Лабораторная работа №3

Практическая линейная авадакедабра

Андрей Зелепугин
2023

Оглавление

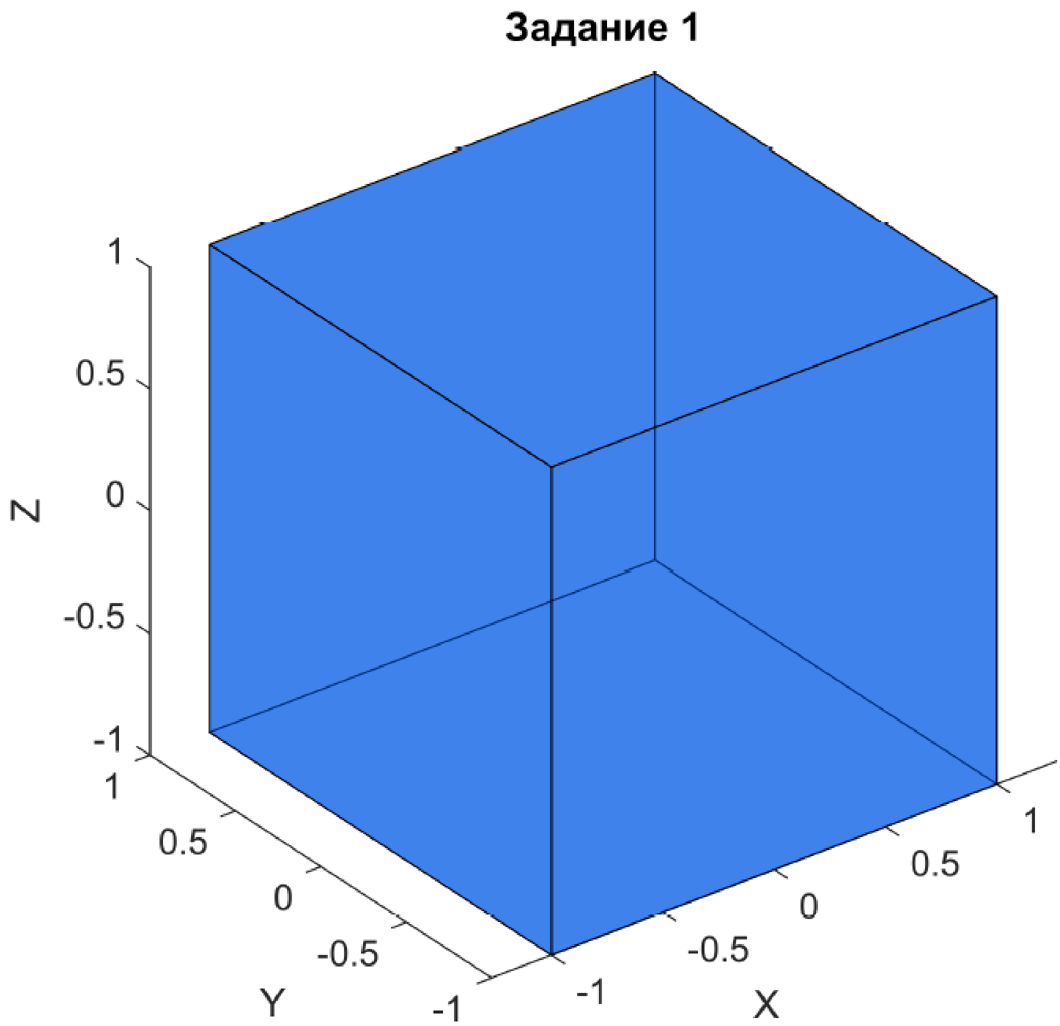
Эта версия не интерактивна, см. наглядный КОД для матлаба со всеми объяснениями

Кто будет запускать, сначала запускаете танк, потом код, не очищая Workspace.

Почувствовав прилив творческих сил, создал куб

Использовал листинг 1, код комментировал, чтобы ближе к строкам быть

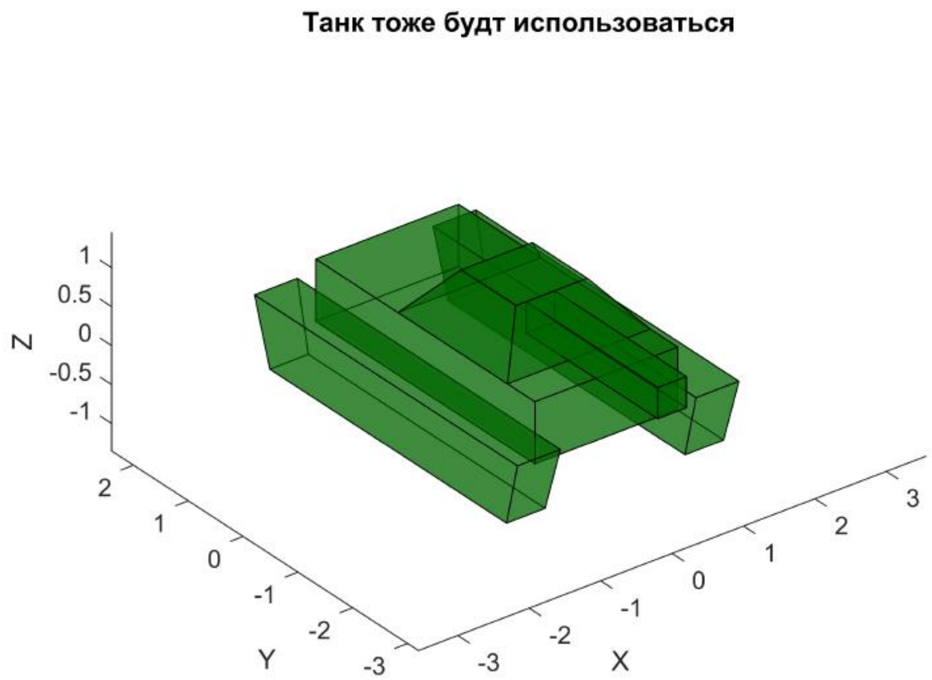
Задали вершины, грани в виде номеров тех вершин, которые они должны соединить, я ещё цвета задал. Куб до синий, после оранжевый. Полупрозрачный, чтобы: а) не такие ядреные цвета, б) видно, что происходит на пересечении.



