ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | Щекин С.В. |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 5 |
| **Работа со списками, текстурой и NURBS-поверхностями** |
| по дисциплине: Компьютерная графика |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 4134к |  |  |  | Костяков Н.А. |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2022

**Цель работы**

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).

**Листинг**

**Main.cpp**

#include <iostream>

#include <cmath>

#include <GL/glut.h>

#define STB\_IMAGE\_IMPLEMENTATION

#include "stb\_image.h"

#define W\_WIDTH 1280

#define W\_HEIGHT 720

// camera

double angle\_x = 0.0;

double angle\_y = -5000.0;

float lx = 0.0f, ly = 0.0f, lz = -1.0f;

float x = 0.0f, y = 0.0f, z = 10.0f;

int refreshMills = 60;

float fraction = -0.5f;

float fraction\_angle = 0.1f;

float light\_fraction = -0.5f;

bool use\_mouse = true;

bool forward = false;

bool back = false;

bool left = false;

bool right = false;

// nurbs

const int size\_numb\_x = 4;

const int size\_numb\_y = 4;

GLfloat ctlpoints[size\_numb\_x][size\_numb\_y][3];

GLfloat knots[size\_numb\_x + size\_numb\_y] = {

0.0, 0.0, 0.0, 0.0,

1.0, 1.0, 1.0, 1.0

};

int showPoints = 0;

GLUnurbsObj\* theNurb;

// lihgt

GLfloat ambientColor\_light[] = { 0.8, 0.8, 0.8, 1.0 };

GLfloat diffuseColor\_light[] = { 1, 1, 1, 1.0 };

GLfloat emis\_light[] = { 1.0, 1.0, 1.0, 1.0 };

// GLfloat light\_position[4] = {10.0, 5.0, -10.0, 1.0};

GLfloat light\_position[4] = { 30.0, 10.0, 0.0, 1.0 };

GLfloat light\_direction[3] = { 0.0, 0.0, 0.0 };

float mat\_dif\_light[] = { 1.0f, 1.0f, 1.0f, 1.0 };

// texture

GLuint theTorus;

GLuint texture;

// fog

bool fog\_bool = false;

GLuint filter;

GLuint fogMode[] = { GL\_EXP, GL\_EXP2, GL\_LINEAR };

GLuint fogfilter = 0;

GLfloat fogColor[4] = { 0.5f, 0.1f, 0.1f, 1.0f };

void fog\_on() {

glEnable(GL\_FOG); // Включает туман (GL\_FOG)

glFogi(GL\_FOG\_MODE, fogMode[fogfilter]);// Выбираем тип тумана

glFogfv(GL\_FOG\_COLOR, fogColor); // Устанавливаем цвет тумана

glFogf(GL\_FOG\_DENSITY, 0.05f); // Насколько густым будет туман

glHint(GL\_FOG\_HINT, GL\_DONT\_CARE); // Вспомогательная установка тумана

glFogf(GL\_FOG\_START, 1.0f); // Глубина, с которой начинается туман

glFogf(GL\_FOG\_END, 5.0f); // Глубина, где туман заканчивается.

}

// functions

void move() {

// camera

if (forward) {

z -= lz \* fraction;

x -= lx \* fraction;

y -= ly \* fraction;

}

if (back) {

z += lz \* fraction;

x += lx \* fraction;

y += ly \* fraction;

}

if (left) {

z += lx \* fraction;

x -= lz \* fraction;

}

if (right) {

z -= lx \* fraction;

x += lz \* fraction;

}

}

void keyUp(unsigned char key, int xx, int yy) {

switch (key) {

// camera

case ('w'):

forward = false;

break;

case ('s'):

back = false;

break;

case ('a'):

left = false;

break;

case ('d'):

right = false;

break;

// fog

case ('f'):

fog\_bool = !fog\_bool;

if (fog\_bool) glDisable(GL\_FOG);

else glEnable(GL\_FOG);

break;

// включить (выключить курсор)

case ('g'):

use\_mouse = !use\_mouse;

break;

// выход

case 27:

//glutDestroyWindow ( Win.id );

exit(0);

break;

}

}

void keyDown(unsigned char key, int xx, int yy) {

switch (key) {

// camera

case ('w'):

forward = true;

break;

case ('s'):

back = true;

break;

case ('a'):

left = true;

break;

case ('d'):

right = true;

break;

