В этой лабораторной вы познакомитесь с особыми матрицами преобразования пространства, которые главным образом используются в 3D-графике. Перед выполнением задания разберитесь с тем, как они устроены. Матрицы преобразования пространства. Матрицы преобразования перспективы (Во втором источнике матрицы транспонированы). Все преобразования координат в данной работе должны быть осуществлены с помощью матриц!

Задание 1. Создайте кубик. Используйте код MATLAB (1) или Python (2) для отрисовки кубика в 3D пространстве. Разберитесь, как работает код, который вы выбрали. Почему мы используем четырехкомпонентный вектор, а не трех? Как задать другие фигуры?

Ожидаемые результаты:

- Приведен результат выполнения программы из приложения.
- Приведен результат выполнения программы для другой фигуры, полученной с помощью замены матриц `vertices_cube` и `faces_cube`.

Задание 2. Измените масштаб кубика. Придумайте матрицу преобразования масштаба по осям S исходя из изученных материалов. Примените эту матрицу ко всем векторам кубика, таким образом получив его образ. Продемонстрируйте, как меняется кубик в зависимости от выбранной матрицы масштабирования.

Ожидаемые результаты:

- Приведена общая структура матриц изменения масштаба по осям x, y и z.
- Приведены две матрицы для выбранных вами значений масштабирования, а также третья полученная с помощью объединения двух прошлых преобразований. Приведены результаты выполнения программы после применения каждой из матриц к вершинам кубика.

Задание 3. Переместите кубик. Придумайте матрицу перемещения T. Разберитесь, как она работает. Исследуйте, как меняется кубик в зависимости от выбранной комбинации матриц перемещения и масштабирования TS и ST. Эквивалентны ли такие преобразования?

Ожидаемые результаты:

- Приведена общая структура матриц перемещения по осям x, y и z.
- Приведены две матрицы для выбранных вами значений перемещения, а также третья полученная с помощью объединения двух прошлых преобразований. Приведены результаты выполнения программы после применения каждой из матриц к вершинам кубика.

- Приведены две матрицы комбинаций TS и ST и результаты выполнения программы после применения каждой из матриц к вершинам кубика.

Задание 4. Выполните вращение кубика. Придумайте вектор v, вокруг которого вы хотите вращать кубик на угол θ. Используйте формулу

$$v = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}, \quad J = \frac{1}{\|v\|} \begin{bmatrix} 0 & -v_z & v_y & 0 \\ v_z & 0 & -v_x & 0 \\ -v_y & v_x & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad R_v(\theta) = e^{J\theta}$$

для получения матрицы поворота $R_v(\theta)$ и примените её к кубику. Исследуйте, как меняется кубик в зависимости от выбранных оси и угла. Разберитесь, как представленная формула работает. На все графики добавьте вектор, коллинеарный оси вращения.

Ожидаемые результаты:

- Приведена общая структура матриц поворота вокруг оси v, проходящей через центр. Приведены формулы матриц поворота вокруг осей x, y и z.
- Приведены две матрицы для выбранных вами произвольных осей и углов поворота ($[v_1, \theta_1]$ и $[v_2, \theta_2]$), а также пара матриц полученная с помощью объединения двух прошлых преобразований двумя способами. Приведены результаты выполнения программы после применения каждой из матриц к вершинам кубика.

Задание 5. Выполните вращение кубика вокруг любой вершины. Найдите матрицу преобразования, которая осуществит вращение кубика таким образом, что центр вращения будет совпадать с одной из его вершин. Продемонстрируйте результат, разместив на одном графике исходный и повернутый кубик (Результат работы должен быть виден).

Ожидаемые результаты:

- Выведена общая формула матриц поворота вокруг оси с направлением v, проходящей через точку $M(x, y, z)$.
- Приведена матрица поворота, выполняющая указанное преобразование. Приведен результат выполнения программы после применения матрицы к вершинам кубика на графике с исходным кубиком.