Тут следовало бы поразмыслить, для каких таких целей нам 4-я координата в кубе. Вначале очевидно посмотреть на код, увидеть *vertices (1:3 ,:) ./ vertices (4,:))'*, тщательно подумать, что такое поэлементное деление первых трёх, которые как раз задают координаты, на 4, и понять для начала, что если в 4 не единица, то точка единичного куба не будет в единицах, то есть, когда будет нарисована поверхность, она окажется наклонённой (если менять не на одно и то же число во всех координатах) (так я сначала узнал координаты танковых запчастей); второе, о чём подумает трезвый человек: это для масштабирования и регулирования чувствительности, мол, разделим на 0,5 и вот он в два раза больше, а про чувствительность см. низ третьего задания; ну а третье, что можно понять, если начать читать мелкий курсивный шрифт в самом начале описания лабораторной, так это то, что это для тех, кто умеет только умножать, но не складывать векторы или выпендривается своими гомогенными координатами.

Также отмечу, что использую полупрозрачные цвета для того, чтобы повысить наглядность.

Вообще, не зря же я старался, держите танк (он же задание 8):

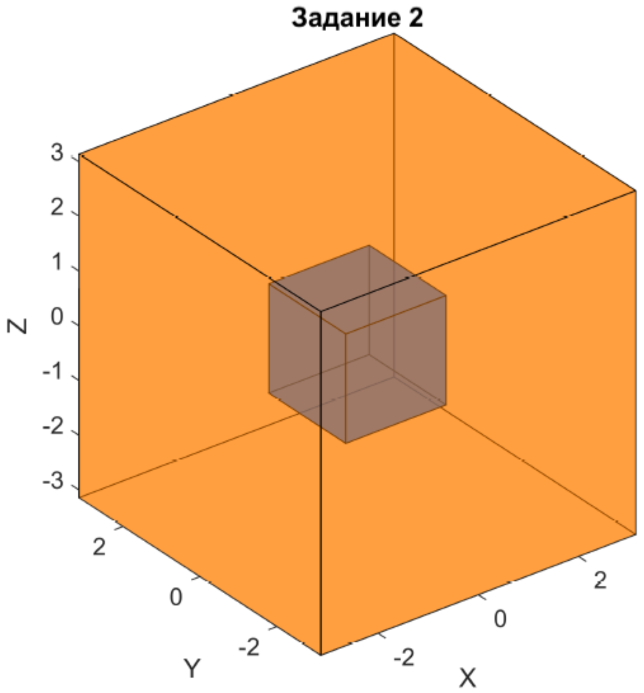


Как создавался танк можно посмотреть по ссылке.

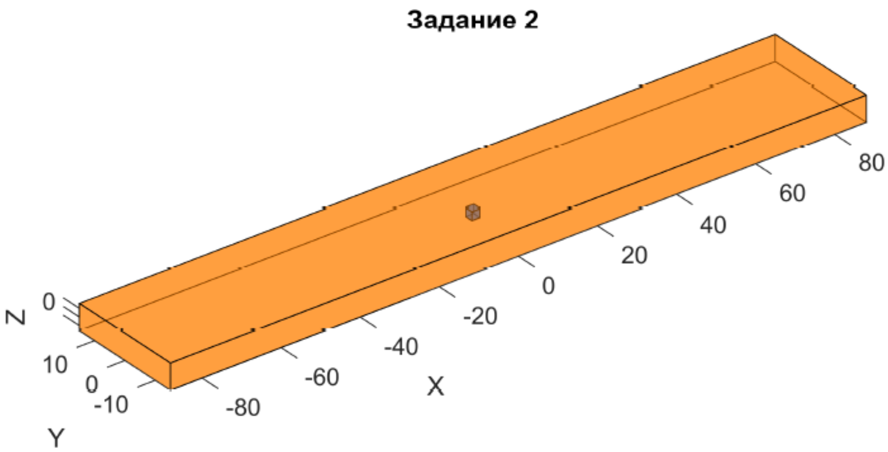
Куб - это хорошо, но мало

Теперь мне предстояло расшириться, ведь куб – это лишь куб, но бо́льший куб даст бо́льшие возможности, например, если увеличить куб в π раз, мы почувствуем, что это совсем нетривиально, а значит мы смелы и сильны. Нули не рисовал. Матрицу A_2 умножаю на матрицу координат вершин куба, так происходит масштабирование.

$$A_2 = \begin{pmatrix} \pi & & & \\ & \pi & & \\ & & \pi & \\ & & & 1 \end{pmatrix}$$

