

Московский авиационный институт
(Национальный исследовательский университет)

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Компьютерная графика»

Лабораторная работа № 4-5

Тема: Ознакомление с технологией OpenGL

Студент: Ильиных Вадим
Максимович

Группа: 80-301

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2021

1. Постановка задачи

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.

Вариант 11:

Прямой усеченный круговой конус

2. Описание программы

Работа выполнена на ЯП Python с использованием библиотек pygame, PyOpenGL и tkinter. С помощью первой реализовано окно программы. Вторая использовалась для отрисовки заданного тела. Третья же была нужна для изменения параметров заданного тела и параметров освещения.

Программа состоит из двух файлов - main.py и widgets.py

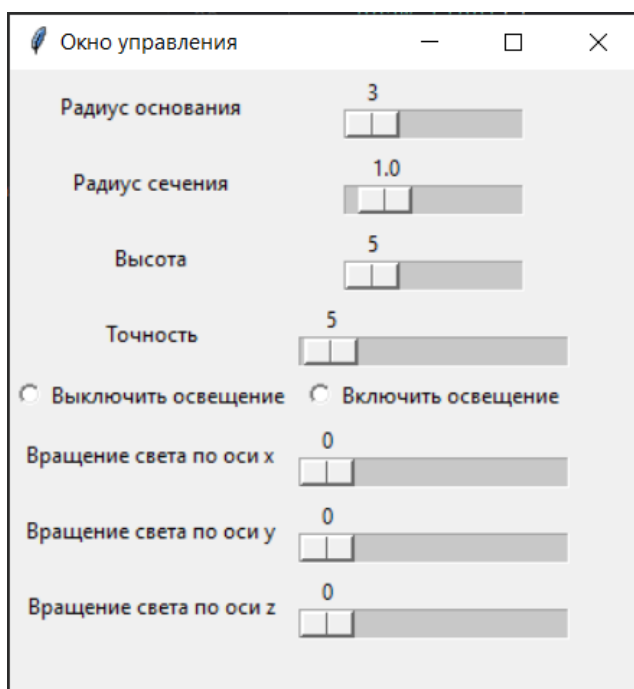
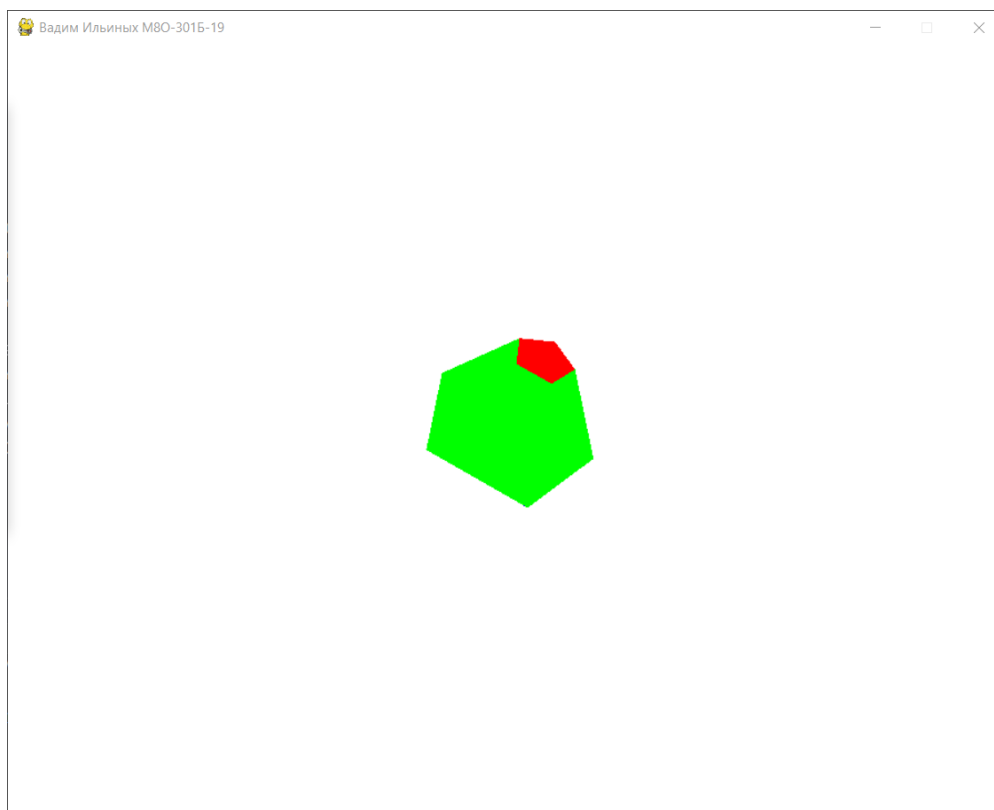
Первая содержит отрисовку тела и света. Во второй строится дополнительное окно для управления параметрами тела и освещения.

