Пациент: Румянцев Алексей Александрович

Группа: R3241

Поток: Прак. Лин. Ал. 1.3

Номер ИСУ: 368731

Лечащий врач: Перегудин Алексей Алексеевич

Диагноз: Отсутствие знаний по Практической Линейной Алгебре

Назначенное лечение: Выполнение лабораторной работы №3

!

Перед началом проверки отчета по пройденному курсу лечения, предлагаю вам посетить мой “репозиторий”, чтобы в процессе проверки вы могли наслаждаться (или нет) анимациями, которые я написал на языке программирования python, используя библиотеку manim. Посмотреть код можно тут (не стоит)

!

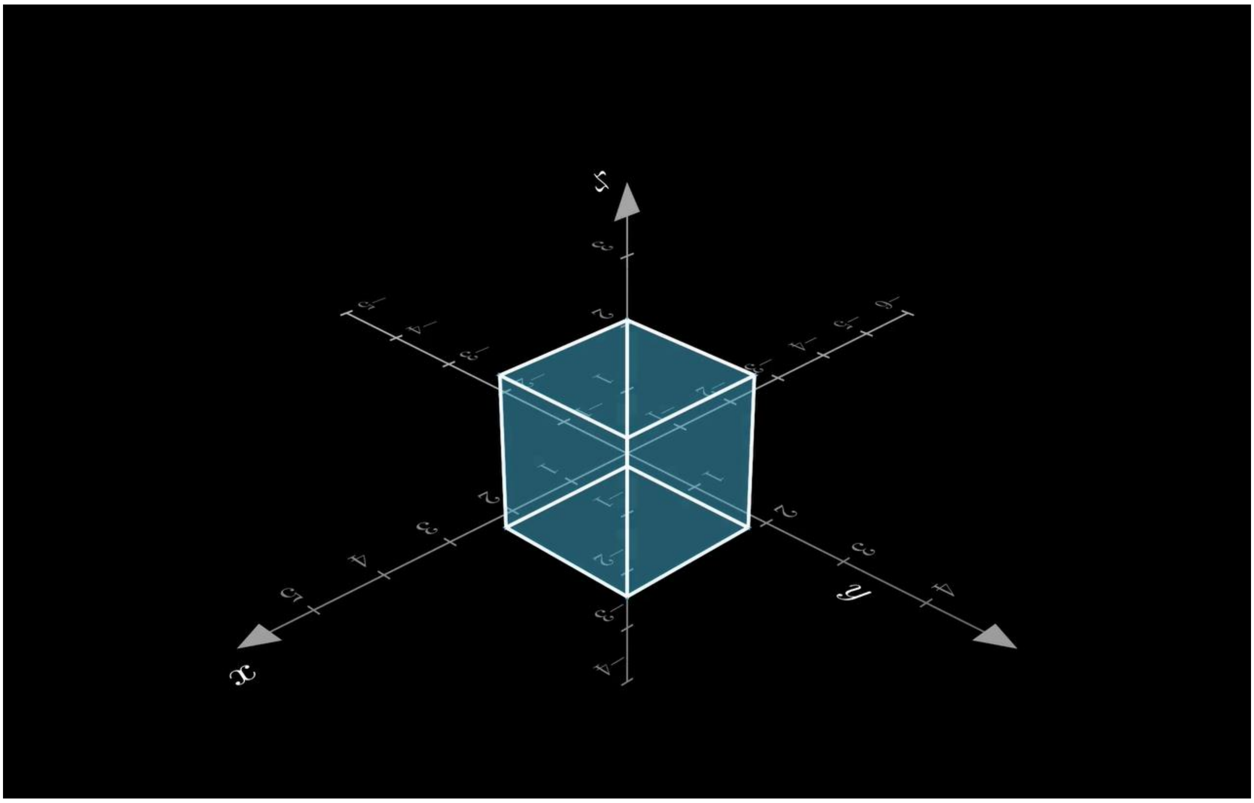
Задание 1

Создадим простой кубик, используя матрицу с координатами вершин “*default_vertices*”, матрицу ребер “*edges*” с номерами вершин, которые необходимо соединить, и матрицу “*faces*” с «будущими плоскостями» с координатами своих вершин (угловых точек) для составления граней

```
default_vertices = np.array([
    [-1,  1,  1, -1, -1,  1,  1, -1],
    [1,  1, -1, -1,  1, 1, -1, -1],
    [1,  1,  1,  1, -1, -1, -1, -1],
    [1, 1, 1, 1, 1, 1, 1, 1]
])

faces = [
    [vertices[0], vertices[1], vertices[2], vertices[3]],
    [vertices[4], vertices[5], vertices[6], vertices[7]],
    [vertices[0], vertices[1], vertices[5], vertices[4]],
    [vertices[2], vertices[3], vertices[7], vertices[6]],
    [vertices[0], vertices[3], vertices[7], vertices[4]],
    [vertices[1], vertices[2], vertices[6], vertices[5]]
]

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)
]
```

P. S. Конкретно создание куба можно посмотреть в коде по ссылке на титульном листе

Нетрудными преобразованиями получим следующее чудо линейной алгебры:



Для отрисовки координатных осей и постановки камеры поду углом я написал следующий код (конечно же, чтобы переиспользовать его и для других сцен)

```
def add_axes_to_scene(scene, phi, theta):
    axes = ThreeDAxes().add_coordinates()
    axes.set_color(GRAY).add(axes.get_axis_labels())
    scene.set_camera_orientation(phi=phi * DEGREES, theta=theta * DEGREES)
    scene.add(axes)
```

Четырехкомпонентный вектор используется для того, чтобы была возможность использовать матрицу перемещения (или трансляции) для задания движения объекта в пространстве при ее воздействии на этот объект. Немного позже будут рассмотрены примеры действия таких матриц

Другие фигуры можно задать, поменяв списки вершин, ребер и граней на соответствующие для нового объекта. В одном из заданий я покажу превращение куба в параллелепипед, однако задать можно и любые другие фигуры

Задание 2

Чтобы увеличить или уменьшить кубик, воспользуемся матрицами масштабирования, которыми подействуем на матрицу координат вершин кубика

```
scale_matrix_A = np.array([
    [0.5, 0, 0, 0],
    [0, 0.5, 0, 0],
    [0, 0, 0.5, 0],
    [0, 0, 0, 0.5]
])

scale_matrix_B = np.array([
    [2, 0, 0, 0],
    [0, 2, 0, 0],
    [0, 0, 2, 0],
    [0, 0, 0, 2]
])

scale_matrix_C = np.array([
    [1.5, 0, 0, 0],
    [0, 1.5, 0, 0],
    [0, 0, 0.7, 0],
    [0, 0, 0, 0.7]
])
```