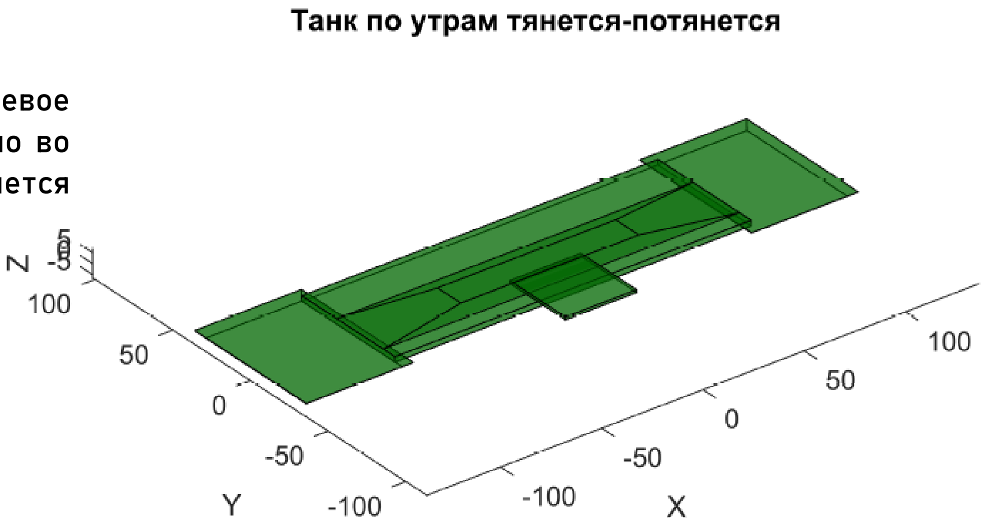


Это, конечно, круто, но до жути банально. Поступим как настоящие панки, и изменим масштаб другой матрицей, а то из куба получать куб… Для этого просто сделаем коэффициенты на диагонали различными.



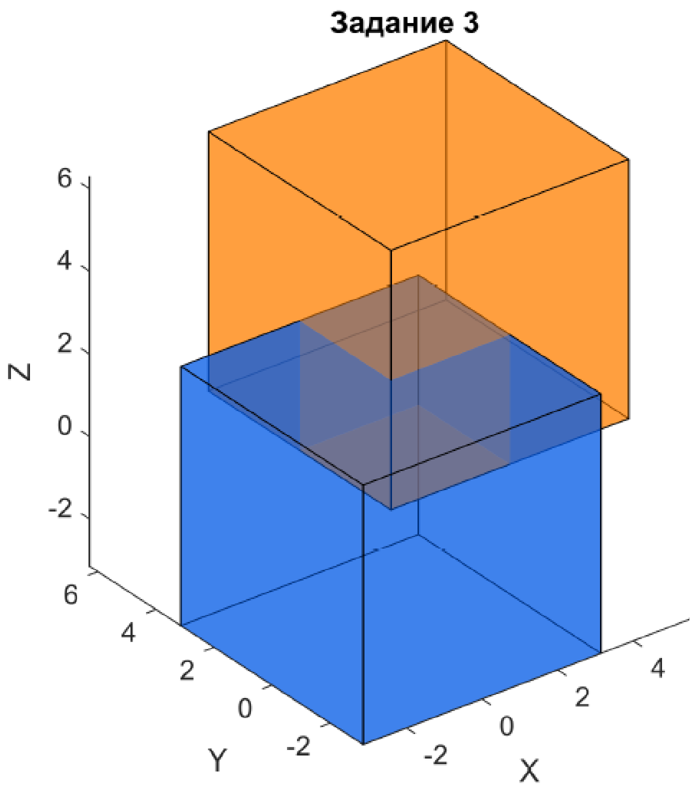
$$A_{21} = \begin{pmatrix} 88 & & & \\ & 15 & & \\ & & \pi & \\ & & & 1 \end{pmatrix}$$

Собственно, единственное ненулевое число в каждой из строк – ровно во столько раз меняется соответствующая координата.



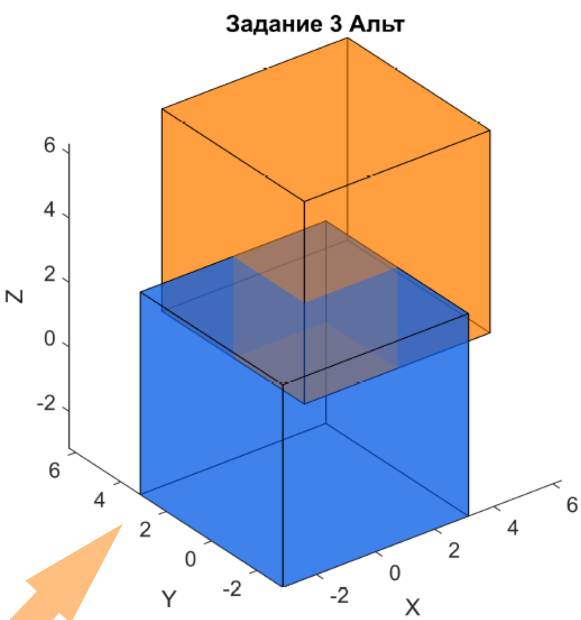
Куб стал таким большим, что загораживает проход, его нужно сместить

Для этих целей решение крайне простое: взять и сдвинуть. Сдвигать буду большой куб из второго задания, и на пи. Куб до преобразований всегда синий, после – оранжевый. Что будем делать? Есть несколько способов, сначала, естественный и очевидный: к каждому столбцу куба прибавить вектор, на который хотим сместить.




```
A3=[pi;pi;pi;0];
Cube3 = Cube2 + A3;
```

$$A_3 = \begin{pmatrix} \pi \\ \pi \\ \pi \\ 0 \end{pmatrix}$$



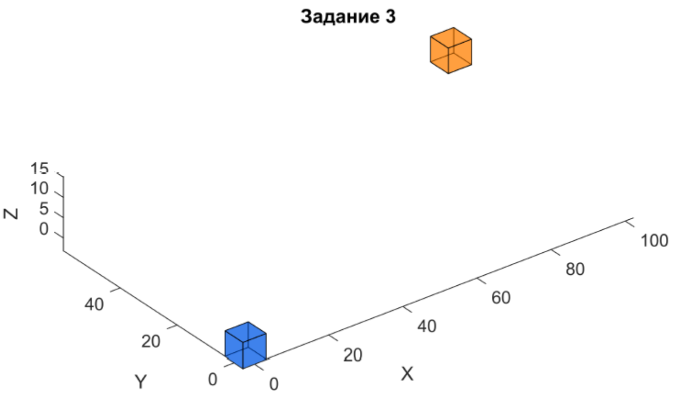
При этом дальняя вершина куба, которая после второго задания оказалась в $-\pi$ по всем координатам, переместится в начало координат, то есть у куба все координаты станут неотрицательными.