Задание 6. Реализация камеры. Разместите несколько разных кубиков на одном графике, таким образом создав сцену аналогичную рисункам 1, 2. Используйте команду MATLAB `view([0 -90])` или Python `ax.view_init(azim=0, elev=-90)`, чтобы посмотреть на график «снизу» (Рис. 1). Выберите матрицы T_c и $R_c(\theta)$, отвечающие за положение и поворот камеры так, чтобы она смотрела на сцену на отдалении под углом. Найдите такую матрицу C^{-1}, которая реализует обратное преобразование положения камеры, переместив её из выбранного положения в начало координат со стандартным поворотом. Примените эту матрицу ко всем объектам сцены.

Ожидаемые результаты:

- Авторское повторение сцены из рисунков 1, 2 с использованием изученных матриц из прошлых пунктов для преобразования кубиков. Приведено два рисунка, под стандартным углом камеры и под углом запрошенным заданием.
- Приведена матрица преобразования пространства C и её обратная. Приведен рисунок, полученный применением этой матрицы ко всем объектам сцены.

Задание 7. Реализация перспективы. Используйте матрицу перспективы P из изученных материалов для преобразования сцены из задания 6. Как эта матрица устроена? Какие параметры нам необходимо выбрать для её реализации? Используйте стандартное вращение графика с помощью команд из предыдущего задания для исследования деформации пространства под её влиянием.

Ожидаемые результаты:

- Приведена общая форма матриц перспективы и частный её случай для самостоятельно выбранных параметров.
- Приведен рисунок демонстрации работы перспективы по сравнению с изометрической камерой. Приведены рисунки, полученные вращением стандартной камеры (команды `view(···)` и `ax.view_init(···)`) для демонстрации деформации пространства.

Задание 8 (необязательное). Почти Blender. Постройте домик или любое другое строение из блоков или других фигур. Используйте матрицы масштабирования, поворота и перемещения, а также задайте камеру и перспективу.

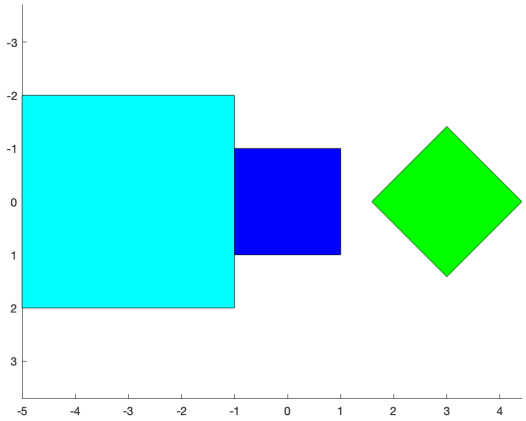

Рис. 1: Сцена из трех кубиков.

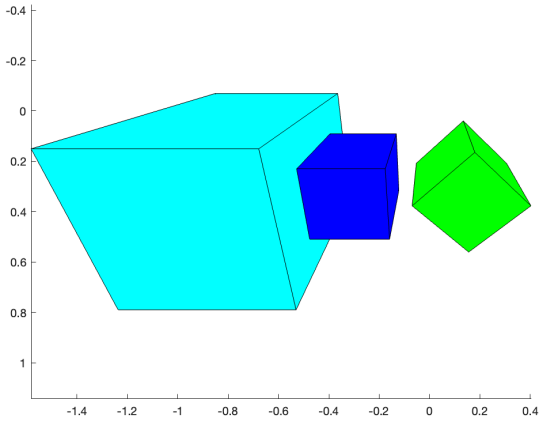

Рис. 2: Результат выполнения задания 7.

Контрольные вопросы по проделанной работе:

