**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 №6**

**по курсу «Компьютерная графика»**

|  |  |
| --- | --- |
| Студент: | Семин А. В. |
| Группа: | М8О-306Б-20 |
| Преподаватель: | Филиппов Г. С. |
| Подпись: |  |
| Оценка: |  |
| Дата: |  |

# 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 20:** Анимация. Изменение цвета источника рассеянного света по синусоидальному закону.

# Описание

В качестве фигуры и всего прочего используются наработки, созданные в 5 лабораторной работе. Для создания анимации создается отдельный поток, который работает постоянно и параллельно нашей программе. Данный поток изменяет цвета фигуры по синусоидальному закону. Формула изменения и все константы в ней подобраны практическим путем.

# Исходный код:

from OpenGL.GL import \*

from OpenGL.GLU import \*

from OpenGL.GLUT import \*

import time

import threading

import math

import sys

import numpy as np

# параметры освещения

light\_pos = (0,10, 10) # положение источника света

light\_intensity = 0 # интенсивность света

# фоновое освещение - окружающее освещеие, которое всегда будет придавать объекту некоторый оттенок

amb\_light = [0.5, 0.4, 0.2]

# диффузное освещение - имитирует воздействие на объект направленного источника света

diff\_light = [1.5, 1.0, 0.0, light\_intensity]

# вращение

x\_rot = 0

y\_rot = -40

z\_rot = 0

# параметры фигуры

approx = 30 # количество образующих

size = 1

a, b, c = 5, 3, 3

def init():

glClearColor(0.44, 0.44, 0.7, 0) # цвет для заднего фона

glEnable(GL\_DEPTH\_TEST) # обновляем буфер глубины

glDepthFunc(GL\_LEQUAL)

glLightModelfv(GL\_LIGHT\_MODEL\_AMBIENT, amb\_light) # определяем текущую модель освещения

glEnable(GL\_LIGHTING) # включаем освещение

glLightModelf(GL\_LIGHT\_MODEL\_TWO\_SIDE, GL\_TRUE) # вершины заднего многоугольника зажигаются с помощью параметров

# заднего материала и имеют обратную норму перед вычислением уравнения освещения

def ellipsoid\_layer():

global a, b, c, approx

longitude\_delta = np.linspace(0, np.pi\*2, approx + 1) #долгота

latitude\_delta = np.linspace(np.pi/4, np.pi\*3/4, approx + 1) #широта

vertices = []

for i in range(approx + 1):

lat = latitude\_delta[i]

for j in range(approx + 1):

lon = longitude\_delta[j]

x = a \* math.sin(lat) \* math.cos(lon)

y = b \* math.sin(lat) \* math.sin(lon)

z = c \* math.cos(lat)

vertices.append([x, y, z]) #создание массива координат вершин

for i in range(approx + 1):

glBegin(GL\_TRIANGLE\_STRIP)

for j in range(approx + 1): # задаем веришны

glVertex3fv(vertices[j + i \* approx])

glVertex3fv(vertices[j + (i + 1) \* approx])

glEnd()

def display():

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

glMatrixMode(GL\_MODELVIEW)

glLoadIdentity()

gluLookAt(7,7, 7, 0, 0, 0, 0, 0, 2)

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

glTranslatef(size, size, size)

init\_lighting()

glRotatef(x\_rot, 1, 0, 0)

glRotatef(y\_rot, 0, 0, 1)

glRotatef(z\_rot, 0, 1, 0)

glPushMatrix() # сохраняем текущее положение "камеры"

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_DIFFUSE, diff\_light)

ellipsoid\_layer()

glPopMatrix() # возвращаем сохраненное положение "камеры"

glutSwapBuffers() # выводим все нарисованное в памяти на экран

def init\_lighting():

glEnable(GL\_LIGHT0) # инициализация источника света

glLightfv(GL\_LIGHT0, GL\_POSITION, light\_pos) # положение источника света

l\_dif = (2.0, 2.0, 3.0, light\_intensity)

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, l\_dif)

l\_dir = (light\_pos[0], light\_pos[1], light\_pos[2], 1.0)

glLightfv(GL\_LIGHT0, GL\_POSITION, l\_dir)

# затухание света

attenuation = float(100 - light\_intensity) / 30.0

distance = math.sqrt(pow(light\_pos[0], 2) + pow(light\_pos[1], 2) + pow(light\_pos[2], 2))

linear\_attenuation = attenuation / (3.0 \* distance)

quadratic\_attenuation = attenuation / (3.0 \* distance \* distance)

# glLightf(GL\_LIGHT0, GL\_CONSTANT\_ATTENUATION, constant\_attenuation)

glLightf(GL\_LIGHT0, GL\_LINEAR\_ATTENUATION, linear\_attenuation)

glLightf(GL\_LIGHT0, GL\_QUADRATIC\_ATTENUATION, quadratic\_attenuation)

def rescale(width, height):

glViewport(0, 0, width, height)

glMatrixMode(GL\_PROJECTION)

glLoadIdentity()

gluPerspective(90, float(width) / float(height), 1.0, 60.0) # 1)Поле угла зрения в градусах в направлении y.;

