МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технологический университет «СТАНКИН»
(ФГБОУ ВО «МГТУ «СТАНКИН»)

Институт информационных систем и технологий

КАФЕДРА ИНФОРМАЦИОННЫХ СИСТЕМ

Программирование специализированных вычислительных устройств

Отчет по лабораторной работе №1

«Моделирование 3D объектов с использованием Visual C++ и OpenGL»

вариант №12

Выполнил студент гр. ИДБ-21-06 Музафаров К.Р.

Проверил Лаверычев М.А.

Москва 2022г.

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

Задачи:

1. Ознакомиться с теоретическими основами.
2. Построить каркасные и сплошные модели фигур и поверхностей, изменить цвет объектов, нанести текстуры.
3. Выполнить задание в соответствии со своим вариантом (табл.)(по согласованию с преподавателем задание может быть изменено). Ввести различное освещение объектов.
4. Составить и защитить отчет по работе.

ПРИМЕНЯЕМЫЕ ТЕХНОЛОГИИ

Библиотека OpenGL, и встроенные в нее функции, так же текстура дерева, взятая из интернета.

РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ЗАДАНИЯ

```
#include <GLUT/GLUT.h>//Подключениебиблиотеки glut.h
#include <iostream>

GLfloat xRotated, yRotated, zRotated;

// параметры для материала
float mat_dif[] = { 0.8,0.8,0.8 };
float mat_amb[] = { 0.2,0.2,0.2 };
float mat_spec[] = { 0.6,0.6,0.6 };
float shininess = 0.7 * 128;

GLuint groundTex;

void specialKeys(int key, int x, int y) {
   if (key == GLUT_KEY_RIGHT) {
   yRotated += 5;
   xRotated -= 5;
   }
   else if (key == GLUT_KEY_LEFT) {
   yRotated -= 5;
   xRotated += 5;
   }
```

```
    else if (key == GLUT_KEY_UP)
    zRotated += 5;
    else if (key == GLUT_KEY_DOWN)
    zRotated -= 5;
    glutPostRedisplay();
}

GLuint LoadTexture(const char* filename)
{//Функция считывания текстуры из файла
    GLuint texture;
    int width, height;
    unsigned char* data;
    FILE* file;
    file = fopen(filename, "rb");
    if (file == NULL) return 0; //Не считывать несуществующий файл
    width = 200; //Размер изображения в пикселях
    height = 200;
    data = (unsigned char*)malloc(width * height * 3);
    fread(data, width * height * 3, 1, file);
    fclose(file);

    for (int i = 0; i < width * height; ++i)
    {
    int index = i * 3;
    unsigned char B, R;
    B = data[index];
    R = data[index + 2];

    data[index] = R;
    data[index + 2] = B;
 }
 glGenTextures(1, &texture);
 glBindTexture(GL_TEXTURE_2D, texture);
 glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);
 glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_LINEAR_MIPMAP_NEAREST);

 glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
GL_LINEAR);
 glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
 glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
 gluBuild2DMipmaps(GL_TEXTURE_2D, 3, width, height, GL_RGB,
GL_UNSIGNED_BYTE, data);
 free(data);
 return texture;

}
```

```
void Ground() {//Отображение поверхности с текстурой снега
    groundTex = LoadTexture("Amaranth.bmp");
    glEnable(GL_TEXTURE_2D);
    glBindTexture(GL_TEXTURE_2D, groundTex);
    glEnable(GL_POLYGON_OFFSET_FILL);

    glColor3f(1.0f, 1.0f, 1.0f);
    glBegin(GL_QUADS);

    glTexCoord2f(0.0f, 0.0f);
