

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
"МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)"

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Лабораторная работа № 6 по курсу
«Компьютерная графика»

Группа: М8о-307Б-18

Студент:

Тояков Артем Олегович

Преподаватель:

Филиппов Глеб Сергеевич

Оценка:

Дата:

Москва, 2021

Оглавление

1.Постановка задачи.....	3
2.Структура программы.....	3
3.Описание программы.....	3
4.Листинг программы.....	4
5.Результат работы.....	11
6.Вывод.....	11

1. Постановка задачи

Реализовать анимационный эффект: вращение относительно оси OZ. Скорость вращения по синусоидному уравнению.

2. Структура программы

1. main.py

3. Описание программы

Использовал язык python: PyOpenGL. Строил двуполостный гиперболоид тело по параметрическим координат, огранивая по оси Z. Таким образом получался слой полости двуполостного гиперболоида.

Параметрическое представление:

$$x = a * \operatorname{sh}(u) * \cos(v)$$

$$y = b * \operatorname{sh}(u) * \sin(v)$$

$$z = \pm c * \operatorname{ch}(v)$$

$$u : (0, 2\pi), v : \mathbb{R}$$

Вращение реализовано таким образом: нажимая горячие клавиши, происходит изменение координат главной оси. Аппроксимация реализована добавлением дополнительных точек, благодаря которым построение становится более четкое и приближенное к необходимой фигуре. Построение происходит по точкам, однако в основе примитива берется многоугольник многоугольник. В качестве света простое зеркальное освещение с зеркальным бликом. Также освещение идет от одной точки. Анимация добавлена таким образом, что изменяется значения осей OX, OY. Значения следующей позиции высчитываются из синусоидного уравнения: таким образом скорость вращения зависит от синусоидного уравнения.

Уравнение: `speed.append((f + s * np.sin(third * u[i] + thurs)) * 0.01)`

`f, s, third, thurs` — константы.

`u = np.linspace(0, 2 * np.pi, int(amountInList))`

4.Листинг программы

```
from OpenGL.GL import *
from OpenGL.GLU import *
from OpenGL.GLUT import *

import numpy as np

import sys

import threading

import time

from itertools import cycle


#ccordinates

xrot = 0

yrot = 0

zrot = 4

#constant

amountInList = 100

f = 4

s = 3

third = 2

thurs = 6

c = 1.5

a = 1.2

z_h = 10

z_l = 5
```

```
intensiv = 10
```

```
reflection = 116
```

```
light_coord = (20, 30, 30)
```

```
size1 = 4
```

```
appr = 40
```

```
def drawBox():
```

```
    global xrot, yrot, reflection, z_h, z_l, appr, a, c
```

```
    glPushMatrix()
```

```
    glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, (0.2, 0.8, 0.0, 0.8))
```

```
    glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, (0.2, 0.8, 0.0, 0.8))
```

```
    glMaterialf(GL_FRONT_AND_BACK, GL_SHININESS, 128 - reflection)
```

```
    draw(z_h, z_l, appr, a, c)
```

```
    glPopMatrix()
```

```
    glutSwapBuffers()
```

```
def init():
```

```
    ambient = (1.0, 1.0, 1.0, 6)
```

```
    lightpos = (1.0, 6.0, 7.0)
```

```
    glClearColor(255, 255, 255, 1.0)
```

```
    glClearDepth(1.0)
```

```
    glEnable(GL_DEPTH_TEST)
```

```
    glShadeModel(GL_FLAT)
```

```

glDepthFunc(GL_LEQUAL)

glEnable(GL_DEPTH_TEST)

glEnable(GL_NORMALIZE)

glHint(GL_POLYGON_SMOOTH_HINT, GL_NICEST)

glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST)

glEnable(GL_LIGHTING)

glLightModelf(GL_LIGHT_MODEL_TWO_SIDE, GL_TRUE)

glEnable(GL_NORMALIZE)

glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambient) # Определяем текущую модель
освещения

glEnable(GL_LIGHTING)                # Включаем освещение

glEnable(GL_LIGHT0)                  # Включаем один источник света

glLightfv(GL_LIGHT0, GL_POSITION, lightpos)

```

```

def draw(z_h, z_l, appr, a, c):

    u = np.linspace(0, 2 * np.pi, int(appr*100))

    v = np.linspace(-1000, 1000, int(appr*100))

    verts = []

    vertsTwo = []

    for i in range(appr*100 - 1):

        temp = c * np.cosh(u[i])

        #print(temp)

        if((z_h > temp) and (temp > z_l)):

            verts.append(((a * np.sinh(u[i]) * np.cos(v[i])), (a * np.sinh(u[i]) * np.sin(v[i])), (c *
np.cosh(u[i])))))

```

```

    #if((-z_h < -temp) and (-temp < -z_l)):

        #vertsTwo.append(((a * np.sinh(u[i]) * np.cos(v[i])),(a * np.sinh(u[i]) * np.sin(v[i])),(-c *
np.cosh(u[i]))))

    glBegin(GL_POLYGON)

    for v in verts:

        glVertex3fv(v)

    #for v in vertsTwo:

        #glVertex3fv(v)

    glEnd()


def reshape(width, height):

    glViewport(0, 0, width, height)

    glMatrixMode(GL_PROJECTION)

    glLoadIdentity()

    gluPerspective(60.0, float(width)/float(height), 1.0, 60.0)

    glMatrixMode(GL_MODELVIEW)

    glLoadIdentity()

    gluLookAt(0.0, 0.0, 0.0, 1.0, -1.0, 1.0, 10.0, 1, 0.0)


def display():

    global size1

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)

    glMatrixMode(GL_MODELVIEW)

    glLoadIdentity()

    gluLookAt(-20, -20, -15, 10, 10, 20, 0, 0, 10)