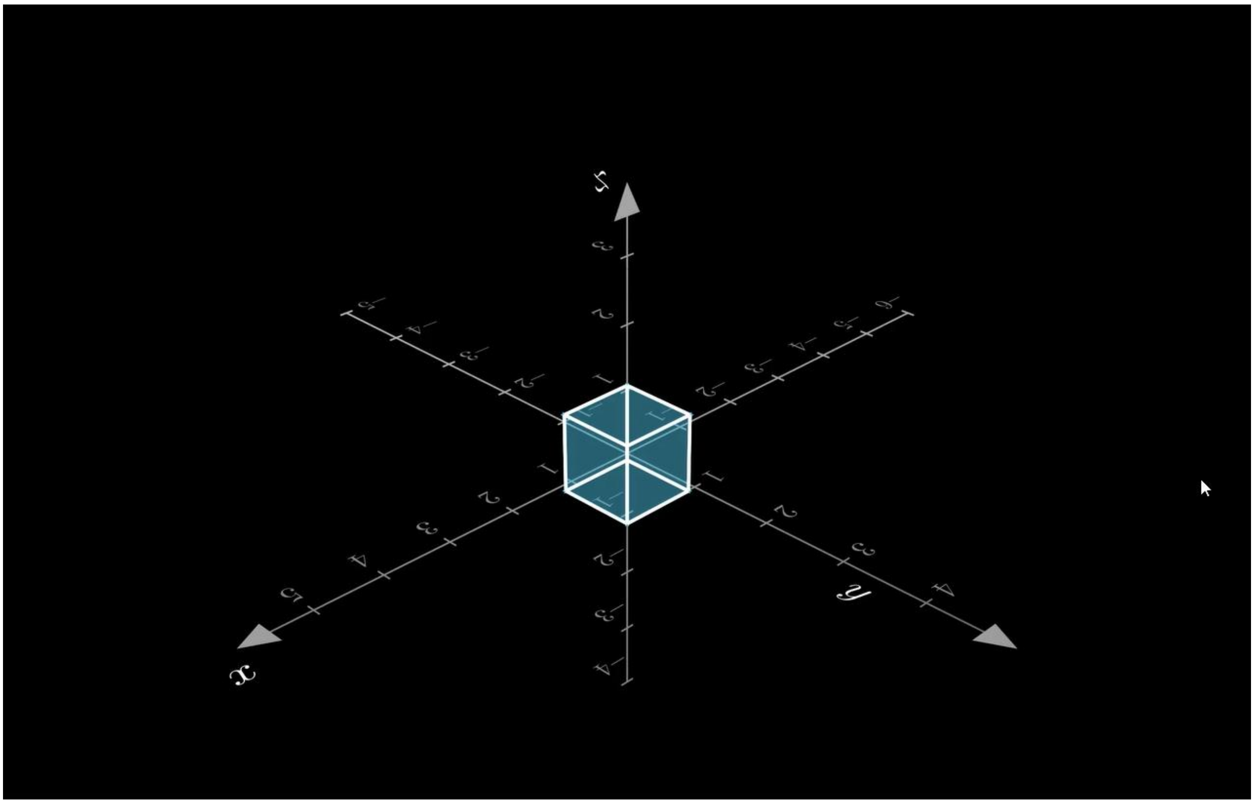
Матрица “A” при воздействии на матрицу координат вершин кубика уменьшит его по всем единичным векторам

Матрица “B” увеличит кубик в два раза по всем единичным векторам

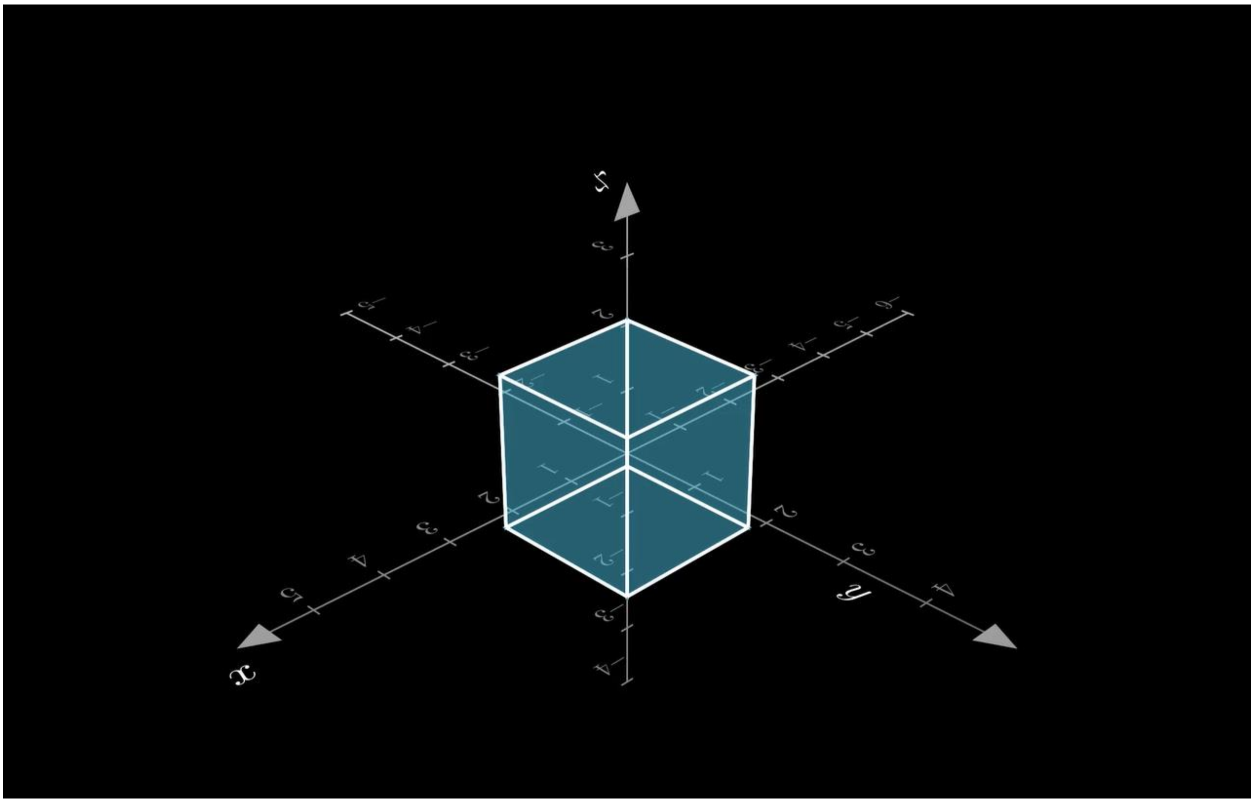
А вот матрица “C” превратит наш кубик в параллелепипед, так как масштабирует объект в не равной степени

Скриншоты отрисовок находятся на следующей странице (изначальный кубик можно посмотреть в первом задании)