    glVertex3f(-0.5, -2.3f, -2.3f);

    glTexCoord2f(0.0f, 1.0f);
    glVertex3f(-0.5, 2.3f, -2.3);

    glTexCoord2f(0.0f, 1.0f);
    glVertex3f(-0.5, 2.3f, 2.3f);

    glTexCoord2f(1.0f, 1.0f);
    glVertex3f(-0.5, -2.3f, 2.3f);

    glEnd();

    glDisable(GL_POLYGON_OFFSET_FILL);
    glBindTexture(GL_TEXTURE_2D, 0);
    glDisable(GL_TEXTURE_2D);
}

void init()
{
    glClearColor(0.3, 0.4, 0.4, 0.f);
    glMatrixMode(GL_PROJECTION);

    glEnable(GL_LIGHTING); // исп освещение

    glLoadIdentity();
    glOrtho(0.0, 10.0, 0.0, 10.0, -1.0, 1.0);

}

void getEnv(void) {
    // очистить буфер рисования.
    // очистить единичную матрицу
    glMatrixMode(GL_MODELVIEW);
    glEnable(GL_DEPTH_TEST);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glDepthFunc(GL_LESS);
```

```
    glLoadIdentity();
    glTranslatef(0.0, -.5, -4.5);
    // используется для рисования белый цвет
    glColor3f(.9, 1., 1.);
    // Поворот по Y
    glRotatef(yRotated, 0.0, 1.0, 0.0);
    // поворот по Z
    glRotatef(zRotated, 0.0, 0.0, 1.0);
    // масштабирование
    glScalef(1.0, 1.0, 1.0);
    glEnable(GL_COLOR_MATERIAL); // вот тут начало
    // использлвание 0 источника света

    glLightf(GL_LIGHT0, 1, GL_CONSTANT_ATTENUATION);
    glEnable(GL_LIGHT0);

    glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, mat_amb);
    glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, mat_dif);
    glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, mat_spec);
    glMaterialf(GL_FRONT, GL_SHININESS, shininess);

}

void drawSnowman(void)
{
    getEnv();
    // функция рисования снеговика
    Ground();
    glutSolidSphere(.5, 20, 20);
    glTranslatef(.7, 0, 0);
    glutSolidSphere(.3, 20, 20);
    glTranslatef(.45, 0, 0);
    glutSolidSphere(.2, 20, 20);
    glColor3f(1, .65, .0);
    glTranslatef(-.0, -.0, .2);
    glutSolidCone(.02, .4, 32, 1);
    glColor3f(0, 0, 0);
    glTranslatef(.05, .07, -.01);
    glutSolidSphere(.02, 20, 20);
    glTranslatef(.0, -.14, -.01);
    glutSolidSphere(.02, 20, 20);
    glTranslatef(-.15, -.01, -.01);
    glutSolidSphere(.02, 20, 20);
    glTranslatef(-0.0, .07, .01);
    glutSolidSphere(.02, 20, 20);
    glTranslatef(0.0, .07, -.01);
    glutSolidSphere(.02, 20, 20);
```

```
    glFlush();
    glutSwapBuffers();

}

void reshapeFunc(int x, int y)
{
    if (y == 0 || x == 0) return;
    // Установлюемо новую проекционную матрицу
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity(); // Угол обзора: 40 градусов
    // Возле плоскости отсечения расстояние: 0,5
    // Дальний отсечения плоскости расстояние: 20,0
    gluPerspective(40.0, (GLdouble)x / (GLdouble)y, 0.5, 20.0);
    glMatrixMode(GL_MODELVIEW);
    glViewport(0, 0, x, y);
    // Использование всего окна для rendering
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(400, 350);
    glutCreateWindow("Снеговичок");
    xRotated = yRotated = zRotated = 90.0;
    xRotated = 33;
    yRotated = 0;
    glClearColor(0.0, 0.0, 0.0, 0.0);
    init();
    glutDisplayFunc(drawSnowman);
    glutReshapeFunc(reshapeFunc);
    glutSpecialFunc(specialKeys);
    glutMainLoop();
    return 0;
}
```