}

}

double sensivity = 0.001;

void gen(int size) {

theTorus = glGenLists(1);

glNewList(theTorus, GL\_COMPILE);

glRotatef(90.0, 1.0, 0.0, 0.0);

for (int i = 1; i < 5; i++) {

// glRotatef(i \* 50, 0.0, 1.0, 0.0);

glTranslatef(0.0f, 2 \* size, 0.0f);

glEnable(GL\_TEXTURE\_2D);

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

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

glBindTexture(GL\_TEXTURE\_2D, texture);

if (i == 1 or i==2) {

glBegin(GL\_QUADS);

glTexCoord2f(0.0f, 0.0f); glVertex3f(1.0f, 1.0f, -1.0f);

glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, 1.0f, -1.0f);

glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, 1.0f, 1.0f);

glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, 1.0f, 1.0f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(1.0f, -1.0f, 1.0f);

glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, -1.0f, 1.0f);

glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, -1.0f, -1.0f);

glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, -1.0f, -1.0f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(1.0f, 1.0f, 1.0f);

glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, 1.0f, 1.0f);

glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, -1.0f, 1.0f);

glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, -1.0f, 1.0f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(1.0f, -1.0f, -1.0f);

glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, -1.0f, -1.0f);

glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, 1.0f, -1.0f);

glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, 1.0f, -1.0f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, 1.0f, 1.0f);

glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, 1.0f, -1.0f);

glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, -1.0f, -1.0f);

glTexCoord2f(1.0f, 0.0f); glVertex3f(-1.0f, -1.0f, 1.0f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(1.0f, 1.0f, -1.0f);

glTexCoord2f(0.0f, 1.0f); glVertex3f(1.0f, 1.0f, 1.0f);

glTexCoord2f(1.0f, 1.0f); glVertex3f(1.0f, -1.0f, 1.0f);

glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, -1.0f, -1.0f);

glEnd();

}

else if (i==3)

{

glTranslatef(-2.0f, 0.0f ,0.0f);

glBegin(GL\_QUADS);

glTexCoord2f(0.0f, 0.0f); glVertex3f(1.0f, 1.0f, -1.0f);

glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, 1.0f, -1.0f);

glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, 1.0f, 1.0f);

glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, 1.0f, 1.0f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(1.0f, -1.0f, 1.0f);

glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, -1.0f, 1.0f);

glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, -1.0f, -1.0f);

glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, -1.0f, -1.0f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(1.0f, 1.0f, 1.0f);

glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, 1.0f, 1.0f);

glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, -1.0f, 1.0f);

glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, -1.0f, 1.0f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(1.0f, -1.0f, -1.0f);

glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, -1.0f, -1.0f);

glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, 1.0f, -1.0f);

glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, 1.0f, -1.0f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, 1.0f, 1.0f);

glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, 1.0f, -1.0f);

glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, -1.0f, -1.0f);

glTexCoord2f(1.0f, 0.0f); glVertex3f(-1.0f, -1.0f, 1.0f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(1.0f, 1.0f, -1.0f);

glTexCoord2f(0.0f, 1.0f); glVertex3f(1.0f, 1.0f, 1.0f);

glTexCoord2f(1.0f, 1.0f); glVertex3f(1.0f, -1.0f, 1.0f);

glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, -1.0f, -1.0f);

glEnd();

glTranslatef(2.0f, 0.0f, 0.0f);

glBegin(GL\_QUADS);

glTexCoord2f(0.0f, 0.0f); glVertex3f(1.0f, 1.0f, -1.0f);

glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, 1.0f, -1.0f);

glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, 1.0f, 1.0f);

glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, 1.0f, 1.0f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(1.0f, -1.0f, 1.0f);

glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, -1.0f, 1.0f);

glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, -1.0f, -1.0f);

glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, -1.0f, -1.0f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(1.0f, 1.0f, 1.0f);

glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, 1.0f, 1.0f);

glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, -1.0f, 1.0f);

glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, -1.0f, 1.0f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(1.0f, -1.0f, -1.0f);

glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, -1.0f, -1.0f);

glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, 1.0f, -1.0f);

glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, 1.0f, -1.0f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, 1.0f, 1.0f);

glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, 1.0f, -1.0f);

glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, -1.0f, -1.0f);

