ГУАП

КАФЕДРА № 42

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| канд. техн. наук, 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. № | 4326 |  |  |  | Г. С. Томчук |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2024

1. Цель работы

Цель данной работы заключается в исследовании теории проективных преобразований в трехмерном пространстве и практических аспектов их реализации.

1. Формулировка задания

Необходимо написать программу, реализующую получение анимированной вращающейся проекции куба, с двухточечной перспективой, ось вращения не параллельна координатным осям (вариант 20).

Куб должен вращаться вокруг указанной оси, не проходящей через сам многогранник и через начало координат, а также не совпадающей с координатными осями. Фигура должна отображаться в контурном виде без удаления невидимых линий. Рисование контура фигур по матрице координат вершин можно осуществлять с помощью специализированных библиотек. Аффинные и проективные преобразования необходимо выполнять только путем матричных вычислений. Использование специализированных программ геометрических преобразований не допускается.

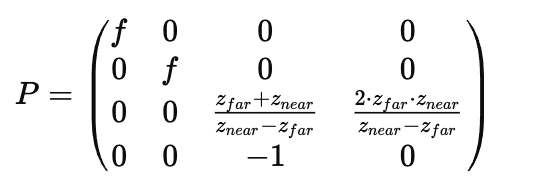
1. Теоретические сведения

**Проективные преобразования.** Проективные преобразования представляют собой обобщение аффинных преобразований, которые описывают взаимосвязь между объектами в проектировании и визуализации. Они играют важную роль в компьютерной графике и визуализации данных, позволяя моделировать трехмерные объекты на двумерных экранах.

Проективные преобразования могут быть описаны с помощью матриц, которые применяются к однородным координатам. В трехмерном пространстве проективные преобразования включают в себя следующие операции:

1. Трансляция — перемещение объектов в пространстве.
2. Вращение — изменение ориентации объекта вокруг заданной оси.
3. Масштабирование — изменение размеров объекта.
4. Параллельные и перспективные проекции — проекция трехмерных объектов на двумерную плоскость, что позволяет моделировать восприятие объектов в реальном мире.

**Проекционная матрица.** Проекционная матрица используется для преобразования координат трехмерных точек в двумерные. Для перспективной проекции важным аспектом является положение наблюдателя (камеры) и точки, на которые он направлен. Матрица проекции для перспективной проекции может быть представлена следующим образом:



где — фокусное расстояние, и — расстояния до ближайшей и дальней плоскостей отсечения.

**Аффинные преобразования.** Проективные преобразования могут быть разбиты на аффинные и перцептивные. Аффинные преобразования сохраняют коллинеарность, отношение длин и параллельность, но не сохраняют углы. В рамках аффинных преобразований основное внимание уделяется выполнению операций, таких как:

1. Сдвиги (трансляции)
2. Масштабирования
3. Вращения

Все аффинные преобразования также могут быть представлены в виде матриц, что позволяет легко комбинировать несколько преобразований в одно.

**Применение проективных преобразований.** Проективные преобразования имеют широкое применение в различных областях, включая:

* Компьютерную графику: для создания 3D-моделей и их визуализации.
* Виртуальную реальность: для создания интерактивных 3D-пространств.
* Геоинформационные системы: для проекции карт и анализа пространственных данных.

Проективные преобразования являются неотъемлемой частью работы с трехмерной графикой и играют важную роль в создании визуальных эффектов и моделировании реальных объектов.

1. Описание алгоритма решения

Для реализации анимации вращающегося куба с использованием двухточечной перспективной проекции, алгоритм решения состоит из нескольких ключевых шагов:

1. Инициализация параметров

* Определение вершин и рёбер куба: Задаются координаты восьми вершин куба и рёбер, которые соединяют эти вершины.
* Определение оси вращения: Указывается точка, через которую проходит ось вращения, и направление этой оси в трехмерном пространстве.

2. Определение матриц преобразования

* Матрица вращения: Разрабатывается функция, которая создает матрицу вращения вокруг заданной оси. Это включает использование формулы для вращения в 3D-пространстве, основанной на угле вращения.
* Матрица перспективной проекции: Создается матрица для двухточечной перспективной проекции, которая преобразует трехмерные координаты в двумерные, с учетом фокусного расстояния и положения камеры.

