ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ _____

ПРЕПОДАВАТЕЛЬ

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

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 4

ПРОЕКТИВНЫЕ ПРЕОБРАЗОВАНИЯ

по курсу: КОМПЬЮТЕРНАЯ ГРАФИКА

РАБОТУ ВЫПОЛНИЛ

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

Санкт-Петербург 2023

СОДЕРЖАНИЕ

1 Цель работы

Изучение проективных преобразований и построение их с помощью языка программирования высокого уровня.

Задание:

На любом языке программирования высокого уровня (из числа изученных в ходе освоения ООП 09.03.02), выполнить проективное преобразование трехмерного объекта на плоскость с использованием графического инструментария.

Выполнить проективное преобразование по трем осям. Рекомендуется брать такие фигуры, чтобы на разных плоскостях получались разные проекции. Также не рекомендуется брать самые типовые фигуры как шар или параллелепипед

2 Теоретические сведения

Проецирование (от лат. projectio – выбрасывать вперед) – это отображение трехмерного объекта на двухмерную картинную плоскость (КП). Получение проекции базируется на методе трассировки лучей: из центра проецирования S (проектора) проводятся лучи через каждую точку объекта до пересечения с КП (рисунок 1). След, образуемый точками пересечения лучей с КП, составляет проекцию объекта. Если фокусное расстояние F во много раз больше габаритов объекта и расстояния d от источника лучей до объекта, то пучок проецирующих лучей можно считать параллельным, а источник – дальним. Такая идеализация отображения объемного мира на плоский экран облегчает построение проекций, за что приходится платить ухудшением реалистичности изображений.

За длительное время развития цивилизации придумано множество способов изображения объемных объектов на плоскости, более или менее точно передающих их геометрическую форму. Поскольку проецирование – не аффинное преобразование и в принципе необратимо, то любая проекция всегда содержит в себе ошибку (лучше сказать – неопределенность) восстановления координат объекта-образа. Назначение того или иного метода проецирования состоит в уменьшении неопределенности восприятия наиболее значимых свойств объекта за счет менее значимых. Например, архитектурный чертеж фасада здания абсолютно точно передает геометрическую ширину и высоту как всего фасада, так и его деталей, за счет полной неопределенности их глубины.

