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»

Лабораторная работа №4-5, 6

по курсу «Компьютерная графика»

Студент:	Хренникова А. С.
Группа:	М8О-308Б-19
Преподаватель:	Филиппов Г. С.
Подпись:	
Оценка:	
Дата:	

Москва, 2021

Лабораторная работа №4-5, 6

Задача: 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. Для поверхности, созданной в л.р. №5, обеспечить выполнение следующего шейдерного эффект.

Вариант эффекта: Анимация. Координата Y изменяется по закону Y = Y*cos(t+Y)**.**

1 Описание

Программа написана на языке программирования Python с использованием библиотек GL для отрисовки трехмерного графика.

Вычисление координат, для аппроксимированной фигуры:

$x = \cos(\frac{2*i*\pi}{n})$

$y = \sin(\frac{2*i*\pi}{n})$

$z = \sum_{i=1}^{n}(\frac{1}{2})^i$, где n - точность аппроксимации, i - целое число от 1 до n.

В программе реализована возможность вращать фигуру с помощью клавиш клавиатуры.

2 Исходный код:

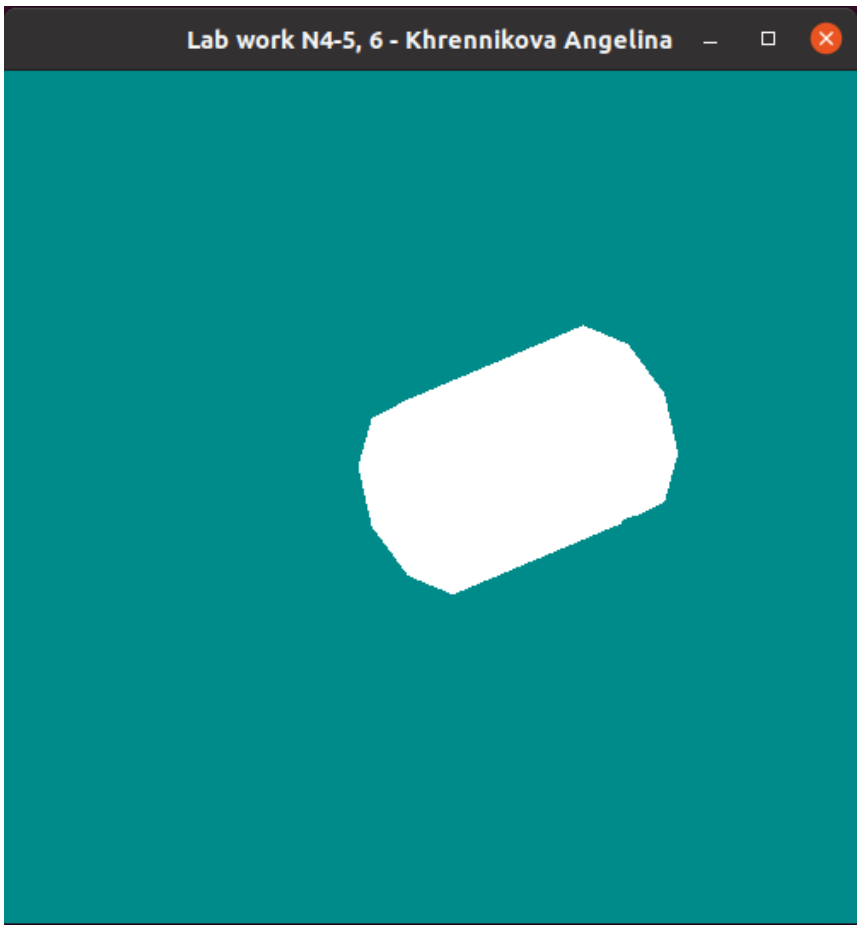
Москва, 2021

```
from OpenGL.GL import *
from OpenGL.GLU import *
from OpenGL.GLUT import *
import numpy as np
from math import sqrt, cos, pi

a = float(input("Радиус: "))
h = float(input("Высота: "))
alpha = int(input("Точность аппроксимации: "))

def init():
	global xrot
	global yrot
	global ambient
	global prismcolor
	global lightpos
	global n
	global points
	global t
	global a
	global h
	global alpha

	n = alpha
	a *= 0.3
	h *= 0.3
	a1 = (sqrt(5) - 1) / 4
	b1 = (sqrt(5) + 5) / 8
	x_1 = np.array([0.5, 0.0]) * a
	x_2 = np.array([x_1[0] * a1 - x_1[1] * b1, x_1[0] * b1 - x_1[1] * a1]) * a
	x_3 = np.array([x_2[0] * a1 - x_2[1] * b1, x_2[0] * b1 - x_2[1] * a1]) * a
	x_4 = np.array([x_3[0] * a1 - x_3[1] * b1, x_3[0] * b1 - x_3[1] * a1]) * a
	x_5 = np.array([x_4[0] * a1 - x_4[1] * b1, x_4[0] * b1 - x_4[1] * a1]) * a
	x_6 = np.array([0.0, 0.0])

	points = np.array([np.hstack((x_1, 0.0)),
                                           np.hstack((x_2, 0.0)),
                                           np.hstack((x_3, 0.0)),
                                           np.hstack((x_4, 0.0)),
                                           np.hstack((x_5, 0.0)),
                                           np.hstack((x_6, [h]))])

	t = 0.
	xrot = 0.0
	yrot = 0.0
	ambient = (1.0, 1.0, 1.0, 1)
	prismcolor = (1, 1, 1, 1)
	lightpos = (1.0, 1.0, 1.0)

	glClearColor(0.0, 0.5451, 0.5451, 1.0)
	gluOrtho2D(-2.5, 2.5, -2.5, 2.5)
	glRotatef(-90, 1.0, 0.0, 0.0)
	glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambient)
	glEnable(GL_LIGHTING)
	glEnable(GL_LIGHT0)
	glLightfv(GL_LIGHT0, GL_POSITION, lightpos)


def specialkeys(key, x, y):
	global xrot
```

```
	global yrot
	if key == GLUT_KEY_UP:
		xrot -= 5.0
	if key == GLUT_KEY_DOWN:
		xrot += 5.0
	if key == GLUT_KEY_LEFT:
		yrot -= 5.0
	if key == GLUT_KEY_RIGHT:
		yrot += 5.0

	glutPostRedisplay()

def TimeFunction(value):
	global t