3. Применение преобразований к вершинам куба

* Смещение вершин: Вершины куба смещаются так, чтобы ось вращения находилась в центре координат для удобства вычислений.
* Вращение куба: Каждую итерацию анимации куб вращается на заданный угол, и к его вершинам применяется матрица вращения.
* Восстановление координат: После вращения осуществляется обратное смещение, чтобы вернуть куб в его исходное положение.

4. Проекция и отрисовка

* Проекция вершин: Применяется матрица перспективной проекции к вращающимся вершинам куба, что позволяет получить их двумерные координаты.
* Отрисовка рёбер: На основе проекций вершин рисуются рёбра куба. Для этого используется функция рисования в библиотеке для графики, которая соединяет пары вершин рёбер.

5. Анимация

* Цикл анимации: Используется механизм анимации, который обновляет графическое представление куба на каждом кадре. Для этого задается количество кадров и временной интервал между ними.
* Очистка графика: В каждом обновлении графика происходит очистка предыдущего кадра, чтобы обеспечить корректное отображение текущего состояния куба.

6. Завершение работы: После завершения анимации программа корректно завершает свою работу, освобождая используемые ресурсы и закрывая окно.

1. Выбор языка программирования и используемых библиотек

Для написания программы я выбрал язык программирования Python. Данный язык имеет богатую экосистему библиотек, в том числе и необходимых для работы с отрисовкой и графикой. Помимо этого, я уже имел опыт работы с данным языком, что ускорило и облегчило рабочий процесс.

Используемые библиотеки:

* NumPy — используется для работы с матрицами и векторами, что упрощает вычисления аффинных преобразований, таких как поворот, сдвиг и масштабирование. Это стандартная библиотека для эффективных математических операций в Python.
* Matplotlib — отвечает за визуализацию и анимацию.

Обе библиотеки широко используются в научных и инженерных вычислениях, что делает их подходящими для задач, связанных с компьютерной графикой и аффинными преобразованиями.

1. Описание разработанной программы

Программа (см. Приложение А) предназначена для визуализации трехмерного вращающегося куба с использованием проективных преобразований. Она позволяет наблюдать эффект двухточечной перспективы, что создает реалистичное восприятие трехмерного объекта на двумерном экране. Основные компоненты и функциональность программы следующие:

1. **Инициализация среды.** Программа использует библиотеку matplotlib для создания графического интерфейса и отрисовки объектов. В начале работы инициализируются необходимые параметры, такие как размеры графика, диапазоны осей координат и начальные координаты вершин куба.

2. **Определение геометрии куба.** Куб описывается как набор из восьми вершин, заданных в трехмерном пространстве, и рёбер, которые соединяют эти вершины. Каждый рёбер представлен как список индексов вершин, что позволяет легко отрисовывать его на экране.

3. **Ось вращения.** В программе задается ось вращения, которая определяет, вокруг какой линии будет происходить вращение куба. Эта ось не проходит через сам куб, что позволяет создавать визуальный эффект вращения вокруг объекта.

4. **Проективные преобразования.** Разработана функция для создания матрицы перспективной проекции, которая преобразует трехмерные координаты в двумерные. Это достигается с помощью матричных операций, что позволяет использовать стандартные методы линейной алгебры для вычислений.

5. **Анимация вращения.** Программа реализует анимацию, которая обновляет положение куба на каждом кадре. В каждую итерацию происходит:

1. Вращение: Куб вращается вокруг заданной оси на небольшой угол, определяемый текущим кадром.
2. Проекция: Вершины куба проецируются на двумерную плоскость с использованием матрицы проекции.
3. Отрисовка: Рёбра куба отрисовываются на графике, создавая эффект трехмерного объекта.

6. **Графический интерфейс.** Для отображения куба и его анимации используется matplotlib. Программа создает окно, в котором производится отрисовка. Пользователь может наблюдать за процессом вращения куба с заданным уровнем детализации и плавности.

7. **Возможность изменения параметров.** Программа предоставляет возможность настраивать параметры, такие как скорость вращения, размеры куба и положение оси вращения. Это позволяет пользователю исследовать различные визуальные эффекты и лучше понять работу проективных преобразований.

1. Скриншоты, иллюстрирующие результаты работы программы

На рис. 1, 2 представлены скриншоты окна программы во время отрисовки двух разных кадров.

