ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ _____

ПРЕПОДАВАТЕЛЬ

Доцент		А.В. Аграновский
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 3

ЗНАКОМСТВО С OPENGL

по курсу: КОМПЬЮТЕРНАЯ ГРАФИКА

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4128		В.А. Воробьев
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2023

СОДЕРЖАНИЕ

1 Цель работы

Изучение открытой графической библиотеки OpenGL; построить динамическую 3D-сцену на языке программирования высокого уровня, поддерживающего библиотеку OpenGL

Задание:

С использованием любого языка программирования высокого уровня (из числа изученных в ходе освоения ООП 09.03.02), поддерживающего библиотеку OpenGL, построить динамическую 3D сцену.

2 Теоретические сведения

OpenGL (Open Graphics Library) является кроссплатформенной спецификацией, определяющей набор обязательных возможностей при создании приложений, использующих двух- и трехмерную графику. В настоящий момент спецификация поддерживается на Mac OS, PlayStation 3, Windows и различных Unix-платформах. По-сути OpenGL описывает набор функций и их поведение независимо от языка программирования. Основной принцип работы OpenGL заключается в получении совокупности графических примитивов (точек, линий и т.п.) и последующую математическую обработку полученных данных с построением растровой картинки.

Преимущества OpenGL:

- повышение производительности в отдельных играх;
- улучшение работы видеокарты;
- наличие расширений;
- наличие дополнительных библиотек;
- независимость от языка программирования.

Недостатки OpenGL:

- Поскольку спецификация является низкоуровневым API – требуется изначально точно задать последовательность шагов;
- сложности в работе с новым железом;
- необходима установка и работа с DirectX.

3 Выполнение работы

Для выполнения лабораторной работы было решено выбрать высокоуровневый язык Python 3. Для взаимодействия с OpenGl было решено выбрать библиотеку pyOpenGl, а для упрощения отрисовки сцены и контроля ввода пользователя также подключим pygame.

В коде программы реализуем вывод на экран фигуры, а также её поворот со временем. Исходный код доступен в Приложении А, а также на GitHub(URL: https://github.com/vladcto/SUAI_homework/blob/20ba27fb305e4d51bc7289350eaac22b48ffb79d/4_semester/CG/3%D0%BB%D1%80/solution.py).

Исходный код снабжен комментариями и легок для понимания, тем не менее обговорим несколько наиболее важных моментов в контексте понимания работы с OpenGl.

На рисунке 1 представлен код, отвечающий за создание окна, в котором будет рисоваться модель и первоначальную настройку OpenGl. gluPerspective – устанавливает матрицу проекции перспективы, а если проще отвечает за угол обзора и соотношение сторон наблюдателя. Далее при помощи функции glTranslatef мы смещаем систему координат для того, чтобы модель полностью влезла в область обзора. Наконец, мы устанавливаем источник освещения (функция glLight), его тип и интенсивность (функция glLightfv).

```
40  def main():
41      pygame.init()
42      display = (1000, 800)
43      pygame.display.set_mode(display, DOUBLEBUF | OPENGL)
44      pygame.display.set_caption("Doshirak")
45      gluPerspective(45, (display[0]/display[1]), 0.1, 50.0)
46      glTranslatef(0.0, 0.0, -5)
47
48      # point light from the left, top, front
49      glLight(GL_LIGHT0, GL_POSITION,  (5, 5, 5, 1))
50      glLightfv(GL_LIGHT0, GL_AMBIENT, (0, 0, 0, 1))
51      glLightfv(GL_LIGHT0, GL_DIFFUSE, (1, 1, 1, 1))
52
53      glEnable(GL_DEPTH_TEST)
```

