ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| асп. |  |  |  | Д.А. Смолиенко |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №5 |
| РАБОТА СО СПИСКАМИ, ТЕКСТУРОЙ И NURBS ПОВЕРХНОСТЯМИ |
| по дисциплине:  КОМПЬЮТЕРНАЯ ГРАФИКА |
|  |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 4631 |  |  |  | С.А. Гришин |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2018

**Задание на лабораторную работу**

Вывести сложный трехмерный объект, сохранить его в виде списка и размножить в виде нескольких копий различного масштаба на экране путем повторного вывода списка. Выводимые объекты располагаются над рельефом, задаваемым NURBS-поверхностью. Наложить текстуру на выводимые объекты, сцена должна быть освещена, необходимо предусмотреть возможность включения эффекта тумана (при помощи клавиатуры или мыши).

**Текст программы**

#include "BmpLoader.h"

#include <GL/glut.h>

#include <iostream>

#include <stdlib.h>

GLuint theCube;

Point CalculateV(float t, Point\* pnts)

{

Point p;

float it = 1.0f - t;

float b0 = t \* t \* t;

float b1 = 3 \* t \* t \* it;

float b2 = 3 \* t \* it \* it;

float b3 = it \* it \* it;

p.x = b0 \* pnts[0].x + b1 \* pnts[1].x + b2 \* pnts[2].x + b3 \* pnts[3].x;

p.y = b0 \* pnts[0].y + b1 \* pnts[1].y + b2 \* pnts[2].y + b3 \* pnts[3].y;

p.z = b0 \* pnts[0].z + b1 \* pnts[1].z + b2 \* pnts[2].z + b3 \* pnts[3].z;

return p;

}

Point Calculate(float u, float v)

{

Point temp[4];

temp[0] = CalculateU(u, 0);

temp[1] = CalculateU(u, 1);

temp[2] = CalculateU(u, 2);

temp[3] = CalculateU(u, 3);

return alculateV(v, temp);

}

void OnReshape(int w, int h)

{

if (h == 0)

h = 1;

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45, (float)w / h, 0.1, 100);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

void drawCubes()

{

glPushMatrix();

glEnable(GL\_TEXTURE\_2D);

glLoadIdentity();

glTranslatef(0.0, 0.0, -10.0);

glRotatef(120, 1.0, 2.0, 0.0);

glScalef(0.9, 0.9, 0.9);

for (int i = 0; i < 6; i++) {

glPushMatrix();

if (i == 1) {

glTranslatef(1.0, 0.0, 0.0);

glRotatef(90, 0.0, 1.0, 0.0);

glTranslatef(1.0, 0.0, 0.0);

}

else if (i == 2) {

glTranslatef(0.0, 0.0, -2.0);

glRotatef(180.0, 0.0, 1.0, 0.0);

}

else if (i == 3) {

glTranslatef(-1.0, 0.0, 0.0);

glRotatef(-90, 0.0, 1.0, 0.0);

glTranslatef(-1.0, 0.0, 0.0);

}

else if (i == 4) {

glTranslatef(0.0, 1.0, 0.0);

glRotatef(-90, 1.0, 0.0, 0.0);

glTranslatef(0.0, 1.0, 0.0);

}

else if (i == 5) {

glTranslatef(0.0, -1.0, 0.0);

glRotatef(90, 1.0, 0.0, 0.0);

glTranslatef(0.0, -1.0, 0.0);

}

glBegin(GL\_QUADS);

glTexCoord2f(1.0, 1.0);

glVertex3f(1.0, 1.0, 0.0);

glTexCoord2f(0.0, 1.0);

glVertex3f(-1.0, 1.0, 0.0);

glTexCoord2f(0.0, 0.0);

glVertex3f(-1.0, -1.0, 0.0);

glTexCoord2f(1.0, 0.0);

glVertex3f(1.0, -1.0, 0.0);

glEnd();

glPopMatrix();

}

glDisable(GL\_TEXTURE\_2D);

glPopMatrix();

glPushMatrix();

glEnable(GL\_TEXTURE\_2D);

glLoadIdentity();

glTranslatef(7, 2.0, -20.0);

glRotatef(120, 1.0, 2.0, 0.0);

glScalef(0.5, 0.5, 0.5);

for (int i = 0; i < 6; i++) {

glPushMatrix();

if (i == 1) {

glTranslatef(1.0, 0.0, 0.0);

glRotatef(90, 0.0, 1.0, 0.0);

glTranslatef(1.0, 0.0, 0.0);

}

else if (i == 2) {

glTranslatef(0.0, 0.0, -2.0);

glRotatef(180.0, 0.0, 1.0, 0.0);

}

else if (i == 3) {

glTranslatef(-1.0, 0.0, 0.0);

glRotatef(-90, 0.0, 1.0, 0.0);

glTranslatef(-1.0, 0.0, 0.0);

}

else if (i == 4) {

glTranslatef(0.0, 1.0, 0.0);

glRotatef(-90, 1.0, 0.0, 0.0);

glTranslatef(0.0, 1.0, 0.0);