Основные методы:

- draw_cone() - построение цилиндра
- glBufferData() - загружает в буфер массив вершин
- glDrawArrays() - строит массив вершин, загруженный из видеокарты

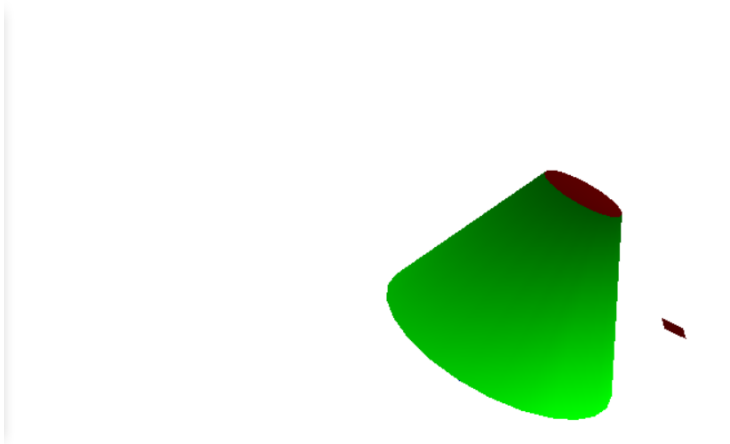
3. Результаты выполнения программы

- рисунок 1



- рисунок 2

Вадим Ильиных М80-3015-19



Окно управления

Радиус основания	4
Радиус сечения	1.3
Высота	8
Точность	25
<input type="radio"/> Выключить освещение	<input checked="" type="radio"/> Включить освещение
Вращение света по оси x	360
Вращение света по оси y	0
Вращение света по оси z	262

4. Листинг программы

main.cpp

```
# Ильиных В.М. М80-301Б-19
# Аппроксимировать заданное тело выпуклым многогранником.
# Точность аппроксимации задается пользователем.
# Обеспечить возможность вращения и масштабирования многогранника
# удаление невидимых линий и поверхностей.
# Реализовать простую модель закраски для случая одного источника
# света.
# Параметры освещения и отражающие свойства материала задаются
# пользователем в диалоговом режиме.
# Вариант 11: Прямой усеченный конус

import pygame
from pygame.locals import *
from OpenGL.GL import *
from OpenGL.GLU import *
import math
import numpy as np
import threading
from ctypes import *
from widgets import *

stop = False
k = 1
vertex = np.array([-1, -1, 1, 1, -1, 1, 1, 1, -1, 1, 1],
    dtype='float32')
vertex_normal = np.array([0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1],
    dtype='float32')
vertex_normal_down = np.array([0, 0, -1, 0, 0, -1, 0, 0, -1, 0, 0,
    -1], dtype='float32')

# длина вектора
def len_vector(x, y):
    return math.sqrt(x*x + y*y)

def init_light():
    glEnable(GL_LIGHTING)
    glEnable(GL_LIGHT0)
    glEnable(GL_COLOR_MATERIAL)
    glColorMaterial(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE)

    glPushMatrix()

    alpha = int(return_rotating_x())
    betta = int(return_rotating_y())
    gamma = int(return_rotating_z())
    intens = 0.3
    intens_diff = 1.0

    glRotatef(alpha, 1, 0, 0)
    glRotatef(betta, 0, 1, 0)
    glRotatef(gamma, 0, 0, 1)

    glLightModelfv(GL_LIGHT_MODEL_AMBIENT, (intens, intens, intens,
1))
    glLightfv(GL_LIGHT0, GL_DIFFUSE, (intens_diff, intens_diff,
intens_diff, 1))
    glLightfv(GL_LIGHT0, GL_POSITION, (0.8 * k, 0.3 * k, 0.6 * k, 1))
```

```

    glTranslatef(0.8 * k, 0.3 * k, 0.6 * k)

    glScalef(0.3, 0.3, 0.3)
    glColor3f(1, 0, 0)
    draw_light()
    glPopMatrix()

def draw_light():
    vbo = glGenBuffers(1)
    glBindBuffer(GL_ARRAY_BUFFER, vbo)
    glBufferData(GL_ARRAY_BUFFER, len(vertex)*4,
(c_float*len(vertex))(*vertex), GL_STATIC_DRAW)
    glBindBuffer(GL_ARRAY_BUFFER, 0)

    glEnableClientState(GL_VERTEX_ARRAY)
    glBindBuffer(GL_ARRAY_BUFFER, vbo)
    glVertexPointer(3, GL_FLOAT, 0, None)
    glDrawArrays(GL_TRIANGLE_FAN, 0, 4)
    glDisableClientState(GL_VERTEX_ARRAY)

def init_shaders():
    global uniforms
    global locations
    global ambient
    global diffuse

    vert = create_shader(GL_VERTEX_SHADER, """
        varying vec4 vertex_color;
        void main(){
            gl_Position = gl_ModelViewProjectionMatrix *
gl_Vertex;
            vertex_color = gl_Color * vec4(0.7, 0.7, 0.7, 1);
        }""")
    fragment = create_shader(GL_FRAGMENT_SHADER, """
        varying vec4 vertex_color;
        void main() {
            gl_FragColor = vertex_color;
        }""")

    program = glCreateProgram()
    glAttachShader(program, vert)
    glAttachShader(program, fragment)
    glLinkProgram(program)
    glUseProgram(program)

def draw_cone():
    s_axis = float(return_small_axis())
    b_axis = float(return_big_axis())
    h = float(return_height())
    h_slide = float(return_height_slide())
    acc = float(return_accuracy())
    global k
    global h_tmp
    k = max(s_axis, b_axis, h)
    z = h_slide
    circle_pts = []
    circle_pts_tmp = []
    circle_pts_mixed = []
    circle_pts_normal = []
    normal = []