Рисунок 1 – код настройки OpenGl

На рисунке 2 представлен код непосредственно контролирующий отрисовку экрана с моделью. С 74 по 78 строчку код очищает буфер окна и устанавливает первоначальные значения для OpenGl. Затем, если мы можем повернуть объект, то вызываем функцию glRotatef и поворачиваем систему координат. 88 по 118 строчки отрисовывают модель, перемещая систему координат и вызывая функцию drawTruncatedPyramid, описание которой будет дальше.

```
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)

    glEnable(GL_LIGHTING)
    glEnable(GL_LIGHT0)
    glColorMaterial(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE)

    if rotate_vector != [0, 0, 0]:
        glRotatef(1, rotate_vector[0], rotate_vector[1], rotate_vector[2])

    # Draw box
    drawTruncatedPyramid(height=0.3, width=0.8, aspect_ratio=0.75)

    # Draw the top border box
    # Draw side borders
    glPushMatrix()
    glTranslatef(0.8, 0.3, 0)
    drawTruncatedPyramid(height=0.1, width=0.1, length=1.1)
    glTranslatef(-1.6, 0, 0)
    drawTruncatedPyramid(height=0.1, width=0.1, length=1.1)
    glPopMatrix()
    # Draw front borders
    glPushMatrix()
    glTranslatef(0, 0.3, 1)
    drawTruncatedPyramid(height=0.1, width=0.899, length=0.099)
    glTranslatef(0, 0, -2)
    drawTruncatedPyramid(height=0.1, width=0.899, length=0.099)
    glPopMatrix()

    # Draw the box lid
    glPushMatrix()
    glTranslatef(0, 0.3, 0)
    drawTruncatedPyramid(height=0.007, width=0.5,
                         length=0.7, aspect_ratio=1.25)
    glPopMatrix()

    # Draw the legs of box
    glPushMatrix()
    glTranslatef(0.4, -0.3, 0)
    drawTruncatedPyramid(height=0.02, width=0.1, length=0.6, aspect_ratio=0.9)
    glTranslatef(-0.8, 0, 0)
    drawTruncatedPyramid(height=0.02, width=0.1, length=0.6, aspect_ratio=0.9)
    glPopMatrix()

    glDisable(GL_LIGHT0)
    glDisable(GL_LIGHTING)
```

Рисунок 2 – код отрисовки модели

Теперь осталось рассмотреть функцию drawTruncatedPyramid (см. рис. 3). Это функция отрисовывают усеченную пирамиду и принимает на вход

коэффициент ширины, высоты и долготы фигуры, а также соотношение верхней и нижней сторон пирамиды. В самой функции в переменной vertices мы храним вершины нашей фигуры, домножая их координаты на соответствующие коэффициенты. Затем мы создаем кортеж из сторон нашей пирамиды, определяя нормаль грани и вершины, которые её образуют. Код с 32 по 37 строчку отрисовывают вершины в режиме отрисовке четырехугольника.

```
7   def drawTruncatedPyramid(height=1, width=1, length=1, aspect_ratio=1):
8       vertices = (
9           (1 * aspect_ratio, -1, -1 * aspect_ratio),
10          (1,  1, -1),
11          (-1,  1, -1),
12          (-1 * aspect_ratio, -1, -1 * aspect_ratio),
13          (1 * aspect_ratio, -1,  1 * aspect_ratio),
14          (1,  1,  1),
15          (-1 * aspect_ratio, -1,  1 * aspect_ratio),
16          (-1,  1,  1),
17      )
18
19      vertices = tuple((x * width, y * height, z * length)
20                       for x, y, z in vertices)
21
22      # normals and edge tuple
23      surfaces = (
24          ((0,  0, -1), (0, 1, 2, 3)),
25          ((-1,  0,  0), (3, 2, 7, 6)),
26          ((0,  0,  1), (6, 7, 5, 4)),
27          ((1,  0,  0), (4, 5, 1, 0)),
28          ((0,  1,  0), (1, 5, 7, 2)),
29          ((0, -1,  0), (4, 0, 3, 6))
30      )
31
32      glBegin(GL_QUADS)
33      for normal, edge in surfaces:
34          glNormal3fv(normal)
35          for vertex in edge:
36              glVertex3fv(vertices[vertex])
37      glEnd()
38
```

Рисунок 3 – код отрисовки пирамиды

Теперь приведем пару примеров работы нашей программы (.gif анимация работы программы доступна на GitHub (URL: https://github.com/vladcto/SUAI_homework/blob/20ba27fb305e4d51bc7289350eaac22b48ffb79d/4_semester/CG/3%D0%BB%D1%80/model_preview.gif)).

