**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Компьютерная графика»

**Лабораторная работа № 4-5**

Тема: Ознакомление с технологией OpenGL.

Студент: Чекушкин Д.И.

Группа: 80-304

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2018

1. Постановка задачи

Задание: Создать графическое приложение с использованием OpenGL. Используя результаты Л.Р.No3, изобразить заданное тело (то же, что и в л.р. No3) 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 задаются пользователем в диалоговом режиме.

12. Прямой круговой цилиндр.

1. Решения задачи

ЯП: Python

ОС: Ubuntu 16.04

Библиотеки: matplotlib.pyplot, numpy, Poly3DCollection, Line3DCollection, matplotlib.colors, mpl\_toolkits.mplot3d, pygame, OpenGL.GL, sys, random

matplotlib.pyplot.figure(*figsize*) - Возвращает matplotlib.figure.Figure

matplotlib.pyplot.add\_subplot(numrows, numcols, fignum) - Добавляет объект для рисования графика по указанным координатам

matplotlib.pyplot.plot(\*args, \*\*kwargs) - Создает график

add\_collection\* - добавить объект типа Collection (объединение большого количества линий или частей многоугольника) на оси (в нее передаются плоскости)

ax.plot\_surface - добавить поверхность на оси

glTranslate - сдвиг в экран

glRotate - вращения объекта вдоль оси

glBlendFunc - смешивания цветов

glVertex3fv - отрисовка векторов

glColor4f(x,y,z,u) - назначение цвета и прозрачности

tksd.askfloat - запуск диалогового окна

glEnable(GL\_LIGHTING) - включить освещение

glEnable(GL\_LIGHT0) - включить 1-ый источник света

glLightfv(GL\_LIGHT0, GL\_POSITION, lightpos) - положение источника света

Ход работы:

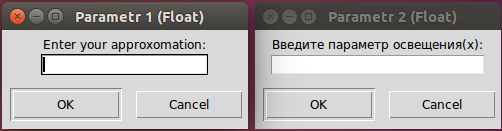
* Вычисляем количество граней, нужных для заданной аппроксимации.
* Вычисляем координаты точек в основании пирамиды.
* Строим многоугольник в основании.
* Строим ребра
* Закрашиванием ребра и основание
* Строим цилиндр
* С помощью функционала OpenGL строим цилиндр
* Добавляем источник света
* Добавляем вращение с использованием стрелочек клавиатуры
* Отрисовываем

При тестировании были выявлены и исправлены незначительные ошибки, связанные с отрисовкой вершин пирамиды.

1. Руководство по использованию программы

Файл 45.py содержит код программы .

На вход подается точность аппроксимации 0<х<1 и положение источника света, на выход – график искомого многогранника и поверхность, освещенную одиночным источником света.



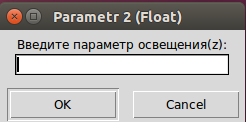
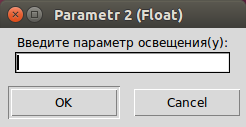


