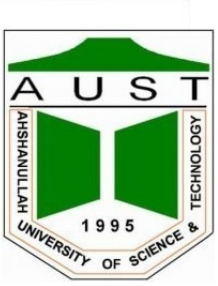
Ahsanullah University of Science & Technology
Department of Computer Science & Engineering



Project: Chichen Itza

CSE 4204: Computer Graphics Lab

Spring 2018

Year: 4th Semester: 2nd

Lab Group: B1

Group Members:

Name: Sabbir Hassan
ID: 14.02.04.058

Name: Nowshin Nawar Arony
ID: 14.02.04.067

Name: Sanjana Farial
ID: 14.02.04.100

Project

To create the structure Chichen Itza in openGL.

Tools Used

OpenGL, GLUT Library

Functions

Some important functions that we have used in our project are described below.

- glutPostRedisplay() glutPostRedisplay() marks the current window as needing to be redisplayed.
- glTranslatef() Multiplies the current matrix by a translation matrix and translates the object.
- glRotatef() This function is used to rotate the structure.
- GLuint loadTexture() We have used this function to set required textures.
- GLfloat ambientColor[]Numerous large light sources are generally positioned in such a way that lights seem to be everywhere in the same amount.
- GLfloat lightPos0[] Adds pointed light where while a point light radiates light out in every direction from it.
- GLfloat lightPos1[] Adds directional light where the light is only coming from a single direction.
- glBegin(GL_QUADS) Used for drawing quadrilateral shapes. We have drawn the temple with this function.
- glutTimerFunc() Registers a timer callback to be triggered in a specified number of milliseconds.
- glutKeyboardFunc() Calls the function when a key that generates an ASCII character is pressed.

- glutReshapeFunc This function is called whenever the window is resized or moved.
- glutSpecialFunc() Sets the special keyboard callback for the current window. The special keyboard callback is triggered when keyboard function or directional keys are pressed.
- glEnable(GL_TEXTURE_2D) We have used this function to draw the backgrounds and brick textures on the temple.

Project Functionality and Key Features

- **Basic Structure:** The basic 3D structure is created using multiple quads.
- **Lighting:** Our project is drawn in both the day and night mode that has been done using suitable lighting and pictures of background.
- **Texture:** We have used pictures of sky and grass as our background which have been set using texture function. The bricks on the temples are also drawn using texture.
- **Rotation:** The structure along with the ground rotates automatically and also by keypress.
- **Keyboard Usage:**
  - We can stop the automatic rotation of the structure along with the ground by pressing the key **s**.
  - Project switches to day mode by keypress **d** and to night mode by keypress **n**.
  - The stucture can be rotated left or right by **left arrow key** or **right arrow key** respectively.