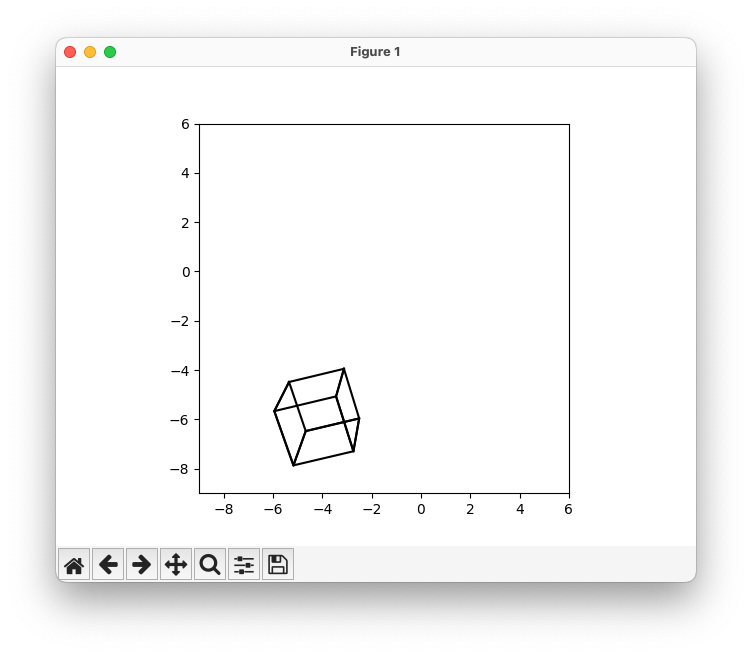


Рисунок 1

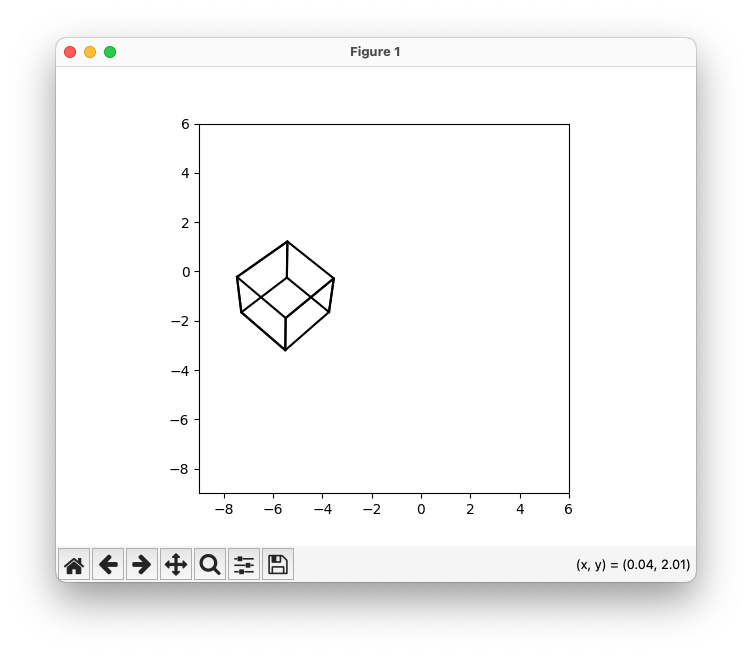


Рисунок 2

На рис. 3 изображен первый кадр анимации (куб в изначальном положении).

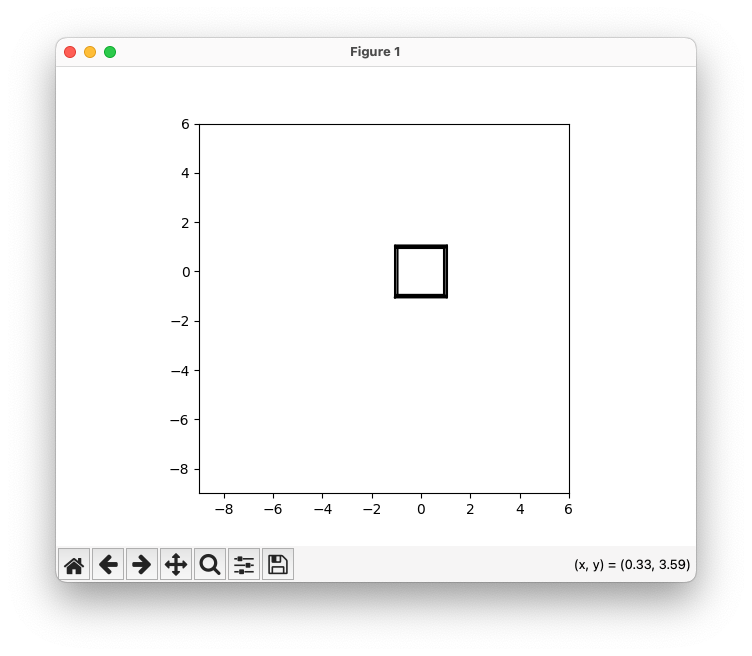


Рисунок 3

1. Вывод

В результате выполнения данной работы была успешно разработана программа для визуализации анимации вращающегося куба с использованием двухточечной перспективной проекции. Работа позволила не только углубить теоретические знания о проективных преобразованиях, но и приобрести практические навыки в их реализации с помощью программирования на Python.

