cette liste pour calculer la force de gravitation.

3.2 Les objets à trajectoire précalculés (KeplerianBody)

En plus de posséder une masse, les planètes en orbite autour du soleil possèdent une trajectoire circulaire. Il est tout à fait possible de simuler la force de gravité sur ces planètes, mais pour réduire la quantité de calculs, nous avons décidé de leur imposer un mouvement circulaire précalculé.

3.3 Simulation des collisions

Nous avons implémenté une simulation de la collision entre deux sphères. Dans notre cas, il n'y a de collisions qu'entre le vaisseau spatial et l'un des différents objets.

Les objets pouvant entrer en collision dérivent de la classe ObjectWithHitbox qui définit une position ainsi que le rayon de la hitbox (qui est sphérique).

Deux objets en collision sont détectés de manière très efficace en comparant la distance entre les centres et la somme des deux rayons.

S'il y a collision, on détermine tout aussi efficacement la normale à la sphère de l'objet et nous réalisons une réflexion du vecteur vitesse du vaisseau spatial. Nous réduisons l'amplitude du vecteur d'un certain pourcentage pour simuler la perte d'énergie lors du rebond.

4

LES ATMOSPHÈRES DES PLANÈTES ET L'EFFET DE GLOW DES SOLEILS

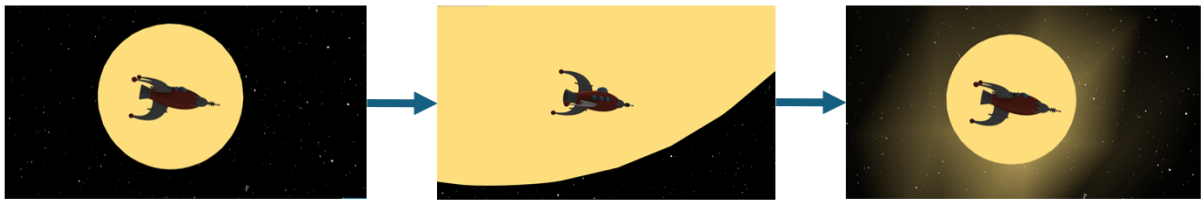

Figure 2 – Les trois étapes de la réalisation de l'effet de glow. À gauche le soleil, au milieu on applique au dessus une grande sphère de même couleur, à droite on applique le shader

4.1 Un shader approprié

L'effet de glow et de d'atmosphère est réalisé de la même manière, il s'agit d'une sphère (ou toute autre forme), un peu plus grande que l'objet. Ensuite, on applique un effet de transparence graduel autour de cette sphère : totalement transparent sur le bord, plus opaque au centre.

Cet effet est réalisé grâce à un shader spécialement conçu pour la sphère. La valeur du alpha à donner à chaque point de la sphère est calculé par le vertex shader. L'idée est de calculer la valeur absolue du produit scalaire entre le vecteur camera-vertex et la normale au vertex. Ce dernier sera très grand au centre de la sphère et petit sur ces bords. On peut ensuite appliquer une fonction à ce produit scalaire pour améliorer l'esthétique du résultat, nous avons décidé d'y appliquer la fonction $x \to x^{10}$.

4.2 Affichage d'objets transparent

Pour afficher correctement les objets transparents, il a fallu activer les options d'OpenGL correspondant (notamment les fonctions de blending). Il faut aussi tracer dans un premier temps les objets opaque, puis les objets transparents du plus lointain au plus proche.

Pour simplifier les calculs et avoir de meilleurs performances, nous trions les objets transparent par rapport à leur centre et pas vertex par vertex. Ceci pourrait résulter en des bugs graphique, mais nos objets transparent étant suffisamment éloignés et sphérique (ou ellipsoïdal), ces bugs ne se produisent pas.

5

CHANGER DE SYSTÈME SOLAIRE À TRAVERS LE TESSERACT

5.1 Le changement de monde

Le tesseract est l'élément central de notre simulation 3D. C'est un cube au travers duquel on peut entrer dans des mondes différents. Le tesseract permet au vaisseau de librement se déplacer entre six mondes (un pour chaque face). Dans un monde donné, le tesseract est fermé sur cinq faces et ouvert sur l'une d'elles. Les objets contenus dans un monde sont visibles à travers les faces internes du tesseract. À l'intérieur du tesseract, le vaisseau peut voir les six mondes en tandem.

Le cube en lui-même est composé d'une armature et de faces extérieures. Il possède aussi des faces internes invisibles mais nécessaires pour les effets d'optique du tesseract.

Les mondes (instances de la classe *World*) sont complètement indépendants les uns des autres au niveau des calculs physiques de collision (ObjectWithHitbox) et de gravité (CelestialBody). Deux objets de deux mondes différents peuvent aussi se retrouver aux mêmes coordonnés. Les mondes possèdent aussi leur propre skybox et le rendu des objets transparents est aussi indépendant.

Le rendu paradoxal des mondes nécessite beaucoup de manipulations sur les buffers de couleur et de profondeur mais surtout du stencil buffer de OpenGL.

La scène est rendue différemment selon si le vaisseau (pour être plus précis, la caméra) se trouve ou non dans le tesseract.

Dans le premier cas, chaque monde est dessiné uniquement sur les pixels visibles de la face idoine. On dessine donc d'abord la face sur le stencil (pas sur les buffers de couleur et de profondeur), puis on autorise uniquement les rendus (en couleur et en profondeur) correspondant à cette partie de l'écran.

Dans le second cas, une manipulation supplémentaire doit être faite sur le stencil pour faire intersecter la face interne d'un monde avec la face de sortie du monde ambiant au vaisseau.

Pour éviter certaines incohérences, l'ordre de rendu des éléments du tesseract (faces extérieures, arêtes, mondes intérieurs) est changé selon que la caméra regarde ou non le cube à travers au niveau d'une face fermée ou bien de la face ouverte. Le monde ambiant est rendu de façon habituelle.

Mise à part le changement d'éclairage sur le vaisseau, la transition d'un mode de rendu à l'autre est imperceptible.

5.2 Les différents monde

En passant au travers du Tesseract, notre astronaute peut découvrir différents mondes. Ils dérivent chacun de la classe World.

- Un système solaire proche du notre
- Un autre composé d'une naine rouge
- Et un avec une naine blanche
- Un soleil noir plongé dans une lumière blanche inspiré de Geidi Prime, planète de l'univers de Dune
- Un problème à trois corps dont le mouvement est simulé