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

Исследование аффинных преобразований в пространстве

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 линейных геометрических преобразований в трехмерном пространстве. Исследование особенностей практической реализации трехмерных аффинных преобразований, теоретических основ линейных геометрических преобразований на плоскости.

2. Задание:

Написать программу на любом языке высокого уровня, реализующую преобразование фигуры в трехмерном пространстве в соответствии с вариантом задания - 17 - Плавная деформация куба путем растяжения (сжатия) с неодинаковыми коэффициентами по координатным осям. Начало координат находится вне куба.

Результат преобразований представляется в экранных координатах. 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 преобразования необходимо выполнять только путем комбинации рассмотренных выше матричных вычислений (можно, но не обязательно, использовать библиотеки программ для матричных вычислений). Использование созданных другими авторами программ аффинных преобразований не допускается.

3. Теоретические сведения:

3.1 Аффинные преобразования в общем

Аффинные преобразования – основные типы линейных преобразований – масштабирование, поворот, перенос, отражение относительно оси. Также отображение плоскости или пространства в себя, при котором параллельные прямые переходят в параллельные прямые, пересекающиеся — в пересекающиеся, скрещивающиеся — в скрещивающиеся.

Отображение в себя – значит, что если мы находились в пространстве R^n, то после образования мы должны остаться в нем же. Например: если мы применили какое-то преобразование к прямоугольнику и получили параллелепипед, то мы вышли из R^2 в R^3. А вот если из прямоугольника у нас получился другой прямоугольник, то все хорошо, мы отобразили исходное пространство в себя. Формально это описывается так: «преобразование f отображает пространство R^n в R^n». Если записать с помощью формул: $f: R^n \to R^n$.

Скрещивающиеся прямые – не лежат в одной плоскости. Мы должны остаться в той же плоскости: значит мы представляем себе 2D декартову систему координат. Здесь

речь идет о нескольких прямых, так что давайте представим 2 параллельных линии. Из определения мы понимаем, что после преобразования эти линии должны остаться параллельными. Просто сдвигаем их куда-нибудь из исходного местоположения. (один из видов аффинных преобразований – сдвиг)

Преобразование плоскости называется аффинным, если оно непрерывно, взаимно однозначно и образом любой прямой является прямая.

Преобразование называется непрерывным, если «близкие точки переходят в близкие». Т.е. иначе - если у нас есть две точки и они находятся рядом, то после преобразования они все равно будут находиться где-то поблизости друг от друга.

Далее - преобразование взаимооднозначно, если разные точки переводятся в разные точки и в каждую точку переводится какая-то точка. Например: если мы отобразили отрезок и он слипся в точку - это не взаимооднозначное преобразование. Из отрезка мы должны получить ровно такой же отрезок, тогда будет взаимооднозначно (если это сработает для всех отрезков, конечно).

Пусть у нас есть исходная система координат. Точка в этой системе характеризуется двумя числами - x и y. Совершить переход к новым координатам x' и y'мы можем с помощью следующей системы:

$$\begin{cases} x' = \alpha x + \beta y + \lambda \\ y' = \gamma x + \delta y + \mu \end{cases}$$

Рисунок 1.1 Система координат

При этом, числа $\alpha, \beta, \gamma, \mu$ должны образовывать невырожденную матрицу:

$$\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}$$

Рисунок 1.2 Невырожденная матрица

Матрица называется невырожденной, если ее определитель не равен нулю, т.е.

$$\begin{vmatrix} \alpha & \beta \\ \gamma & \delta \end{vmatrix} \neq 0$$

Рисунок 1.3 Определитель матрицы равен 0

Можно записать и в более общем в виде.

Аффинное преобразование $f: R^n \rightarrow R^n$- преобразование вида $f(x) = Mx + v$, где M- обратимая матрица, а $v \in R^n$. В данном случае x, само собой, n-мерный вектор.

3.2. Композиция аффинных преобразований на плоскости

Математически сложное аффинное преобразование выражается через последовательное умножение матриц, представляющих каждое простое преобразование. В результате получается итоговая матрица преобразований.

Суперпозиция аффинных преобразований сохраняет прямые линии, отношения длин отрезков на одной прямой или параллельных прямых, а также отношения площадей фигур. Также она гарантирует, что параллельные прямые останутся параллельными.

Важно отметить, что все рассмотренные преобразования выполняются относительно начала координат или координатной оси. Если необходимо выполнить аналогичные преобразования относительно другой точки или прямой, сначала нужно совместить их с началом координат или осью с помощью сдвига и, возможно, поворота.