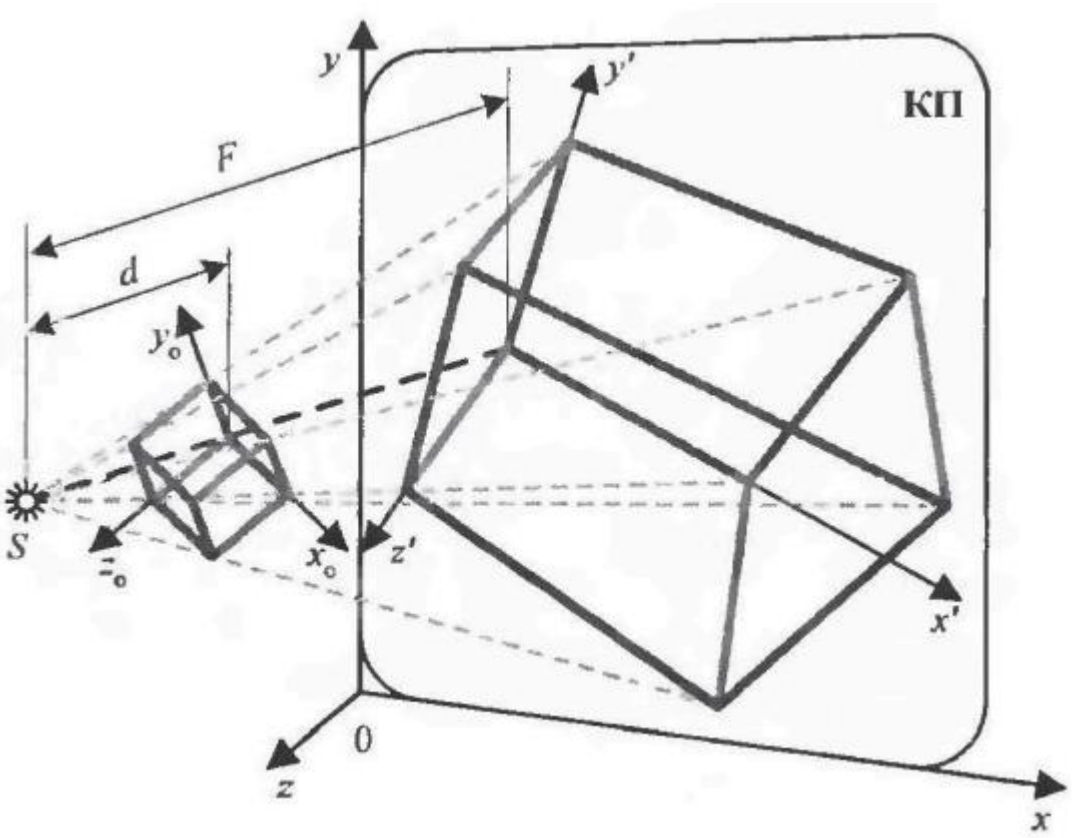

Рисунок 1 - Получение проекции методом трассировки лучей

К числу важнейших свойств метода проецирования относится достоверность восприятия объекта наблюдателем по его проекции, т. е. узнаваемость объекта по его плоскому изображению. Если свойство достоверности восприятия объявляется доминирующим, то наилучшей может стать «живая» проекция с далеко не минимальной геометрической неопределенностью. Например, изометрическая проекция несет искаженную информацию о размерах объекта во всех трех измерениях, но лучше передает форму объекта в целом по сравнению с изображением объекта в «фас».

Рисунок 2 - Классификация проекций

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, её проецирование на 3 плоскости, а также её отображение в изометрической проекции. Исходный код доступен в Приложении А, а также на GitHub(URL: https://github.com/vladcto/SUAI_homework/blob/967c5b75f0405bbbeef8c93525820cb5cb784468/4_semester/CG/4%D0%BB%D1%80/solution.py).

Исходный код снабжен комментариями и легок для понимания. В программе предусмотрено изменение режима проецирования:

- Для проецирования на плоскость oYZ нажмите СТРЕЛКУ ВПРАВО.
- Для проецирования на плоскость oXZ нажмите СТРЕЛКУ ВВЕРХ.
- Для проецирования на плоскость oXY нажмите СТРЕЛКУ ВЛЕВО.
- Для изометрического проецирования нажмите СТРЕЛКУ ВНИЗ.

Также в программе доступно включение/выключение источников света, для этого достаточно нажать ПРОБЕЛ.

Приведем примеры работы программы ниже, приложив скриншоты.