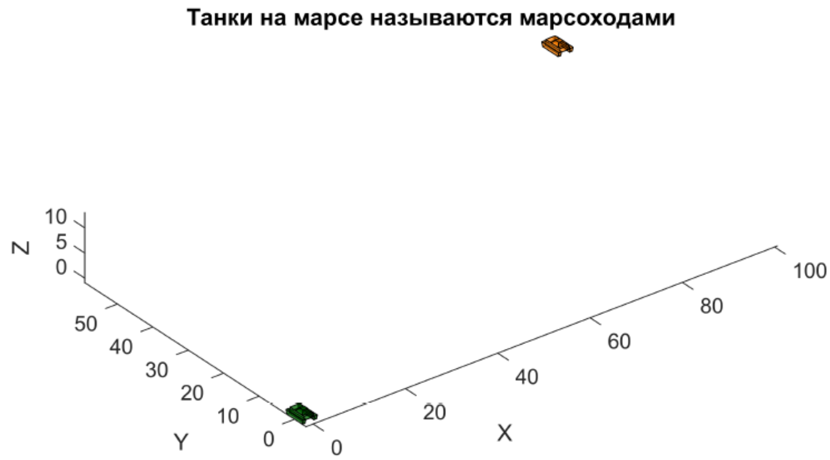
Конечно, можно задать и умножением матриц, для чего, собственно, и дана нам четвертая координата, которую поначалу легко принять за пятую ногу. Далее я покажу, как будет выглядеть умножение, но только в одном примере просто потому, что считаю сложение матриц, во-первых, более интуитивной операцией, во-вторых, более дешёвой для ресурсов компа, которые все и напрочь скушал матлаб, с другой же стороны, путём умножения нескольких матриц получится изменить состояние только тогда, когда мы двигаем именно умножая на матрицу.

```
A30=[1,0,0,pi;
    0,1,0,pi;
    0,0,1,pi;
    0,0,0,1];
Cube30 = A30*Cube2;
```

$$A_3 = \begin{pmatrix} 1 & & & \pi \\ & 1 & & \pi \\ & & 1 & \pi \\ & & & 1 \end{pmatrix}$$

Но это снова не так интересно, ведь так мы долго будем лететь до Марса. Чтобы приземлиться прямёхонько туда, возьмем новую мтарицу, и сместим дальше до 2035 года.

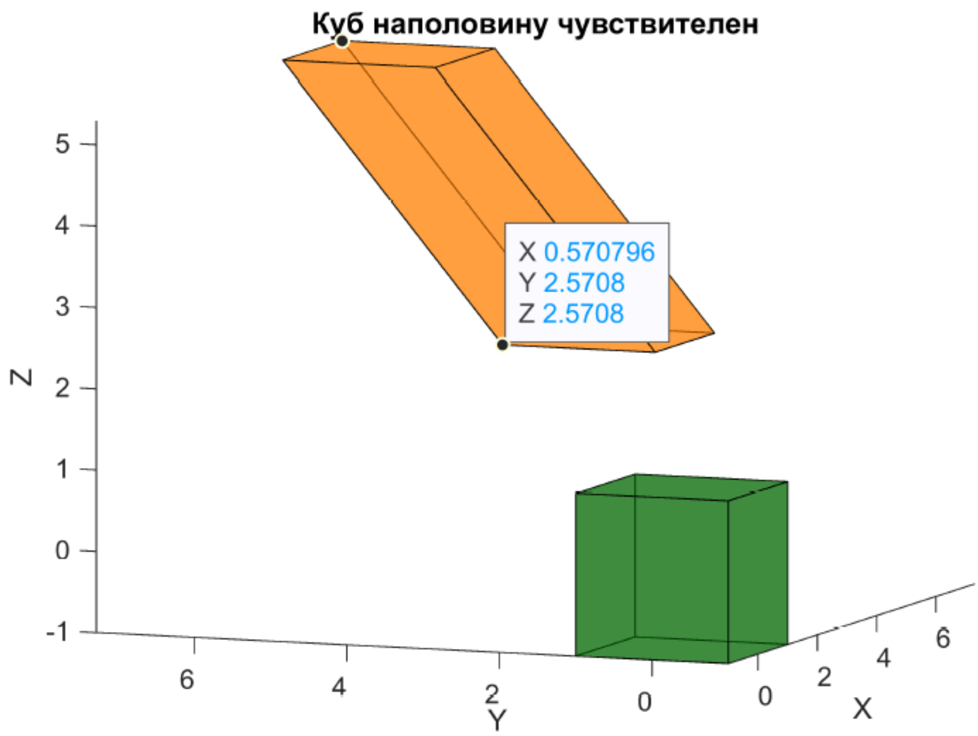




Совсем другое дело! Более того, из-за такого расстояния размерами теперь можно пренебречь, и вот она, задачка про материальные точки из курса школьной физики. Хотя я потом посмотрел на результат и, для наглядности, 9999 и 568 заменил на 99 и 56,8.

