Московский авиационный институт

(Национальный исследовательский университет)

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Компьютерная графика»

Лабораторная работа № 2

Тема: Каркасная визуализация выпуклого многогранника. Удаление невидимых линий

Студент: Ильиных Вадим Максимович

Группа: 80-301

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2021

1. Постановка задачи

Разработать формат представления многогранника и процедуру его каркасной отрисовки в ортографической и изометрической проекциях. Обеспечить удаление невидимых линий и возможность пространственных поворотов и масштабирования многогранника. Обеспечить автоматическое центрирование и изменения размеров изображения при изменении размеров окна.

Вариант 7

4-гранная прямая правильная призма

2. Описание программы

Программа написана на *Python*. Многогранник построен при помощи библиотеки *PyOpenGl*, его вращение также обеспечивается ею. Пользовательское окно и отклик программы на нажатия клавиш созданы при помощи библиотеки *pygame.*

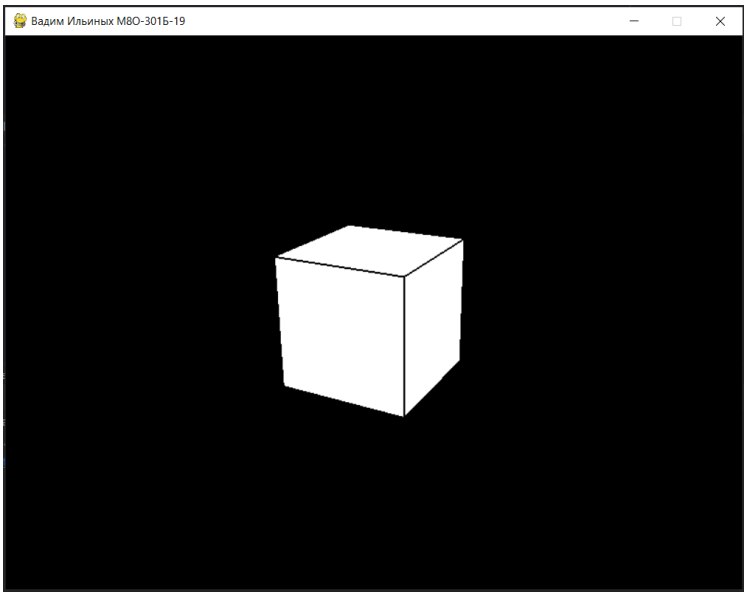
Программа состоит из двух файлов. Первый содержит координаты вершин, ребёр и поверхностей. Второй файл – основная программа.

Основные методы:

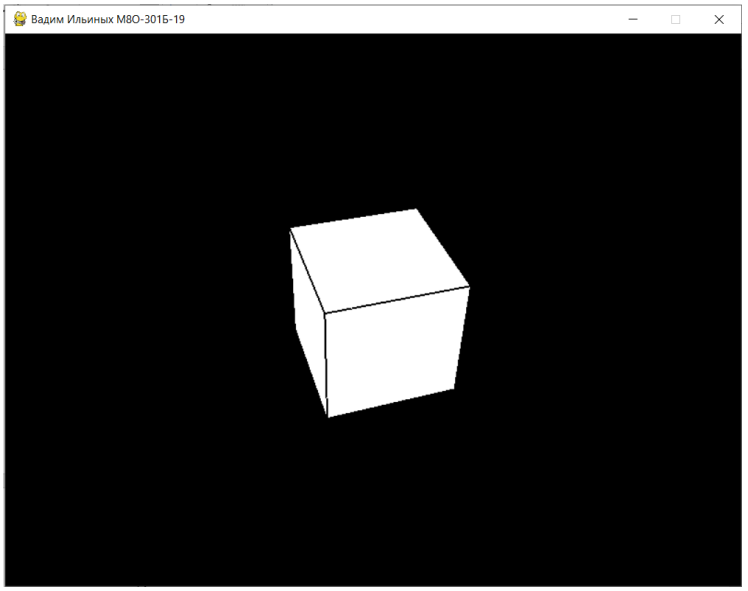
- cube(): отрисовка многогранника
- actions_with_cube(): создание окна и действия с многогранником

4. Результаты выполнения тестов

1)



2)



5. Листинг программы

coordinates.py

```
vertices = (
    (1, -1, -1),
    (1, 1, -1),
    (-1, 1, -1),
    (-1, -1, -1),
    (1, -1, 1),
    (1, 1, 1),
    (-1, -1, 1),
    (-1, 1, 1)
)

edges = (
    (0, 1),(0, 3),(0, 4),
    (2, 1),(2, 3),(2, 7),
    (6, 3),(6, 4),(6, 7),
    (5, 1),(5, 4),(5, 7)
)

surfaces = (
    (0, 1, 2, 3),
    (3, 2, 7, 6),
    (6, 7, 5, 4),
    (4, 5, 1, 0),
    (1, 5, 7, 2),
    (4, 0, 3, 6)
)
```

main.py

```
import pygame
from pygame.locals import *
```

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

def cube():
    glBegin(GL_QUADS)
    for surface in surfaces:
        for vertex in surface:
            glColor3fv((1, 1, 1))
            glVertex3fv(vertices[vertex])
    glEnd()

    glBegin(GL_LINES)
    for edge in edges:
        for vertex in edge:
            glColor3fv((0, 0, 0))
            glVertex3fv(vertices[vertex])
    glEnd()

def actions_with_cube():
    pygame.init()
    display = (800, 600)
    pygame.display.set_caption('Вадим Ильиных М8О-301Б-19')
    pygame.display.set_mode(display, DOUBLEBUF | OPENGL)
    glEnable(GL_DEPTH_TEST)
    gluPerspective(45, (display[0] / display[1]), 0.1, 50.0)
    glTranslatef(0.0, 0.0, -10)
    glLineWidth(2)

    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_r]:
            glLoadIdentity()
            gluPerspective(45, (display[0] / display[1]), 0.1, 50.0)
            glTranslatef(0.0, 0.0, -10)

        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
        cube()

        pygame.display.flip()
        pygame.time.wait(10)

if __name__ == "__main__":
    actions_with_cube()
```

6. Вывод

OpenGl позволяет моделировать многогранники без особых усилий, а с pygame можно взаимодействовать с построенными на базе OpenGl объектами так же легко.

ЛИТЕРАТУРА

1. Справочник по PyOpenGl и Pygame [Электронный ресурс]. URL: https://pythonist.ru/vvedenie-v-opengl-i-pyopengl-chast-i-sozdanie-vrashhayushhegosya-kuba/ (дата обращения: 3.10.2021).

2. Справочник по Python [Электронный ресурс]. URL: https://jenyay.net/Matplotlib/Widgets (дата обращения: 3.10.2021).