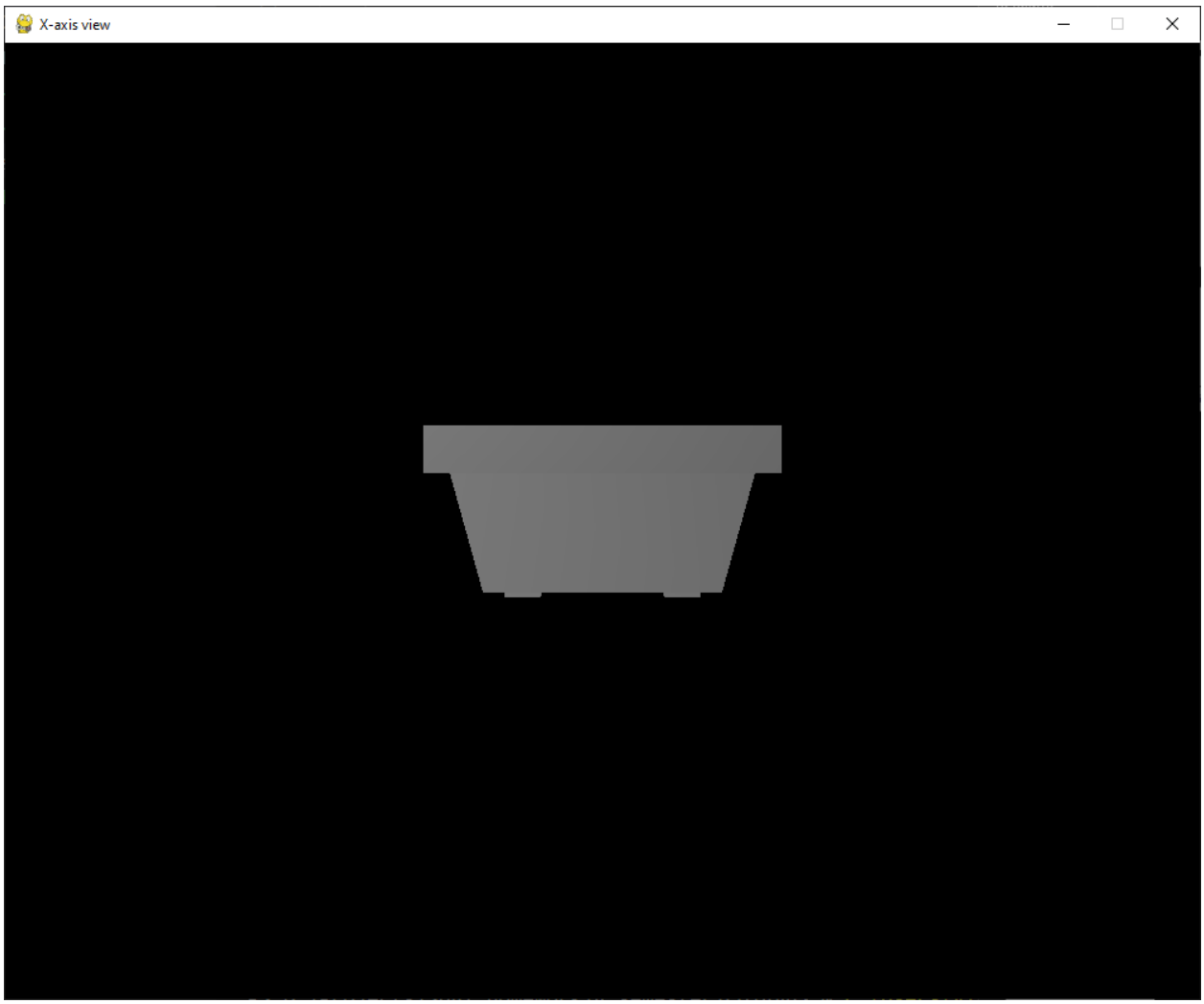

Рисунок 3 – Проекция на плоскость oyz

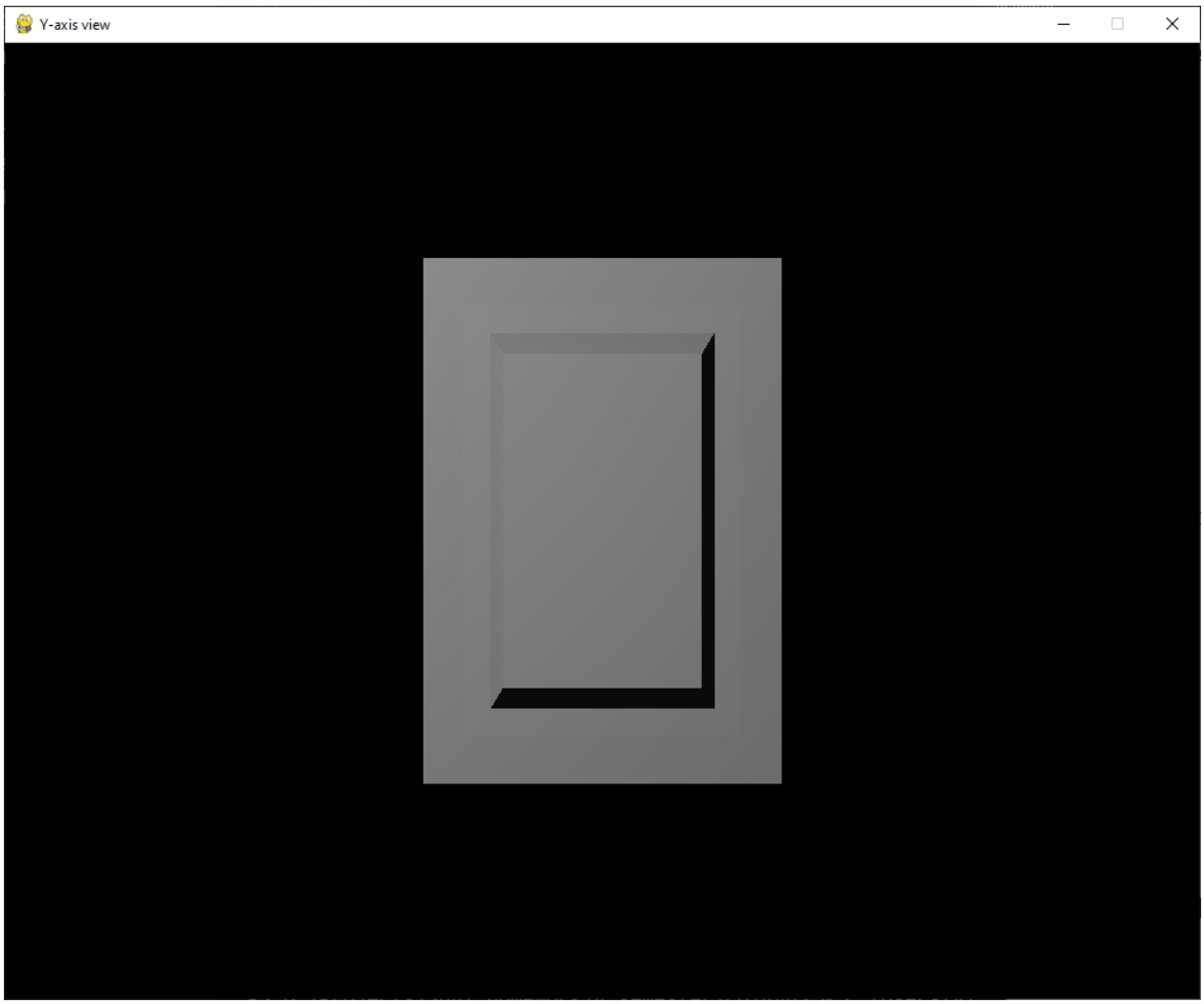

Рисунок 4 – Проекция на плоскость oxz

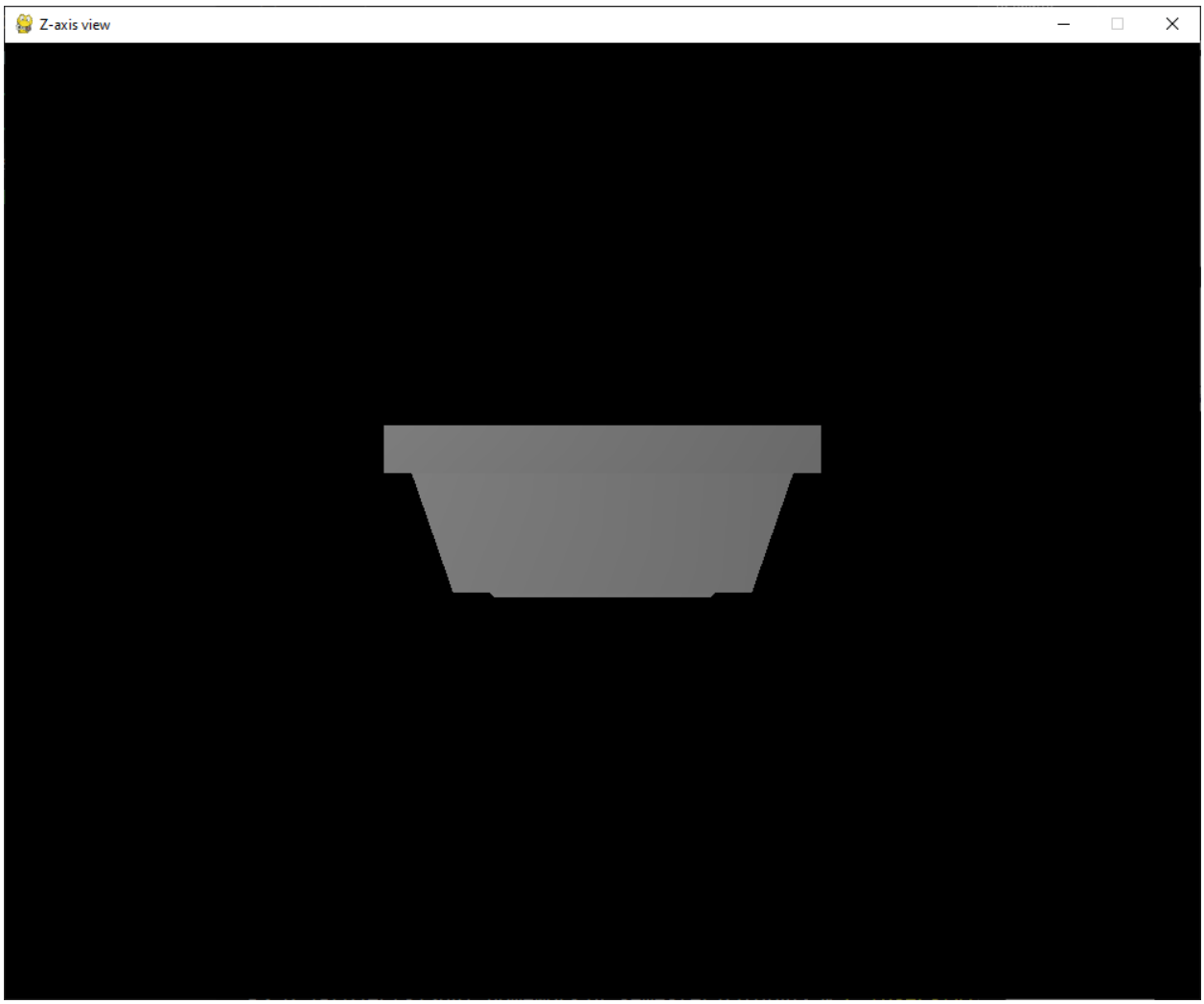

