Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Московский государственный технический университет имени Н.Э.Баумана (национальный исследовательский университет)»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Джабаров Р.А. ИУ9-32Б

Компьютерная графика.

Лабораторная работа №2

Москва

2023

CODE:

import math

from OpenGL.GL import \*

from OpenGL.GLU import \*

from OpenGL.GLUT import \*

import numpy as np

object\_matrix = np.identity(4)

last\_mouse\_x = None

last\_mouse\_y = None

def handle\_mouse\_down(button, state, x, y):

global mouse\_down, last\_mouse\_x, last\_mouse\_y

if button == GLUT\_LEFT\_BUTTON and state == GLUT\_DOWN:

last\_mouse\_x = x

last\_mouse\_y = y

def handle\_mouse\_move(x, y):

global object\_matrix, last\_mouse\_x, last\_mouse\_y

delta\_x = x - last\_mouse\_x

delta\_y = y - last\_mouse\_y

rotation\_y = np.identity(4)

rotation\_x = np.identity(4)

rotation\_y[0, 0] = np.cos(delta\_x / 100.0)

rotation\_y[2, 0] = -np.sin(delta\_x / 100.0)

rotation\_y[0, 2] = np.sin(delta\_x / 100.0)

rotation\_y[2, 2] = np.cos(delta\_x / 100.0)

rotation\_x[1, 1] = np.cos(delta\_y / 100.0)

rotation\_x[2, 1] = np.sin(delta\_y / 100.0)

rotation\_x[1, 2] = -np.sin(delta\_y / 100.0)

rotation\_x[2, 2] = np.cos(delta\_y / 100.0)

object\_matrix = np.dot(rotation\_x, object\_matrix)

object\_matrix = np.dot(rotation\_y, object\_matrix)

last\_mouse\_x = x

last\_mouse\_y = y

glutPostRedisplay()

scale = 1.0

def mouse\_wheel\_callback(wheel, direction, x, y):

global scale

if direction > 0:

scale += 0.1

else:

scale -= 0.1

glutPostRedisplay()

n = 8

def draw\_scene():

global n

global object\_matrix, scale

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

glMatrixMode(GL\_MODELVIEW)

glLoadIdentity()

glPushMatrix()

glMultMatrixf(object\_matrix)

glScalef(scale\*0.25, scale\*0.25, scale\*0.25)

globalMas = []

vectors = []

for i in range(n+1):

theta = 2 \* math.pi \* i / n

for j in range(n+1):

phi = math.pi \* j / n

x = math.cos(theta) \* math.sin(phi)

y = math.sin(theta) \* math.sin(phi)

z = math.cos(phi)

vectors.append([x, y, z])

#glVertex3f(x, y, z)

#print(vectors, "\n\n")

globalMas.append(vectors)

vectors = []

globalMas = np.array(globalMas)

#print(globalMas)

print("\n\n")

for i in range(len(globalMas) - 1):

glColor3f(0, i % 2, 1)

for j in range(len(globalMas[0]) - 1):

glBegin(GL\_POLYGON)

glVertex3f(globalMas[i + 1][j][0], globalMas[i + 1][j][1], globalMas[i + 1][j][2])

glVertex3f(globalMas[i][j][0], globalMas[i][j][1], globalMas[i][j][2])

glVertex3f(globalMas[i][j + 1][0], globalMas[i][j + 1][1], globalMas[i][j + 1][2])

glVertex3f(globalMas[i + 1][j + 1][0], globalMas[i + 1][j + 1][1], globalMas[i + 1][j + 1][2])

glEnd()

glPopMatrix()

glutSwapBuffers()

# глобальная переменная-флаг

wireframe\_mode = False

def handle\_key\_press(key, x, y):

global n, scale

if key == b'q':

n -= 1

if key == b'e':

n += 1

global wireframe\_mode

if key == b'x':

wireframe\_mode = not wireframe\_mode

if wireframe\_mode:

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_LINE)

glDisable(GL\_CULL\_FACE)

else:

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_FILL)

glEnable(GL\_CULL\_FACE)

elif key == b'o':

scale -= 0.1

elif key == b'p':

scale += 0.1

glutPostRedisplay()

def reshape(width, height):

aspect = float(width) / float(height)

glViewport(0, 0, width, height)

glMatrixMode(GL\_PROJECTION)

glLoadIdentity()

if aspect > 1:

glOrtho(-aspect, aspect, -1.0, 1.0, -1.0, 1.0)

else:

glOrtho(-1.0, 1.0, -1.0/aspect, 1.0/aspect, -1.0, 1.0)

glMatrixMode(GL\_MODELVIEW)

glLoadIdentity()

glutInit()

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH)

glutInitWindowSize(600, 600)

glutCreateWindow(b"PyOpenGL Example")

glutReshapeFunc(lambda w, h: glViewport(0, 0, w, h))

glutMouseFunc(handle\_mouse\_down)

glutMotionFunc(handle\_mouse\_move)

glutKeyboardFunc(handle\_key\_press)

glutReshapeFunc(reshape)

glEnable(GL\_CULL\_FACE)

glutDisplayFunc(draw\_scene)

glutMainLoop()

**Задача**:

Написать программу на Python, используя библиотеки OpenGL, для визуализации шара. Реализовать возможность вращения шара с помощью мыши, масштабирование с помощью колеса мыши, изменение количества разбиений шара и переключение между режимом отображения полигонов и режимом отображения линий.

**Теория**:

OpenGL (Open Graphics Library) - кросс-платформенная библиотека для создания приложений, работающих с 2D и 3D графикой. Она позволяет создавать сложные трехмерные сцены, анимации, игры и прочее.

glMatrixMode() - функция, устанавливающая текущую матрицу для преобразований. glLoadIdentity() - функция, загружающая единичную матрицу для текущей матрицы.

glPushMatrix() и glPopMatrix() - функции для сохранения и восстановления матрицы преобразования.

glViewport() - функция, устанавливающая область вывода на экран.

glOrtho() - функция, устанавливающая ортографическую проекцию.

glPolygonMode() - функция, устанавливающая режим отображения полигонов (линии или полигоны).

glEnable() и glDisable() - функции, включающие и выключающие определенные функции OpenGL (например, отсечение граней или освещение).

Практическая реализация:

Данный код на языке Python использует библиотеки [OpenGL.GL](https://opengl.gl/), OpenGL.GLU и OpenGL.GLUT для визуализации шара. В основе кода лежит матричная трансформация объекта с помощью мыши и масштабирование с помощью колеса мыши. В цикле отрисовки draw\_scene() генерируется набор вершин для шара и отображается в виде полигонов с помощью функций OpenGL.

**Заключение**:

Данный код демонстрирует простую реализацию визуализации с использованием библиотек OpenGL. Он также демонстрирует использование матричных операций для трансформации объекта и масштабирования его размера. Код может быть использован как основа для более сложных визуализаций на основе OpenGL.