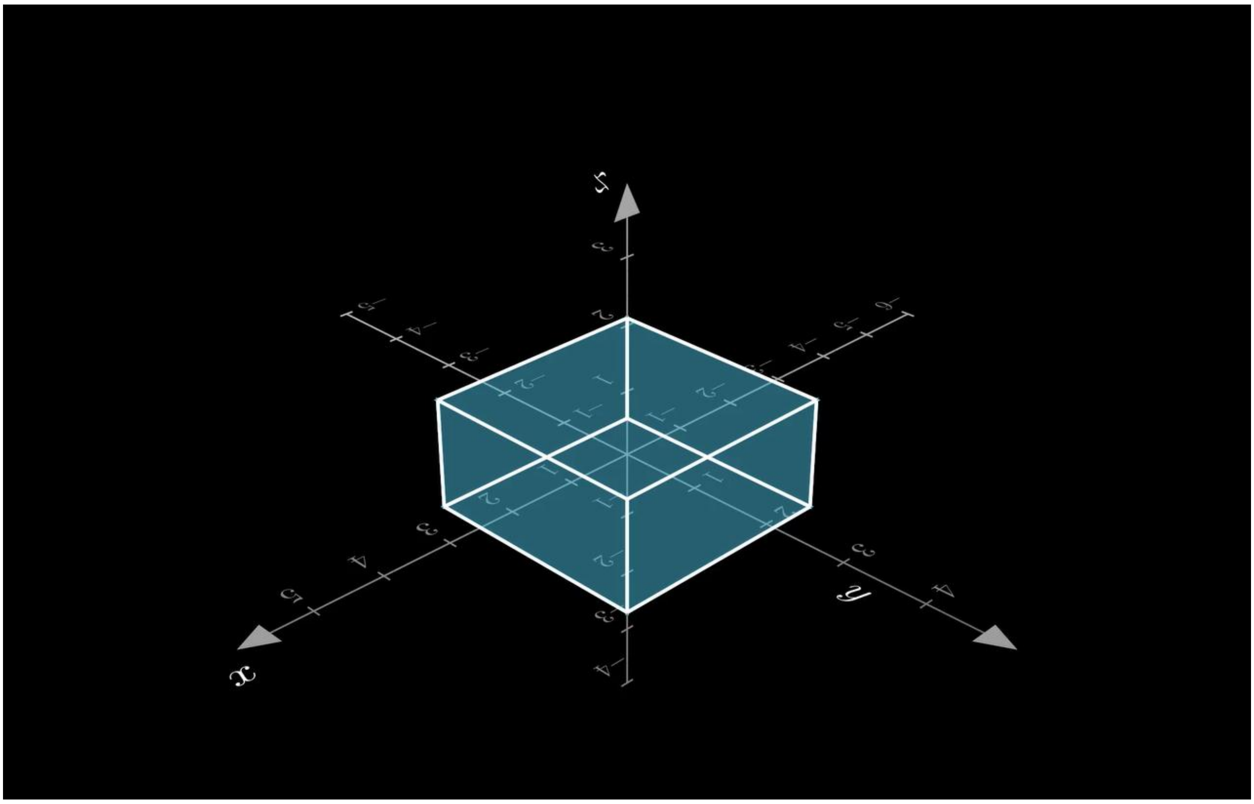
Воздействие матрицы “A”



Воздействие матрицы “B” (вернет кубик в изначальный размер, так как сначала «поделили на два», а потом «умножили на два»



Воздействие матрицы “C” (Появилась другая фигура – параллелепипед)



Задание 3

Как раз таки в этом задании мы и будем использовать те самые матрицы перемещения (или матрицы трансляции). Пример такой матрицы:

```
translation_matrix_example = np.array([
    [1, 0, 0, Tx],
    [0, 1, 0, Ty],
    [0, 0, 1, Tz],
    [0, 0, 0, 1]
])
```

Tx, Ty и Tz отвечают за сдвиг объекта вдоль соответствующих осей (x, y, z). На одном из сайтов, предложенных к рассмотрению перед выполнением лабораторной работы, находится изображение, которое в достаточной степени объясняет принцип работы с такой матрицей:

$$\begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x + T_x \\ y + T_y \\ z + T_z \\ 1 \end{pmatrix}$$

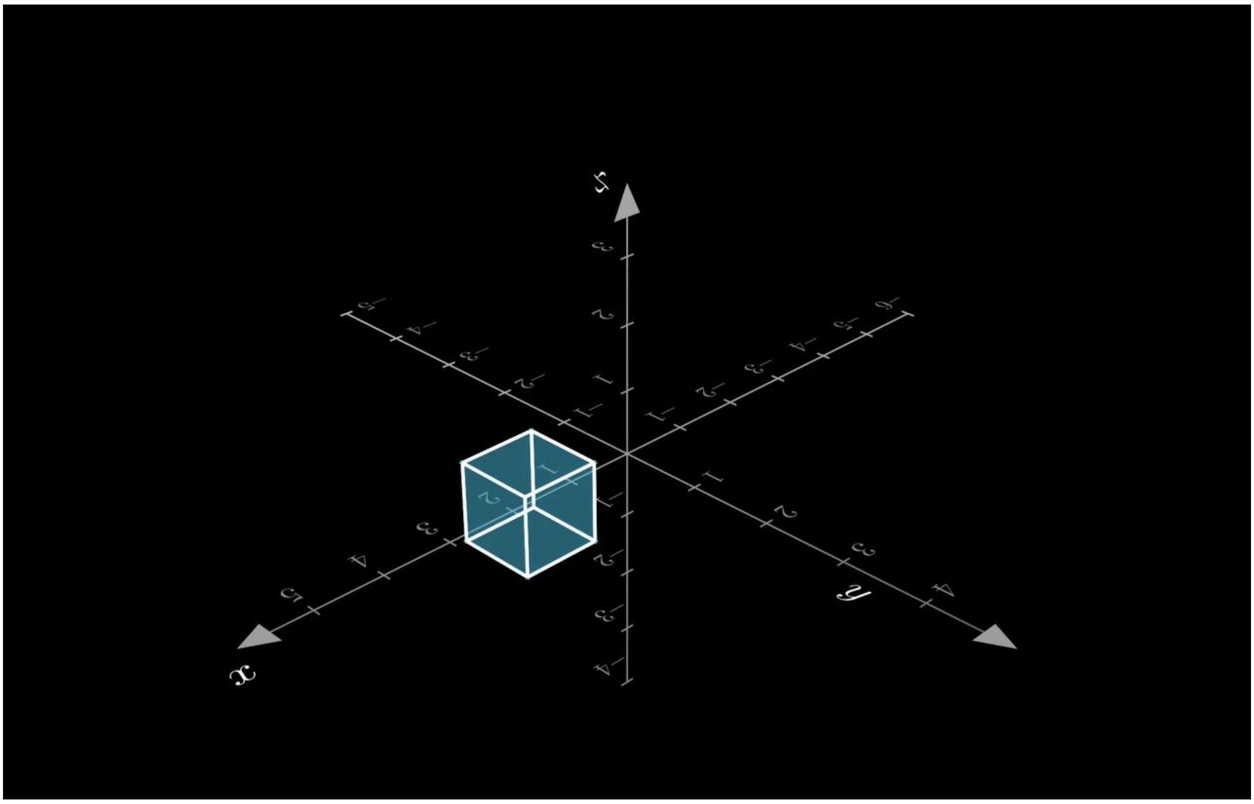
Исследуем перемещение нашего кубика с помощью следующих матриц:

```
translation_matrix_A = np.array([
    [1, 0, 0, 3],
    [0, 1, 0, 0],
    [0, 0, 1, 0],
    [0, 0, 0, 1]
])

translation_matrix_B = np.array([
    [1, 0, 0, -2.5],
    [0, 1, 0, -3.5],
    [0, 0, 1, 0],
    [0, 0, 0, 1]
])

translation_matrix_C = np.array([
    [1, 0, 0, -2],
    [0, 1, 0, 2.5],
    [0, 0, 1, 2.5],
    [0, 0, 0, 1]
])

translation_matrix_D = np.array([
    [1, 0, 0, -3],
    [0, 1, 0, 2.5],
    [0, 0, 1, -4],
    [0, 0, 0, 1]
])

translation_matrix_E = np.array([
    [1, 0, 0, 5],
    [0, 1, 0, -0.5],
    [0, 0, 1, -1.5],
    [0, 0, 0, 1]
])
```