glTexCoord2f(1.0f, 0.0f); glVertex3f(-1.0f, -1.0f, 1.0f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(1.0f, 1.0f, -1.0f);

glTexCoord2f(0.0f, 1.0f); glVertex3f(1.0f, 1.0f, 1.0f);

glTexCoord2f(1.0f, 1.0f); glVertex3f(1.0f, -1.0f, 1.0f);

glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, -1.0f, -1.0f);

glEnd();

glTranslatef(2.0f, 0.0f, 0.0f);

glBegin(GL\_QUADS);

glTexCoord2f(0.0f, 0.0f); glVertex3f(1.0f, 1.0f, -1.0f);

glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, 1.0f, -1.0f);

glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, 1.0f, 1.0f);

glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, 1.0f, 1.0f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(1.0f, -1.0f, 1.0f);

glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, -1.0f, 1.0f);

glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, -1.0f, -1.0f);

glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, -1.0f, -1.0f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(1.0f, 1.0f, 1.0f);

glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, 1.0f, 1.0f);

glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, -1.0f, 1.0f);

glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, -1.0f, 1.0f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(1.0f, -1.0f, -1.0f);

glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, -1.0f, -1.0f);

glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, 1.0f, -1.0f);

glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, 1.0f, -1.0f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, 1.0f, 1.0f);

glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, 1.0f, -1.0f);

glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, -1.0f, -1.0f);

glTexCoord2f(1.0f, 0.0f); glVertex3f(-1.0f, -1.0f, 1.0f);

glTexCoord2f(0.0f, 0.0f); glVertex3f(1.0f, 1.0f, -1.0f);

glTexCoord2f(0.0f, 1.0f); glVertex3f(1.0f, 1.0f, 1.0f);

glTexCoord2f(1.0f, 1.0f); glVertex3f(1.0f, -1.0f, 1.0f);

glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, -1.0f, -1.0f);

glEnd();

glTranslatef(-2.0f, 0.0f, 0.0f);

}

else if (i == 4) {

glutSolidTeapot(2);

}

;

}

glRotatef(-90.0, 1.0, 0.0, 0.0);

glTranslatef(0.0f, -2 \* size, 0.0f);

glEndList();

}

void mouseMove(int xx, int yy) {

if (use\_mouse) {

angle\_x -= (W\_WIDTH / 2 - xx) \* sensivity;

if ((angle\_y + (W\_HEIGHT / 2 - yy) \* sensivity < -4998.3) && (angle\_y + (W\_HEIGHT / 2 - yy) \* sensivity > -5001.3)) {

angle\_y += (W\_HEIGHT / 2 - yy) \* sensivity;

}

lx = sin(angle\_y) \* sin(angle\_x);

ly = -cos(angle\_y);

lz = -sin(angle\_y) \* cos(angle\_x);

glutWarpPointer(

W\_WIDTH / 2,

W\_HEIGHT / 2

);

}

}

void init\_surface(void) {

int u, v;

int m = 50; // 2

for (u = 0; u < size\_numb\_y; u++) {

for (v = 0; v < size\_numb\_x; v++) {

ctlpoints[u][v][0] = m \* ((GLfloat)u - 1.5);

ctlpoints[u][v][1] = m \* ((GLfloat)v - 1.5);

if ((u == 1 || u == 2) && (v == 1 || v == 2))

ctlpoints[u][v][2] = 3.0;

// else

// ctlpoints[u][v][2] = -1.0;

}

}

}

void display(void) {

glLoadIdentity();

move();

gluLookAt(x, y, z,

x + lx, y + ly, z + lz,

0.0f, 1.0f, 0.0f);

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

glClearColor(0.5f, 0.1f, 0.1f, 1.0f);

glPushMatrix();

glEnable(GL\_DEPTH\_TEST);

glEnable(GL\_BLEND);

glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, diffuseColor\_light);

glLightfv(GL\_LIGHT0, GL\_AMBIENT, ambientColor\_light);

glLightfv(GL\_LIGHT0, GL\_EMISSION, emis\_light);

glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position);

glLightfv(GL\_LIGHT0, GL\_SPOT\_DIRECTION, light\_direction);

glTranslatef(light\_position[0], light\_position[1], light\_position[2]);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, mat\_dif\_light);

glutSolidSphere(0.2, 32, 32);

glTranslatef(-light\_position[0], -light\_position[1], -light\_position[2]);