```

```

# init_shaders()

for i in range(int(acc) + 1):
    angle = 2 * math.pi * (i / acc)
    x_bottom = b_axis * math.cos(angle)
    y_bottom = b_axis * math.sin(angle)

    x_top = s_axis * math.cos(angle)
    y_top = s_axis * math.sin(angle)

    if abs(x_bottom) < 10 ** (-10):
        x_bottom = 0
    if abs(y_bottom) < 10 ** (-10):
        y_bottom = 0
    pt = (x_bottom, y_bottom, z)
    circle_pts_normal.append(pt)
    circle_pts.append(x_top), circle_pts.append(y_top),
circle_pts.append(z)
    circle_pts_tmp.append(x_bottom),
circle_pts_tmp.append(y_bottom), circle_pts_tmp.append(0)
    circle_pts_mixed.append(x_top),
circle_pts_mixed.append(y_top), circle_pts_mixed.append(z)
    circle_pts_mixed.append(x_bottom),
circle_pts_mixed.append(y_bottom), circle_pts_mixed.append(0)

i = 0
for (x, y, z) in circle_pts_normal:
    x1 = (circle_pts_normal[i][0] + circle_pts_normal[(i + 1) %
int(acc)][0]) / 2.0
    y1 = (circle_pts_normal[i][1] + circle_pts_normal[(i + 1) %
int(acc)][1]) / 2.0
    length = len_vector(x1, y1)
    normal.append(x1 / length), normal.append(y1 / length),
normal.append(0)
    i += 1

vbo1 = glGenBuffers(1)
glBindBuffer(GL_ARRAY_BUFFER, vbo1)
glBufferData(GL_ARRAY_BUFFER, len(circle_pts)*4,
(c_float*len(circle_pts))(*circle_pts), GL_STATIC_DRAW)
glBindBuffer(GL_ARRAY_BUFFER, 0)

vbo2 = glGenBuffers(1)
glBindBuffer(GL_ARRAY_BUFFER, vbo2)
glBufferData(GL_ARRAY_BUFFER, len(circle_pts_tmp)*4,
(c_float*len(circle_pts_tmp))(*circle_pts_tmp), GL_STATIC_DRAW)
glBindBuffer(GL_ARRAY_BUFFER, 0)

vbo3 = glGenBuffers(1)
glBindBuffer(GL_ARRAY_BUFFER, vbo3)
glBufferData(GL_ARRAY_BUFFER, len(circle_pts_mixed)*4,
(c_float*len(circle_pts_mixed))(*circle_pts_mixed), GL_STATIC_DRAW)
glBindBuffer(GL_ARRAY_BUFFER, 0)

glEnableClientState(GL_VERTEX_ARRAY)

glBindBuffer(GL_ARRAY_BUFFER, vbo1)
glVertexPointer(3, GL_FLOAT, 0, None)
glNormalPointer(GL_FLOAT, 0, vertex_normal)
glColor(1, 0, 0)
glDrawArrays(GL_TRIANGLE_FAN, 0, int(acc+1))

glBindBuffer(GL_ARRAY_BUFFER, vbo2)
glVertexPointer(3, GL_FLOAT, 0, None)

