Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский Авиационный Институт»

Национальный Исследовательский Университет

Институт №8 «Информационные технологии и прикладная математика»

Кафедра 806 «Вычислительная математика и программирование»

Лабораторные работы №4-5-6

по курсу «Компьютерная графика»

Студент:	Попов И. П.
Группа:	М8О-306Б-20
Преподаватель:	Филиппов Г. С.
Подпись:	
Оценка:	
Дата:	

Лабораторные работы №4-5

Тема: Ознакомление с технологией OpenGL.

Задание: Создать графическое приложение с использованием OpenGL. Используя результаты Л.Р.№3, изобразить заданное тело (то же, что и в л.р. №3) с использованием средств OpenGL 2.1. Использовать буфер вершин. Точность аппроксимации тела задается пользователем. Обеспечить возможность вращения и масштабирования многогранника и удаление невидимых линий и поверхностей. Реализовать простую модель освещения на GLSL. Параметры освещения и отражающие свойства материала задаются пользователем в диалоговом режиме.

Лабораторная работа №6

Тема: Создание шейдерных анимационных эффектов в OpenGL 2.1

Задание: Для поверхности, созданной в л.р. №5, обеспечить выполнение следующего шейдерного эффекта:

Вариант: Анимация. Расстояние от вершины до заданной точки меняется по синусоиде

1 Описание

Программа написана на языке программирования Python с использованием библиотек GL для отрисовки трехмерного графика. В программе реализована возможность вращать фигуру с помощью клавиш клавиатуры.

