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

Исследование проективных преобразований

по курсу:

Компьютерная графика

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. № 4329		Д.С. Шаповалова
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2024

Содержание

1. Цель работы:

Изучение теоретических основ проективных преобразований в трехмерном пространстве. Исследование особенностей практической реализации проективных преобразований.

2. Задание:

Написать программу на любом языке высокого уровня, реализующую получение анимированной проекции фигуры в соответствии с вариантом задания – 17: Фигура – Куб, Проекция – Кабинетная, Ось вращения – Параллельна OX.

Заданный многогранник должен вращаться вокруг указанной оси, не проходящей через сам многогранник и через начало координат, а также не совпадающей с координатными осями. Для всех вариантов фигура должна отображаться в контурном виде без удаления невидимых линий. Рисование контура фигур по матрице координат вершин можно (но не обязательно) осуществлять с помощью специализированных библиотек. Аффинные и проективные преобразования необходимо выполнять только путем рассмотренных выше матричных вычислений (можно, но не обязательно, использовать библиотеки программ для матричных вычислений). Использование специализированных программ геометрических преобразований не допускается.

3. Теоретические сведения:

3.1 Типы проективных преобразований

Проектирование – это отображение точек из системы координат N в систему координат меньшей размерности n (n<N). Мы будем рассматривать проекции трехмерных объектов на плоскость, соответствующие отображению реального мира на экране (рис. 1.1).

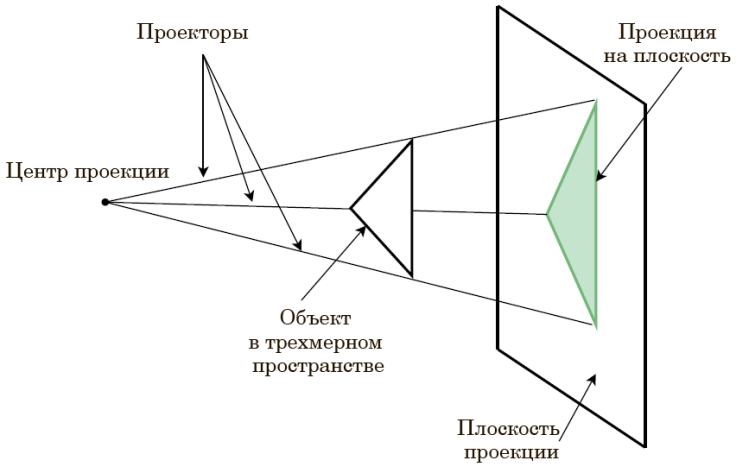

Рисунок 1.1 – Проецирование на плоскость

Проекторами называют отрезки прямых, идущие из центра проекции через точки объекта до пересечения с плоскостью. В зависимости от расположения центра проекции выделяют:

– центральное (перспективное) проецирование с конечным центром;

– параллельное проецирование с бесконечным центром и параллельными лучами.

В большинстве графических систем параллельное и перспективное проецирование рассматриваются как два разных типа с соответствующими функциями обработки и параметрами (рис. 1.2).