Рис 1 - Диалоговые окна

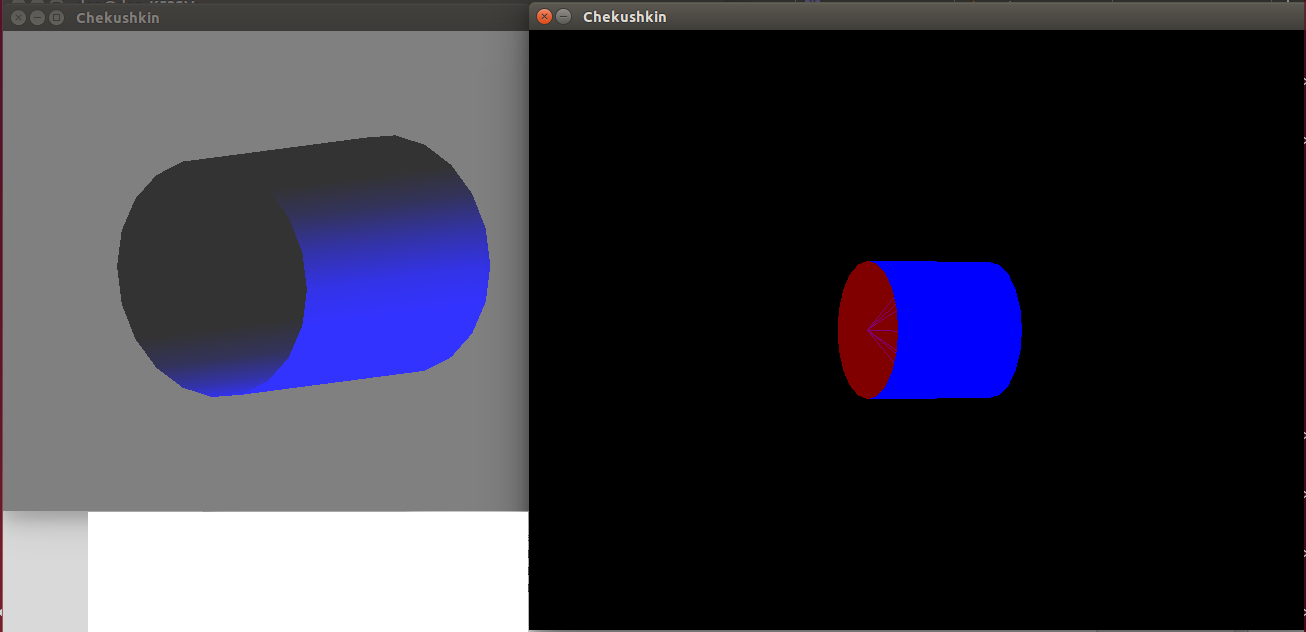


Рис 2 - Вывод программы

1. Листинг программы

#coding=utf-8

import pygame

from pygame.locals import \*

from OpenGL.GL import \*

from OpenGL.GLU import \*

from OpenGL.GLUT import \*

import math

import sys

import random

try:

# for Python2

import Tkinter as tk

import tkSimpleDialog as tksd

except:

# for Python3

import tkinter as tk

import tkinter.simpledialog as tksd

#def show\_message():

# tkMessageBox.showinfo("Approximation", message.get())

#root = Tk()

#root.title("Approximation")

#root.geometry("300x250")

#message = StringVar()

#message\_entry = Entry(textvariable=message)

#val = message\_entry.get()

#int(val)

#message\_entry.place(relx=.5, rely=.1, anchor="c")

#message\_button = Button(text="Send", command=show\_message)

#message\_button.place(relx=.5, rely=.5, anchor="c")

#root.mainloop()

global d

root = tk.Tk()

ap = tksd.askfloat("Parametr 1 (Float)", "Enter your approxomation:",

parent=root, minvalue=0.0001)

d=ap

print(ap)

#print("Введите прочность аппроксимации eps(0<eps<1):")

#eps=float(input()) #Точность аппроксимации

eps=d

n=3 #Мин количество сторон

r=1 #Радиус круга в основании

alpha = 2\*math.pi/n;

length\_of\_side = r \* math.sqrt(2-2\*math.cos(alpha))

height = length\_of\_side/2/math.tan(alpha/2)

while r - height > eps: #Аппроксимация

n += 1

alpha = 2\*math.pi/n

length\_of\_side = r \* math.sqrt(2-2\*math.cos(alpha))

height = length\_of\_side/2/math.tan(alpha/2)

#n=float(input())

g=360/(n)

print("g=",g)

lx=[] #Список координат многогранника в основании(Ось Х)

ly=[] #Список координат многогранника в основании(Ось У)

ga=g

while (ga <= 360): #Расчет координат

x=math.cos(math.radians(ga))

y=math.sin(math.radians(ga))

lx.append(x)

ly.append(y)

ga=ga+g

#print("x=",x)

#print("y=",y)

#print("ga=",ga)

if (ga>360): break

lx.append(lx[0])

ly.append(ly[0])

#print(lx)