# 2) Пропорции, определяющие поле представления в направлении x - отношение x/y;

# 3) расстояние от наблюдателя до ближней плоскости; 4) до дальней плоскости

glMatrixMode(GL\_MODELVIEW)

glLoadIdentity()

gluLookAt(0, 0.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1, 0)

def keys(key, x, y):

global x\_rot, y\_rot, z\_rot, size, approx, light\_intensity

# вращаем на +-5 градусов по оси X

if key == b'w':

x\_rot += 5

if key == b's':

x\_rot -= 5

# вращаем на +-5 градусов по оси Y

if key == b'a':

y\_rot += 5

if key == b'd':

y\_rot -= 5

# вращаем на +-5 градусов по оси Z

if key == b'q':

z\_rot += 5

if key == b'e':

z\_rot -= 5

# размер фигуры +-1

if key == b'=':

size += 1

if key == b'-':

size -= 1

# изменение числа образующих на +-1

if key == b'.':

approx += 1

if key == b',':

approx -= 1

approx = max(10, approx)

# интенсивность света +-5

if key == b'0':

light\_intensity += 5

light\_intensity = min(100, light\_intensity)

if key == b'9':

light\_intensity -= 5

light\_intensity = max(-100, light\_intensity)

glutPostRedisplay() # помечает, что текущее окно требует повторного отображения

def change\_color():

global amb\_light, diff\_light, light\_intensity

i = 0.01

while True:

diff\_light = [np.sin(i \* np.pi), np.sin(i +0.02\* np.pi), np.sin(i+ 0.13 \* np.pi), light\_intensity]

amb\_light = [np.sin(i\* np.pi), np.sin(i + 0.02 \* np.pi), 0.2]

i = i + 0.01

glutPostRedisplay()

time.sleep(0.01)

def main():

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

glutInitWindowSize(500, 500)

glutInit(sys.argv)

glutCreateWindow("lab 6")

glutDisplayFunc(display) # функция отрисовки

glutReshapeFunc(rescale) # функция масштабирования

glutKeyboardFunc(keys) # функция обработки нажатий

init()

t = threading.Thread(target=change\_color)

t.daemon = True

t.start()

glutMainLoop()

print("Оx: W S")

print("Oy: A D")

print("Oz: Q E")

print("Приближение/удаление фигуры: - +")

print("Изменение аппроксимации(en): < >")

print("Изменение интенсивности света: 9 0")

main()

# Работа программы:

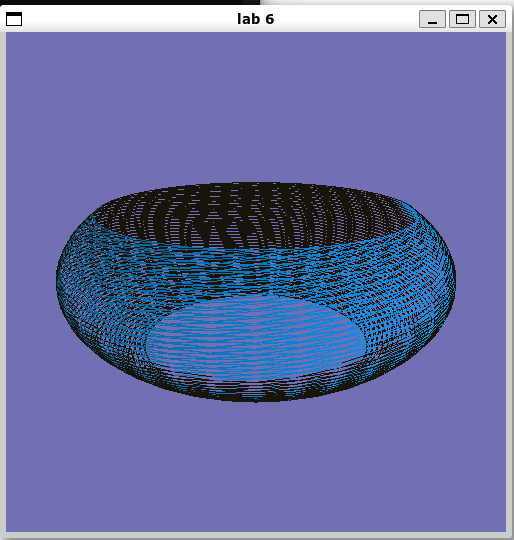


Рис.1.

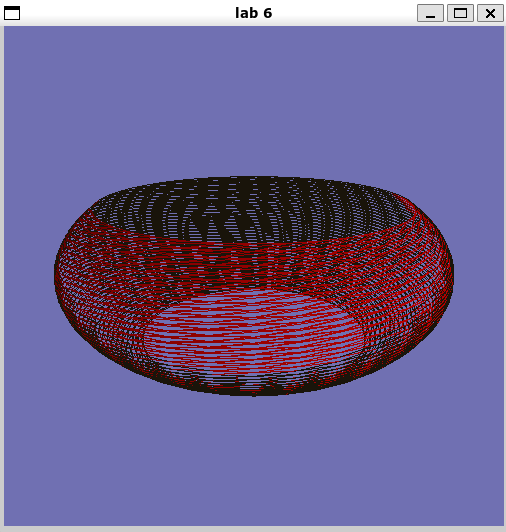


Рис.2.

1. **Выводы:**

В ходе выполнения данной лабораторной работы была написана программа на языке Python для создания анимации на слое эллипсоида в трехмерном пространстве с использованием библиотеки OpenGL. В процессе выполнения работы я приобрел опыт работы с потоками, а также опыт более тщательной работы с освещением фигуры.