Заодно покажу, что я имел ввиду под тем, что 4-я координата регулирует чувствительность (как задан зелёный куб см. ниже), а я его сместил на π:

```
PsevdoCube = [
-0.5, 0.5, 0.5,-0.5,-2, 2, 2 ,-2;
-0.5,-0.5, 0.5, 0.5,-2,-2, 2, 2;
-0.5,-0.5,-0.5,-0.5, 2, 2, 2, 2;
0.5, 0.5, 0.5, 0.5, 2, 2, 2, 2];
```



Подул ветер, и куб завертелся

Использую матрицу преобразования для поворота кубика. Три. Именно столько видов независимых возможных вращений может существовать в 3D пространстве. Вокруг Ox, Oy, Oz. Если добавить к ним смещение по этим же осям, мы получим 6. Это число так и называют: 6 степеней свободы. Поворотам соответствуют матрицы, состоящие из блока обычной матрицы поворота 2x2, значения cos в которой расположены на главной диагонали матрицы поворота 3x3, также на этой диагонали находится 1, ярко выражающая отсутствие преобразований по той оси, координаты по которой не меняются. Далее пойдет несколько примеров, по которым легко понять, что именно происходит.

Вокруг X:

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

Вокруг У:

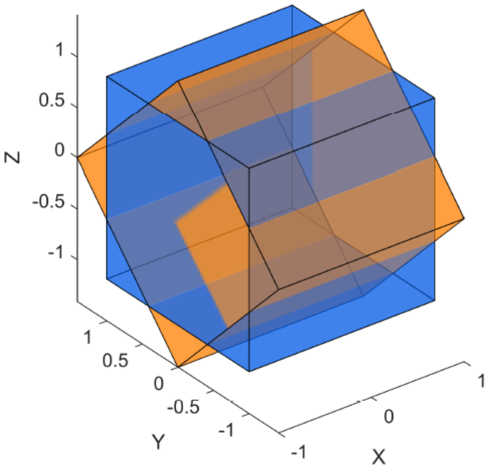
$$\begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta \cdot x + \sin\theta \cdot z \\ y \\ -\sin\theta \cdot x + \cos\theta \cdot z \\ 1 \end{pmatrix}$$

Вокруг Z:

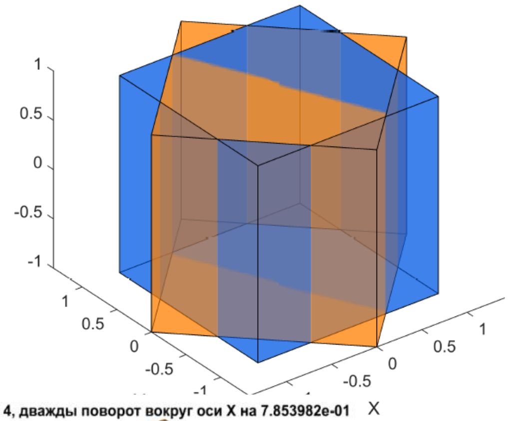
$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta \cdot x - \sin\theta \cdot y \\ \sin\theta \cdot x + \cos\theta \cdot y \\ z \\ 1 \end{pmatrix}$$

А далее берём и просто умножаем матрицу с вершинами куба на матрицу поворота, получаем:

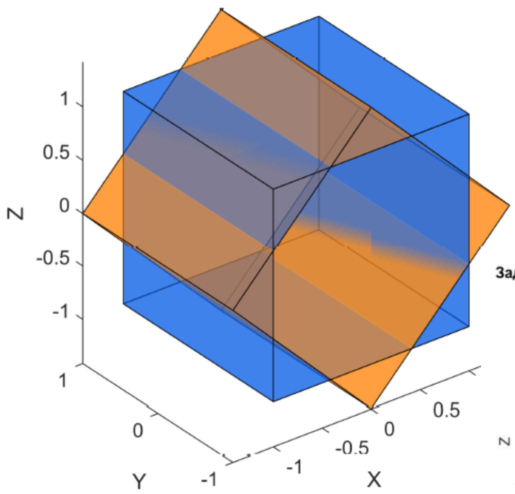
Задание 4, поворот вокруг оси X на 7.853982e-01



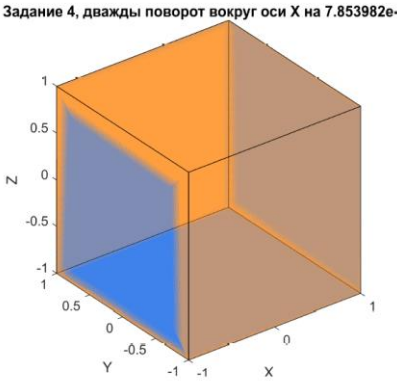
Задание 4, поворот вокруг оси Z на 7.853982e-01



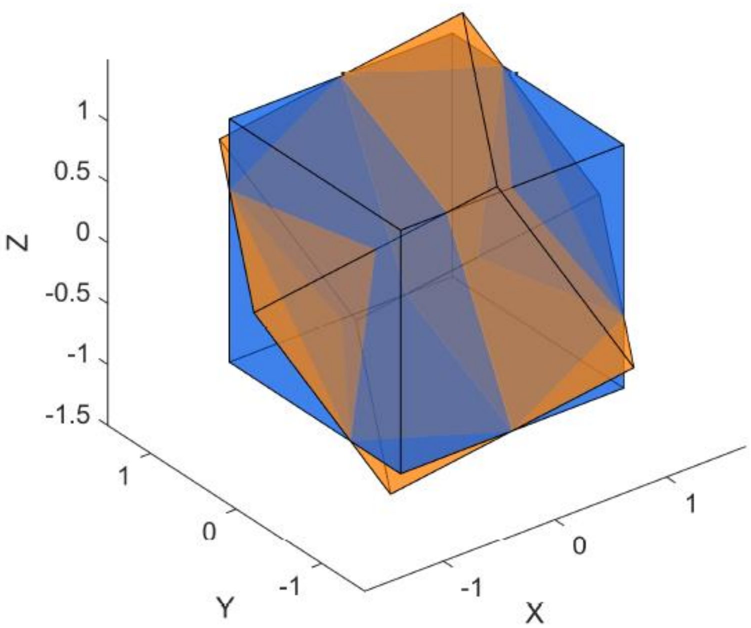
Задание 4, поворот вокруг оси У на 7.853982e-01