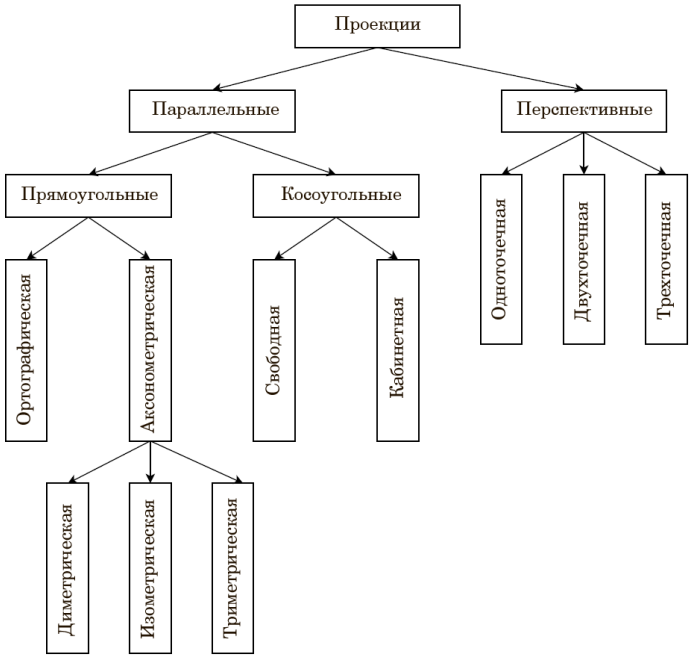

Рисунок 1.2 - Классификация проекций

Параллельные проекции делятся на два типа:

Прямоугольные (ортографические и аксонометрические) с совпадающими направлениями проектора и нормали;

Косоугольные с несовпадающими направлениями проектора и нормали.

Все проективные преобразования вырождены, восстановить объект по одной проекции невозможно.

3.2. Ортографические проекции

Ортографическая проекция представляет собой отображение трехмерного объекта на двумерную поверхность, где линии проекции (или лучи) перпендикулярны к этой поверхности. Это означает, что все точки объекта проецируются вертикально на плоскость проекции, и масштаб остается постоянным.

Основные характеристики:

Отсутствие искажений: В ортографической проекции расстояния между объектами и их размеры не искажаются. Это позволяет точно измерять длины и площади.

Количество видов: Для полного представления объекта часто применяется несколько проекций (например, фронтальная, боковая и сверху).

Координаты: В ортографической проекции трехмерные координаты объекта (x,y,z) преобразуются в двумерные (x′, y′) без учета координаты z : x′ = x, y′ = y.

Ортографические проекции могут быть описаны при помощи матричного преобразования. Для проекции на плоскость z=0 используется соответствующая матрица (рис. 1.3)

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Рисунок 1.3 – Матрица для ортографической проекции

3.3. Аксонометрические проекции

Аксонометрическая проекция позволяет отображать трехмерные объекты таким образом, чтобы сохранить их трехмерное восприятие, но при этом представление остается двумерным. Основное отличие от ортографических проекций заключается в том, что в аксонометрии применяются наклонные оси, что придаёт изображению объем и глубину - плоскость проекции в них не совпадает с координатными плоскостями и не параллельна им.

Основные типы аксонометрических проекций:

Изометрическая проекция:

В изометрической проекции угол между осями X, Y и Z составляет 120 градусов, что делает все три оси равнозначными. Масштаб по всем осям одинаков, что позволяет точно передавать размеры объектов.

Диметрическая проекция:

В диметрической проекции используются две оси с одинаковым масштабом, а третья ось имеет другой масштаб, что приводит к искажению одного из направлений. Углы между осями также варьируются, что создает различные визуальные эффекты.

Триметрическая проекция:

В этой проекции каждая ось представляется с разным масштабом, создавая наиболее выраженное искажение. Эта проекция используется для передачи наиболее детального представления объекта, хотя и менее удобно воспринимается.

Математическое представление:

Для создания аксонометрической проекции используется матричное преобразование. Пример матрицы (рис. 1.4) для изометрической проекции может выглядеть так:

$$M = \begin{pmatrix} \frac{1}{\sqrt{3}} & 0 & -\frac{1}{\sqrt{3}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Рисунок 1.4 – Матрица для аксонометрической проекции

3.4 Косоугольные проекции

Косоугольные проекции создаются путем проекции объекта под углом к плоскости проекции, что приводит к искажению формы и размера объектов. В отличие от ортографических и аксонометрических проекций, в которых размеры и пропорции остаются более или менее постоянными, в косоугольной проекции эти параметры могут варьироваться, что придаёт изображению особый стиль.

Основные характеристики

Угол наклона: Объекты проецируются под произвольным углом, чаще всего 30 или 45 градусов, что позволяет создать эффект глубины.