Рисунок 5 – Проекция на плоскость oxy

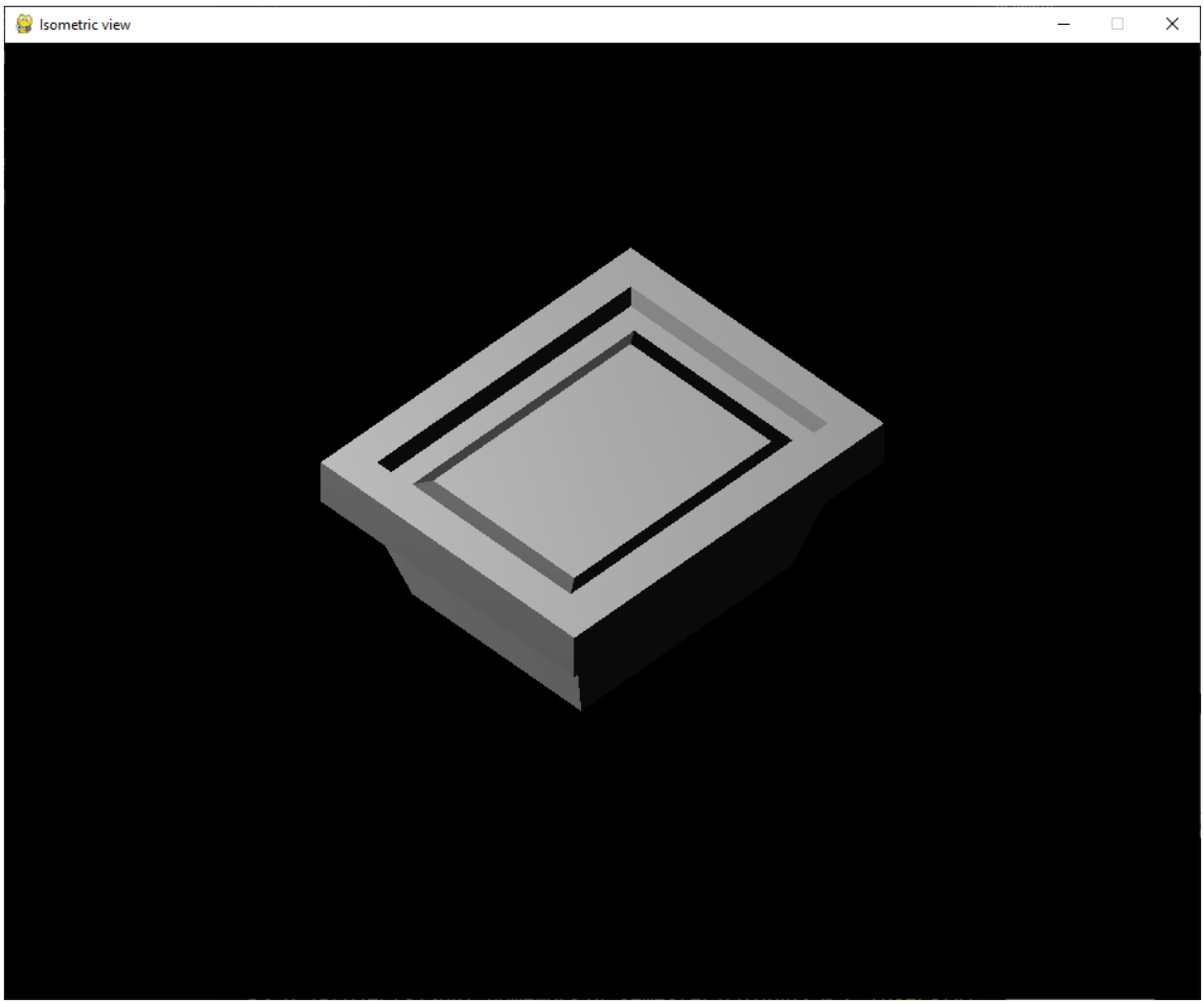

Рисунок 6 – Изометрическая проекция

Также для рисунка 6 мне бы хотелось для достоверности привести в сравнение модель с центральной проекцией(см. рисунок 7), полученной в лабораторной работе № 3 (URL на отчет: https://github.com/vladcto/SUAI_homework/blob/%D0%A1G/4lr/4_semester/CG/3%D0%BB%D1%80/%D0%92%D0%BE%D1%80%D0%BE%D0%B1%D1%8C%D0%B5%D0%B24128%D0%9B%D0%A03.pdf). Как мы можем видеть, изометрическая проекция отображает достоверное представление длины, ширины и долготы объекта, но за это приходится платить отсутствием глубины изображения.