glTranslatef(0.0f, 0.0f, 0.0f);

glRotatef(-90.0, 1.0, 0.0, 0.0);

gluBeginSurface(theNurb);

glDisable(GL\_TEXTURE\_2D);

glBindTexture(GL\_TEXTURE\_2D, texture);

gluNurbsSurface(theNurb,

size\_numb\_x + size\_numb\_y, knots, size\_numb\_x + 1.13\*size\_numb\_y, knots,

size\_numb\_x \* 2, 3, &ctlpoints[0][0][0],

4, 4, GL\_MAP2\_VERTEX\_3);

gluEndSurface(theNurb);

glTranslatef(10.0f, 0.0f, 0.0f);

glCallList(theTorus);

glTranslatef(10.0f, 0.0f, -10.0f);

glScaled(2,2,2);

glCallList(theTorus);

glTranslatef(10.0f, 0.0f, -10.0f);

glScaled(2, 2, 2);

glCallList(theTorus);

glTranslatef(10.0f, 0.0f, -10.0f);

glScaled(2, 2, 2);

glCallList(theTorus);

glPopMatrix();

glDisable(GL\_TEXTURE\_2D);

glFlush();

}

void init(void) {

glClearColor(0.0, 0.0, 0.0, 0.0);

glEnable(GL\_MAP2\_VERTEX\_3);

glMapGrid2f(20, 0.0, 1.0, 20, 0.0, 1.0);

glEnable(GL\_DEPTH\_TEST);

glShadeModel(GL\_FLAT);

}

void reshape(int w, int h) {

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(40.0, (GLfloat)w / h, 1, 100.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glViewport(0, 0, (GLsizei)w, (GLsizei)h);

}

void timer(int value) {

glutPostRedisplay();

glutTimerFunc(1000 / refreshMills, timer, 0);

}

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

std::cout << "HELLO WORLD" << std::endl;

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGB | GLUT\_DEPTH);

glutInitWindowPosition(150, 50);

glutInitWindowSize(W\_WIDTH, W\_HEIGHT);

glutCreateWindow("nurbs");

glutReshapeFunc(reshape);

glutDisplayFunc(display);

glutSetKeyRepeat(GLUT\_KEY\_REPEAT\_OFF);

int width, height, nrChannels;

unsigned char\* data = stbi\_load("side.png", &width, &height, &nrChannels, 0);

glGenTextures(1, &texture);

std::cout << texture << std::endl;

glBindTexture(GL\_TEXTURE\_2D, texture);

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

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

glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGBA, width, height, 0, GL\_RGBA, GL\_UNSIGNED\_BYTE, data);

// stbi\_image\_free(data);

init();

fog\_on();

gen(1);

init\_surface();

GLfloat mat\_diffuse[] = { 0.9, 0.9, 0.9, 1.0 };

GLfloat mat\_specular[] = { 0.5, 0.5, 0.5, 1.0 };

GLfloat mat\_emission[] = { 0.5, 0.5, 0.5, 1.0 };

// GLfloat mat\_specular[] = { 0.0, 0.0, 0.0, 1.0 };

GLfloat mat\_shininess[] = { 128.0 };

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, mat\_diffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, mat\_specular);

glMaterialfv(GL\_FRONT, GL\_EMISSION, mat\_emission);

glMaterialfv(GL\_FRONT, GL\_SHININESS, mat\_shininess);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glEnable(GL\_DEPTH\_TEST);

glEnable(GL\_AUTO\_NORMAL);

glEnable(GL\_NORMALIZE);

theNurb = gluNewNurbsRenderer();

gluNurbsProperty(theNurb, GLU\_SAMPLING\_TOLERANCE, 30.0);

gluNurbsProperty(theNurb, GLU\_DISPLAY\_MODE, GLU\_FILL);

gluNurbsProperty(theNurb, GLU\_AUTO\_LOAD\_MATRIX, false);

glutKeyboardFunc(keyDown);

glutKeyboardUpFunc(keyUp);

glutPassiveMotionFunc(mouseMove);