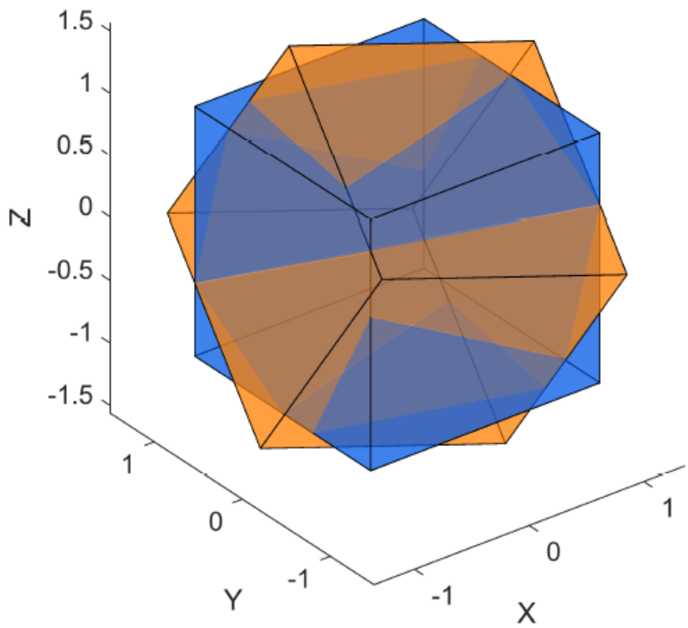
Задание 4, дважды поворот вокруг оси X на 7.853982e-01



вокруг X, потом вокруг Y, потом снова X

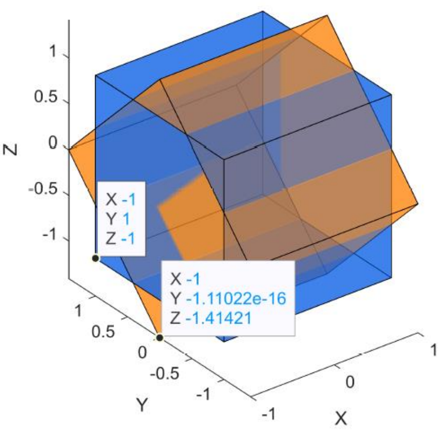


вокруг X, потом Y, снова X, опять Y

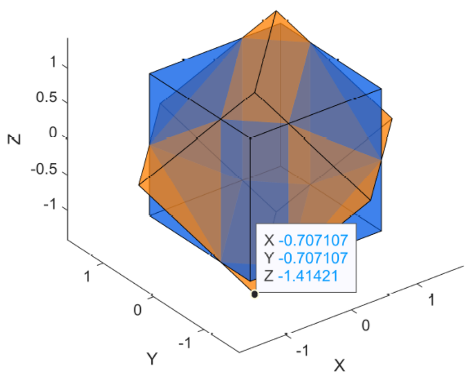


Заметно, что мы так и не пришли в исходное состояние. Это легко представить, разложив сложный поворот по действиям:

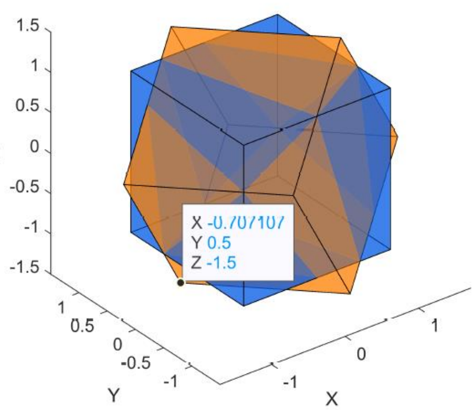
Поворот вокруг X



Поворот вокруг Z



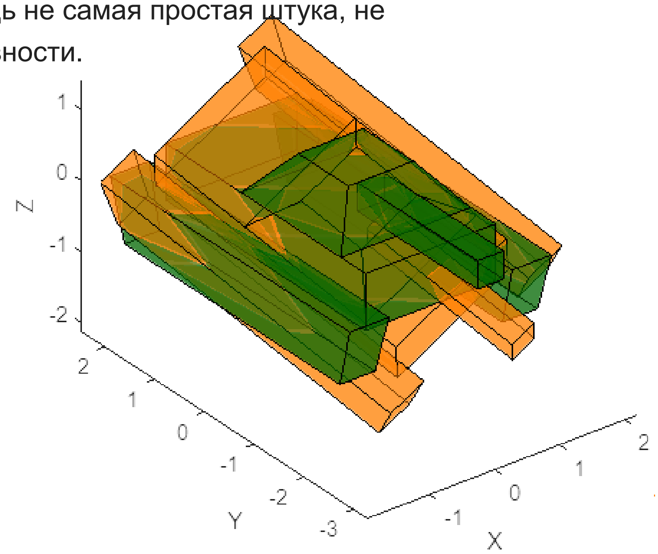
Поворот вокруг X, но ось X на своём месте, а ребро куба - нет



Из этого можно сделать вывод, что повороты - отнюдь не самая простая штука, не обладающая 3D пространстве свойством коммутативности.