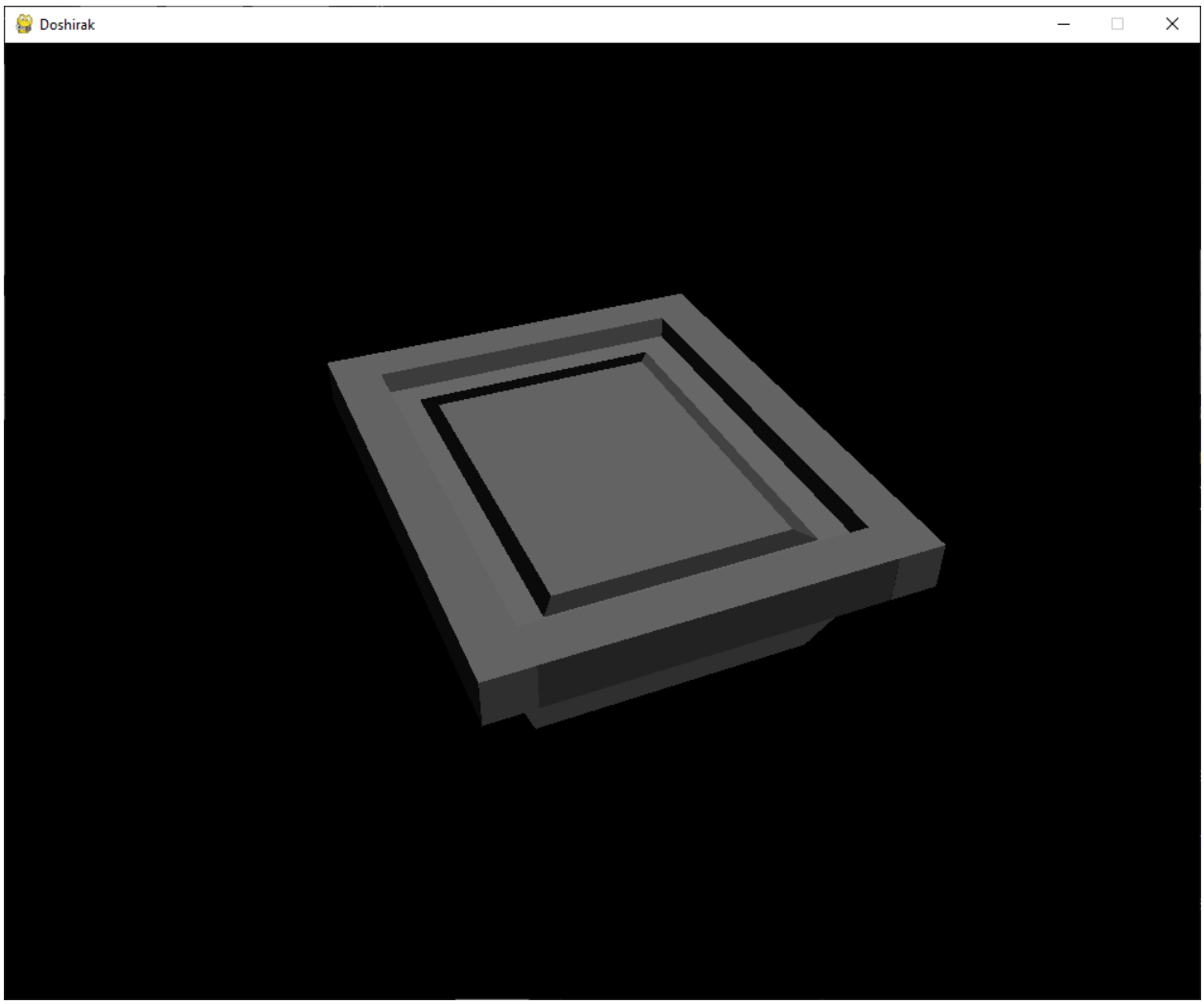

Рисунок 7 – Центральная проекция

4 Вывод

В ходе выполнения лабораторной работы узнали о типах проецирования и методах проецирования в компьютерной графике.

Проецирование – процесс получения изображения предмета (пространственного объекта) на какой-либо поверхности с помощью световых или зрительных лучей (лучей, условно соединяющих глаз наблюдателя с какой-либо точкой пространственного объекта), которые называются проецирующими.

Можно выделить два основных метода проецирования.