Искажение: Способ проекции приводит к изменению длины сторон объектов, что может использоваться для достижения художественных эффектов или акцентирования внимания на определённых частях объекта.

Простота восприятия: Несмотря на искажения, косоугольные проекции легко воспринимаются, что делает их популярными для ряда приложений.

Математическое представление

Косоугольная проекция может быть описана с использованием матриц преобразования. Например, для косоугольной проекции с углом 45 градусов и с использованием изометрической проекции может быть следующая матрица (рис. 1.5):

$$M = \begin{pmatrix} 1 & 0 & -\frac{1}{2} & 0 \\ 0 & 1 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Рисунок 1.5 – Матрица для косоугольной проекции

Выделяют два типа косоугольных проекций:

– свободную (рис 1.6),

– кабинетную (рис. 1.7).

Свободную проекцию иногда называют косоугольной изометрией. При свободной проекции угол между проекторами и плоскостью проекции равен 45°, т.е. $\alpha = \beta = Cos\frac{\pi}{4}$

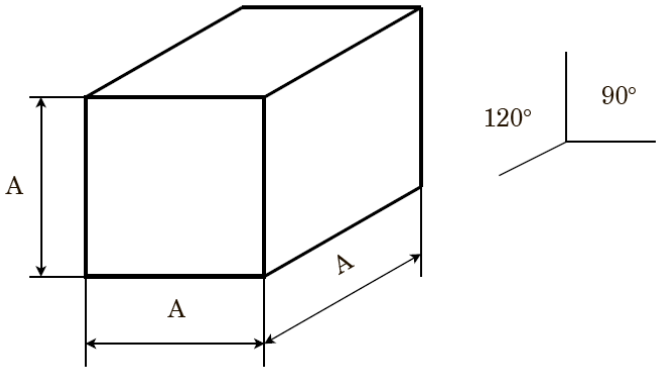

Рисунок 1.6 – Свободная проекция куба

Кабинетная проекция, иногда называемая косоугольной диметрией – частный случай свободной проекции, при котором $\alpha = \beta = \frac{1}{2}Cos\frac{\pi}{4}$

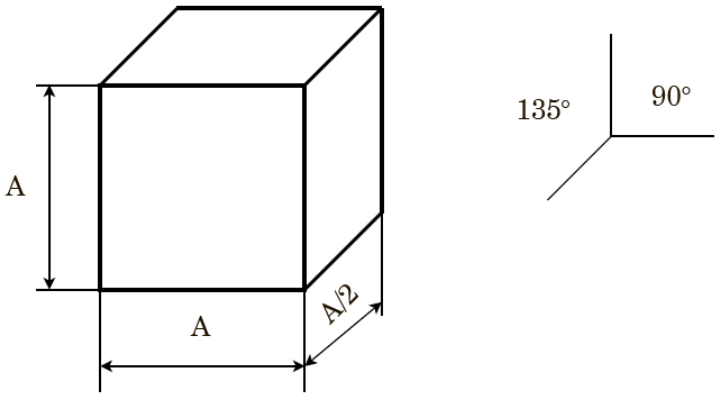

Рисунок 1.7 – Кабинетная проекция куба

4. Алгоритм преобразований:

1. Определение вершин куба.

Код создает массив вершин куба в трехмерном пространстве. Вершины задаются с координатами x, y и z, что позволяет нам иметь полное представление о положении куба в пространстве.

2. Вращение куба.

Исполняющийся алгоритм включает матрицу вращения по оси X, что позволяет кубу вращаться вокруг нее. Это важный аспект отрисовки, поскольку куб будет менять свою ориентацию, что наглядно демонстрирует трехмерное движение.

3. Кабинетная проекция.

Проекция осуществляется путем умножения координат вершин на матрицу проекции. В данной реализации, используется простая матрица, которая наклоняет проекцию вдоль оси Z (глубины), тем самым создавая эффекты искажения.

В результате, каждая вершина проецируется на двумерную плоскость (основу), что создает эффект глубины за счет уменьшения координаты Z, что и соответствует принципам кабинетной проекции - вся глубина (координаты Z) уменьшается в 0.5 раз, что создает элизионный эффект.

