GPU : Illumination
5ETI – IMI 2024-2025

CPE Lyon

Noms, Prénoms : BRUSSIEUX Guillaume, ZEMAN Mathieu

Date : 16 Décembre 2024

1 Introduction

Ce TP a pour objectif d'exploiter la puissance de la programmation GPU en mettant en œuvre des techniques de rendu avancées, permettant d'améliorer la qualité visuelle des images générées. Plus précisément, nous allons :

- Implémenter et observer l'illumination de Phong, un modèle d'éclairage classique combinant des composantes ambiante, diffuse et spéculaire.
- Utiliser plusieurs textures dans un même shader, permettant la combinaison de différentes cartes de matériaux pour enrichir le rendu.
- Exploiter des cartes d'occlusion ambiante (Ambient Occlusion) pour simuler les effets d'ombre dans les zones peu accessibles à la lumière.
- Intégrer des cartes d'élévation (displacement maps) pour modifier dynamiquement la géométrie des objets, ajoutant ainsi des détails visibles sans augmenter le nombre de sommets.
- Travailler dans l'espace tangent pour appliquer des techniques avancées comme le *normal mapping*, qui simule des détails de surface en modifiant l'orientation des normales.

Ces approches mettent en lumière le pipeline graphique de la programmation GPU, illustrant comment les données géométriques et texturales peuvent être transformées pour produire des rendus réalistes en temps réel.

2 Illumination de Phong

Dans la vraie vie, les phénomènes d'éclairage sont d'une grande complexité, car ils dépendent d'une multitude de variables physiques et environnementales. Ces interactions sont souvent impossibles à calculer en temps réel avec les capacités de traitement dont nous disposons. Ainsi, avec OpenGL, on s'appuie sur des modèles simplifiés pour simuler la lumière de manière crédible tout en respectant les contraintes de faisabilité numérique. Parmi ces modèles, celui de Phong est très utilisé.

2.1 Composantes de l'illumination de Phong

L'illumination de Phong repose sur trois composantes principales : une lumière ambiante, une lumière diffuse et une lumière spéculaire.

- **Ambiante :** ajoute une base lumineuse uniforme qui imite la lumière environnante émise par une source de faible intensité. Cette composante garantit que l'objet ne soit jamais complètement noir, même en l'absence d'une lumière directe visible.
- **Diffuse :** dépend de l'orientation de la surface de l'objet éclairé par rapport à la source lumineuse. C'est cette composante qui donne le plus de relief visuel, car elle accentue les zones qui font directement face à la lumière et atténue celles qui en sont éloignées.

— **Spéculaire :** responsable des reflets brillants que l'on observe sur des matériaux réfléchissants ou lisses. Ces reflets dépendent de l'alignement entre la surface de l'objet, la source lumineuse, et la position de l'observateur, et apportent des détails.

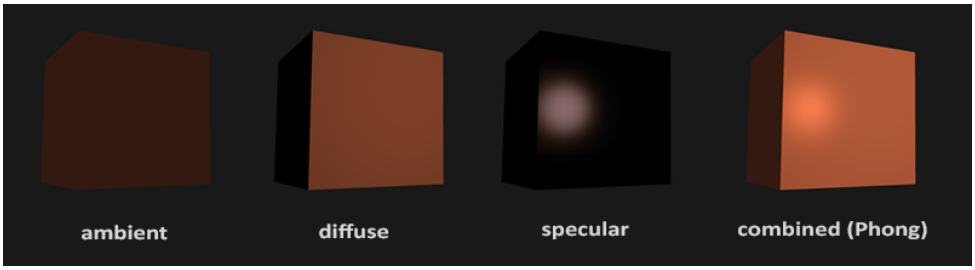

FIGURE 1 – Les différentes composantes de l'illumination de Phong.

Source : `https://learnopengl.com/Lighting/Basic-Lighting`

2.2 Pipeline graphique GPU et étapes de calculs

2.2.1 Étape 1 : Transformation des sommets (Vertex Shader)

— **Entrées** :
  — Les positions des sommets en espace objet.
  — Les normales des sommets.
  — Les coordonnées de texture.
  — Les matrices de transformation (modèle, vue, projection).
— **Traitements** :
  — Conversion des positions et normales des sommets dans l'espace vue pour des calculs cohérents avec la lumière et la caméra.
  — Transmission des données interpolées (normales, positions, coordonnées UV) vers l'étape suivante.
— **Sorties** :
  — Les coordonnées des sommets transformées pour l'affichage (`gl_Position`).
  — Les normales et positions interpolées vers les fragments.

2.2.2 Étape 2 : Calcul de l'éclairage (Fragment Shader)

— **Entrées** :
  — Normales interpolées en espace vue.
  — Position du fragment en espace vue.
  — Direction de la lumière en espace vue.
  — Direction vers la caméra (ou l'œil).
  — Couleur de la lumière et propriétés du matériau (k_a, k_d, k_s, *shininess*).
— **Traitements** :
  — **Composante ambiante** : Calculée comme une fraction fixe (k_a) de la couleur du matériau.
  — **Composante diffuse** : Basée sur le produit scalaire entre la normale ($\vec{N}$) et la direction de la lumière ($\vec{L}$).

$$I_{\text{diffuse}} = k_d \cdot \max(\vec{N} \cdot \vec{L}, 0) \cdot C_{\text{mat}}$$

  — **Composante spéculaire** : Calculée en utilisant la réflexion de la lumière ($\vec{R}$) et la direction de la caméra ($\vec{V}$).

$$I_{\text{specular}} = k_s \cdot (\max(\vec{R} \cdot \vec{V}, 0))^{\text{shininess}} \cdot C_{\text{light}}$$

  — Combinaison des trois composantes pour obtenir la couleur finale.
— **Sorties** :
  — La couleur finale pour le fragment, affichée à l'écran.