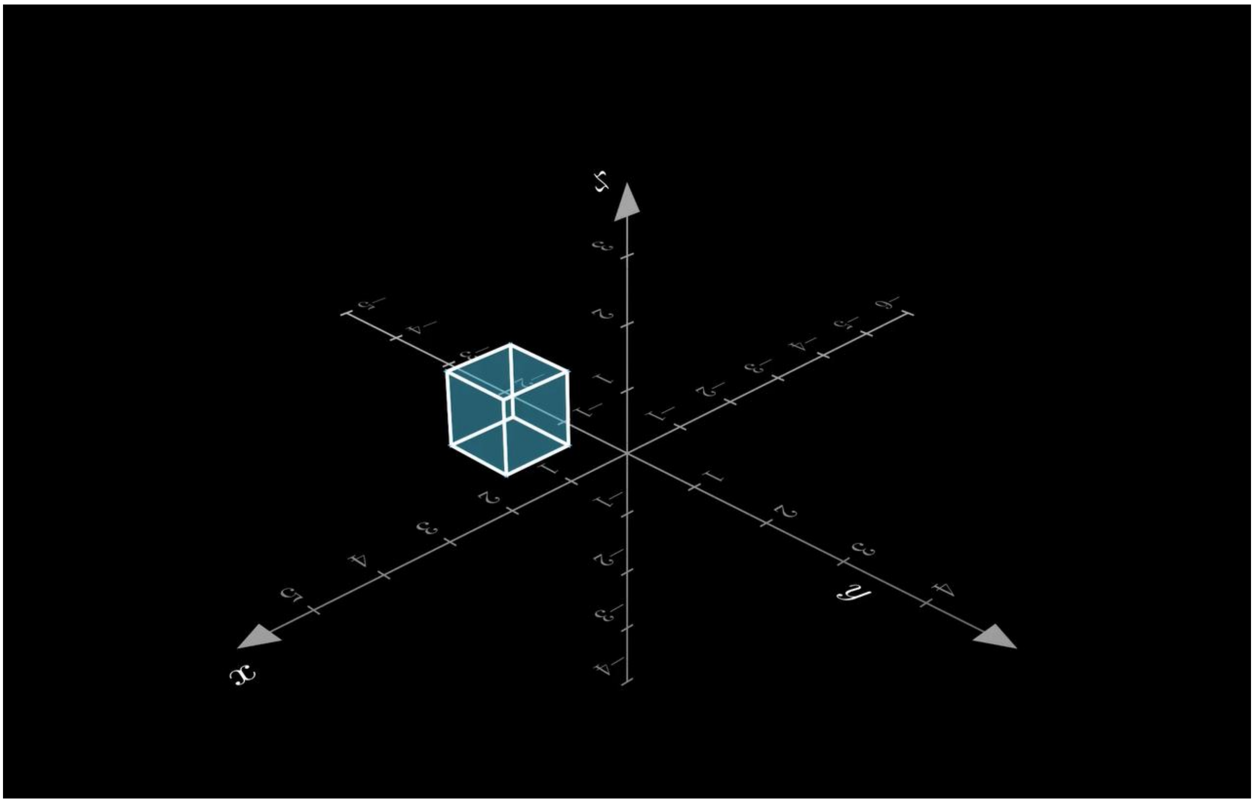
Матрица “A” сдвинет куб на 3 единицы по координате “*X*”, матрица “B” сдвинет объект -2.5 единицы по координате “*X*” и -3.5 по “*Y*” и так далее

Кубик остался на тех же координатах, что и в задании 1, однако я уменьшил его в два раза для удобного рассмотрения различных перемещений в пространстве (*Nota Bene* перемещение кубика в пространстве высчитывается относительно его заданной изменяемой матрицы, а не относительно центра системы координат)

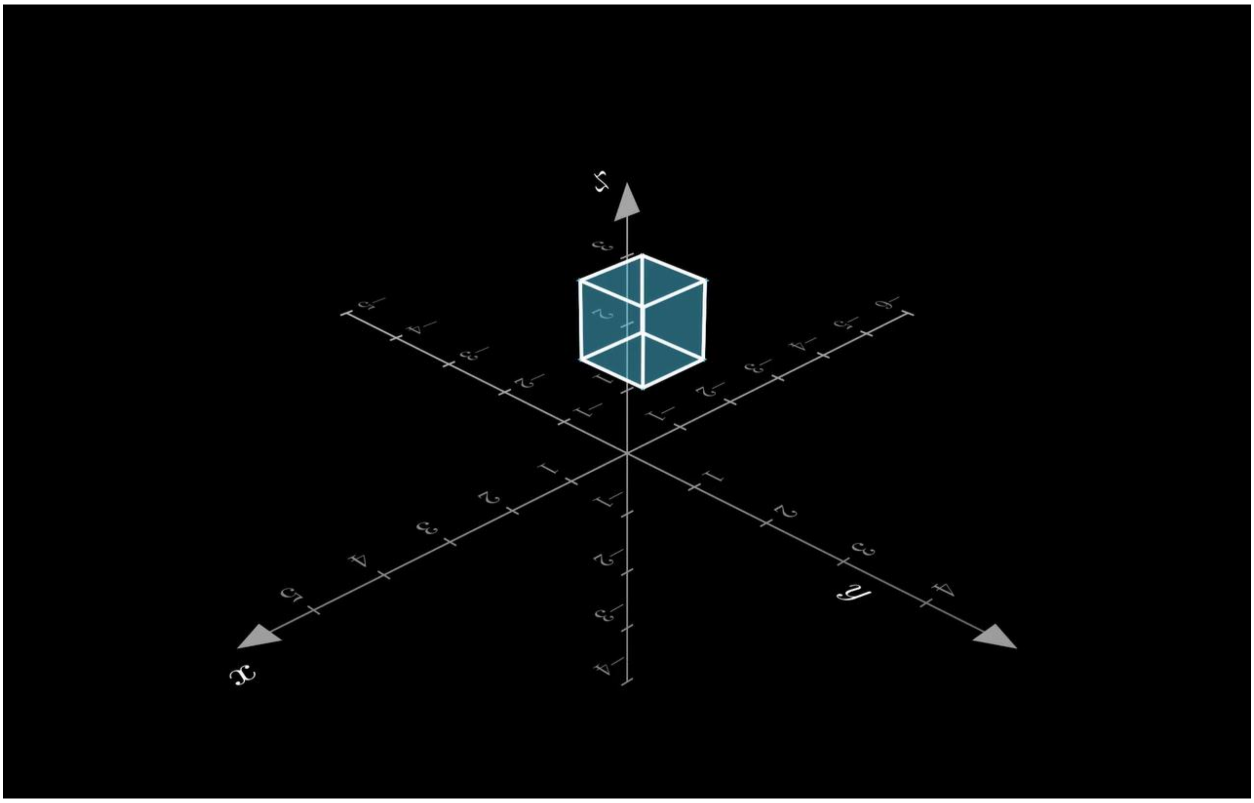
Воздействие матрицы “A”