4. Визуализация.

Объект куба отображается в 2D, но с учетом изменений координат, которые создают иллюзию трехмерности. Линии, соединяющие проецированные вершины, образуют видимые рёбра куба в плоскости, что помогает лучше воспринимать его форму.

5. Язык программирования и используемые библиотеки

В качестве языка программирования был выбран язык Python. В коде используются следующие библиотеки:

1) numpy (import numpy as np):

Библиотека Python, которую применяют для математических вычислений: начиная с базовых функций и заканчивая линейной алгеброй. В данном случае используется для создания и манипулирования массивами, а именно, для создания двумерного массива - матрицы, благодаря которой можно представить пиксели в трёхмерной системе координат, в последствии отображать проекцию в двухмерной системе координат.

2) matplotlib ((import matplotlib.pyplot as plt) и (import matplotlib.animation as animation)):

Библиотека на языке Python для визуализации данных. В ней можно построить двумерные и трехмерные графики. В данном случае используется для отображения перемещений точек – вершин куба, а также отрисовки его рёбер. В свою очередь «matplotlib.animation» используется для анимации перемещений.

6. Описание разработанной программы:

1. Импорт библиотек.

   В начале кода импортируются необходимые библиотеки, такие как NumPy для работы с массивами и Matplotlib для создания графиков и анимации.

2. Функция - определение вершин.

   Функция возвращает массив, содержащий координаты восьми вершин куба. r = 1 обозначает, что куб имеет длину ребра 2 (от -1 до 1 по всем осям).

3. Матрица вращения по оси X.

Функция создает матрицу вращения вокруг оси X. Параметр theta — это угол вращения в радианах. В результате матрицы c и s хранят косинус и синус угла, что позволяет правильно поворачивать координаты куба.

4. Кабинетная проекция.

Функция принимает вершины куба и применяет к ним кабинетную проекцию. Мы используем матрицу проекции, которая наклоняет (сдвигает) координаты по оси Z. Это создаёт эффект уменьшения значения Z (глубины) куба, что типично для кабинетной проекции.

5. Создание графика и настройка видимости.

Создается новое окно графика и задаются границы осей, чтобы куб помещался в видимое пространство.

6. Определение рёбер куба.

Создаём массив подмассивов с парами координат x, y. Определяем, какие вершины соединяются ребрами. Каждый подмассив содержит индексы вершин, которые составляют линии.

7. Функция обновления для анимации.

Функция вызывается для каждого кадра анимации. Она очищает текущее состояние графика, рассчитывает новое положение вершин куба (проводя его вращение) и применяет к ним кабинетную проекцию. Затем рисуются линии (рёбра) между проекциями вершин.

8. Создание анимации.

Создается анимация, которая обновляется, вызывая функцию update. frames=np.arange(0, 360, 5) генерирует углы от 0 до 360 градусов с шагом 5 градусов. plt.show() отображает окно с анимацией.

7. Скриншоты, иллюстрирующие результаты работы программы:

На данных иллюстрациях мы можем наглядно наблюдать кабинетную проекцию вращающегося куба параллельно оси OX. Представлены отображения куба по прошествии 1 секунды (рис. 2.1), 2 секунд (рис. 2.2), 3 секунд (рис. 2.3).

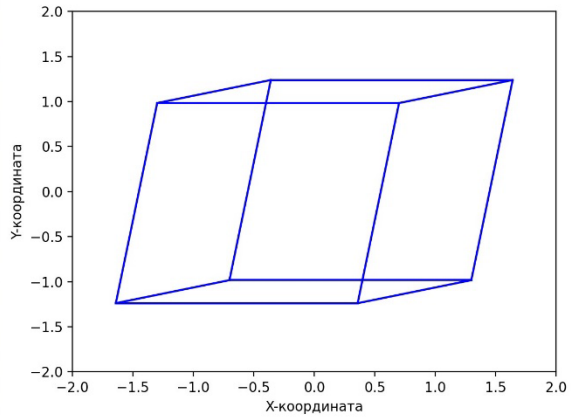

Рисунок 2.1 – Проекция вращающегося куба, с начала работы программы прошла 1 секунда.