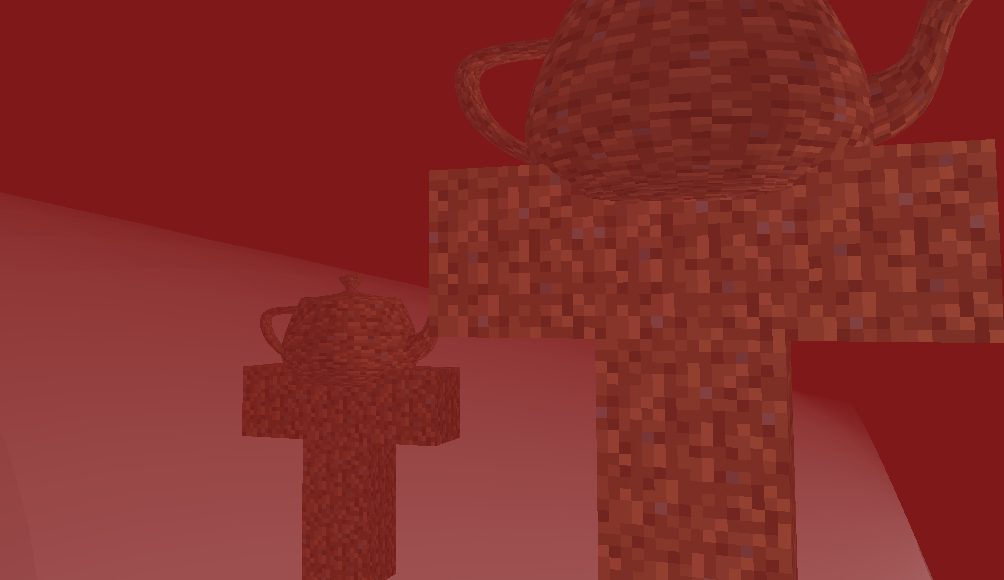
glutTimerFunc(0, timer, 0);

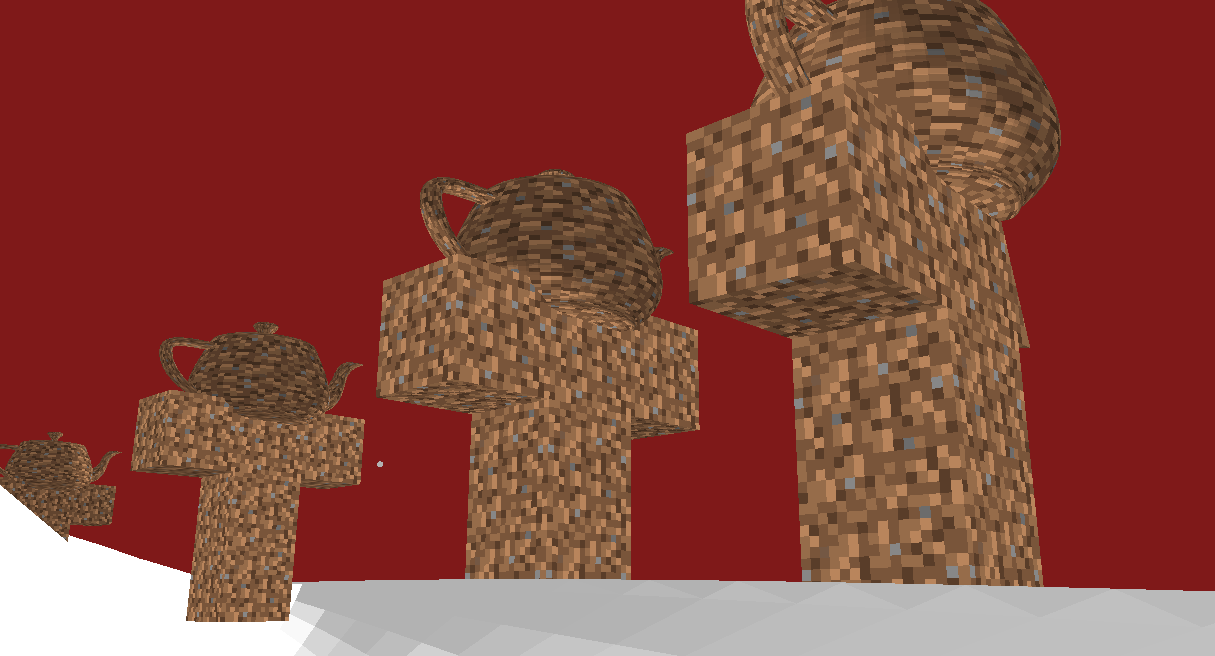
glutMainLoop();

return 0;

}

**Результат работы программы**





Вывод

Я освоил работу с nurb поверхностями и листами объектов