Techniques d'ombrage

CPE — ETI5–IMI

1 Introduction

Lors de ce TP, nous allons implémenter différentes façons de réaliser des ombres avec OpenGL.

Afin de se focaliser uniquement sur les techniques d'ombrage, et pas sur les shaders ou autre spécificités récentes d'OpenGL, nous utiliserons des fonctions assez anciennes déjà présentes dans OpenGL 1.x, ce que l'on appelle le *pipeline fixe*

Quelques unes des fonctions permettant de paramétrer le *pipeline fixe* seront expliquées durant le TP.

Afin d'éviter les cas particuliers dûs aux polyèdres trop réguliers, on utilisera un personnage animé au format md2 comme forme de base. Ce format permet de stocker un personnage avec différentes animations et éventuellement plusieurs "skins". Il était utilisé dans des jeux au début des années 2000 pour sa simplicité de mise en œuvre.

FIGURE 1 – Le personnage sans son ombre

Les programmes proposés permettent d'afficher un tel modèle, de changer le point de vue, la skin, l'animation, etc. Le `Makefile` fourni permet de compiler le tout et le programme créé doit être lancé dans le même répertoire.

Voici les commandes principales de ce programme :
- Bouton gauche de la souris : rotation de la scène
- Bouton gauche de la souris + SHIFT : déplacement de la scène
- Bouton central de la souris : zoom
- Bouton droit de la souris : affichage du menu des animations / skins
- Touches fléchées : déplacement de la source lumineuse (inutilisée au début du TP)

2 Techniques simples

2.1 Projection orthogonale

La manière la plus simple d'obtenir une ombre est de dessiner le personnage deux fois, une fois normalement, et une fois sans texture, en projetant les positions des sommets sur un plan (habituellement le plan $z = 0$).

FIGURE 2 – Le personnage avec une ombre du type z zéro

Question 1 : *Récupérez l'archive* `ombres-projetées.tar.gz`*, décompactez, compilez. Vous devriez avoir une image ressemblant à la figure 1. Identifiez chaque fichier afin de comprendre comment fonctionne le tout. La gestion du clavier et de la souris est dans le fichier* `src/main.cpp`*. L'autre fichier important est* `src/md2_model.cpp` *qui contient le code que vous devez modifier.*

Question 2 : *Remplissez le vecteur* `positions_ombres` *en mettant la composante z à zéro pour obtenir une ombre (pensez à bouger la scène pour voir ce qui se passe).*

Question 3 : *Les pieds du personnages ne sont pas à l'altitude 0, il faut donc modifier la position de l'ombre pour avoir quelque chose de réaliste. À quelle altitude faut-il mettre l'ombre ? Quelle est l'équation du plan devant recevoir l'ombre pour avoir une image ressemblant à la figure 2.*

2.2 Projection centrale

Question 4 : *Connaissant le plan de projection et la position de la lumière (*`light_pos`*), vous pouvez calculer la matrice de projection de l'ombre sur le plan (donnée dans le cours). Modifiez le calcul des positions de l'ombre afin d'utiliser cette matrice. Vous devez obtenir une image semblable à la figure 3. Vérifiez que l'ombre dépend de la position de la lumière en la déplaçant à l'aide des touches flêchées.*

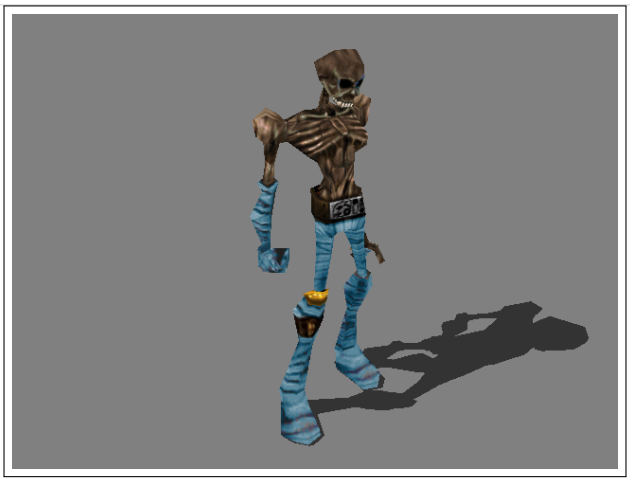

FIGURE 3 – Le personnage avec une ombre projetée (sans arme)