2.2.3 Paramètres nécessaires

- **Uniformes** :
  - Matrices de transformation (modèle, vue, projection).
  - Position et intensité de la lumière.
  - Propriétés du matériau (k_a, k_d, k_s, *shininess*).
- **Textures** :
  - Une texture diffuse pour représenter la couleur du matériau.
- **Données interpolées** :
  - Normales, coordonnées UV et positions calculées dans le vertex shader.

2.2.4 Résultats Visuels

FIGURE 2 – Illumination de Phong sur une sphère à texture rocheuse

2.3 Importance du nombre de points sur la sphère

Une sphère avec un maillage dense (nombre élevé de sommets) produit des transitions plus douces entre les zones éclairées et les ombres. À l'inverse, une sphère avec peu de sommets montre des artefacts visibles (effet angulaire ? La sphère est moins ronde, on vois des faces.).

3 Utilisation de multiples textures

Pour intégrer plusieurs textures au sein d'un même shader, On utilise des unités de texture, qui permettent de stocker jusqu'à 16 textures dans des espaces dédiés et d'y accéder via leurs index. La texture de couleur est ensuite combinée avec une carte d'occlusion ambiante pour assombrir les zones moins accessibles à la lumière. Le pipeline effectue :

- **Entrées :** Coordonnées UV pour échantillonner les textures.
- **Traitement :** Les valeurs de la carte AO modulent la composante ambiante de l'éclairage.
- **Sortie :** Une image où les zones moins accessibles apparaissent visuellement plus sombres.

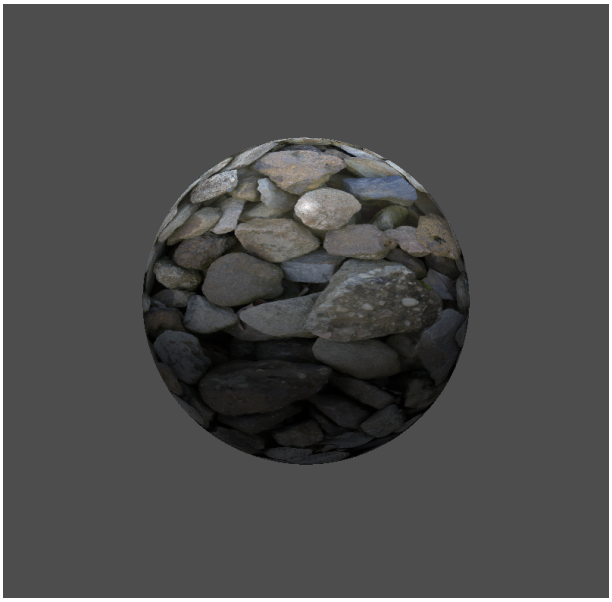

FIGURE 3 – Carte d'occlusion

4 Modification de la géométrie avec une carte de hauteur

Une carte de déplacement (displacement map) modifie la position des sommets en fonction de leurs normales. Le pipeline GPU procède ainsi :

- **Entrées :** Texture de hauteur échantillonnée à l'aide des coordonnées UV.
- **Traitement :** Les sommets sont déplacés le long de leurs normales proportionnellement à la valeur de hauteur. Un facteur d'échelle (0.2 dans ce cas) contrôle l'intensité du déplacement.
- **Sortie :** Une géométrie modifiée où les détails sont visibles directement dans le maillage.

FIGURE 4 – Carte de hauteur

5 Normal Mapping

Le *normal mapping* utilise une carte de normales pour simuler des détails de surface. Ces normales dûe a déplacement des points précédent sont ré orienter et re définies dans l'espace TBN (tangent, bi-tangent,

normal), ce qui modifient l'interaction entre la lumière et la surface.

- **Entrées :** Texture de normales et bases TBN (Tangente, Bitangente, Normale).
- **Traitement :** Les normales de la texture sont utilisées pour ajuster les calculs de l'illumination sans modifier la géométrie réelle.
- **Sortie :** Une surface visuellement plus détaillée sans augmenter la complexité géométrique.

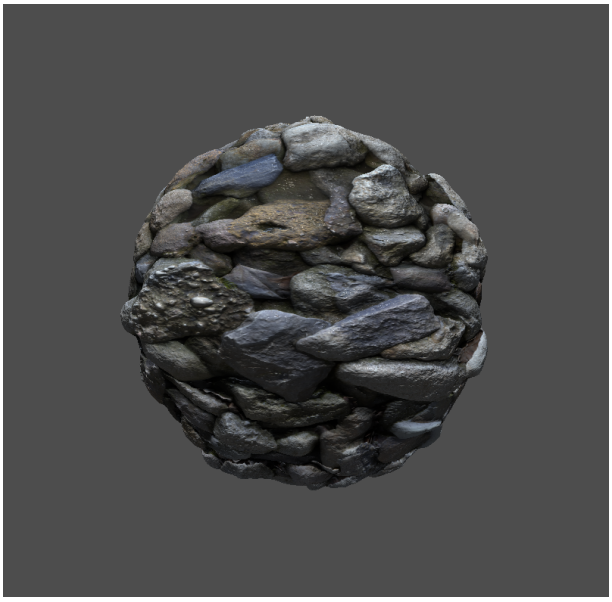

FIGURE 5 – Normal mapping

6 Références

- `https://learnopengl.com` (Learn OpenGL)
- `https://foundationsofgameenginedev.com/`(Foundations of Game Engine Development)