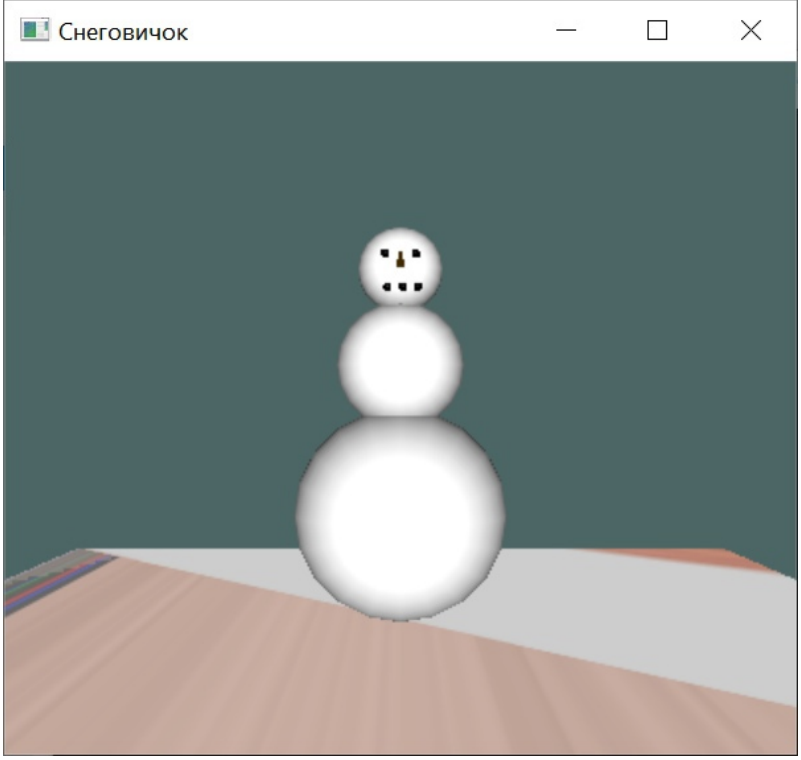

Рис.1. Результат выполнения кода

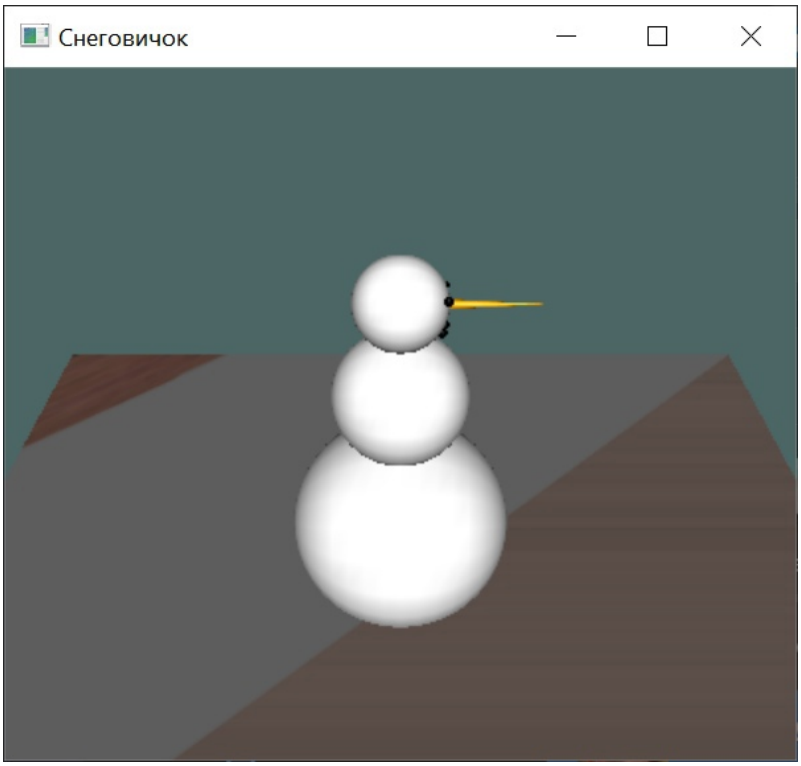

Рис.2. Результат выполнения кода