```

```
glTranslatef(size1, size1, size1)
```

```
glRotatef(xrot, 1, 0, 0)
```

```
glRotatef(yrot, 0, 0, 1)
```

```
glRotatef(zrot, 0, 1, 0)
```

```
drawBox()
```

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

```
    global xrot, yrot, zrot, size1
```

```
    if key == b'w':
```

```
        glRotate(+1.0, 1, 0, 0)
```

```
    elif key == b's':
```

```
        xrot -= 2
```

```
    elif key == b'a':
```

```
        yrot += 2
```

```
    elif key == b'd':
```

```
        yrot -= 2
```

```
    elif key == b'q':
```

```
        zrot += 2
```

```
    elif key == b'e':
```

```
        zrot -= 2
```

```
    elif key == b'=':
```

```
        size1 += 1
```

```
    elif key == b'-':
```

```
        size1 -= 1
```

```
    elif key == b'c':
```



```

    app_change(appr + 1)
elif key == b'v':
    app_change(appr - 1)
elif key == b'p':
    exit(0)
glutPostRedisplay()

```

```

def rotate():
    global zrot, xrot, yrot, f, s, third, thurs, amountInList
    u = np.linspace(0, 2 * np.pi, int(amountInList))
    speed = []
    for i in range(amountInList):
        speed.append((f + s * np.sin(third * u[i] + thurs)) * 0.01)
    for val in cycle(speed):
        print(val)
        xrot += val
        yrot += val
        glutPostRedisplay()

```

```

def app_change(x):
    global appr
    appr = x
    glutPostRedisplay()
    return 0

```

```
def main():

    glutInit(sys.argv)

    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH)

    glutInitWindowSize(500, 500)

    glutInitWindowPosition(0, 0)

    glutCreateWindow("")

    glutDisplayFunc(display)

    glutReshapeFunc(reshape)

    glutKeyboardFunc(specialkeys)

    init()

    t = threading.Thread(target=rotate)

    t.daemon = True

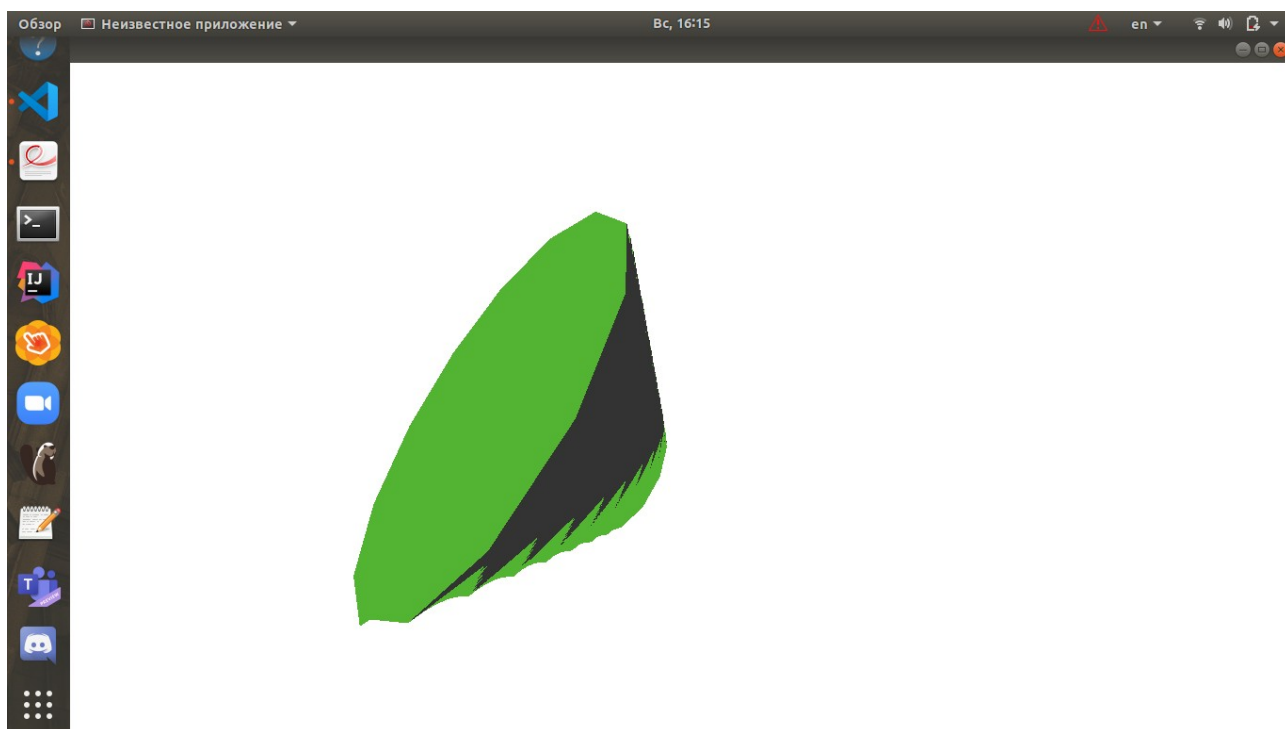
    t.start()

    glutMainLoop()


if __name__ == "__main__":

    main()
```

5.Результат работы



6.Вывод

Благодаря данной лабораторной работе я получил базовые навыки построения, используя технологии OpenGL. Познакомился с простейшими инструментами и функциями фреймворка. Также удалось создать автоматизированное изменение координат оси(вращение).