Танк, а дальше все закружилось, завертелось и помчалось кувырком:

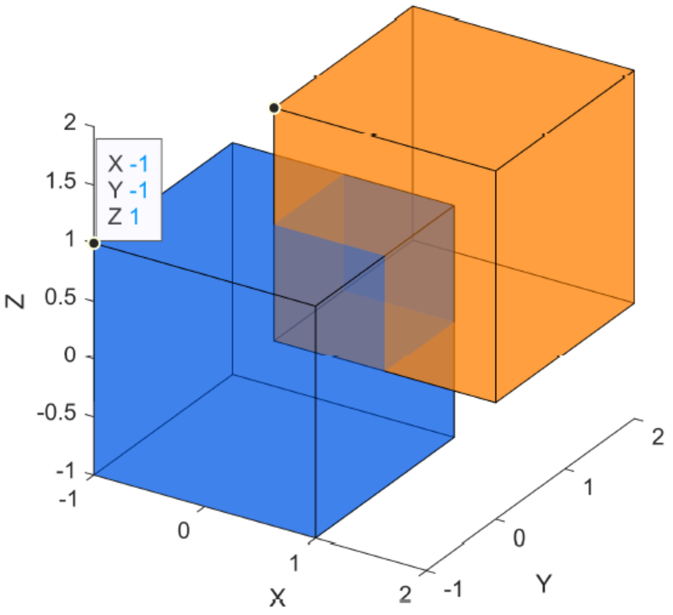
*коллизию не прописывал



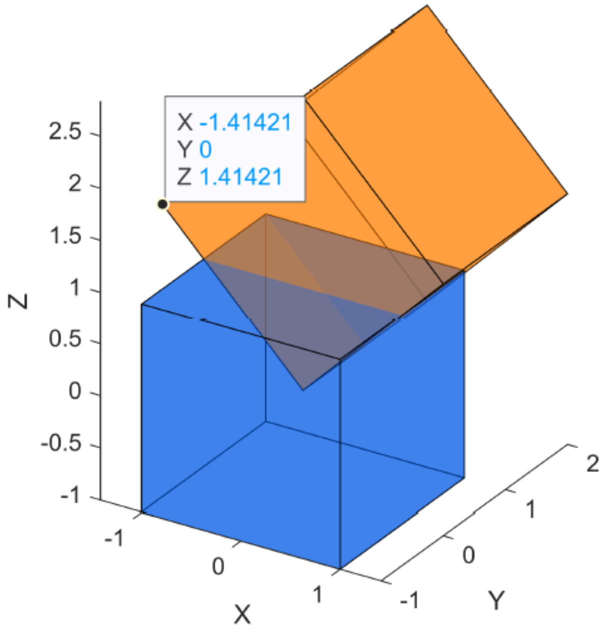
Чтобы куб не крутился, я пригвоздил одну вершину

Вокруг неё он всё ещё может вращаться. Дабы не заниматься всякой слишком сложной математикой, вспомним, что мы уже умеем вращать кубы вокруг осей, умеем двигать кубы. Возьмем и сдвинем его так, чтобы спокойно вращать вокруг Ox, Oy, Oz, а потом поставим на место.