1. Что такое однородные координаты и зачем они нужны? Как преобразовать координаты точки в декартовы?
2. Какие основные виды преобразований встречаются в 3D-графике? Что такое аффинное преобразование и чем оно отличается от линейного?
3. Как получить матрицу масштабирования вдоль какого-нибудь вектора?
4. Коммутируют ли матрицы S и T между собой? Коммутируют ли две матрицы T между собой или две S? Как (не)коммутивность можно объяснить геометрически для этих матриц?
5. Что такое кососимметрические матрицы, какие у них собственные числа? Какие собственные числа симметрических матриц?
6. Пусть вещественная матрица имеет комплексные собственные числа. Какую особенность имеют её собственные значения и векторы?
7. Что такое аналитическая функция? Как определяются матричные функции? Как определяется матричная экспонента и какое её применение?
8. Какое собственное число всегда будет присутствовать у матриц $SO(3)$? Как это объяснить, зная свойства кососимметрической матрицы? Связь с теоремой о вращении Эйлера.
9. Как восстановить ось вращения, зная лишь матрицу поворота в 3D пространстве?
10. Какие матрицы называются подобными? Чем схожи и чем различаются две подобные друг-другу матрицы?
11. Какой смыл обратного преобразования матрицы камеры шестого задания? Что происходит с пространством под её влиянием?
12. Какие виды проекций использовались в этой работе. Как перспектива преобразует пространство геометрически? Как перспектива влияет на четвертую координату алгебраически?

```
vertices_cube = [
    -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_cube = [
         1, 2, 6, 5;
         2, 3, 7, 6;
         3, 4, 8, 7;
         4, 1, 5, 8;
         1, 2, 3, 4;
         5, 6, 7, 8
        ];

DrawShape(vertices_cube, faces_cube, 'blue')
axis equal;
view(3);

function DrawShape(vertices, faces, color)
    patch('Vertices', (vertices(1:3,:)./vertices(4,:))', 'Faces', faces, '
   FaceColor', color);
end
```

Листинг 1: Код для отрисовки кубика на языке MATLAB.

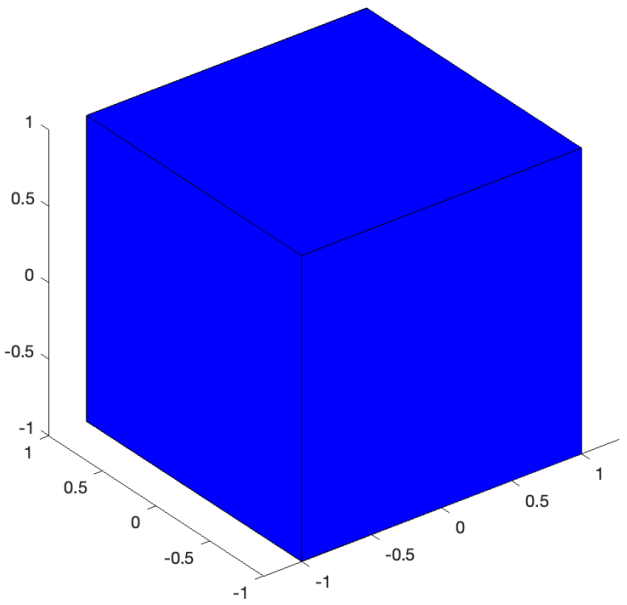

Рис. 3: Результат выполнения программы.

```python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d.art3d import Poly3DCollection

def draw_shape(ax, vertices, faces, color):
    vertices = (vertices[:3, :] / vertices[3, :]).T
    ax.add_collection3d(Poly3DCollection(vertices[faces], facecolors=color
    , edgecolors='k', linewidths=0.2))

vertices_cube = 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_cube = np.array([
    [0, 1, 5, 4],
    [1, 2, 6, 5],
    [2, 3, 7, 6],
    [3, 0, 4, 7],
    [0, 1, 2, 3],
    [4, 5, 6, 7]
])

fig = plt.figure()
ax = fig.add_subplot(projection='3d', proj_type = 'ortho')

draw_shape(ax, vertices_cube, faces_cube, 'blue')

ax.set_box_aspect([1,1,1])
ax.set_xlim(-1, 1); ax.set_ylim(-1, 1); ax.set_zlim(-1, 1)
ax.view_init(azim=-37.5, elev=30)
ax.set_xticks(np.linspace(-1, 1, 5)); ax.set_yticks(np.linspace(-1, 1, 5))
    ; ax.set_zticks(np.linspace(-1, 1, 5))

plt.show()
```

Листинг 2: Код для отрисовки кубика на языке Python.

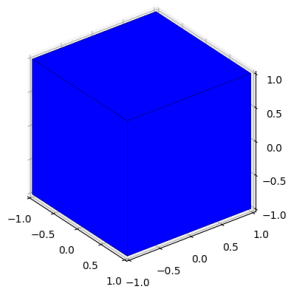

Рис. 4: Результат выполнения программы.