В процессе работы была достигнута цель — создание наглядной анимации, позволяющей наблюдать эффекты проекции трехмерных объектов на двумерную плоскость. Реализация программы включает:

* Определение и визуализацию геометрии куба с рёбрами и вершинами.
* Применение матричных преобразований для вращения объекта вокруг оси, которая не проходит через сам куб, что добавляет реалистичности анимации.
* Реализацию двухточечной перспективной проекции, позволяющей создать эффект глубины и трехмерности на плоском экране.

В процессе выполнения работы были получены следующие знания и навыки:

1. Углубленное понимание проективных преобразований и их применения в компьютерной графике.
2. Освоение методов линейной алгебры, необходимых для работы с матрицами и векторными преобразованиями.
3. Практические навыки работы с библиотекой matplotlib для создания графических интерфейсов и визуализации данных.
4. Опыт решения задач, связанных с анимацией и графическим представлением объектов, что имеет большое значение для дальнейшей работы в области компьютерной графики и программирования.

В процессе разработки программы возникли определенные проблемы, которые требовали внимания и анализа:

1. Первоначально некоторые рёбра пропадали из-за неправильной реализации алгоритма отрисовки. Эта проблема была решена путем тщательной проверки индексов вершин и улучшения логики проекции.
2. На начальных этапах ось вращения вращалась вместе с кубом, что не соответствовало поставленной задаче. Этот момент был исправлен путём разделения логики вращения куба и отрисовки оси, что позволило добиться нужного эффекта.
3. Изменение масштаба и размера куба требовало доработки, чтобы все элементы были хорошо видны на экране. В результате была установлена оптимальная система координат и пропорции для корректного отображения.

В результате преодоления этих проблем была создана стабильная программа, которая выполняет поставленную задачу и демонстрирует знания о проективных преобразованиях и методах графического программирования.

Таким образом, работа не только позволила реализовать проект, связанный с визуализацией трехмерных объектов, но и способствовала развитию критического мышления и навыков решения проблем. Полученные знания и опыт будут полезны в будущих проектах и научных исследованиях.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Аграновский, А. В. Использование методов преобразования координат для формирования растровых изображений. Учебно-методическое пособие / А. В. Аграновский. — СПб.: Редакционно-издательский центр ГУАП, 2024. — 40 с.
2. Петров, А. Н., Смирнов, И. В. Основы программирования на Python: Учебное пособие. – Санкт-Петербург: Издательство СПб ГУАП, 2022. – 320 с.
3. Бобылев, С. И. Основы компьютерной графики: учебное пособие / С. И. Бобылев. — М.: Издательство МГТУ им. Н. Э. Баумана, 2010. — 224 с.
4. Ласков, А. В. Математические основы компьютерной графики / А. В. Ласков, Ю. Г. Лоскутов. — СПб.: Питер, 2008. — 352 с.
5. NumPy. Documentation. URL: https://numpy.org/doc/stable/ (дата обращения: 27.10.2024).
6. Matplotlib. Documentation. URL: https://matplotlib.org/stable/contents.html (дата обращения: 27.10.2024).

ПРИЛОЖЕНИЕ A

import numpy as np  
import matplotlib.pyplot as plt  
from matplotlib.animation import FuncAnimation  
  
*# Вершины куба*cube\_vertices = np.array([  
 [-1, -1, -1], [1, -1, -1], *# Нижняя грань* [1, 1, -1], [-1, 1, -1], *# Верхняя грань* [-1, -1, 1], [1, -1, 1], *# Нижняя грань* [1, 1, 1], [-1, 1, 1] *# Верхняя грань*])  
  