3.3. Создание анимированных изображений

При создании анимированных изображений с помощью аффинных преобразований важно учитывать, что для плавного движения фигур необходимо, чтобы частота смены кадров была не менее 24 кадров в секунду. Если продолжительность анимации составляет T секунд, то для достижения этой частоты преобразование нужно разделить на 24*T итераций. Каждая итерация должна занимать 1/24 секунды.

Можно использовать метод двойной буферизации кадра. При этом изображение выводится на экран из одного буфера, а расчет нового кадра происходит в другом буфере. Затем буферы меняются местами с нужной частотой кадров. При использовании двойной буферизации рекомендуется синхронизировать смену кадров с началом каждой новой итерации. Также важно помнить об удалении фигуры с предыдущей позиции перед началом следующей итерации.

3.4 Особенности применения аффинных преобразований на плоскости

Любое аффинное преобразование имеет обратное преобразование, которое также является аффинным и описывается обратной матрицей. Произведение прямой и обратной матриц дает единичную матрицу, то есть все пиксели остаются на своих местах.

Однако, математически точное описание аффинных преобразований для обработки цифровых изображений затруднительно из-за их растровой природы. Большинство преобразований могут привести к нецелым значениям x' и y', которые нужно округлять. При повороте изображения могут возникать неопределенные пиксели, которые не преобразуются ни в один другой пиксель.

Эти проблемы не возникают при преобразованиях, где каждый пиксель отображается в один пиксель (например, отражение относительно оси или поворот на 90

или 180 градусов). В остальных случаях приходится использовать более сложные алгоритмы, такие как методы интерполяции яркости пикселя или алгоритм поворота растровых изображений Оуэна и Македона.

3.5 Особенности применения аффинных преобразований в трёхмерном пространстве.

Аффинные преобразования в трёхмерном пространстве сложнее, чем на плоскости. Чтобы показать их на экране компьютера, нужно сделать двумерное изображение трёхмерной модели.

Как и в двумерном пространстве, любое аффинное преобразование в трёхмерном можно представить как комбинацию вращений, растяжений, отражений и переносов.

Точка в трёхмерном пространстве (x, y, z) с помощью нормированных однородных координат записывается как [x, y, z, 1].

Аффинное преобразование такой точки можно описать так:

[x', y', z', 1] = [x, y, z, 1] * T, где матрица T имеет размер 4x4.

Чтобы выполнить сложное изменение формы, его делят на несколько простых шагов, как и в случае с двумерными фигурами. При этом нужно помнить, что простые изменения формы в трёхмерном пространстве связаны с началом координат, осями или плоскостями.

Можно выполнять изменения формы последовательно или сначала посчитать итоговую таблицу изменений.

В трёхмерном пространстве можно менять форму некоторых фигур. Если это многогранник, то проще всего задать его как таблицу с координатами вершин и список рёбер. Например, куб, одна из вершин которого находится в начале координат, можно задать таблицей с координатами.

Описание этой фигуры определяется набором рёбер, который содержит все пары точек, соединённых рёбрами. Номера, указанные в этом наборе, соответствуют порядку координат соответствующих точек в матрице однородных координат многоугольника.

Для упрощения создания исходной трёхмерной фигуры иногда будут использоваться так называемые платоновы тела — правильные многогранники, состоящие из одинаковых правильных многоугольников и обладающие пространственной симметрией.

Платоновы тела имеют следующие характеристики:

1. Они являются выпуклыми, то есть все точки на их поверхности расположены по одну сторону от плоскости каждого плоского многоугольника.

2. Все грани — это одинаковые правильные многоугольники.
3. В каждой вершине сходится одинаковое количество рёбер.

Количество вершин (В), граней (Г) и рёбер (Р) любого платонова тела связано формулой Эйлера: В + Г = Р + 2.

Доказано, что в трёхмерном пространстве существует всего пять платоновых тел, которые названы в соответствии с количеством граней: тетраэдр, гексаэдр (куб), октаэдр, додекаэдр и икосаэдр.

Свойства платоновых тел

Название многогранника	Число вершин	Число ребер	Число граней	Число сторон у грани	Число ребер, примыкающих к вершине
Тетраэдр	4	6	4	3	3
Октаэдр	6	12	8	3	4
Гексаэдр	8	12	6	4	3
Икосаэдр	12	30	20	3	5
Додекаэдр	20	30	12	5	3

Рисунок 1.4 – Свойства платоновых тел

4. Алгоритм преобразований:

4.1. Определение исходной геометрии:

Начнем с определения восьми вершин исходного куба. Эти вершины могут быть представлены в виде координат, где каждая координата включает в себя x, y и z.