	glutPostRedisplay()
	glutTimerFunc(60, TimeFunction, 1)
	t += 0.01 * 2 * pi
	if (t == pi):
		t = 0

def make_sides(p):
	global n

	k = []
	s = []

	sides = [[p[0], p[3], p[2*n+1]], [p[1], p[2], p[2*n]], [p[3], p[2*n], p[2*n+1]], [p[2], p[3], p[2*n]]]

	for i in range(2, 2*n-1, 2):
		sides = sides + [[p[0], p[i+1], p[i+3]], [p[1], p[i], p[i+2]], [p[i], p[i+1], p[i+3]], [p[i], p[i+2],
p[i+3]]]

	for i in range(1, n+1):
		k += [p[i+1]]
		s += [p[i+1+n]]

	sides += [k, s]

	return sides

def approximate(list_of_points):
	global n

	h = list_of_points[0][5][2]
	R = 2 * list_of_points[0][0][0]

	list_of_points1 = []
	list_of_points1 += [np.hstack((0.0, 0.0, [h])), np.hstack((0.0, 0.0, 0.0))]

	x_1 = np.array([0., 0.])
	for i in range(1, n + 1):
		x_1[0] = np.cos(2 * np.pi * i / n) * R
		x_1[1] = np.sin(2 * np.pi * i / n) * R
		z = sum([0.5 ** i for i in range(1, n - 4)]) * h

		list_of_points1 += [np.hstack((x_1, 0.0)), np.hstack((x_1, [z]))]
	return list_of_points1

def draw():
```

```
        global xrot
        global yrot
        global lightpos
        global prismcolor
        global points
        global n

        glClear(GL_COLOR_BUFFER_BIT)
        glPushMatrix()
        glRotatef(xrot, 1.0, 0.0, 0.0)
        glRotatef(yrot, 0.0, 1.0, 0.0)

        glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, prismcolor)

        prisms = []
        list_of_points = [points]
        list_of_points1 = approximate(list_of_points)
        prisms += [make_sides(list_of_points1)]

        glBegin(GL_POLYGON)
        glColor3f(0., 0., 0.)
        for q in range(len(prisms)):
                for i in range(len(prisms[q])):
                        for j in range(len(prisms[q][i])):
                                x = prisms[q][i][j][0]
                                y = prisms[q][i][j][1] + cos(t)
                                z = prisms[q][i][j][2]
                                glVertex3f(x, y, z)
        glEnd()

        glPopMatrix()
        glutSwapBuffers()

#сама работа OpenGL
def main():
        glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB)
        glutInitWindowSize(500, 500)
        glutInitWindowPosition(523, 150)
        glutInit(sys.argv)
        glutCreateWindow(b"Figure")
        glutDisplayFunc(draw)
        glutSpecialFunc(specialkeys)
        glutTimerFunc(600, TimeFunction, 1)
        init()
        glutMainLoop()

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

3 Работа программы:



4 Выводы:

В ходе выполнения данной лабораторной работы была написана программа на языке Python для аппроксимации цилиндра пятигранной прямой правильной пирамиды (где координата Y меняется по закону Y = Y*cos(t+Y)) в трехмерном пространстве с использованием библиотеки OpenGL. Я познакомилась как с самой библиотекой, так и с принципами построения анимационных эффектах в рамках данной работы.

Москва, 2021