Département d'Informatique et de Recherche Opérationnelle
Université de Montréal

Mikhail Bessmeltsev
Bureaux : AA3357
bmpix@iro.umontreal.ca

IFT3355 : Infographie
Travail Pratique 3 : Un jeu 3D incomplet
Shading, collisions et animation

Disponible :	Mardi 2 novembre 2021	23 :59
Remise :	Lundi 15 novembre 2021	23 :59

1 Introduction

L'objectif principal de ce TP est d'explorer l'éclairage, l'ombrage et l'animation basée sur la physique des modèles 3D. Pour la partie 1, vous simulerez quatre styles d'ombrage (shading) différents : Gouraud, Phong, Blinn-Phong et les textures. Pour la partie 2, vous compléterez la simulation physique et l'animation des modèles 3D du jeu. Vous pouvez commencer par la partie 1 ou 2 (au choix).

2 Template

- Le répertoire `glsl` contient les vertex et fragment shaders. C'est dans ces fichiers que vous aurez à faire la majorité du travail pour la partie 1.
- Le fichier `TP3_PhysicsUtils.js` contient le code pour la physique et l'animation. C'est dans ce fichier que vous aurez à faire la majorité du travail pour la partie 2.
- Le fichier `TP3.html` est le launcher du jeu.
- Le fichier `TP3.js` contient le code utilisé pour construire la scène et rendre l'environnement du jeu. Vous pourrez faire des changements pour compléter la partie license créative.
- Le fichier `Test.html` est le launcher de la scene test avec references (`.png` et `.mp4`).
- Les répertoires `js` et `build` contient les librairies *JavaScript* requises. Vous n'avez pas à modifier ces fichiers.
- Le répertoire `models` contient les géométries et textures chargés dans la scène.

3 Règles importantes

Tout est permis sauf prendre du code qui ne vous appartient pas. Donc pas de librairies externes, mais vous pouvez utiliser toutes les fonctions de `Three.js` dans le TP sauf les matériaux prédéfinis (eg. `MeshPhongMaterial`). Vous devez toujours utiliser `ShaderMaterial`.

4 Contrôles

- Les touches de clavier R et E réinitialisent (de façon différente) la position des boules de billards.
- Les touches de clavier WASD bougent la position de la lumière.
- La touche de clavier Q réinitialise la position de la lumière.
- Le bouton gauche de la souris vous permet d'appliquer une impulsion à la boule de billard choisi. (La collision cercle-segment interne doit être implémenté pour ce fonctionnement)

- Le bouton droit de la souris vous permet de bouger la boule de billard choisi. (La collision cercle-segment interne doit être implémenté pour ce fonctionnement)
- Maintenir le bouton gauche/droite de la souris pour bouger la caméra.
- Le bouton `space` pour démarrer la simulation physique dans la scene test (`Test.html`). Vous devez au moins avoir complété la fonction `applyMovement(...)` pour voir du mouvement.

5 Travail à réaliser

1. **Partie 1 : Ombrage (Shading)** (70 points)
   Les boules de billard 1 à 5 + boule blanche vont être ombré avec la méthode Gouraud, les boules de billard 6 à 10 vont être ombré avec la méthode Phong et 11 à 15 avec Blinn-Phong.
   (a) **Ombrage de Gouraud** (15 pts)
   Dans cette partie, vous commencerez avec un modèle d'éclairage simple et rapide. L'ombrage de Gouraud calcule l'éclairage d'un objet à chaque sommet (dans le vertex shader). Le pipeline graphique standard interpolera ensuite la couleur entre les sommets pour obtenir les couleurs des fragments individuels. Le calcul de l'éclairage à un sommet devrait être basé sur le modèle de réflexion de Phong standard discuté en classe. Vous devez tenir en compte tous les différents composants (diffus, spéculaire et ambiant) du matériau ! Référez-vous aux commentaires dans les fichiers .glsl et les notes de cours pour plus de détail.
   (b) **Ombrage de Phong** (15 pts)
   Cette partie est très similaire à la partie précédente, mais cette fois, l'ombrage est calculé dans le fragment shader au lieu du vertex shader. Calculez tout les vecteurs nécessaires pour l'ombrage de Phong dans le vertex shader et passez cette information au fragment shader. N'oubliez pas de normaliser !
   (c) **Ombrage de Blinn-Phong** (10 pts)
   Encore une fois, modifiez l'ombrage de phong pour calculer l'ombrage de Blinn-Phong. La différence est dans l'utilisation du vecteur mi-chemin.
   (d) **Textures** (5 pts)
   Appliquez la texturisation sur les ombrages précédents. Vous êtes encouragé mais pas obligé de tenir compte de la distorsion de perspective dans la texture. Utilisez les coordonnées UV fournis par le système de rendu dans le vertex shader. Passez ces coordonnées au fragment shader. Les textures fournis sont sous forme de masque (mask) noir et blanc, mais utilisez-les pour l'instant comme une texture qui décrit la couleur des boules. Puisque les masques sont en noir et blanc, il se peut que l'ombrage est peu visible.
   (e) **Textures coloriés** (15 pts)
   Maintenant, utilisez la fonction `mix(...)` de OpenGL pour changer la couleur de la boule en fonction du masque. Chaque boule de billard peut avoir un masque et/ou couleur différent. Référez-vous aux commentaires dans les fichiers .glsl pour plus de détail. N'oubliez pas l'ombrage !
   (f) **License créative** (10 pts + Bonus)
   Implémentez une méthode d'ombrage de votre choix plus complexe que l'ombrage de Blinn-Phong. Idées possibles : Ombrage anisotrope, bump mapping, displacement mapping, matériel diélectrique (verre transparent), matériel métallique, ombrage cel (toon shading), PBR, etc.