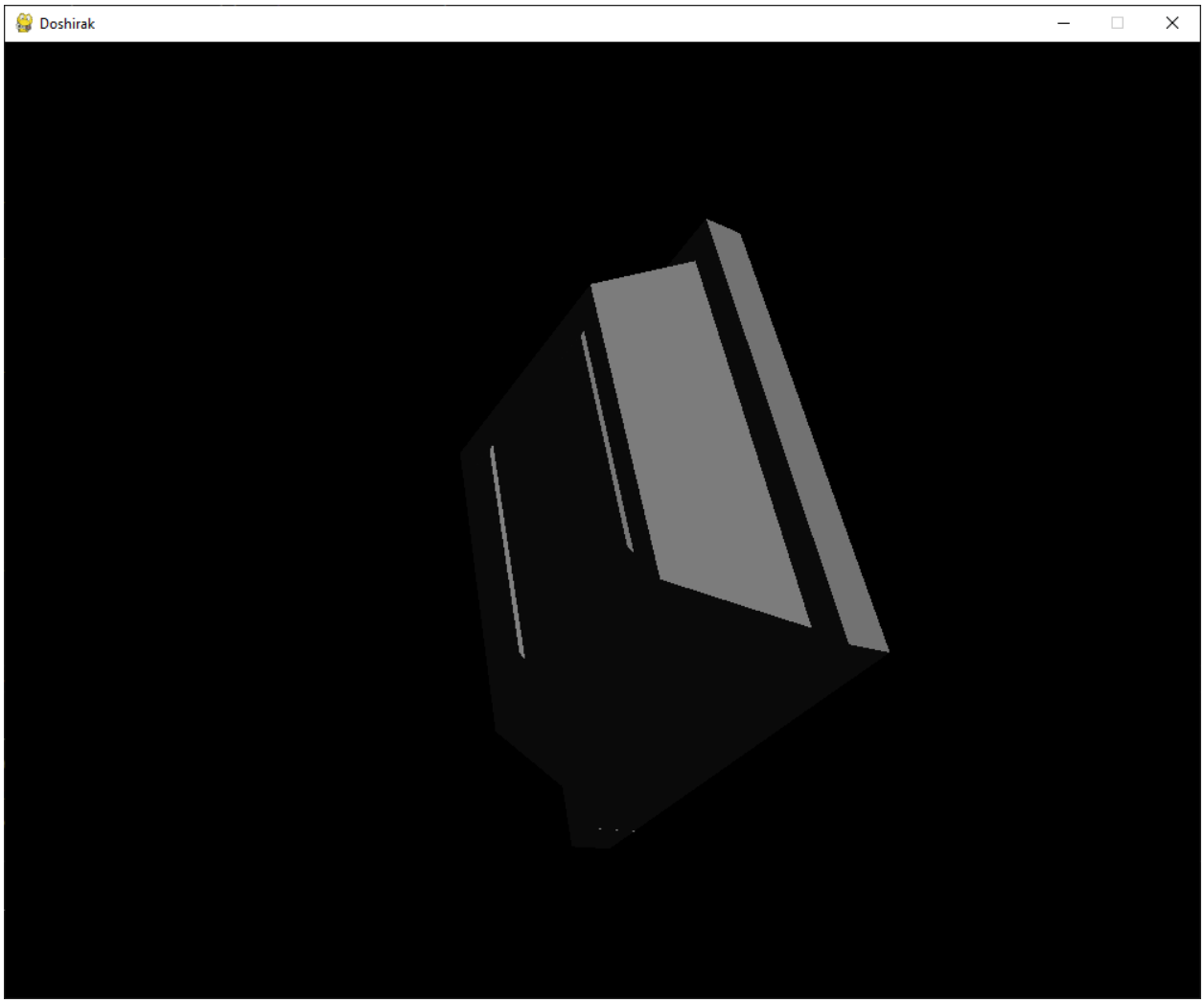

Рисунок 4 – скриншот работы программы

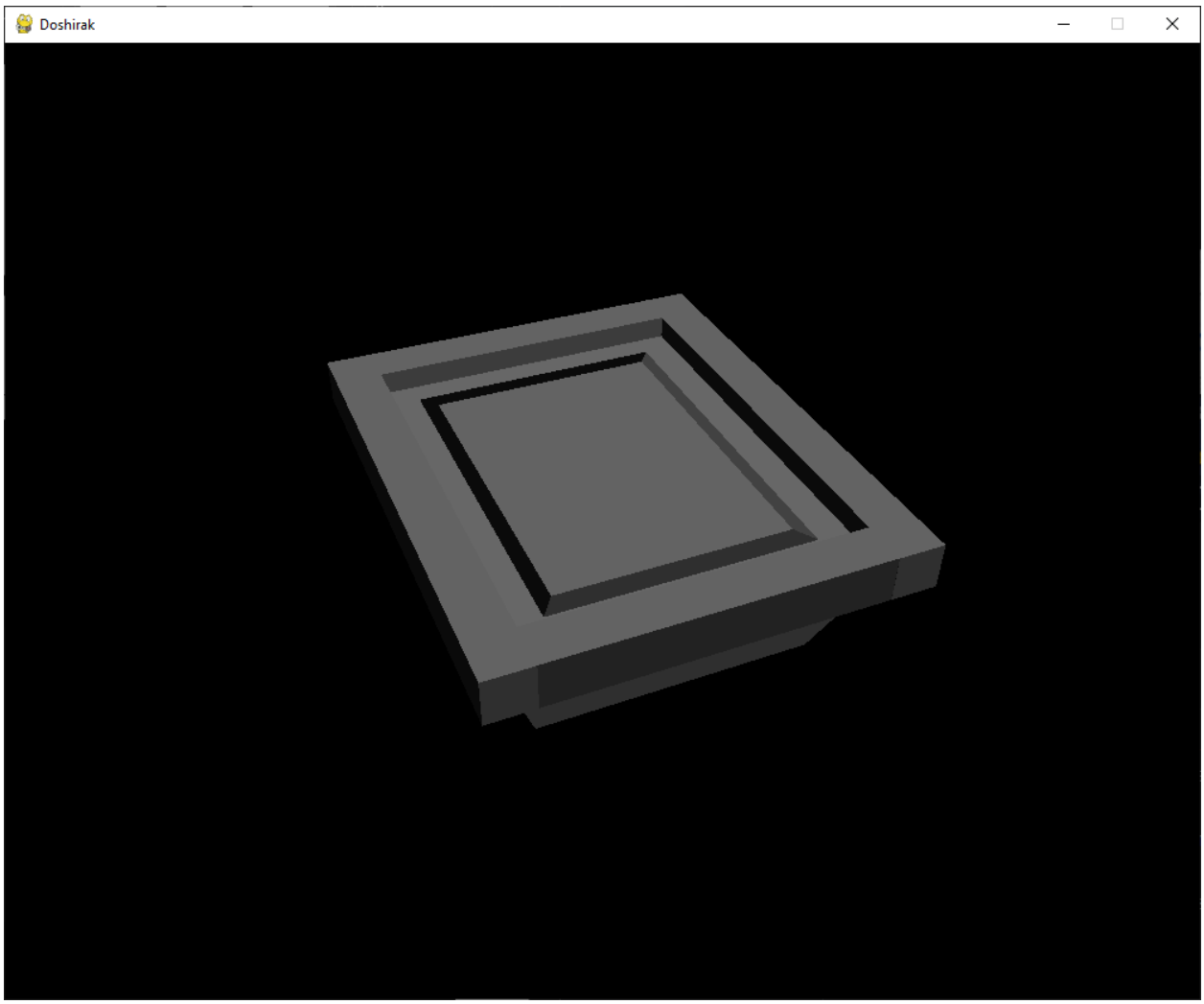

Рисунок 5 – скриншот работы программы

4 Вывод

В результате выполнения работы была изучена открытая графическая библиотеки OpenGL. OpenGL является мощной библиотекой для создания простых примитивов, освещения, 3D моделей и т.д. Он предоставляет богатый набор графических инструментов, которые позволяют создавать и отображать различные примитивы и модели в 3D пространстве. Он также предоставляет инструменты для создания и настройки освещения. Все это делает OpenGL мощным инструментом для создания 3D графики.

Также мы построили динамическую 3D-сцену на языке программирования высокого уровня, поддерживающего библиотеку OpenGL. В ходе написания кода, мы использовали такие функции билиотеки:

- Работа с матрицами: glTranslatef, gluPerspective, glRotatef, glPushMatrix, glPopMatrix.
- Работа с освещением: glLight, glLightfv, glColorMaterial.
- Работа с отрисовкой примитивов: glBegin, glVertex3fv, glEnd.
- Настройка OpenGl: glDisable, glEnable

Полученные навыки можно применять при создании графики 2D или 3D приложений разного рода задач: анализ данных, виртуальная реальность и другое.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1) Райт, Р.С.-мл., Липчак Б. OpenGL. Суперкнига, 3-е издание. — М.: Издательский дом «Вильямс», 2006. - 1040 с. (дата обращения: 14.03.2023)
2) Pythonisit: Введение в PyOpenGl: сайт. – URL: https://pythonist.ru/vvedenie-v-opengl-i-pyopengl-chast-i-sozdanie-vrashhayushhegosya-kuba/ (дата обращения: 14.03.2023)
3) Stackoverflow: как добавить свет в PyOpenGl: сайт. – URL: https://stackoverflow.com/questions/56514791/how-to-correctly-add-a-light-to-make-object-get-a-better-view-with-pygame-and-py (дата обращения: 14.03.2023)
4) PyOpenGl: документация: сайт. – URL: https://pyopengl.sourceforge.net/documentation/index.html (дата обращения: 14.03.2023)