*# Грани куба определяются с помощью рёбер*cube\_edges = [  
 [cube\_vertices[0], cube\_vertices[1], cube\_vertices[2], cube\_vertices[3]], *# Нижняя грань* [cube\_vertices[4], cube\_vertices[5], cube\_vertices[6], cube\_vertices[7]], *# Верхняя грань* [cube\_vertices[0], cube\_vertices[1], cube\_vertices[5], cube\_vertices[4]], *# Боковая грань* [cube\_vertices[2], cube\_vertices[3], cube\_vertices[7], cube\_vertices[6]], *# Боковая грань* [cube\_vertices[1], cube\_vertices[2], cube\_vertices[6], cube\_vertices[5]], *# Боковая грань* [cube\_vertices[4], cube\_vertices[7], cube\_vertices[3], cube\_vertices[0]], *# Боковая грань* [cube\_vertices[0], cube\_vertices[4]] *# Дополнительное ребро*]  
  
*# Параметры оси вращения*rotation\_axis\_start = np.array([2, 2, 2]) *# Начальная точка оси вращения*rotation\_axis\_direction = np.array([1, 0, -1]) *# Направление оси вращения  
  
# Матрица проекции для двухточечной перспективы*def perspective\_projection\_matrix(d=10):  
 *# Возвращает матрицу проекции для двухточечной перспективы* return np.array([  
 [1, 0, 0, 0],  
 [0, 1, 0, 0],  
 [0, 0, 1, -1/d], *# Удаление глубины* [0, 0, 0.5/d, 1] *# Применение коэффициента перспективы* ])  
  
*# Функция создания матрицы вращения вокруг произвольной оси*def rotation\_matrix(axis, theta):  
 *# Нормализация оси вращения* axis = axis / np.linalg.norm(axis)  
 a = np.cos(theta / 2.0)  
 b, c, d = -axis \* np.sin(theta / 2.0)  
 *# Возвращает матрицу вращения* return np.array([  
 [a\*a + b\*b - c\*c - d\*d, 2\*(b\*c + a\*d), 2\*(b\*d - a\*c), 0],  
 [2\*(b\*c - a\*d), a\*a + c\*c - b\*b - d\*d, 2\*(c\*d + a\*b), 0],  
 [2\*(b\*d + a\*c), 2\*(c\*d - a\*b), a\*a + d\*d - b\*b - c\*c, 0],  
 [0, 0, 0, 1] *# Четвертая строка и столбец для однородных координат* ])  
  
*# Применение вращения и проекции*def transform(vertices, rotation, projection):  
 *# Вращение вершин куба* rotated = np.dot(vertices, rotation[:3, :3].T)  
 *# Проекция на двумерную плоскость* projected = np.dot(np.c\_[rotated, np.ones(len(rotated))], projection.T)  
 *# Нормализация координат* projected = projected / projected[:, -1:]  
 return projected[:, :2] *# Возвращаем только x и y  
  
# Настройка графика*fig, ax = plt.subplots()  
ax.set\_aspect('equal') *# Сохраняем пропорции*ax.set\_xlim(-9, 6) *# Установка пределов по оси x*ax.set\_ylim(-9, 6) *# Установка пределов по оси y  
  
# Функция для обновления анимации*def update(frame):  
 ax.cla() *# Очистка текущего кадра* ax.set\_aspect('equal') *# Сохранение пропорций* ax.set\_xlim(-9, 6) *# Пределы по оси x* ax.set\_ylim(-9, 6) *# Пределы по оси y  
  
 # Создание матрицы вращения на основе текущего кадра* rotation = rotation\_matrix(rotation\_axis\_direction, frame)  
 *# Смещение вершин перед вращением* shifted\_vertices = cube\_vertices + rotation\_axis\_start  
 *# Вращение смещенных вершин* rotated\_vertices = np.dot(shifted\_vertices, rotation[:3, :3].T)  
 *# Обратное смещение для восстановления положения* restored\_vertices = rotated\_vertices - rotation\_axis\_start  
  
 *# Применение проекции к восстановленным вершинам* projected\_vertices = transform(restored\_vertices, np.eye(4), perspective\_projection\_matrix())  
  
 *# Отрисовка куба* for edge in cube\_edges:  
 *# Проекция и отрисовка каждого ребра куба* edge\_proj = [projected\_vertices[cube\_vertices.tolist().index(list(vertex))] for vertex in edge]  
 ax.plot(\*zip(\*edge\_proj), color='black') *# Рисование ребер в черном цвете  
  
# Создание анимации*ani = FuncAnimation(fig, update, frames=np.linspace(0, 2 \* np.pi, 120), interval=50, repeat=True)  
plt.show() *# Отображение графика*