```

```

glNormalPointer(GL_FLOAT, 0, vertex_normal_down)
glColor(0, 0, 1)
glDrawArrays(GL_TRIANGLE_FAN, 0, int(acc+1))

glBindBuffer(GL_ARRAY_BUFFER, vbo3)
glVertexPointer(3, GL_FLOAT, 0, None)
glNormalPointer(GL_FLOAT, 0, normal)
glColor(0, 1, 0)
glDrawArrays(GL_TRIANGLE_STRIP, 0, int((acc+1)*2))

glDisableClientState(GL_VERTEX_ARRAY)

def create_shader(shader_type, source):
    shader = glCreateShader(shader_type)
    glShaderSource(shader, source)
    glCompileShader(shader)
    return shader

t1 = threading.Thread(target=make_window)
t1.start()

pygame.init()
(width, height) = (900, 700)
screen = pygame.display.set_mode((width, height), OPENGL | DOUBLEBUF)
pygame.display.set_caption('Вадим Ильиных М80-301Б-19')
gluPerspective(45, (width / height), 0.1, 50.0)
glTranslatef(0.0, 0.0, -30)
glEnable(GL_DEPTH_TEST)

glLineWidth(2)
clock = pygame.time.Clock()

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            quit()
        if event.type == pygame.MOUSEMOTION:
            pressed = pygame.mouse.get_pressed(3)
            if pressed[0]:
                glRotatef(2, event.rel[1], event.rel[0], 0)
        if event.type == pygame.MOUSEBUTTONDOWN:
            if event.button == 4:
                glScalef(1.1, 1.1, 1.1)
            elif event.button == 5:
                glScalef(0.9, 0.9, 0.9)

    key = pygame.key.get_pressed()

    if key[pygame.K_LEFT]:
        glRotatef(1, 0, -1, 0)
    if key[pygame.K_RIGHT]:
        glRotatef(1, 0, 1, 0)
    if key[pygame.K_UP]:
        glRotatef(1, -1, 0, 0)
    if key[pygame.K_DOWN]:
        glRotatef(1, 1, 0, 0)
    if key[pygame.K_q]:
        glRotatef(1, 0, 0, -1)
    if key[pygame.K_e]:
        glRotatef(1, 0, 0, 1)
    if key[pygame.K_KP_PLUS] or key[pygame.K_PLUS]:

```



```

        glScalef(1.1, 1.1, 1.1)
    if key[pygame.K_MINUS] or key[pygame.K_KP_MINUS]:
        glScalef(0.9, 0.9, 0.9)
    if key[pygame.K_r]:
        glLoadIdentity()
        gluPerspective(45, (width / height), 0.1, 50.0)
        glTranslatef(0.0, 0.0, -40)
        glLineWidth(2)

    glClearColor(1, 1, 1, 1)
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)

    light_change = int(return_light_change())
    if light_change:
        init_light()

    draw_cone()

    glDisable(GL_LIGHT0)
    glDisable(GL_LIGHTING)
    glDisable(GL_COLOR_MATERIAL)

    pygame.display.flip()
    clock.tick(60)

```

widgets.py

```

import tkinter as tk
from tkinter import *

radius = 3
height = 5
height_slide = 5
accuracy = 3
small_axis = 0
big_axis = 3
light_change = 0
rotating_x = 0
rotating_y = 0
rotating_z = 0

def return_height():
    return height

def return_height_slide():
    return height_slide

def return_big_axis():
    return big_axis

def return_small_axis():
    return small_axis

def return_accuracy():
    return accuracy