2 Исходный код:

```
'''
Popov Ilya
M80-306Б-20

ЛР 45
Тема: Ознакомление с технологией OpenGL.
Задание: Создать графическое приложение с использованием OpenGL. Используя
результаты Л.Р.№3, изобразить
заданное тело (то же, что и в л.р. №3) с использованием средств OpenGL 2.1.
Использовать буфер вершин. Точность
```

```
аппроксимации тела задается пользователем. Обеспечить возможность вращения и масштабирования многогранника
и удаление невидимых линий и поверхностей. Реализовать простую модель освещения на GLSL.
Параметры освещения и отражающие свойства материала задаются пользователем в диалоговом режиме.

Прямой цилиндр, основание – сектор параболы.

ЛР 6
Тема: Создание шейдерных анимационных эффектов в OpenGL 2.1
Задание: Для поверхности, созданной в л.р. №5, обеспечить выполнение следующего шейдерного эффекта:

Анимация. Расстояние от вершины до заданной точки меняется по синусоиде
'''

from OpenGL.GL import *
from OpenGL.GLU import *
from OpenGL.GLUT import *
import numpy
import sys
import threading
import time
from itertools import cycle

xrot = 0
yrot = 0
zrot = 0
h = 3.25
app = 4
intensive = 10
reflection = 116
light_coord = (20, 30, 30)
zoom = 4

def SetFigure(app, h):
    x = numpy.linspace(-1, 1, app)
    y = 2*x**2
    verts = []

    #координаты вершин, на которые действуют правила аппроксимации
    for i in range(len(x) - 1):
        tmp = []
        tmp.append((x[i], y[i], 0))
        tmp.append((x[i], y[i], h))
        tmp.append((x[i+1], y[i+1], h))
        tmp.append((x[i+1], y[i+1], 0))
        verts.append(tmp)
```

```
    #задняя стенка
    tmp = []
    tmp.append((x[-1], y[-1], 0))
    tmp.append((x[-1], y[-1], h))
    tmp.append((x[0], y[0], h))
    tmp.append((x[0], y[0], 0))
    verts.append(tmp)

    glBegin(GL_QUADS) #задаем примитивы прямоугольнииками
    for v in verts:
        n = numpy.cross(numpy.array(v[3]) - numpy.array(v[1]),
                        numpy.array(v[0]) - numpy.array(v[1]))

        glNormal3fv(n)# задаем нормаль

        # задаем 4 координаты прямоугольника
        glVertex3fv(v[0])
        glVertex3fv(v[1])
        glVertex3fv(v[2])
        glVertex3fv(v[3])
    glEnd()

    l = [(x[i], y[i], 0) for i in range(len(x))]
    coord_centr = numpy.array([0, 1, 0])
    l2 = [(x[i], y[i], h) for i in range(len(x))]

    glBegin(GL_TRIANGLES) #задаем примитивы треугольнииками
    for i in range(0, len(l)):
        n = numpy.cross(coord_centr - numpy.array(l[i]),
                        numpy.array(l[i - 1]) - numpy.array(l[i]))
        n = -n
        glNormal3fv(n)
        glVertex3fv(l[i - 1])
        glVertex3fv(l[i])
        glVertex3fv(coord_centr)

    coord_centr = numpy.array([0, 1, h])
    for i in range(0, len(l2)):
        n = numpy.cross(coord_centr - numpy.array(l2[i]),
                        numpy.array(l2[i - 1]) - numpy.array(l2[i]))
        glNormal3fv(n)
        glVertex3fv(l2[i - 1])
        glVertex3fv(l2[i])
        glVertex3fv(coord_centr)
    glEnd()

def RotateFn():
    global zrot
    speed = [1 / 100000]
    for s in cycle(speed):
```

```
        begin = time.time()
        while time.time() - begin < 1:
            zrot += s
            glutPostRedisplay()

def DrawFn(): #работает с включённой двойной буфферизацией
    global xrot, yrot, app, reflection, h
    # Сохраняем текущее состояние в стек
    glPushMatrix()

    #аппроксимируемая часть GL_DIFFUSE(чем глубже, тем темнее)
    glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, (0.0, 0.0, 2.0, 1.0))

    # основания
    glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, (0.0, 0.0, 2.0, 1.0))

    #задаем рефлекс
    glMaterialf(GL_FRONT_AND_BACK, GL_SHININESS, 128 - reflection)
    SetFigure(app + 1, h)

    # Возвращаемся в сохранённое состояние
    glPopMatrix()

    glutSwapBuffers()

def IntensiveChangeFn(x):
    global intensive
    intensive = x
    glutPostRedisplay() #помечает текущее окно как требующее повторного
отображения
    return 0

def ApproximationChangeFn(x):
    global app
    app = x
    glutPostRedisplay() #помечает текущее окно как требующее повторного
отображения
    return 0

def LightingFn():
    global light_coord

    glEnable(GL_LIGHT0)
    # интенсивность цветов
    light_intensity = (1.0, 1.0, 1.0)

    glLightfv(GL_LIGHT0, GL_DIFFUSE,  light_intensity)
    light_position = (light_coord[0], light_coord[1], light_coord[2], 1.0)

    glLightfv(GL_LIGHT0, GL_POSITION, light_position)
```

```
    attenuation = float(101 - intensive) / 25.0 #задаем новую интенсивность

    # обработка затухания
    # Расстояние между положением источника света и вершиной
    distance = numpy.sqrt(pow(light_coord[0], 2) +
                          pow(light_coord[1], 2) + pow(light_coord[2], 2))

    kQ = attenuation / (3.0 * distance * distance)
    kL = attenuation / (3.0 * distance)
    kC = attenuation / 3.0

    glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, kC)
    glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, kL)
    glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, kQ)

def keys(key, x, y):
    global xrot, yrot, zrot, zoom
    if key == b'w':
        xrot += 2
    elif key == b's':
        xrot -= 2

    elif key == b'a':
        yrot += 2
    elif key == b'd':
        yrot -= 2

    elif key == b'q':
        zrot += 2
    elif key == b'e':
        zrot -= 2

    elif key == b'b':
        zoom += 1
    elif key == b'n':
        zoom -= 1

    elif key == b'z':
        IntensiveChangeFn(intensive + 5)
        LightingFn()
    elif key == b'x':
        IntensiveChangeFn(intensive - 5)
        LightingFn()

    elif key == b'c':
        ApproximationChangeFn(app + 1)
    elif key == b'v':
        ApproximationChangeFn(app - 1)
    # Перерисовка изображения
    glutPostRedisplay()
```

```
def init():
    # свет
    glClearColor(255, 255, 255, 1.0)

    glClearDepth(1.0)
    #glEnable(GL_DEPTH_TEST)
    glShadeModel(GL_FLAT)

    # Фрагмент проходит тест, если его значение глубины меньше либо равно
хранимому в буфере
    glDepthFunc(GL_LEQUAL)
    glEnable(GL_DEPTH_TEST)
    glEnable(GL_NORMALIZE)

    # уменьшаем ступенчатость прямых за счёт увеличение пикселей
    glHint(GL_POLYGON_SMOOTH_HINT, GL_NICEST)
    glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST)

    # Включаем освещение
    glEnable(GL_LIGHTING)
    glLightModelf(GL_LIGHT_MODEL_TWO_SIDE, GL_TRUE)

    #glEnable(GL_NORMALIZE)

def DisplayFn():
    global zoom
    # очищаем цветовой и глубинный буферы
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)

    # Переходим в режим просмотра (работать будем с просмотром)
    glMatrixMode(GL_MODELVIEW)

    # считываем текущую матрицу
    glLoadIdentity()

    # задаём координаты точки просмотра, центра, направдение вертикального вектора
    gluLookAt(10, 10, 10, 0, 0, 0, 0, 0, 1)
    glTranslatef(zoom, zoom, zoom)
    LightingFn()

    # Вращение вокруг осей
    glRotatef(xrot, 1, 0, 0) #умножает текущую матрицу на матрицу вращения
    glRotatef(yrot, 0, 1, 0)
    glRotatef(zrot, 0, 0, 1)
    DrawFn()

def ReshapeFn(width, height):
    # устанавливаем область просмотра
    glViewport(0, 0, width, height)
```

```
    # Переходим в режим проекта - для взаимодействия с окном окном проекта
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    # задаём угол поля зрения, соотношение сторон, расстояние до ближайшей
плоскости и дальшей плоскости
    gluPerspective(60.0, float(width) / float(height), 1.0, 60.0)

    # Переходим в режим просмотра (работать будем с просмотром)
    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()

    # задаём координаты точки глаза наблюдателя, коорнаты центра экрана,
направление вектора задающего поворот сцеры
    gluLookAt(0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1, 0.0)

def main():
    glutInit(sys.argv)
    glutInitWindowSize(600, 400)
    glutInitWindowPosition(300, 150)
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH)

    glutCreateWindow(b"lab456")
    glutDisplayFunc(DisplayFn)
    glutReshapeFunc(ReshapeFn)
    glutKeyboardFunc(keys)

    init()
    t = threading.Thread(target=RotateFn)

    t.daemon = True #работает в фоне
    t.start()

    glutMainLoop()


if __name__ == "__main__":
    main()
```

