Phase 3 - Projet 8INF935

Alaoui, Mohamed-Wajih, Delort Tristan, Miras Lucas

Novembre 2021

1 Introduction

L'objectif dans cette phase a été d'implémenter les classes RigidBody, et toute la physique de rotation qui va avec. Il faut aussi créer des force generators et des demos simples pour montrer que tout fonctionne.

2 GUI

Le rendu graphique de la phase 2 introduisait de la profondeur mais sans être très attrayant et il était dur de discerner l'orientation d'un objet car il était d'une couleur uni dans ombre ni contraste. De plus le seul objet 3D était un Cube. Après un peu de lecture de fichier .obj, nous avons rajoutés un ObjMesh qui supporte des fichier .obj simples (le cas ou les attributs de face sont sous la forme XX/XX/XX n'est pas supporté). Un fichier teapot.obj est fournis avec le rendu et il suffit de le passer en argument à l'exécutable pour l'utiliser.

Ensuite, une lumière simple a été ajoutée (avec un point fictif émetteur de lumière). Le fichier .obj utilisé ne fournissait pas les normales des faces. Plutôt que d'en trouvé un nouveau, j'ai décider de faire le calcul des normales dans mon tout premier Geometry Shader. L'algorithme simple peut se retrouver ici : `https://www.khronos.org/opengl/wiki/Calculating_a_Surface_Normal`. Il reste encore quelques petits défaut avec ce shader, on le voit en faisant pivoter la caméra autour d'un cube, la lumière change en fonction de la caméra ce qui n'est pas normal.

3 Matrix & Quaternion

L'implémentation des Matrices est très classique mais la difficulté que nous nous somme rajouter est lors de l'utlisation d'une template. En effet, les classes Matrix3x4 et Matrix3x3 héritent d'une classe mère `Matrix<T, N, M>` avec T un type, et N et M les dimensions. Ainsi, on peut avoir une implémentation générique du constructeur sans tableau dynamiques (donc pas d'allocations ou de vectors). Pour les Quaternions, peu de difficultés car il suffisait de suivre le cours.

4 RigidBody

Le Rigidbody reprend en grande partie les concepts d'une particule simple qu'on a créés pour le premier travail mais apporte tout de même plusieurs nouvelles précisions. Le Rigidbody étant plus complexe qu'une particule, les forces doivent être appliqué sur une zone monde pour intégrer les forces angulaires pour la rotation de cet objet "complexe" (torque) en supplément des forces linéaires pour sa translation. Les objets Rigidbody ont donc une vélocité et une rotation à intégrer dans la simulation. Le modèle présenté en cours et dans le livre de Millington a été utilisé pour créer cet objet, la démarche a été donc grandement été simplifier.

5 Force Generators

L'implémentation des Forces Generators s'est faite assez simplement en reprenant le modèle proposé dans le cours et en généralisant le principe appliqué à la particule aux RigidBodys. Nous avons implémenté la force de gravité ainsi que la force de rappel d'un ressort. Ensuite, la mécanique rotationnelle découle de ce qui a déjà été implémenté : En utilisant la fonction `AddForceAtBodyPoint` qui prend en arguments une force et le point d'application de la force, on peut générer les couples et moments d'inertie.

6 Demo

Lors des 2 dernières phase, tout le moteur physique était manipulé dans le `main` et le code était difficilement lisible. Nous avons donc commencé par rajouter un singleton PhysicsEngine pour regrouper les étapes d'itération, de calcul des forces

et de résolution des contacts pour tous les objets en un seul appel à une fonction `Update()`. Le choix du patron de conception du singleton paraissait logique étant donné que dans un moteur de jeu, un moteur physique serait un "Système" (S dans ECS) et on veut toujours un et un seul système physique. Ensuite, une classe `Demo` fait office de "Jeu" en regroupant toutes les variables nécessaires à la démonstration. Dans cette classe, 3 scènes de démonstration ont été créées pour mettre en avant chacun des scripts développés durant cette phase. Il est possible de naviguer entre ces 3 scène et de les redémarrer à volonté. Les 3 scènes sont :

- Sample Demo : Un Maillage reçoit une impulsion (une force multipliée par un grand facteur sur une seule frame) à un point aléatoire en coordonnées monde entre (-1, 0, -1) et (1, 0, 1).
- Collision Demo : Deux boites de poids différent se dirige l'une vers l'autre, et quand la distance entre les deux est nulle, une impulsion dans la direction donnée par les deux centres de gravité des deux boites simule un collision.
- Spring Demo : 2 objets sont attachés entre eux par un ressort. L'objet le plus haut est aussi relié à une ancre (et non un objet) au dessus de lui.

Enfin, les contrôles de la caméra ont été améliorés avec notamment la possibilité de la déplacer. Les 3 valeurs sont en coordonnées cylindriques (angle, hauteur, et rayon). Il est aussi possible de changer le point vers lequel la caméra est tourné, et dans les 2 premières démonstrations il est même possible de suivre un des objets de la scène.