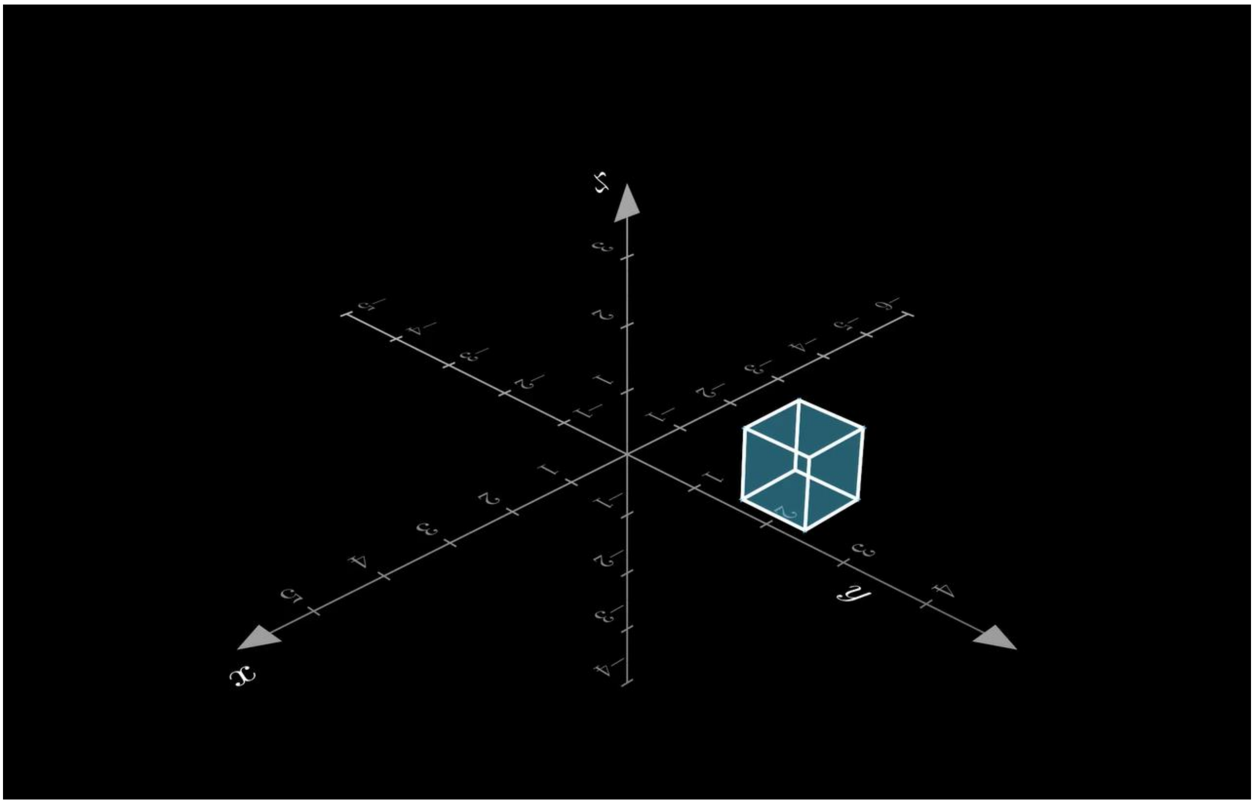
Воздействие матрицы “B”



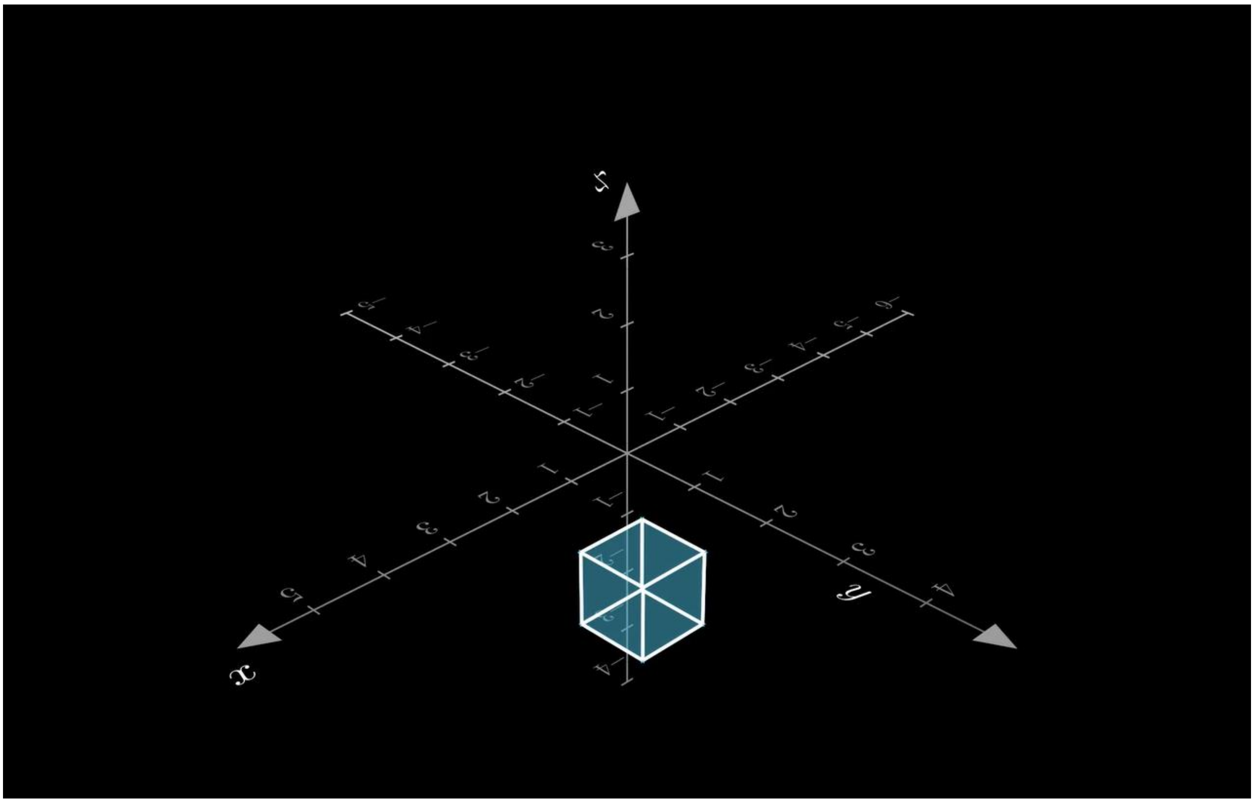
Воздействие матрицы “C”



Воздействие матрицы “D”



Воздействие матрицы “E”



Задание 4

В 3D пространстве существует всего 3 вида независимых возможных вращений – вокруг оси “*X*”, вокруг оси “*Y*” и вокруг оси “*Z*”. Рассмотрим соответствующие им матрицы:

Rotation around the X-axis:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ \cos\theta \cdot y - \sin\theta \cdot z \\ \sin\theta \cdot y + \cos\theta \cdot z \\ 1 \end{pmatrix}$$

Rotation around the Y-axis:

$$\begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta \cdot x + \sin\theta \cdot z \\ y \\ -\sin\theta \cdot x + \cos\theta \cdot z \\ 1 \end{pmatrix}$$