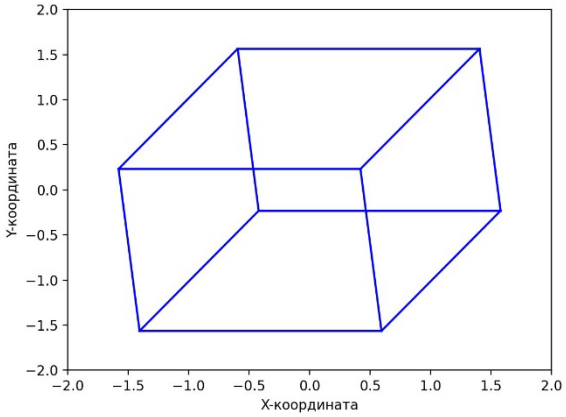

Рисунок 2.2 – Проекция вращающегося куба, с начала работы программы прошло 2 секунды.

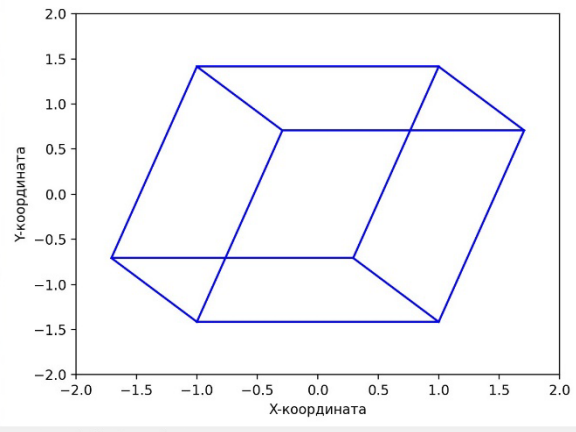

Рисунок 2.3 – Проекция вращающегося куба, с начала работы программы прошло 3 секунды.

8. Вывод:

8.1 Полученные теоретические знания и навыки:

В процессе выполнения этой работы были получены следующие теоретические знания и навыки:

1) Проективные преобразования:

Были изучены основы проективных преобразований в трёхмерном пространстве, особенности практической реализации отображения кабинетной проекции. Были исследованы особенности практической реализации проективных преобразований. Была вспомнена линейная алгебра, матрицы, векторы. Были освоены способы их переноса на компьютерный код, реализация манипуляций с пикселями в компьютерной программе.

2) Язык программирования высокого уровня Python:

В процессе выполнения работы был использован язык программирования Python. Был изучен алгоритм вращения куба, отображение кабинетной проекции, следовательно изучена визуализация алгоритма, посредством графика и математических операций с матрицами.

Так же были подробнее изучены некоторые библиотеки Python, такие как numpy и matplotlib, которые в будущем могут пригодиться в работе с графикой и с которыми мы работали в прошлых лабораторных.

3) Библиотеки numpy, matplotlib:

Использование этих библиотек позволило реализовать алгоритм перемещения часовых стрелок и визуализировать это в отдельном окошке. А конкретнее, модуль numpy был использован чтобы вычислять новые координаты точек для вектора. В общем позволяет нам работать с многомерными массивами данных (матрицы). В свою очередь matplotlib был использован для визуализации вычисленных данных на протяжении установленного количества кадров(длины анимации).

8.2 Возникшие проблемы и пути их решения:

В процессе выполнения работы возникали определенные трудности, а именно:

1) Разные версии Matplotlib:

Matplotlib, использованная в программе – это большая библиотека с обширным функционалом и большим количеством функций, существующая достаточно лет, чтобы некоторая созданная справочная информация стала слегка устаревшей, из-за чего возникали ошибки и изучать особенности конкретной версии и подстраивать полученную информацию под них. – Ранее импорт писали как import matplotlib.pyplot as plt; from mpl_toolkits.mplot3d import Axes3D; from matplotlib.animation import FuncAnimation. Теперь

импорт пишем как import matplotlib.pyplot as plt; import matplotlib.animation as animation. С виду банальность, но поначалу с толку сбивает.