#print(ly)

verticies=() ## Кортеж - буфер вершин

edges = ()## Кортеж - буфер сторон

verticies=verticies + ((0,0,1),)##

v=[]

z=0

while z<(len(lx)-1):#Список ребер = вершина + одна из точек основания

v.append([ [lx[z],ly[z],0], [0,0,1]])

verticies=verticies +((lx[z],ly[z],0),)## Кортеж - все вершины

z=z+1

edges = edges + ((0,z),)## Кортеж - боковые ребра

if (z>1):

edges = edges + ((z-1,z),)## Кортеж - нижние ребра

print('\_\_\_')

print(z)

print('\_\_\_')

edges = edges + ((z,1),)

print(verticies)

print(edges)

v2=[]

j=0

while j < (len(lx)-1):#Нижние ребра

v2.append([[lx[j],ly[j],0],[lx[j+1],ly[j+1],0]])

j=j+1

v.append(v2)

def draw\_cylinder(radius, height, num\_slices):

r = radius

h = height

n = float(num\_slices)

circle\_pts = []

for i in range(int(n) + 1):

angle = 2 \* math.pi \* (i/n)

x = r \* math.cos(angle)

y = r \* math.sin(angle)

pt = (x, y)

circle\_pts.append(pt)

glBegin(GL\_TRIANGLE\_FAN)#Нижнее основание цилиндра

#glColor(1, 0, 0)

glColor4f(1.0, 0.0, 0.0, 0.5);

glVertex(0, 0, h/2.0)

for (x, y) in circle\_pts:

z = h/2.0

glVertex(x, y, z)

glEnd()

glBegin(GL\_TRIANGLE\_FAN)#Верхнеее основание цилиндра

#glColor(0, 0, 1)

glColor4f(0.0, 0.0, 1.0, 0.5);

glVertex(0, 0, h/2.0)

for (x, y) in circle\_pts:

z = -h/2.0

glVertex(x, y, z)

glEnd()

glBegin(GL\_TRIANGLE\_STRIP)#Сам цилиндр

#glColor(0, 1, 0)

glColor4f(0.0, 0.0, 1.0, 1);

for (x, y) in circle\_pts:

z = h/2.0

glVertex(x, y, z)

glVertex(x, y, -z)

glEnd()

def Pyramid():

glBegin(GL\_LINES)

for edge in edges:

for vertex in edge:

glVertex3fv(verticies[vertex])

glEnd()

pygame.init()

pygame.display.set\_caption('Chekushkin')

(width, height) = (800, 600)

screen = pygame.display.set\_mode((width, height), OPENGL | DOUBLEBUF)

clock = pygame.time.Clock()

rotation = 0.0

while True:

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

sys.exit()

rotation += 1.0

glClear(GL\_COLOR\_BUFFER\_BIT)

glClear(GL\_DEPTH\_BUFFER\_BIT)

glEnable(GL\_DEPTH\_TEST)

glMatrixMode(GL\_PROJECTION)

glLoadIdentity()

gluPerspective(10, float(width)/height, 1, 1000)

glMatrixMode(GL\_MODELVIEW)

glLoadIdentity() #считывает текущую матрицу

glTranslate(0, 0, -50)#сдвиг в экран на -50

glRotate(rotation, 0, 1, 0)#вращения объекта вдоль оси

glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA); #смешивания цветов

glEnable(GL\_BLEND);

Pyramid()

draw\_cylinder(1, 2, 20)

pygame.display.flip()

clock.tick(60)

glDisable(GL\_BLEND);

#2-2-0

global q

global w

global e

#Параметры освещения и отражающие свойства материала задаются пользователем в диалоговом режиме.

root = tk.Tk()

q = tksd.askfloat("Dialog (Float)", "Введите параметр освещения(x):",

parent=root, minvalue=0)

w = tksd.askfloat("Dialog (Float)", "Введите параметр освещения(y):",

parent=root, minvalue=0)

e = tksd.askfloat("Dialog (Float)", "Введите параметр освещения(z):",

parent=root, minvalue=0)

a=q

b=w

c=e

global xrot # Величина вращения по оси x

global yrot # Величина вращения по оси y

global ambient # рассеянное освещение

global cylcolor # Цвет цилиндра

global lightpos # Положение источника освещения

def init():