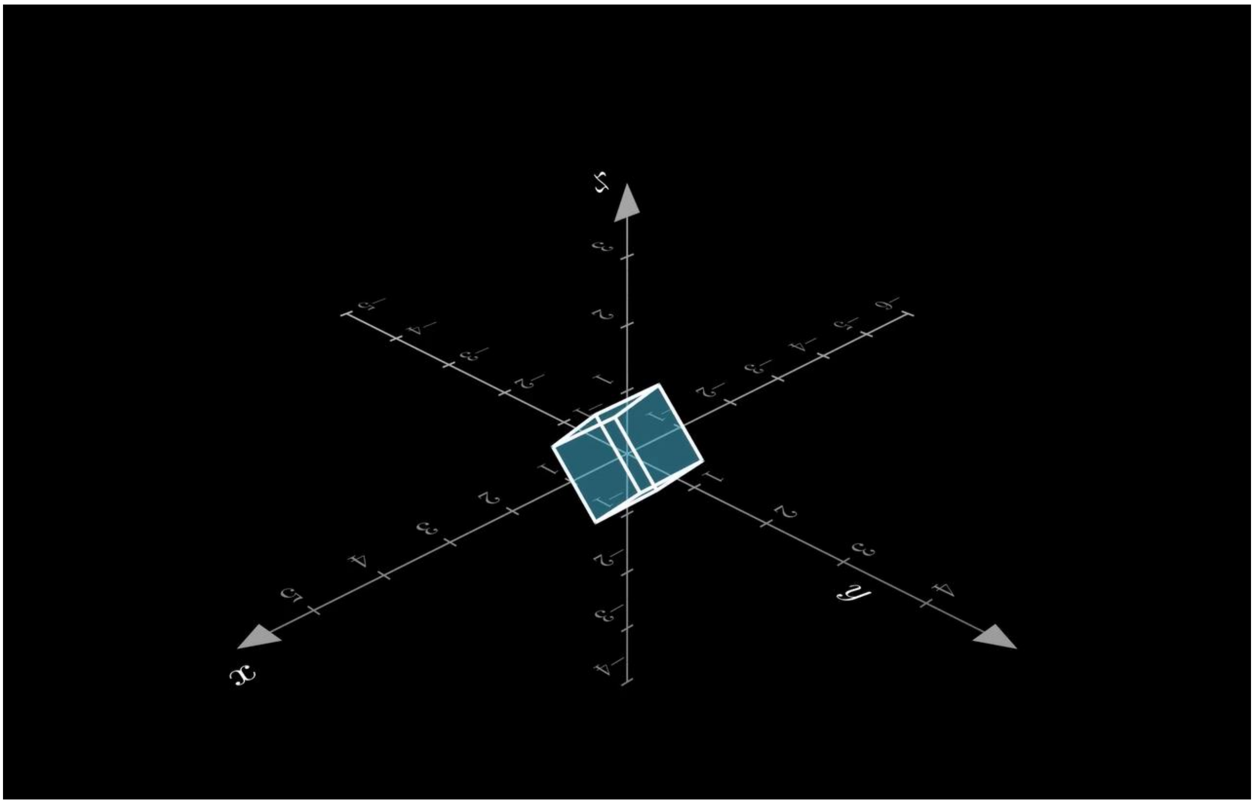
Rotation around the Z-axis:

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta \cdot x - \sin\theta \cdot y \\ \sin\theta \cdot x + \cos\theta \cdot y \\ z \\ 1 \end{pmatrix}$$

Перед рассмотрением полученных преобразований, ответим на вопрос – повороты в 3D пространстве не обладают свойством коммутативности (поворот вокруг двух разных осей будет выглядеть по разному в зависимости от порядка, в котором матрицы поворота действуют на матрицу объекта), а повороты в 2D – обладают, так как не имеет значения, рассмотрим мы сначала поворот около одной точки, а потом около другой или наоборот. При выполнении данного исследования я составил следующие матрицы:

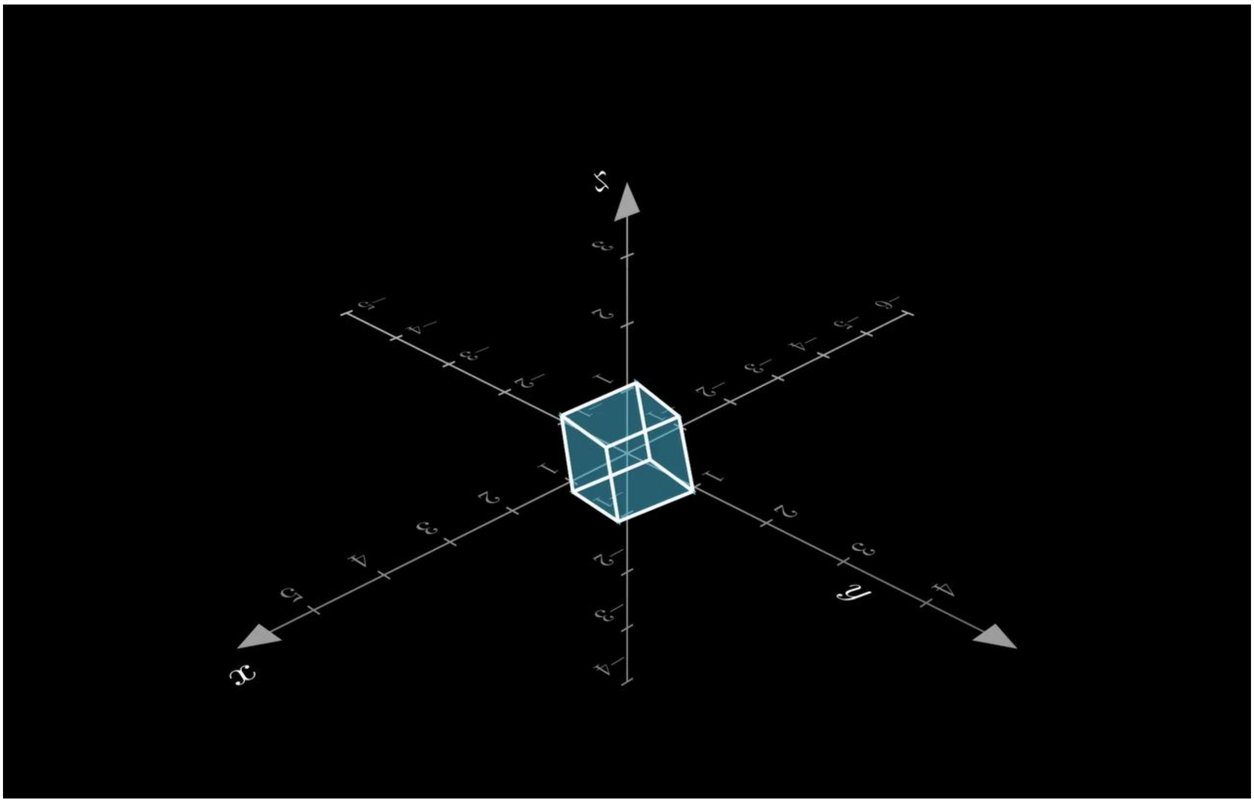
```
rotate_matrix_X_axis = np.array([
    [1, 0, 0, 0],
    [0, np.cos(np.pi / 4), -np.sin(np.pi / 4), 0],
    [0, np.sin(np.pi / 4), np.cos(np.pi / 4), 0],
    [0, 0, 0, 1]
])

rotate_matrix_Y_axis = np.array([
    [np.cos(np.pi / 4), 0, np.sin(np.pi / 4), 0],
    [0, 1, 0, 0],
    [-np.sin(np.pi / 4), 0, np.cos(np.pi / 4), 0],
    [0, 0, 0, 1]
])

rotate_matrix_Z_axis = np.array([
    [np.cos(np.pi / 4), -np.sin(np.pi / 4), 0, 0],
    [np.sin(np.pi / 4), np.cos(np.pi / 4), 0, 0],
    [0, 0, 1, 0],
    [0, 0, 0, 1]
])
```