Code

```
#include <windows.h>
#include <iostream>
#include <stdlib.h>

//Include OpenGL header files,
```

```
//so that we can use OpenGL
#ifdef __APPLE__
#include <OpenGL/OpenGL.h>
#include <GLUT/glut.h>
#else
#include <GL/glut.h>
#endif

using namespace std;

//Initializes 3D rendering
void initRendering()
{
    //Makes 3D drawing work when something
    //is in front of something else
    glEnable(GL_DEPTH_TEST);
}

//Called when the window is resized
void handleResize(int w, int h)
{
    //Tell OpenGL how to convert
    //from coordinates to pixel values
    glViewport(0, 0, w, h);

    //Switch to setting the camera perspective
    glMatrixMode(GL_PROJECTION);

    //Set the camera perspective
    glLoadIdentity(); //Reset the camera
    gluPerspective(45.0,    //The camera angle
    (double)w / (double)h, //The width-to-height
      ratio
    1.0,                   //The near z clipping
      coordinate
    200.0);                //The far z clipping
      coordinate
}

//Draws the 3D scene
void drawScene()
```

```
{
    //Clear information from last draw
    glClear(GL_COLOR_BUFFER_BIT |
      GL_DEPTH_BUFFER_BIT);

    glMatrixMode(GL_MODELVIEW);
    //Switch to the drawing perspective

    glLoadIdentity();
    //Reset the drawing perspective

    //Translated to left
    glTranslatef(-1.25f, 0.0f, 0.0f);
    //glTranslatef(2.25f, 0.0f, 0.0f);

    glBegin(GL_TRIANGLES);
    //Triangle Door
    glColor3f(1,1,1);
    glVertex3f(-0.25f, -1.0f, -5.0f);
    glVertex3f(0.25f,-1.0f, -5.0f);
    glVertex3f(0.0f,-0.4f, -5.0f);

    glEnd();

    glBegin(GL_QUADS);
    //Begin quadrilateral coordinates

    //Rectangle Middle part
    glColor3f(0.8,0.498039,0.196078);
    glVertex3f(-0.5f, -1.0f, -5.0f);
    glVertex3f(0.5f, -1.0f, -5.0f);
    glVertex3f(0.5f, 0.5f, -5.0f);
    glVertex3f(-0.5f, 0.5f, -5.0f);

    glEnd(); //End quadrilateral coordinates

    glBegin(GL_POLYGON);
    glColor3f(1.0,1.0,0.0);

    glVertex3f(-0.75f, -1.25f,-5.0f);
    glVertex3f(0.0f, -1.65f,-5.0f);
```

```
glVertex3f(0.75f,-1.25f,-5.0f);
glVertex3f(0.5f,-1.0f,-5.0f);
glVertex3f(-0.5f,-1.0f,-5.0f);
glEnd();

glBegin(GL_POLYGON);
glColor3f(1.0,0.0,0.0);

glVertex3f(-1.0f, -1.25f,-5.0f);
glVertex3f(0.0f, -2.0f,-5.0f);
glVertex3f(1.0f,-1.25f,-5.0f);
glVertex3f(0.5f,-1.0f,-5.0f);
glVertex3f(-0.5f,-1.0f,-5.0f);
glEnd();

glBegin(GL_TRIANGLES);
//Begin triangle coordinates

//Triangle Top
glColor3f(0.623529,0.623529,0.372549);
glVertex3f(-0.5f, 0.5f, -5.0f);
glVertex3f(0.5f, 0.5f, -5.0f);
glVertex3f(0.0f, 1.75f, -5.0f);

glEnd(); //End triangle coordinates

glBegin(GL_TRIANGLES);
//Begin triangle coordinates

//Triangle Upper Left Wing
glColor3f(1,0.5,0);
glVertex3f(-0.85f, 0.1f, -5.0f);
glVertex3f(-0.5f, 0.1f, -5.0f);
glVertex3f(-0.5f, 0.5f, -5.0f);

//Triangle Upper Right Wing
glVertex3f(0.5f, 0.1f, -5.0f);
glVertex3f(0.85f, 0.1f, -5.0f);
glVertex3f(0.5f, 0.5f, -5.0f);

//Triangle Lower Left Wing
```

```
        glVertex3f(-0.85f, -1.0f, -5.0f);
        glVertex3f(-0.5f, -1.0f, -5.0f);
        glVertex3f(-0.5f, -0.2f, -5.0f);

        //Triangle Lower Right Wing
        glVertex3f(0.5f, -1.0f, -5.0f);
        glVertex3f(0.85f,-1.0f, -5.0f);
        glVertex3f(0.5f,-0.2f, -5.0f);

        glEnd(); //End triangle coordinates

        glutSwapBuffers();
        //Send the 3D scene to the screen

}

int main(int argc, char** argv)
{
        //Initialize GLUT
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_DOUBLE |
                            GLUT_RGB | GLUT_DEPTH);
        glutInitWindowSize(700, 600);
        //Set the window size

        //Create the window
        glutCreateWindow("Rocket");
        initRendering(); //Initialize rendering

        //Set handler functions for drawing,
        //keypresses, and window resizes
        glutDisplayFunc(drawScene);
        glutReshapeFunc(handleResize);

        glutMainLoop(); //Start the main loop.
                        //glutMainLoop doesn't return.
        return 0;
         //This line is never reached
}
```

Output

Figure 1: Front (Day) View

Figure 2: Side (Day) View

Figure 3: Night View