3 Stencil et ombre

En plus des tampons de couleurs et de profondeurs (voir la fin du TP sur le rendu projectif), OpenGL dispose également d'un tampon *Stencil*. Ce tampon est destiné à contenir des entiers et est généralement utilisé en deux temps :

— la première phase dessine dans le stencil sans modifier le tampon de couleur
— une deuxième phase utilise le contenu du stencil pour limiter le dessin dans le tampon de couleur

3.1 Quads d'ombre

Dans cette partie, vous devrez afficher les quads d'ombres pour une scène donnée et avoir un affichage ressemblant à la figure 4.

Question 5 : *Récupérez l'archive* `ombres-stencil.tar.gz` *qui contient à peu près le même programme que précédemment, avec des fonctions en plus (pour le calcul des quads d'ombre). Compilez, exécutez de façon à avoir un personnage dans son décor (mais sans ombre). Prenez connaissance des nouvelles fonctions.*

Question 6 : *Dans la fonction* `draw_quads_in_stencil()`*, ajouter ce qu'il faut pour dessiner directement les quads (sans utiliser de stencil pour l'instant, de façon à avoir une scène ressemblant à la figure 4. L'utilisation de blending (optionnel) peut rendre les choses plus ou moins jolies.*

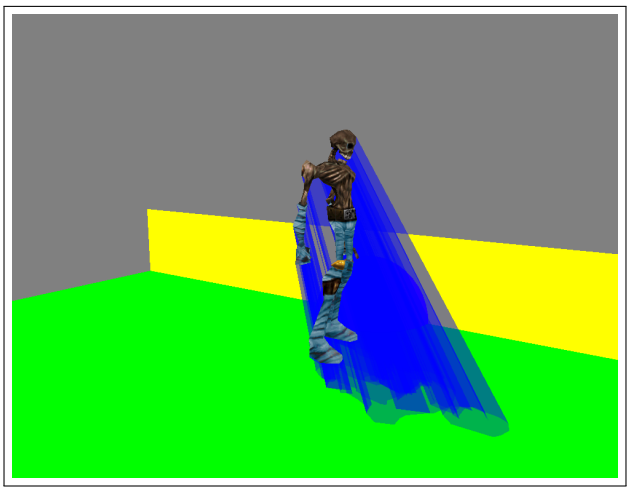

FIGURE 4 – Le personnage entouré de ses quads d'ombre

3.2 Affichage de l'ombre en utilisant le stencil

Il ne reste plus qu'à implémenter l'algorithme présenté en cours et indiqué dans le programme pour obtenir des véritables ombres stencil.

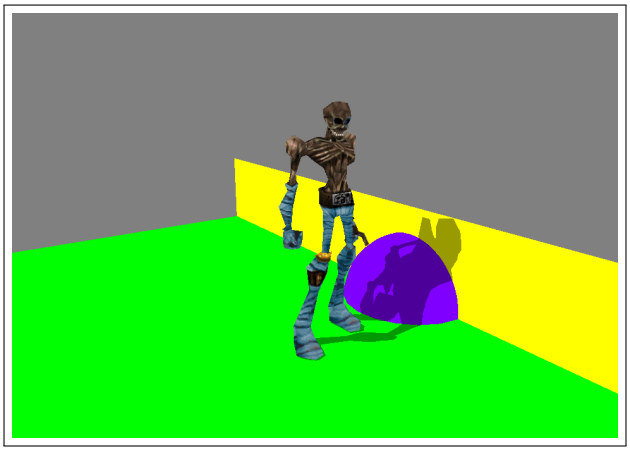

FIGURE 5 – Le personnage avec une ombre portée sur des éléments du décor

Question 7 : *Implémentez l'algorithme donné à la fin de la fonction* `draw_quads_in_stencil()`*. Chaque ligne correspond à un ou deux appels OpenGL. Vous devriez obtenir une image ressemblant à la figure 5. Bien évidemment, vous pouvez modifier le décor si le cœur vous en dit.*