# Graphique 3D - Compte-rendu

Maxime Jolivel, Emmanuel Leroux, Jean-Baptiste Gaeng

May 3, 2018

Figure 1: Une capture d'écran de la scène 3D finale

## 1 Présentation du projet

Au sein de notre projet de Graphique 3D, nous avons pu implémenter une scène contenant un modèle 3D de dinosaure, au milieu d'un environnement de type savane, auquel nous avons ajouté des modèles 3D d'arbres. Le lancement du projet s'effectue en éxécutant viewer.py.

### 1.1 Le Dinosaure Raptor

Le modèle 3D du raptor que nous avons utilisé est l'un des dinosaure fourni au sein du projet, à savoir dinosaurus-roar.dae du dossier dino. C'est un fichier au format Collada contenant une animation de rugissement, le modèle 3D étant skinné. Nous avons pu importer la texture de base du dinosaure, elle aussi fournie.

En appuyant sur la touche "r" pour "rugir" du clavier, l'animation du Raptor se lance.

On remarquera aussi que le matériau utilisé sur le Raptor est de type Phong, et que la lumière est placée à la position correspondante à celle du Soleil de l'image de la SkySphere.

### 1.2 Le Sol - Algorithme de Diamant-Carré

Nous avons utilisé un algorithme pour générer le mesh correspondant au sol, appelé algorithme de Diamond Square. Il s'agit de calculer récursivement la position de points afin de former un terrain fractal. On trouvera le détail de l'algorithme sur la page Wikipédia suivante :

https://fr.wikipedia.org/wiki/Algorithme_Diamant-Carre

Notre implémentation de cet algorithme se trouve dans le script Python DiamondSquare.py.

### 1.3 La forêt d'arbre

Les modèles 3D utilisés sont tirés du site internet free3D. Nous avons utilisé Maya afin de créer une forêt avec les différents arbres récupérés, puis nous les avons inclus au sein du projet à l'aide de la fonction load-textured. Les fichiers .mtl permettent de bien importer les textures.

### 1.4 La Skysphere

L'image utilisée pour le ciel est un fichier issu du site :

http://www.hdrlabs.com/sibl/archive.html

Le site contient des fichiers au format .hdr, permettant de faire des éclairages basés sur image. C'est d'ailleurs l'une des extensions que nous pensions réaliser au départ (cf. plus loin). Finalement, nous avons utilisé également Maya pour projeter l'image à l'intérieur de la sphere. Cela nous a permit d'éviter les problèmes liés au calcul des coordonnées de textures sur la sphère en OpenGL.

## 2 Réponse aux critères de l'énoncé

### 2.1 Modélisation

Nous avons modélisé toute la scène du projet à l'aide de la modélisation hiérarchique. Nous avons de plus importé des modèles 3D d'arbres et modélisé un sol non linéaire à l'aide de l'algorithme de Diamant-Carré. Ceci rempli bien les 3 points requis de Modélisation.

### 2.2 Rendu

De plus, nous avons utilisé une lumière pour modéliser la lumière du Soleil, et un matériau Phong sur le Raptor. Nous avons de plus utilisé des textures pour tous les objets de la scène. On trouvera le nouvel effet de rendu à la section suivante.

### 2.3 Animation

L'animation du Raptor utilise la classe Keyframe implémentée au cours des TPs, de plus nous utilisons aussi la technique du skinning. Nous intéragissons aussi avec le clavier puisque la touche "r" permet de faire rugir le Raptor.

### 2.4 Effet supplémentaire - Son

Nous avons ajouté des effets sonores à l'aide de librairies Python permettant la lecture des fichiers audios. Ainsi, l'appui sur la touche "r" du clavier lance l'animation de rugissement mais aussi lit le fichier son inclu dans les fichiers du projet.

## 2.5 Effet de rendu supplémentaire - Normal Mapping

**NB : Pour appliquer/enlever l'effet de normal mapping sur le Raptor, il suffit d'appuyer sur la touche "n" du clavier pour "Normal mapping"**

Nous avons hésité entre 3 effets de rendu supplémentaires, à savoir l'illumination basée sur image (avec des fichiers HDR), l'implémentation de l'occlusion ambiante et l'implémentation du normal mapping. C'est finalement ce dernier sujet que nous avons choisi, en nous basant sur le site internet suivant :

http://www.opengl-tutorial.org/fr/intermediate-tutorials/tutorial-13-normal-mapping/

Cette implémentation se passe en deux temps. Tout d'abord, les vecteurs tangents et bitangents sont calculés dans la fonction load-skinned(). Pour chaque triangle de la figure, nous allons déterminer ces deux vecteurs à l'aide des cotés de ce triangle en utilisant les formules fournies par le lien ci-dessus. Ces vecteurs seront ensuite passés en liste dans un buffer pour arriver jusqu'au shader.

Dans les shaders, nous calculons la matrice TBN afin de calculer la normale prise dans la texture de normales dans l'éspace tangent. Nous l'utilisons ensuite dans le modèle de Phong implémenté précedemment.

L'effet a été compliqué et long à mettre en place, mais est implémenté. Il reste cependant encore à régler certains paramètres pour que le rendu soit optimal. Voici ce que donne le rendu en testant la valeur des paramètres :

Figure 2: Une capture d'écran de la scène 3D finale avec l'effet de normal mapping (test)