4.2. Составление матрицы преобразования:

Аффинные преобразования в 3D описываются с помощью матрицы 4x4. Она включает в себя трансляцию, масштабирование и вращение. Для масштабирования используем диагональную матрицу, где Sx , Sy , и Sz — коэффициенты масштабирования по осям X, Y и Z соответственно.

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

Рисунок 2.1 – Матрица для аффинных преобразований в 3D

4.3 Моделирование деформации:

Для создания анимации деформации куба, мы должны динамически изменять коэффициенты масштабирования. Например, можно использовать линейную

интерполяцию между начальными и конечными значениями масштабов, аналогично для Sy(t) и Sz(t).

$$s_x(t) = s_{x_{\text{start}}} + t \cdot (s_{x_{\text{end}}} - s_{x_{\text{start}}})$$

Рисунок 2.2 – линейная интерполяция

4.4 Применение матрицы преобразования:

Каждая вершина куба представляется в виде однородного вектора [x,y,z,1]T . Для применения аффинного преобразования к вершинам куба, мы умножаем матрицу преобразования M на вектор каждой вершины:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = M \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Рисунок 2.3 – Применение матрицы преобразования

Это позволит нам получить новые координаты деформированного куба.

4.5 Визуализация результата:

После применения матрицы преобразования к вершинам куба, мы строим рёбра, соединяющие новые позиции вершин.

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.

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. Создание куба:

Определяется функция create_cube, которая генерирует координаты вершин куба определенного размера. Для этого в функции используются трехмерные координаты, которые затем преобразуются в двумерный массив.

3. Деформация куба:

В функции deform_cube описывается, как куб может изменять свои размеры. Куб деформируется на основе входных коэффициентов масштабирования, что достигается путём умножения координат куба на эти коэффициенты.

4. Определение рёбер куба:

Создается список пар индексов, представляющих рёбра куба, чтобы можно было визуализировать его структуру.

5. Создание фигуры:

После этого создаются графические элементы. Создается фигура и трехмерные оси, к которым будет добавляться анимация.

6. Функция анимации:

Основная логика анимации реализована в функции animate, которая обновляет отображение куба в зависимости от текущего кадра. Здесь происходит линейная интерполяция для вычисления текущего масштаба.

7. Отрисовка рёбер:

После деформации куба в цикле отрисовываются его рёбра с помощью координат вершин, обеспечивая визуализацию формы.

8. Настройки и создание анимации:

Настраиваются начальные и конечные размеры куба, количество кадров и запускается анимация, финализируя создание динамического отображения деформации куба.

7. Скриншоты, иллюстрирующие результаты работы программы:

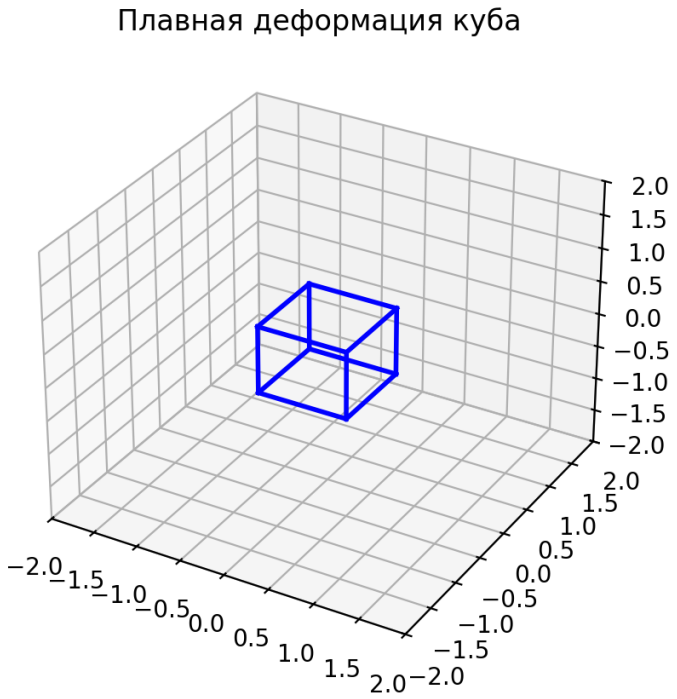

Рис 3.1. Изображение куба, прошло 0 секунд.