- **Центральное** проецирование использует лучи, исходящие от центра проекции (перспективы) и проецирующие объекты на плоскость. Это проецирование позволяет сделать изображение похожим на то, как оно выглядело бы в реальном мире, а также может достоверно отображать глубину изображения, нарушая при этом ясность длины, ширины и долготы объекта.
- **Параллельное** проецирование использует лучи, идущие в одном направлении (параллельно) и проецирующие объекты на плоскость. Если направление проецирования перпендикулярно плоскости проекций, то проецирование называется прямоугольным или ортогональным, а если не перпендикулярно, то проецирование называется косоугольным. Это проецирование дает менее достоверное отображение глубины модели, тем не менее позволяет корректно отобразить характеристики размера объекта. Также можем сказать, что этот тип проецирования является более быстрым и эффективным.

Полученные знания закрепили, написав программу с использованием языка Python 3 и модулей PyOpengl и pygame. Программа реализует параллельное проецирование модели в трех плоскостях, а также позволяет

отобразить модель в изометрическом виде. Функционал программы был протестирован и приложен в виде скриншотов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1) Райт, Р.С.-мл., Липчак Б. OpenGL. Суперкнига, 3-е издание. — М.: Издательский дом «Вильямс», 2006. - 1040 с. (дата обращения: 15.03.2023)
2) Pythonisit: Введение в PyOpenGl: сайт. – URL: https://pythonist.ru/vvedenie-v-opengl-i-pyopengl-chast-i-sozdanie-vrashhayushhegosya-kuba/ (дата обращения: 15.03.2023)
3) GoogleGroups: Поворот для изометрической проекции: сайт. – URL: https://groups.google.com/g/comp.graphics.api.opengl/c/9fS5o2QtyPc (дата обращения: 15.03.2023)
4) BSU: Лекция машинной графики: сайт. – URL: https://bsu.by/upload/page/353583.pdf (дата обращения: 15.03.2023)
5) StudFiles: Основные виды проецирования: сайт. – URL: https://studfile.net/preview/2814520/ (дата обращения: 15.03.2023)
6) PyOpenGl: документация: сайт. – URL: https://pyopengl.sourceforge.net/documentation/index.html (дата обращения: 15.03.2023)

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")

    glEnable(GL_DEPTH_TEST)
```

```
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    glOrtho(0, 6, 4, 0, 0.1, 50)
    glTranslatef(3, 2, -5)
    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()
    # Make the model more logical.
    glRotatef(180,0,0,1)
    glPushMatrix()

    # 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))

    enable_light = False
    while True:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                quit()
            if event.type == pygame.KEYDOWN and event.key ==
pygame.K_SPACE:
                enable_light = not enable_light
```

```
        if enable_light:
            glEnable(GL_LIGHTING)
            glEnable(GL_LIGHT0)
        else:
            glDisable(GL_LIGHTING)

# input events to axis
pressed = pygame.key.get_pressed()
if pressed[pygame.K_UP]:
    pygame.display.set_caption("Y-axis view")
    glPopMatrix()
    glPushMatrix()
    glRotatef(90,1,0,0)
if pressed[pygame.K_DOWN]:
    pygame.display.set_caption("Isometric view")
    glPopMatrix()
    glPushMatrix()
    glRotatef(56.600269334224741,
              0.5902844985873289,
              0.76927373575385605,
              0.24450384497347222)
if pressed[pygame.K_RIGHT]:
    pygame.display.set_caption("X-axis view")
    glPopMatrix()
```

```
            glPushMatrix()
        if pressed[pygame.K_LEFT]:
            pygame.display.set_caption("Z-axis view")
            glPopMatrix()
            glPushMatrix()
            glRotatef(90,0,1,0)

        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
        glColorMaterial(GL_FRONT_AND_BACK,
GL_AMBIENT_AND_DIFFUSE)

        # 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()

        pygame.display.flip()
```

```
        pygame.time.wait(10)


main()
```