global xrot

global yrot

global ambient

global cylcolor

global lightpos

xrot = 0.0

yrot = 0.0

ambient = (1.0, 1.0, 1.0, 1)

cylcolor = (0, 0, 1, 0.8)

lightpos = (a, b, c)

glClearColor(0.5, 0.5, 0.5, 1.0) # Серый цвет для первоначальной закраски

gluOrtho2D(-1.0, 1.0, -1.0, 1.0) # Определяем границы рисования по горизонтали и вертикали

glRotatef(-90, 1.0, 0.0, 0.0) # Сместимся по оси Х на 90 градусов

glLightModelfv(GL\_LIGHT\_MODEL\_AMBIENT, ambient) # Определяем модель освещения

glEnable(GL\_LIGHTING) # Включаем освещение

glEnable(GL\_LIGHT0) # Включаем один источник света

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

# Процедура обработки стрелочек

def specialkeys(key, x, y):

global xrot

global yrot

if key == GLUT\_KEY\_UP: # Клавиша вверх

xrot -= 2.0 # Уменьшаем угол вращения по оси Х

if key == GLUT\_KEY\_DOWN: # Клавиша вниз

xrot += 2.0 # Увеличиваем угол вращения по оси Х

if key == GLUT\_KEY\_LEFT: # Клавиша влево

yrot -= 2.0 # Уменьшаем угол вращения по оси Y

if key == GLUT\_KEY\_RIGHT: # Клавиша вправо

yrot += 2.0 # Увеличиваем угол вращения по оси Y

glutPostRedisplay() # перерисовка

# Процедура перерисовки

def draw():

global xrot

global yrot

global lightpos

global greencolor

global cylcolor

glClear(GL\_COLOR\_BUFFER\_BIT) # Очищаем экран и заливаем серым цветом

glPushMatrix() # Сохраняем текущее положение "камеры"

glRotatef(xrot, 1.0, 0.0, 0.0) # Вращаем по оси X на величину xrot

glRotatef(yrot, 0.0, 1.0, 0.0) # Вращаем по оси Y на величину yrot

glLightfv(GL\_LIGHT0, GL\_POSITION, lightpos) # Источник света вращаем вместе с елкой

# Рисуем цилиндр

# Устанавливаем материал: рисовать с 2 сторон, рассеянное освещение, синий цвет

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, cylcolor)

glTranslatef(0.0, 0.0, -0.7) # Сдвинемся по оси Z на -0.7

# Рисуем цилиндр с радиусом 0.1, высотой 0.2

# Последние два числа определяют количество полигонов

glutSolidCylinder(0.5, 1, 20, 20)

glPopMatrix() # Возвращаем сохраненное положение "камеры"

glutSwapBuffers() # Выводим все нарисованное в памяти на экран

# Здесь начинается выполнение программы

# Использовать двойную буферизацию и цвета в формате RGB (Красный, Зеленый, Синий)

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB)

# Указываем начальный размер окна (ширина, высота)

glutInitWindowSize(300, 300)

# Указываем начальное положение окна относительно левого верхнего угла экрана

glutInitWindowPosition(50, 50)

# Инициализация OpenGl

glutInit(sys.argv)

glutCreateWindow(b"Chekushkin")

# Определяем процедуру, отвечающую за перерисовку

glutDisplayFunc(draw)

# Определяем процедуру, отвечающую за обработку клавиш

glutSpecialFunc(specialkeys)

# Вызываем нашу функцию инициализации

init()

# Запускаем основной цикл

glutMainLoop()

Выводы: благодаря проделанной работе мне удалось освоить 3д-графику в Python, методы закраски и отрисовки плоскостей с помощью технологий OpenGL

Список литературы:

1. Самоучитель по графике [Электронный ресурс] Url: https://pythonworld.ru/novosti-mira-python/scientific-graphics-in-python.html
2. OpenGL manual [Электронный ресурс] Url: http://pyopengl.sourceforge.net/documentation/manual-3.0/
3. Методическое пособие по графике в OpenGL [Электронный ресурс] Url: https://rsdn.org/article/opengl/ogltut2.xml