```

```

def return_light_change():
    return light_change

def return_rotating_x():
    return rotating_x

def return_rotating_y():
    return rotating_y

def return_rotating_z():
    return rotating_z

def change_big_axis(value):
    global big_axis
    big_axis = value

def change_small_axis(value):
    global small_axis
    global height
    global height_slide
    b_axis = return_big_axis()
    coef = float(value) / float(b_axis)
    if height_slide > 1 or small_axis > value:
        height_slide = height - (height * coef)
        small_axis = value

def change_height(value):
    global height
    global height_slide
    global big_axis
    global small_axis
    coef = float(small_axis) / float(big_axis)
    height = float(value)
    if height_slide > 1 or small_axis > value:
        height_slide = height - (height * coef)

def change_accuracy(value):
    global accuracy
    accuracy = value

def change_rotating_x(value):
    global rotating_x
    rotating_x = value

def change_rotating_y(value):
    global rotating_y
    rotating_y = value

def change_rotating_z(value):
    global rotating_z
    rotating_z = value

def light_on():

```

```

global light_change
light_change = 1

def light_off():
    global light_change
    light_change = 0

def make_window():
    window = tk.Tk()
    window.title("Окно управления")
    window.geometry('350x350')

    var1 = IntVar()

    rad1 = Radiobutton(window, text="Выключить освещение", value=1,
variable=var1, command=light_off)
    rad2 = Radiobutton(window, text="Включить освещение", value=2,
variable=var1, command=light_on)
    rad1.grid(row=4, column=0)
    rad2.grid(row=4, column=1)

    scale_big_axis = tk.Scale(window, from_=3, to=15,
orient=HORIZONTAL, command=change_big_axis)
    label_big_axis = tk.Label(window, text="Радиус основания")
    label_big_axis.grid(row=0, column=0)
    scale_big_axis.grid(row=0, column=1)

    scale_small_axis = tk.Scale(window, from_=0, to=9.9,
resolution=0.1, orient=HORIZONTAL, command=change_small_axis)
    label_small_axis = tk.Label(window, text="Радиус сечения")
    label_small_axis.grid(row=1, column=0)
    scale_small_axis.grid(row=1, column=1)

    scale_height = tk.Scale(window, from_=5, to=20, orient=HORIZONTAL,
command=change_height)
    label_height = tk.Label(window, text="Высота")
    label_height.grid(row=2, column=0)
    scale_height.grid(row=2, column=1)

    scale_accuracy = tk.Scale(window, from_=3, to=100,
orient=HORIZONTAL, length=150, command=change_accuracy)
    label_accuracy = tk.Label(window, text="Точность")
    label_accuracy.grid(row=3, column=0)
    scale_accuracy.grid(row=3, column=1)

    scale_rotating_x = tk.Scale(window, from_=0, to=360,
orient=HORIZONTAL, length=150, command=change_rotating_x)
    label_rotating_x = tk.Label(window, text="Вращение света по оси
x")
    label_rotating_x.grid(row=5, column=0)
    scale_rotating_x.grid(row=5, column=1)

    scale_rotating_y = tk.Scale(window, from_=0, to=360,
orient=HORIZONTAL, length=150, command=change_rotating_y)
    label_rotating_y = tk.Label(window, text="Вращение света по оси
y")
    label_rotating_y.grid(row=6, column=0)
    scale_rotating_y.grid(row=6, column=1)

    scale_rotating_z = tk.Scale(window, from_=0, to=360,
orient=HORIZONTAL, length=150, command=change_rotating_z)

```

```
label_rotating_z = tk.Label(window, text="Вращение света по оси  
z")  
label_rotating_z.grid(row=7, column=0)  
scale_rotating_z.grid(row=7, column=1)  
  
tk.mainloop()
```

5. Вывод

В OpenGL освещения как такового нет: при включении режима освещения, следует добавить нормали к рисуемым поверхностям, после чего задать положение источника света, и в зависимости его отклонения от нормали, к цвету поверхности добавляется или удаляется белая составляющая. Буфер вершин используется для оптимизации работы: вершины прогружаются не в оперативную память, а сразу в видеопамять.

ЛИТЕРАТУРА

1. Справочник по PyOpenGL и Pygame [Электронный ресурс]. URL: [Введение в OpenGL и PyOpenGL. Часть I: создание вращающегося куба](#) (дата обращения: 05.11.2021).
2. Справочник по Python [Электронный ресурс]. URL: <https://jenyay.net/Matplotlib/Widgets> (дата обращения: 05.11.2021).
3. Справочник по OpenGL [Электронный ресурс]. URL: <https://youtu.be/sRpXMnaOAcU> (дата обращения: 05.11.2021)