République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université des Sciences et de la Technologie d'Oran – Mohamed Boudiaf
Faculté des Mathématiques et Informatique
Département d'informatique

Modélisation 3D

Rapport TP
Modélisation 3D

Domaine : **Mathématiques – Informatique**
Filière : **Informatique**
Spécialité : **M1 IAA**

Présenté Par :

- BELHADJ Mohamed

- KEBIRI Issam Dine

Supervisé par:

- ELHANNACHI.S, Département Informatique - USTO.

Ce rapport contient les objectives du TP, les outils nécessaires pour atteindre nos objectives, on expliquera aussi les étapes pour réaliser notre projet ainsi que des morceaux de code source pour mieux comprendre. Au final on visualisera le résultat.

Objectives

L'objective de ce TP était de réaliser un modèle de mosquée 3D, à l'aide d'un environnement de développement et de différentes bibliothèques très utiles. On se basera sur des formes bien connu et déjà utiliser dans ce deuxième semestre de master 1 intelligence artificielle dans le module modélisation 3D pour concevoir notre mosquée.

Modélisation tridimensionnelle

La modélisation tridimensionnelle est l'étape en infographie tridimensionnelle qui consiste à créer, dans un logiciel de modélisation 3D, un objet en trois dimensions, par ajout, soustraction et modifications de ses constituants. La révolution consiste à faire tourner un profil 2D autour d'un axe 3D : on obtient ainsi un volume de révolution. C'est la technique majoritairement utilisée dans le jeu vidéo, et le cinéma d'animation. La modélisation polygonale induit une marge d'erreur de proportions et de dimensions le plus souvent invisible à l'œil nu. Dans le cinéma d'animation, les modèles 3D organiques sont le plus souvent lissés. Le lissage consiste à subdiviser un maillage (une itération correspond à une subdivision de chaque arête, soit dans le cas de face à quatre côtés, une subdivision en quatre faces) et arrondir les faces obtenues selon différents algorithmes, afin de gommer l'effet anguleux des modèles obtenus par modélisation polygonale.

Environnement de développement

On a choisi Code::Blocks, Code::Blocks est un environnement de développement intégré libre et multiplateforme. Il est écrit en C++ et utilise la bibliothèque wxWidgets. Code::Blocks est orienté C et C++, Code::Blocks se veut simple, voire

intuitif, d'utilisation pour un programmeur. Il se révèle néanmoins fort complet au fur et à mesure qu'on en explore les options. Son architecture de plug-ins permet de l'étendre et de le personnaliser, tout en n'y incluant que ce que l'on souhaite utiliser. La plupart étant inclus dans l'archive et l'installeur, il n'est cependant pas nécessaire de les installer un à un.

OpenGL

OpenGL (Open Graphics Library) est un ensemble normalisé de fonctions de calcul d'images 2D ou 3D lancé par Silicon Graphics en 19922. Cette interface de programmation est disponible sur de nombreuses plateformes où elle est utilisée pour des applications qui vont du jeu vidéo jusqu'à la CAO en passant par la modélisation2. OpenGL permet à un programme de déclarer la géométrie d'objets sous forme de points, de vecteurs, de polygones, de bitmaps et de textures. OpenGL effectue ensuite des calculs de projection en vue de déterminer l'image à l'écran, en tenant compte de la distance, de l'orientation, des ombres, de la transparence et du cadrage3,4. L'interface regroupe environ 250 fonctions différentes qui peuvent être utilisées pour afficher des scènes tridimensionnelles complexes à partir de simples primitives géométriques. Du fait de son ouverture, de sa souplesse d'utilisation et de sa disponibilité sur toutes les plates-formes, elle est utilisée par la majorité des applications scientifiques, industrielles ou artistiques 3D et certaines applications 2D vectorielles. Cette bibliothèque est également utilisée dans l'industrie du jeu vidéo où elle est souvent en rivalité avec la bibliothèque de Microsoft : Direct3D. Une version nommée OpenGL ES a été conçue spécifiquement pour les applications embarquées (téléphones portables, agenda de poche, consoles de jeux…).