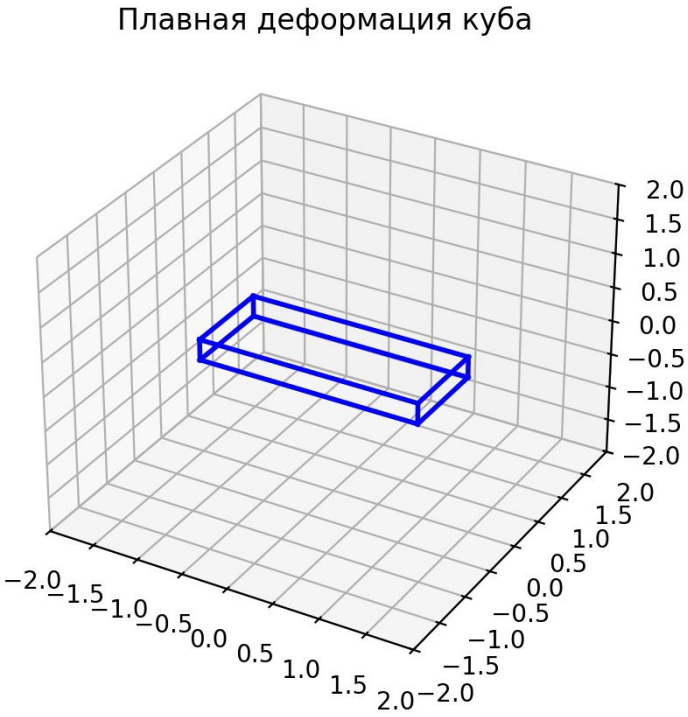

Рис 3.2. Изображение деформированного куба путём растяжения, прошло 7 секунд.

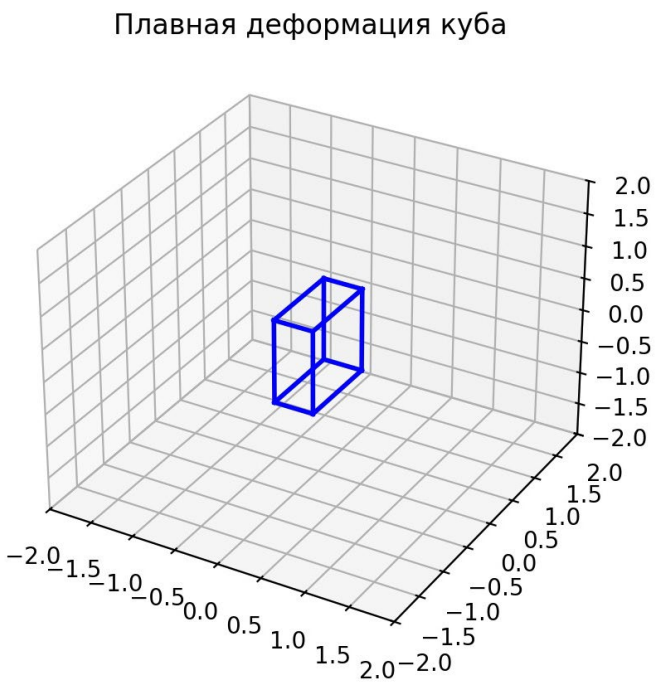

Рис 3.3. Изображение деформированного куба путём сжатия, прошло 7 секунд.

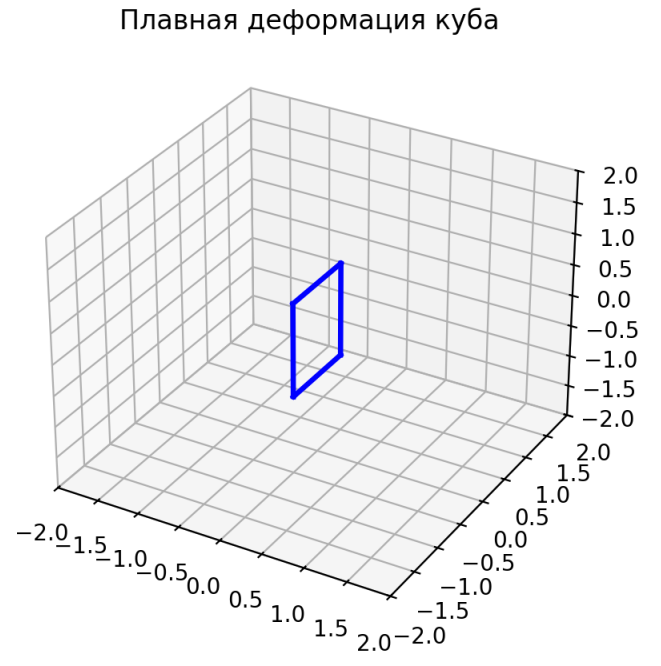

Рис 3.4. Изображение деформированного куба путём сжатия, прошло 12 секунд.