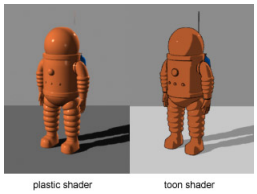

FIGURE 1 – Un exemple de *toon shading* (Wikipedia)

2. **Partie 2 : Animation et collisions** (30 pts)

Le travail à faire dans la partie 2 se trouve uniquement dans `TP3_PhysicsUtils.js`.

(a) **Animation** (10 pts)

Implémentez le mouvement linéaire et la rotation des boules de billard en fonction de leur vélocité et du temps. Les deux fonctions à compléter sont `applyMovement(...)` et `applyRotation(...)`.

Le mouvement linéaire d'un objet dans l'espace est décrite par l'intégrale suivante :

$$\vec{x}(t) = \int \vec{v}(t)dt + \vec{x}_0$$

Pour résoudre cet équation, utilisez la méthode d'Euler. Au lieu de décrire la position absolue d'un objet en fonction du temps $\vec{x}(t)$, nous allons plutôt décrire le changement de position $\Delta\vec{x}$ en fonction du temps t.

$$\Delta\vec{x}(t) = \int_{t_0}^{t_1} \vec{v}(t)dt$$

Si on assume que $\vec{v}$ ne change pas durant le temps Δt : (C'est une approximation, puisque ce n'est pas toujours vrai !)

$$\vec{x}_{t+1} = \vec{v}_t\Delta t + \vec{x}_t$$

Pour la rotation, calculez la circonférence d'une boule de billard de rayon 2 (unités arbitraires) et appliquez une rotation dans la direction du mouvement proportionnelle à $\Delta\vec{x}$. Il est possible que vous ayez à implémenter les textures pour bien voir la rotation des boules de billard.

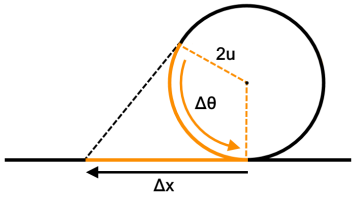

FIGURE 2 – Diagramme de rotation

(b) **Collisions** (20 pts)

Implémentez la détection de collision entre deux boules de billard et la collision d'une boule de billard sur un segment de mur. Les trois fonctions à compléter sont `checkCollisionCircleCircle(...)`, `checkCollisionCircleSegmentOuter(...)` et `checkCollisionCircleSegmentInner(...)`.

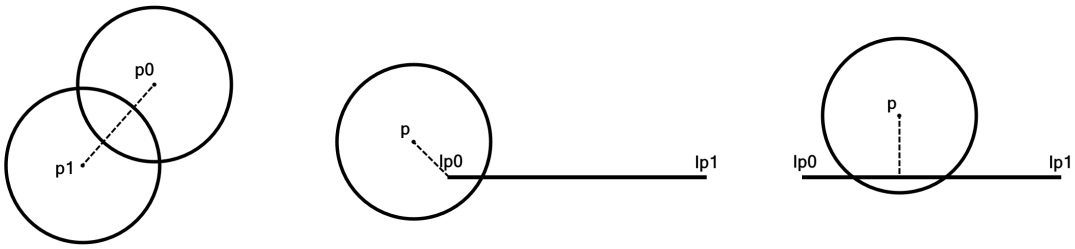

FIGURE 3 – Trois types de collision. De gauche à droite : Collision cercle-cercle, collision cercle-segment externe (outer), collision cercle-segment interne (inner).

6 Remise

Créez un repertoire VOTRENOM_TP3 et placez-y les fichiers sources du projet en respectant la hiérarchie du template. Placez-y également un fichier README.txt contenant votre nom ainsi que toute information que vous aimeriez fournir au correcteur. Compressez ce répertoire avant de le rendre sur StudiUM.

Faites une copie de votre TP avant de commencer la partie de license créative. Lors de la remise, remettez deux versions de votre TP, une avec la partie license créative et une sans license créative. Ceci vous évite de perdre des points dans les autres parties si votre license créative écrase (overwrite) certains shaders ou ne marche pas correctement !

Pour ce TP, n'utilisez pas du code sur l'internet. (Zéro tolérance au plagiat !)

7 Détails

Ce TP utilise les modules javascripts intoduits dans la 6^e édition du standard *ECMAScript* (ES6). Si vous utilisez Webstorm dans le premier travail de la session, vous ne devrez pas avoir de problème. Par contre, si vous utilisez un serveur local (tel qu'avec la commande python -m http-server), vous pourrez avoir des problème. Deux options s'offrent à vous :

1. Utiliser le script python fourni dans le répertoire principal du travail avec la commande python server.py
2. Utiliser Webstorm