Воздействие матрицей поворота относительно координаты “*X*”

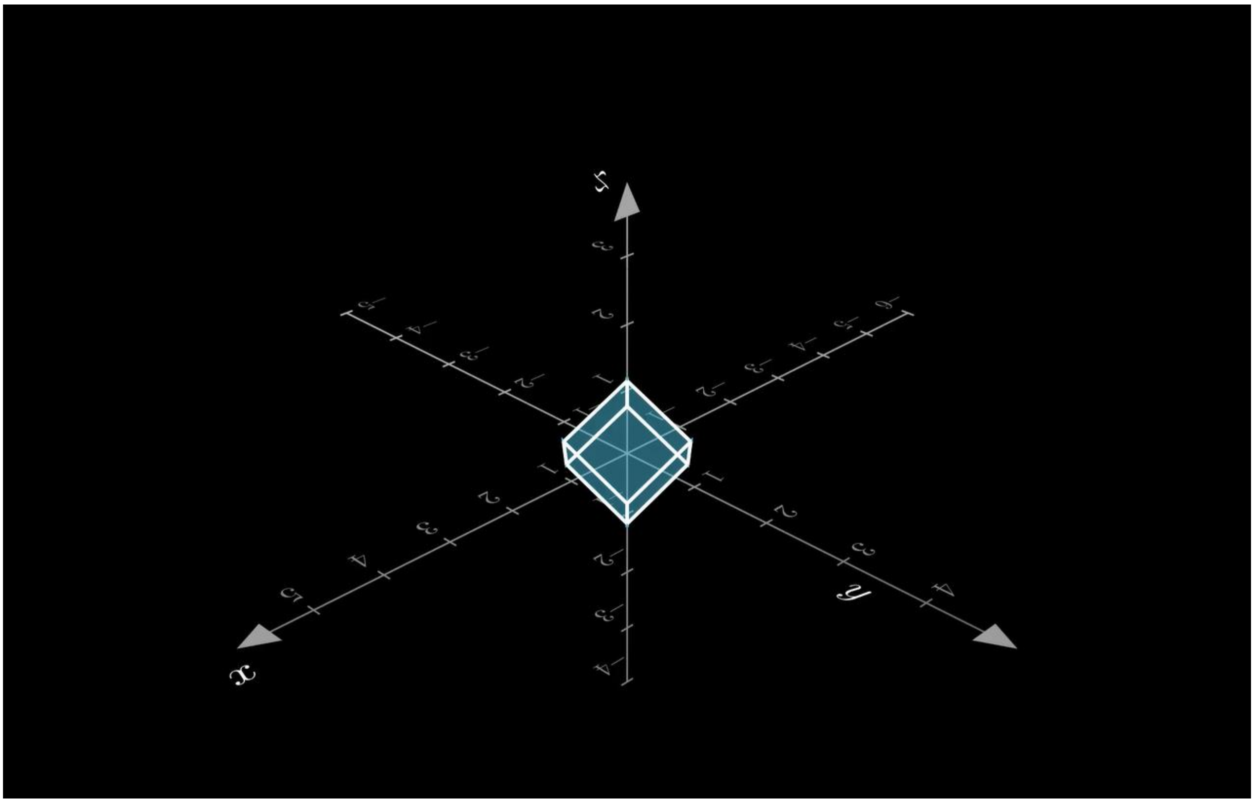


P. S. Кубик задан как в 1 задании, только меньше в 2 раза

Воздействие матрицей поворота относительно координаты “*Y*”



Воздействие матрицей поворота относительно координаты “Z”



Задание 5

Рассмотрим вращение относительно вершины (1, 1, 1). Используем матрицу трансляции, чтобы переместить куб так, чтобы вершина (1, 1, 1) встала в центр системы координат. Выполним поворот куба, после чего используем обратную матрицу к матрице трансляции точки (1, 1, 1) в центр СК, таким образом вернув ее в изначальное положение. Вращать будем вокруг осей “*X*”, “*Y*” и “*Z*”. Воспользуемся следующими матрицами:

```
rotate_matrix_X_axis = np.array([
    [1, 0, 0, 0],
    [0, np.cos(np.pi / 4), -np.sin(np.pi / 4), 0],
    [0, np.sin(np.pi / 4), np.cos(np.pi / 4), 0],
    [0, 0, 0, 1]
])

rotate_matrix_Y_axis = np.array([
    [np.cos(np.pi / 4), 0, np.sin(np.pi / 4), 0],
    [0, 1, 0, 0],
    [-np.sin(np.pi / 4), 0, np.cos(np.pi / 4), 0],
    [0, 0, 0, 1]
])

rotate_matrix_Z_axis = np.array([
    [np.cos(np.pi / 4), -np.sin(np.pi / 4), 0, 0],
    [np.sin(np.pi / 4), np.cos(np.pi / 4), 0, 0],
    [0, 0, 1, 0],
    [0, 0, 0, 1]
])
```

Матрицы поворота вокруг осей возьмем из предыдущего задания. Сначала подействуем матрицей трансляции на матрицу куба. Потом последовательно матрицами поворота, а в конце матрицей, обратной к матрице трансляции точки в центр СК. В силу ассоциативности произведения матриц, данные операции можно записать так:

```
temp_1 = np.dot(inversed_translation_matrix_v_1_1_1, rotate_matrix_Z_axis)
temp_2 = np.dot(temp_1, rotate_matrix_Y_axis)
temp_3 = np.dot(temp_2, rotate_matrix_X_axis)
transformation_matrix = np.dot(temp_3, translation_matrix_v_1_1_1)
```

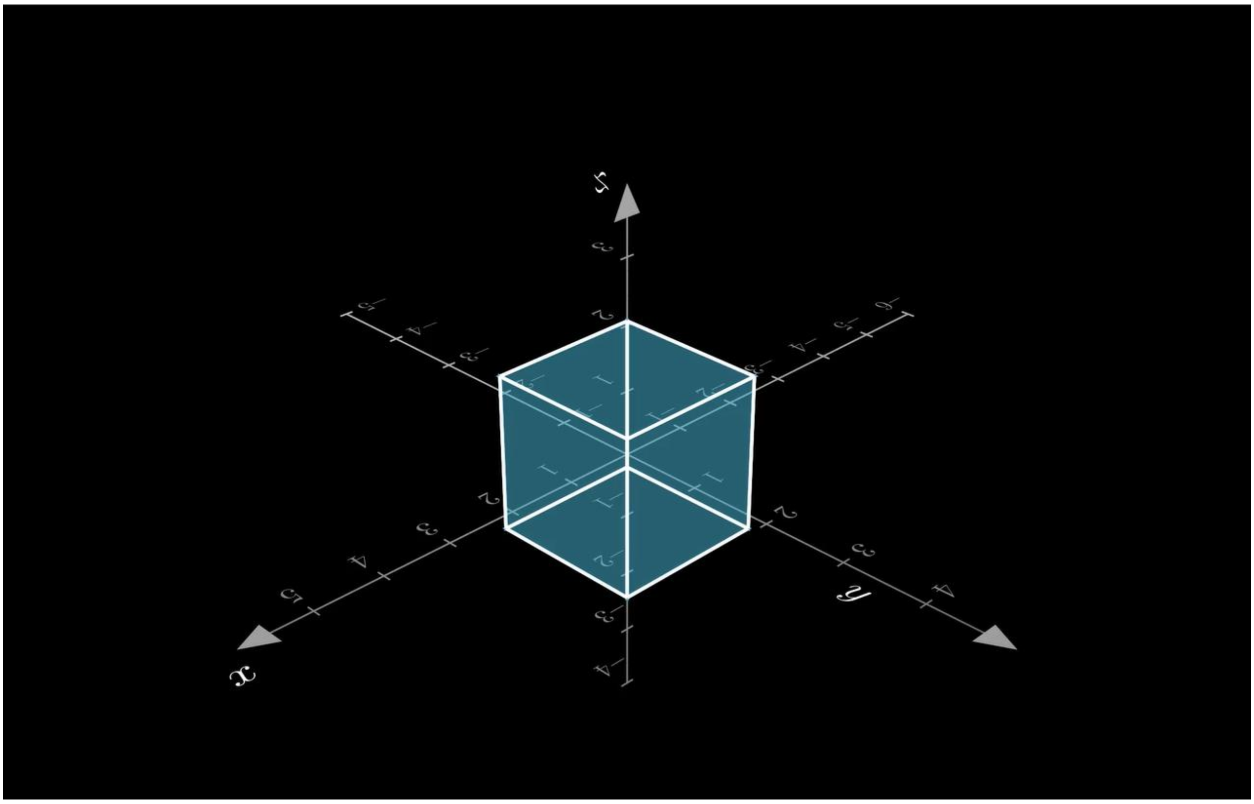
Наша “transformation_matrix” выполняет сразу все преобразования, то есть вращает куб вокруг одной вершины – (1, 1, 1). Сама матрица имеет следующие значения:

```
transformation_matrix = [[ 0.5, -0.14644661, 0.85355339, -0.20710678]
                         [ 0.5, 0.85355339, -0.14644661, -0.20710678]
                         [-0.70710678, 0.5, 0.5, 0.70710678]
                         [ 0, 0, 0, 1]]
```