Exemple de code source de la mosquée 3D

- Code source de la porte :

```
glColor3d(0,250,250);
```

```
 glPushMatrix();
      glTranslated(0,0.9,-12);
      glRotated(90,1,0,0);
      glScaled(0.8,0.1,1);
      glutSolidCube(3);
   glPopMatrix();
```

- Code source de la Som3a :

```
glColor3d(1,1,1);
   glPushMatrix();
      glTranslated(0.1,3.5,0.5);
      glRotated(0,0,1,0);
      glScaled(0.1,0.1,0.9);
      glutSolidTorus(12,10,100,50);
   glPopMatrix();
```

- Code source de la courbe :

```
glColor3d(0.45,0.45,0.45);
 glPushMatrix();
      glTranslated(-0.2,1.5,0);
      glRotated(90,0,1,0);
      glScaled(8,2,4);
      glutSolidCube(3);
   glPopMatrix();
```

- Code source carre kouba :

```
glColor3d(0,1,1);
 glPushMatrix();
      glTranslated(-0.0,4.3,-6.5);
      glRotated(90,0,1,0);
      glScaled(2.7,1,2.7);
      glutSolidCube(3);
   glPopMatrix();
```

- Code source cube Som3a :

```
glColor3d(1,1,0);
 glPushMatrix();
       glTranslated(-4.7,11.5,10.5);
      glRotated(90,0,1,0);
      glScaled(1.05,1,1.05);
      glutSolidCube(3);
```

- Code source pilier de porte :

```
glColor3d(1,1,0);
 glPushMatrix();
      glTranslated(-4.7,19.1,10.5);
      glRotated(90,0,1,0);
      glScaled(0.04,0.5,0.04);
      glutSolidCube(3);

   glPopMatrix();



    glPushMatrix();
      glTranslated(-4.7,20,10.5);
      glRotated(0,1,0,0);
      glRotated(a,0,0,1);
      glutSolidTorus(0.05,0.2,slices,stacks);
   glPopMatrix();
```

- Code source escalier :

```
glColor3d(0,0,0);
 glPushMatrix();
       glTranslated(-0.2,-1.4,-15);
      glRotated(90,0,1,0);
      glScaled(2,0.13,4);
      glutSolidCube(3);
   glPopMatrix();

glColor3d(0,0,1);
```

```
glPushMatrix();
       glTranslated(-0.2,-1.2,-14.5);
      glRotated(90,0,1,0);
      glScaled(1.6,0.13,3.6);
      glutSolidCube(3);
  glPopMatrix();


  glColor3d(0,1,0);
glPushMatrix();
       glTranslated(-0.2,-0.8,-13.9);
      glRotated(90,0,1,0);
      glScaled(1.2,0.13,3.2);
      glutSolidCube(3);

  glPopMatrix();
```

- Code source fenetres :

```
  glColor3d(12,250,0);
glPushMatrix();
      glTranslated(6,1.7,7);
      glRotated(90,1,0,0);
      glScaled(0.1,1,1);
      glutSolidCube(3);
  glPopMatrix();


  glColor3d(12,250,0);
glPushMatrix();
      glTranslated(6,1.7,0);
      glRotated(90,1,0,0);
      glScaled(0.1,1,1);
      glutSolidCube(3);
  glPopMatrix();


   glColor3d(12,250,0);
```

```
glPushMatrix();
      glTranslated(6,1.7,-7);
      glRotated(90,1,0,0);
      glScaled(0.1,1,1);
      glutSolidCube(3);
   glPopMatrix();



    glColor3d(12,250,0);
glPushMatrix();
      glTranslated(-6.2,1.7,7);
      glRotated(90,1,0,0);
      glScaled(0.1,1,1);
      glutSolidCube(3);
   glPopMatrix();


   glColor3d(12,250,0);
glPushMatrix();
      glTranslated(-6.2,1.7,0);
      glRotated(90,1,0,0);
      glScaled(0.1,1,1);
      glutSolidCube(3);
   glPopMatrix();


    glColor3d(12,250,0);
glPushMatrix();
      glTranslated(-6.2,1.7,-7);
      glRotated(90,1,0,0);
      glScaled(0.1,1,1);
      glutSolidCube(3);
   glPopMatrix();
```

- Code source du gazon :

```
glColor3d(0,128,0);
```

```
glPushMatrix();
        glTranslated(0,-3,0);
       glRotated(90,0,1,0);
       glScaled(20,0.1,20);
       glutSolidCube(3);
   glPopMatrix();
```

Résultat final

Exécuter le code source pour visualiser