2) Формулировка задания

В связи с ранее более понятным принципом построения анимации, представляется более сложным строить трёхмерную фигуру в трёх осях, но отображать лишь проекцию, в двух осях. Проблема решается лишь выполнением задания так, как поняли и надеждой на правильность понимания.

В результате выполнения этой работы, был получен практический опыт в создании алгоритма проективных преобразований, вращения куба и отображение кабинетной проекции на основании математических операций с функциями и визуализации всего этого с использованием языка Python и его библиотек. Такой опыт может пригодиться для решения других задач в сфере информационных технологий, науке о данных и других областях, где используется вычислительное моделирование и визуализация, анимация посредство компьютерных мощностей.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Д. Роджерс – Математические основы машинной графики / Д. Роджерс, Дж. Адамс: Пер. с англ. – П.А. Монахова Г.В. Олохтоновой, Д.В. Волкова – М.: Мир, 2001. – 604с., ил.
2. Шабанов П.А. Научная графика в python [Электронный ресурс]. URL: https://github.com/whitehorn/Scientific_graphics_in_python (31.08.2015).
3. Аграновский А. В. Использование методов преобразования координат для формирования растровых изображений: учеб .- метод. Пособие / А. В. Аграновский. – СПб.: ГУАП, 2024. – 40 с.
4. Косоугольная диметрия | Начертательная геометрия - nGeo.FXYZ.ru – URL: https://ngeo.fxyz.аксонометрические_проекции/построение_аксонометрических_проекций/косоугольная_диметрия/ (дата обращения 27.10.2024)

ПРИЛОЖЕНИЕ А

Листинг Программы

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation

# Определяем вершины куба
def cube_vertices():
    r = 1
    return np.array([[r, r, r],
                     [-r, r, r],
                     [-r, -r, r],
                     [r, -r, r],
                     [r, r, -r],
                     [-r, r, -r],
                     [-r, -r, -r],
                     [r, -r, -r]])

# Матрица вращения вокруг оси X
def rotation_matrix_x(theta):
    c, s = np.cos(theta), np.sin(theta)
    return np.array([[1, 0, 0],
                     [0, c, -s],
                     [0, s, c]])

# Кабинетная проекция
def cabinet_projection(vertices):
    scale = 1  # Коэффициент масштабирования
    projection_matrix = np.array([[1, 0, -0.5],
                                  [0, 1, -0.5]])
    proj_vertices = vertices @ projection_matrix.T
    return proj_vertices * scale

# Анимация куба
fig = plt.figure()
ax = fig.add_subplot(111)
ax.set_xlim([-2, 2])
ax.set_ylim([-2, 2])

vertices = cube_vertices()
edges = [[0, 1], [1, 2], [2, 3], [3, 0],
         [4, 5], [5, 6], [6, 7], [7, 4],
         [0, 4], [1, 5], [2, 6], [3, 7]]

def update(frame):
    ax.cla()  # Очистка осей
    ax.set_xlim([-2, 2])
    ax.set_ylim([-2, 2])
    ax.set_xlabel('X-координата')
    ax.set_ylabel('Y-координата')

    # Вращение куба
    theta = frame * np.pi / 180
    rotated_vertices = vertices @ rotation_matrix_x(theta)

    # Применяем проекцию
    projected_vertices = cabinet_projection(rotated_vertices)

    # Рисуем куб
    for edge in edges:
        ax.plot(*zip(projected_vertices[edge[0]],
projected_vertices[edge[1]]), color='b')
```

```
# Анимация
ani = animation.FuncAnimation(fig, update, frames=np.arange(0, 360, 5),
interval=100)
plt.show()
```