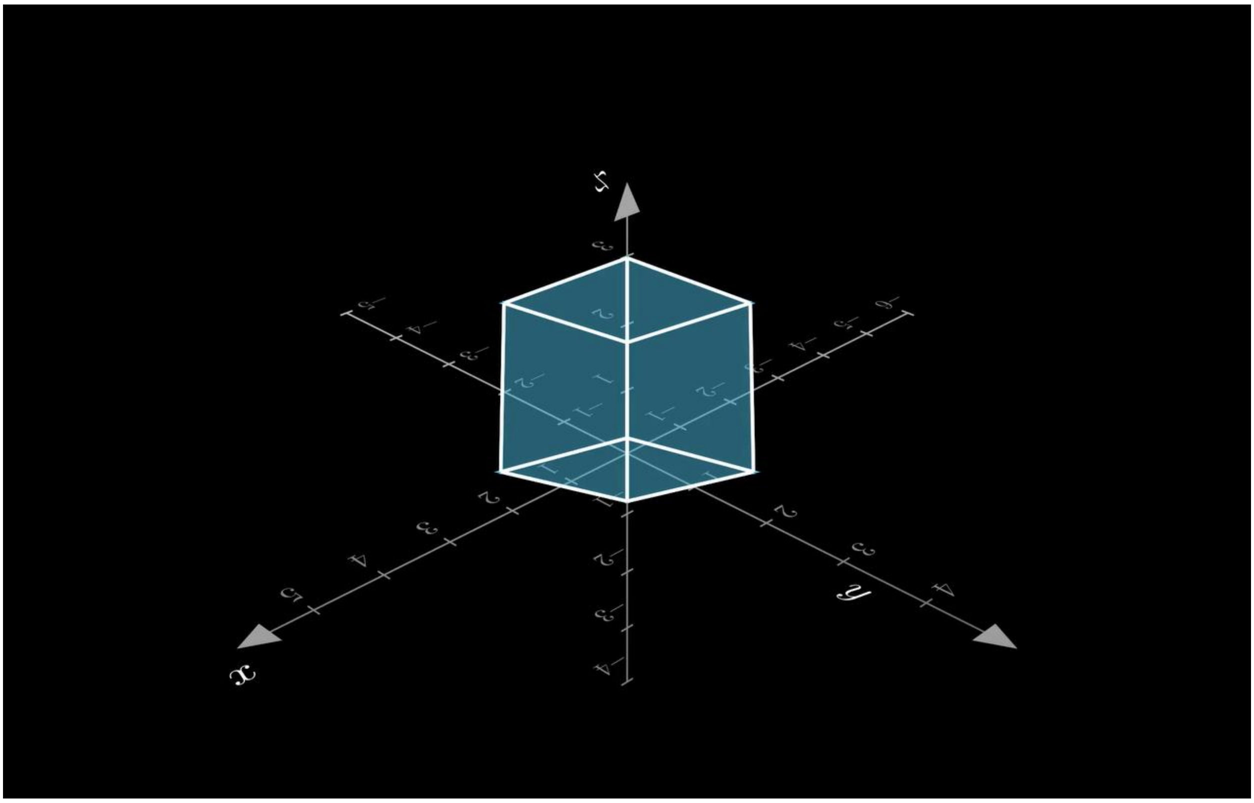
Для наглядности преобразования я покажу изначальный куб и преобразованный

Преобразования на этом листе уже не поместились

Изначальный куб



После воздействия матрицей поворота вокруг вершины



Как видим, вершина (1, 1, 1) осталась на месте (по СК не похоже на вершину с единицами – мне не удалось в маниме настроить СК так, чтобы размеры совпадали, sorry)

Задание 6

Для выполнения этого задания я разместил три кубика разного размера в разных частях пространства. Для того, чтобы переместить кубики, я использовал матрицы трансляции: “A” для куба среднего размера, “B” для большого куба и “C” для маленького

```
cube_1 = Cube(default_vertices*0.5)
cube_1 = cube_1.copy().get_transformed(translation_matrix_A)

cube_2 = Cube(default_vertices*0.7)
cube_2 = cube_2.copy().get_transformed(translation_matrix_B)

cube_3 = Cube(default_vertices*0.3)
cube_3 = cube_3.copy().get_transformed(translation_matrix_C)
```

Как обычно, я зафиксировал вид камеры на 60 градусов по вертикали и 45 градусов по горизонтали, чтобы был хороший и удобный обзор на кубик в центре СК

```
add_axes_to_scene(self, phi=90, theta=0)
```

Теперь реализуем преобразование сцены – поворачивание системы координат под вертикальным углом в 90 градусов и горизонтальным в 0 градусов, чтобы кубики было видно со всех сторон сбоку…

```
self.begin_ambient_camera_rotation(rate=90 * DEGREES, about="theta")
```

К сожалению, я неправильно задание, поэтому использовал метод из библиотеки manim, позволяющий сделать такое преобразование камеры (п. с. лаба по умножению матриц, да-да). Однако, как я узнал позже, нужно переместить всю сцену так, чтобы камера оказалась в начале координат. Для этого необходимо умножить каждый компонент сцены на обратную матрицу перемещения и поворота камеры

Однако на часиках 2:41… Предлагаю читающему выполнить преобразование устно в качестве небольшой разминки! Ну или можно посмотреть на вращающуюся СК на гугл диске, тоже своеобразный развлекательный контент

Вместо 7 и 8 заданий предлагаю вспомнить легендарное видео, а я вынужден откланяться, то есть пришвартоваться к кровати. Всем радости, веселья, удачи, сна, денег и новый год