}

else if (i == 5) {

glTranslatef(0.0, -1.0, 0.0);

glRotatef(90, 1.0, 0.0, 0.0);

glTranslatef(0.0, -1.0, 0.0);

}

glBegin(GL\_QUADS);

glTexCoord2f(1.0, 1.0);

glVertex3f(1.0, 1.0, 0.0);

glTexCoord2f(0.0, 1.0);

glVertex3f(-1.0, 1.0, 0.0);

glTexCoord2f(0.0, 0.0);

glVertex3f(-1.0, -1.0, 0.0);

glTexCoord2f(1.0, 0.0);

glVertex3f(1.0, -1.0, 0.0);

glEnd();

glPopMatrix();

}

glDisable(GL\_TEXTURE\_2D);

glPopMatrix();

}

void display()

{

glClear(GL\_DEPTH\_BUFFER\_BIT | GL\_COLOR\_BUFFER\_BIT);

glFogi(GL\_FOG\_MODE, fog\_mode);

if (FogAble) {

glEnable(GL\_FOG);

}

else {

glDisable(GL\_FOG);

}

glLoadIdentity();

gluLookAt(4, 30, 20, 0, 0, 0, 0, 1, 0);

glPushMatrix();

glColor3f(1, 0, 1);

glPushMatrix();

glBegin(GL\_LINES);

for (int i = 0; i != LOD; ++i) {

float u = (float)i / (LOD - 1);

for (int j = 0; j != LOD; ++j) {

float v = (float)j / (LOD - 1);

Point p = Calculate(u, v);

Point q = Calculate((float)(i + 1) / (LOD - 1), (float)(j + 1) / (LOD - 1));

glVertex3f(p.x, p.y, p.z);

glVertex3f(q.x, q.y, q.z);

}

}

glEnd();

glPopMatrix();

glCallList(theCube);

glPopMatrix();

glutSwapBuffers();

}

void Init()

{

theCube = glGenLists(1);

glNewList(theCube, GL\_COMPILE);

drawCubes();

glEndList();

glFogf(GL\_FOG\_START, near\_distance);

glFogf(GL\_FOG\_END, far\_distance);

glFogfv(GL\_FOG\_COLOR, fog\_colour);

glFogf(GL\_FOG\_DENSITY, fog\_density);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glEnable(GL\_DEPTH\_TEST);

glDepthFunc(GL\_LEQUAL);

}

void LoadTexture(const char\* filename)

{

BmpLoader bl(filename);

glGenTextures(1, &ID);

glBindTexture(GL\_TEXTURE\_2D, ID);

glPixelStorei(GL\_UNPACK\_ALIGNMENT, 1);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_REPEAT);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_REPEAT);

gluBuild2DMipmaps(GL\_TEXTURE\_2D, GL\_RGB, bl.iWidth, bl.iHeight, GL\_RGB,

GL\_UNSIGNED\_BYTE, bl.textureData);

}

void Keyboard(unsigned char key, int, int)

{

if (key == 'f') {

FogAble = !FogAble;

}

glutPostRedisplay();

}

int main(int argc, char\*\* argv)

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DEPTH | GLUT\_RGBA | GLUT\_DOUBLE);

glutInitWindowSize(640, 480);

glutCreateWindow("Lab\_5");

LoadTexture("image.bmp");

glutDisplayFunc(display);

glutReshapeFunc(OnReshape);

glutKeyboardFunc(Keyboard);

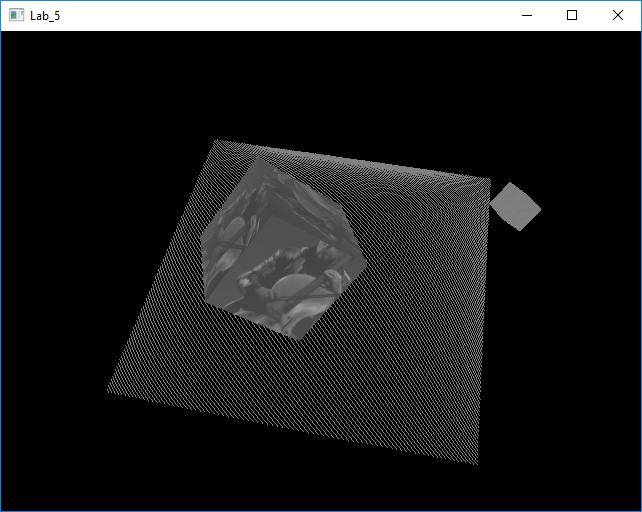
Init();

glutMainLoop();

return 0;

}

**Результат работы программы**



**Выводы**

В результате лабораторной работы разработано приложение, выводящее сложный трехмерный объект, сохранённый в виде списка и размноженный в виде нескольких копий различного масштаба на экране путем повторного вывода списка. Выводимые объекты располагаются над рельефом, задаваемым NURBS-поверхностью. Наложена текстура на выводимые объекты.