ПРИЛОЖЕНИЕ А

ЛИСТИНГ ПРОГРАММЫ

```
import pygame
from pygame.locals import *
from OpenGL.GL import *
from OpenGL.GLU import *



def drawTruncatedPyramid(height=1, width=1, length=1, aspect_ratio=1):
    verticies = (
        (1 * aspect_ratio, -1, -1 * aspect_ratio),
        (1,  1, -1),
        (-1,  1, -1),
        (-1 * aspect_ratio, -1, -1 * aspect_ratio),
        (1 * aspect_ratio, -1,  1 * aspect_ratio),
        (1,  1,  1),
        (-1 * aspect_ratio, -1,  1 * aspect_ratio),
        (-1,  1,  1),
    )

    verticies = tuple((x * width, y * height, z * length)
                      for x, y, z in verticies)

    # normals and edge tuple
```

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

    glBegin(GL_QUADS)
    for normal, edge in surfaces:
        glNormal3fv(normal)
        for vertex in edge:
            glVertex3fv(verticies[vertex])
    glEnd()



def main():
    pygame.init()
    display = (1000, 800)
    pygame.display.set_mode(display, DOUBLEBUF | OPENGL)
    pygame.display.set_caption("Doshirak")
    gluPerspective(45, (display[0]/display[1]), 0.1, 50.0)
    glTranslatef(0.0, 0.0, -5)
```

```
# point light from the left, top, front
glLight(GL_LIGHT0, GL_POSITION,  (5, 5, 5, 1))
glLightfv(GL_LIGHT0, GL_AMBIENT, (0, 0, 0, 1))
glLightfv(GL_LIGHT0, GL_DIFFUSE, (1, 1, 1, 1))

glEnable(GL_DEPTH_TEST)

rotate_vector = [5, 3, 1]
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            quit()
    # input events to rotate vector
    pressed = pygame.key.get_pressed()
    if pressed[pygame.K_UP]:
        rotate_vector = [1, 0, 0]
    if pressed[pygame.K_DOWN]:
        rotate_vector = [5, 3, 1]
    if pressed[pygame.K_RIGHT]:
        rotate_vector = [0, 0, 1]
    if pressed[pygame.K_LEFT]:
        rotate_vector = [0, 1, 0]
```

```
        if pressed[pygame.K_SPACE]:
            rotate_vector = [0, 0, 0]

        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)

        glEnable(GL_LIGHTING)
        glEnable(GL_LIGHT0)
        glColorMaterial(GL_FRONT_AND_BACK,
GL_AMBIENT_AND_DIFFUSE)

        if rotate_vector != [0, 0, 0]:
            glRotatef(1, rotate_vector[0], rotate_vector[1], rotate_vector[2])

        # Draw box
        drawTruncatedPyramid(height=0.3, width=0.8, aspect_ratio=0.75)

        # Draw the top border box
        # Draw side borders
        glPushMatrix()
        glTranslatef(0.8, 0.3, 0)
        drawTruncatedPyramid(height=0.1, width=0.1, length=1.1)
        glTranslatef(-1.6, 0, 0)
        drawTruncatedPyramid(height=0.1, width=0.1, length=1.1)
        glPopMatrix()
```

```
        # Draw front borders
        glPushMatrix()
        glTranslatef(0, 0.3, 1)
        drawTruncatedPyramid(height=0.1, width=0.899, length=0.099)
        glTranslatef(0, 0, -2)
        drawTruncatedPyramid(height=0.1, width=0.899, length=0.099)
        glPopMatrix()

        # Draw the box lid
        glPushMatrix()
        glTranslatef(0, 0.3, 0)
        drawTruncatedPyramid(height=0.007, width=0.5,
                             length=0.7, aspect_ratio=1.25)
        glPopMatrix()

        # Draw the legs of box
        glPushMatrix()
        glTranslatef(0.4, -0.3, 0)
        drawTruncatedPyramid(height=0.02, width=0.1, length=0.6,
aspect_ratio=0.9)
        glTranslatef(-0.8, 0, 0)
        drawTruncatedPyramid(height=0.02, width=0.1, length=0.6,
aspect_ratio=0.9)
        glPopMatrix()
```

```
        glDisable(GL_LIGHT0)
        glDisable(GL_LIGHTING)

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


main()
```