3 Работа программы:

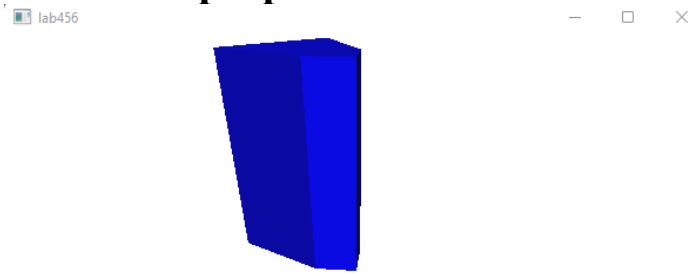

Рис.1.

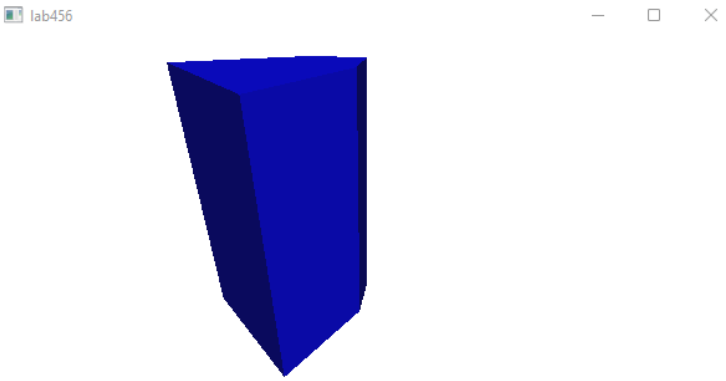

Рис.2.

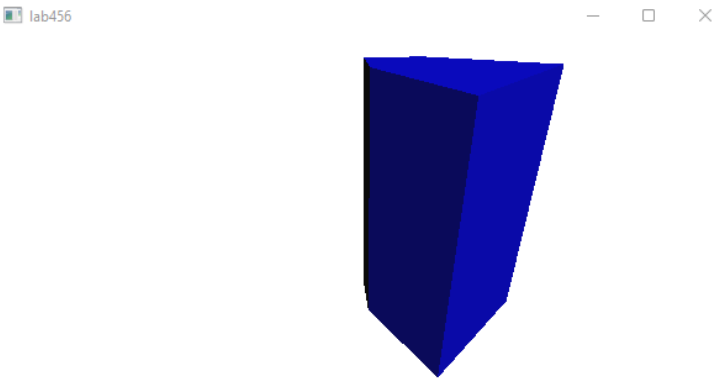

Рис.3.

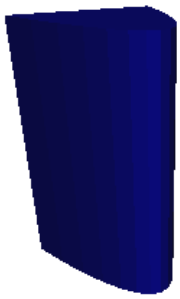

Рис.4.
На рисунках 1-3 точность аппроксимации равна 5
На рисунке 5 точность аппроксимации равна 100

4 Выводы:

В ходе выполнения данной лабораторной работы была написана программа на языке Python для аппроксимации прямого цилиндра, основанием которого является сектор параболы в трехмерном пространстве с использованием библиотеки OpenGL. В процессе данной работы я получил опыт работы с библиотекой GL и принципами построения анимационных эффектов.