8. Вывод:

8.1 Полученные теоретические знания и навыки:

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

1) Аффинные преобразования:

Были изучены основы линейных геометрических преобразований в пространстве, особенности практической реализации трёхмерных аффинных преобразований их виды(поворот, сдвиг, растяжение-сжатие), главным образом - линейная интерполяция. 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 деформации куба и визуализации 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).

4) Вставка формул в Microsoft Word

В процессе написания отчёта была выявлена необходимость вставлять формулы адекватно. Потому пару минут было затрачено на получение этого навыка, хотя бы в базовом виде. Пригодится в дальнейших работах.

8.2 Возникшие проблемы и пути их решения:

В процессе выполнения работы возникали определенные трудности, а именно:

1) Визуализация только вершин куба, не прорисованные рёбра.

Проблема возникала в связи с использованием нерабочего метода scatter. Пришлось отдельно добавить массив с рёбрами куба – отрезками, и также как к вершинам

применять функцию деформации, а затем отрисовывать уже не с помощью scatter, а используя Line3DCollection.

2) 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.

В результате выполнения этой работы, был получен практический опыт в создании алгоритма деформации куба на основании математических операций с функциями, аффинных преобразований в 3D 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. Что за зверь — аффинные преобразования? – URL: https://habr.com/ru/articles/539420/ (дата обращения 28.09.2024)
2. 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.
3. Шабанов П.А. Научная графика в python [Электронный ресурс]. URL: https://github.com/whitehorn/Scientific_graphics_in_python (31.08.2015).
4. Аграновский А. В. Использование методов преобразования координат для формирования растровых изображений: учеб .- метод. Пособие / А. В. Аграновский. – СПб.: ГУАП, 2024. – 40 с.
5. Компьютерная графика :: Теория 3D :: Аффинные преобразования пространства – URL: https://compgraphics.info/3D/3d_affine_transformations.php (дата обращения 12.10.2024)

ПРИЛОЖЕНИЕ А

Листинг Программы

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

# Функция для создания куба
def create_cube(size):
    r = [-size / 2, size / 2]
    X, Y, Z = np.meshgrid(r, r, r)
    return np.array([X.flatten(), Y.flatten(), Z.flatten()])

# Функция для деформации куба
def deform_cube(cube_points, scale_factors):
    return cube_points * scale_factors[:, np.newaxis]

# Определение рёбер куба
edges = [
    [0, 1], [0, 2], [0, 4],
    [1, 3], [1, 5], [2, 3],
    [2, 6], [3, 7], [4, 5],
    [4, 6], [5, 7], [6, 7]
]

# Создание фигуры
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Функция анимации
def animate(i):
    ax.clear()
    t = i / frames
    scale_factors = (scale_start + t * (scale_end - scale_start))  # Линейная
интерполяция масштабов для растяжения
    #scale_factors = (#scale_end + t * (scale_start - scale_end))  # Линейная
интерполяция масштабов для сжатия
    # просто закомментить ненужное

    # Деформация куба
    deformed_cube = deform_cube(cube, scale_factors)

    # Рисуем рёбра куба
    for start, end in edges:
        xs = [deformed_cube[0, start], deformed_cube[0, end]]
        ys = [deformed_cube[1, start], deformed_cube[1, end]]
        zs = [deformed_cube[2, start], deformed_cube[2, end]]
        ax.plot(xs, ys, zs, 'b', linewidth=2)

    ax.set_xlim([-2, 2])
    ax.set_ylim([-2, 2])
    ax.set_zlim([-2, 2])
    ax.set_title("Плавная деформация куба")

# Настройки
cube_size = 1
scale_start = np.array([1, 1, 1])
scale_end = np.array([2, 1, 0.5])
frames = 100

# Куб в исходных координатах
cube = create_cube(cube_size)
```

```
# Анимация
ani = animation.FuncAnimation(fig, animate, frames=1024, interval=50)
plt.show()
```