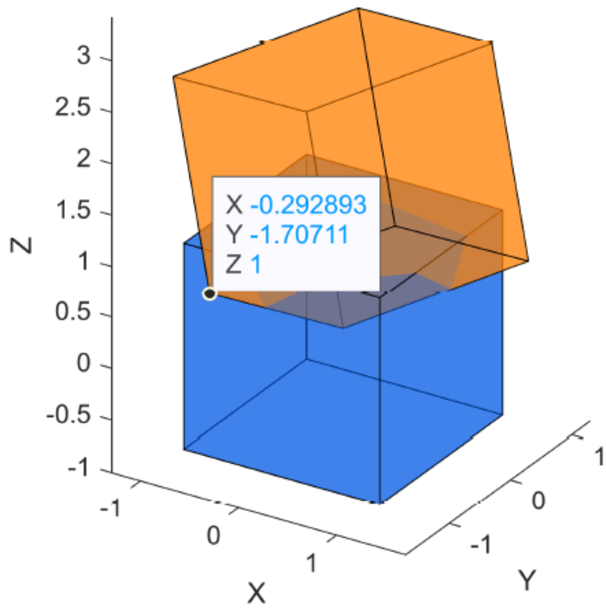
Задание 5 Действие 1



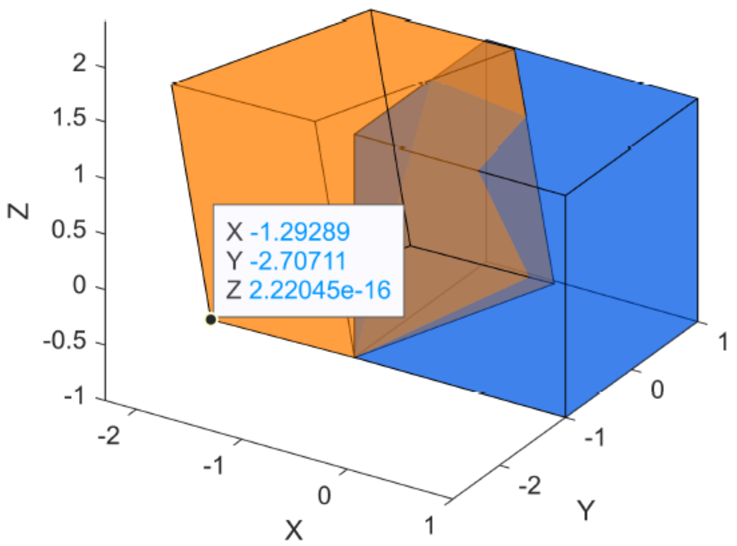
Задание 5 Действие 2



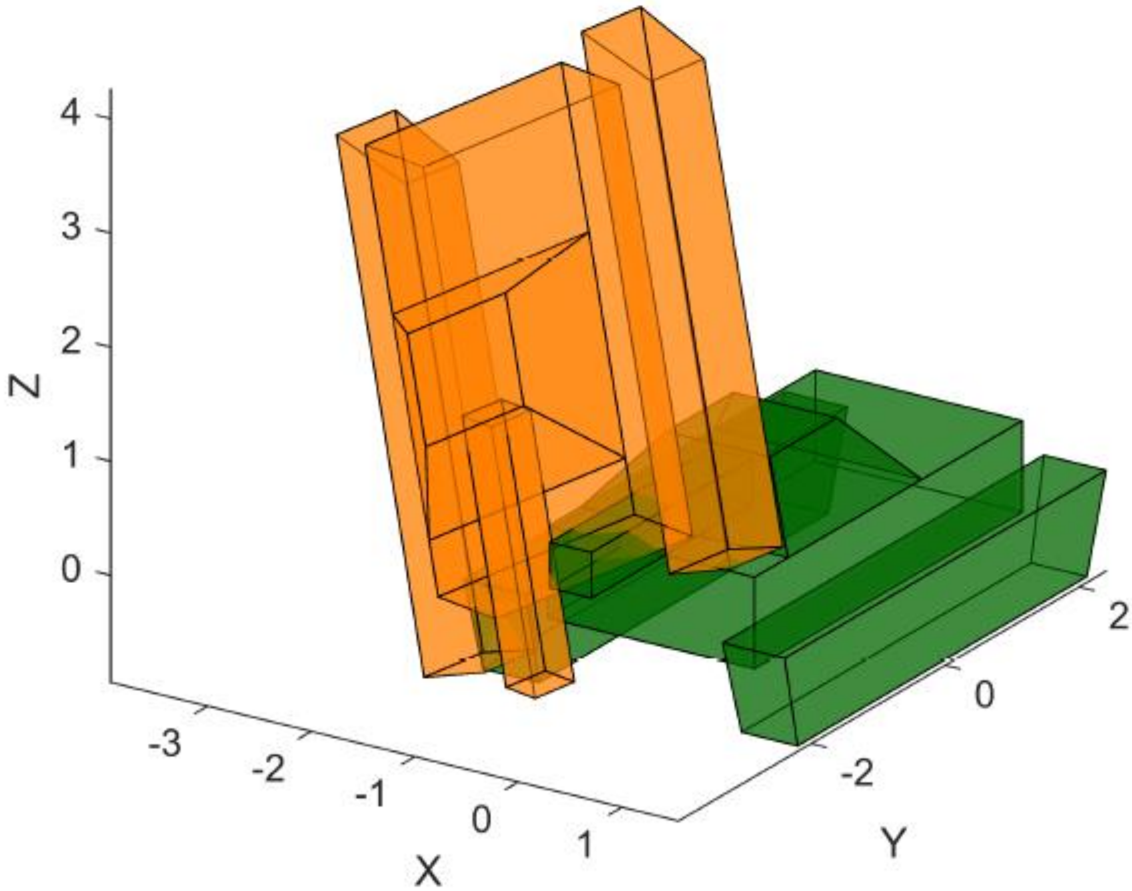
Задание 5 Действие 3



Задание 5 Действие 4



И я ещё сделал так, чтобы **Танк повернулся вокруг гусеницы**



Вернее, вокруг крайней правой нижней точки

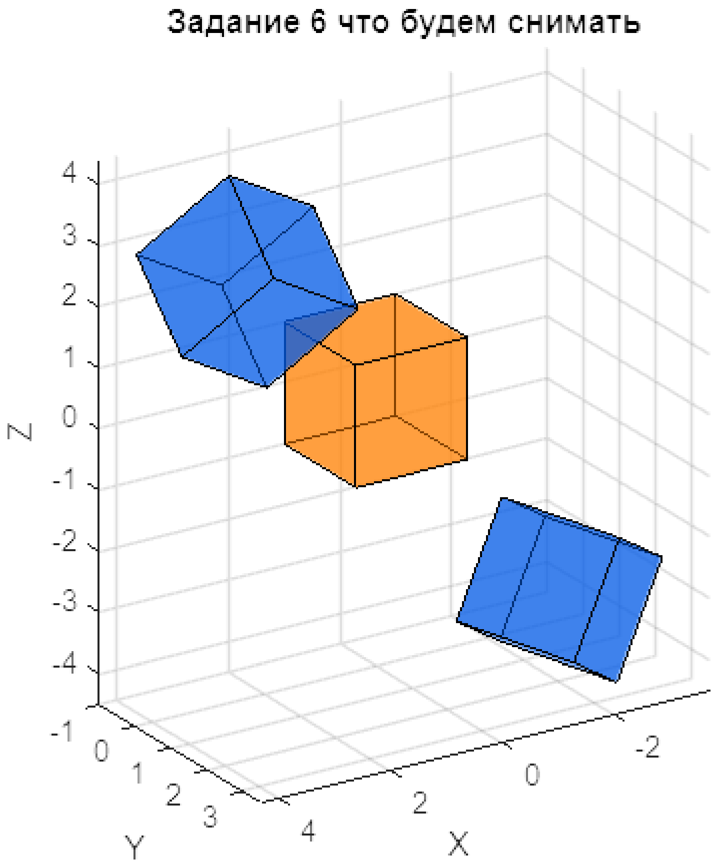
[В коде можно увидеть, как именно это было сделано.]

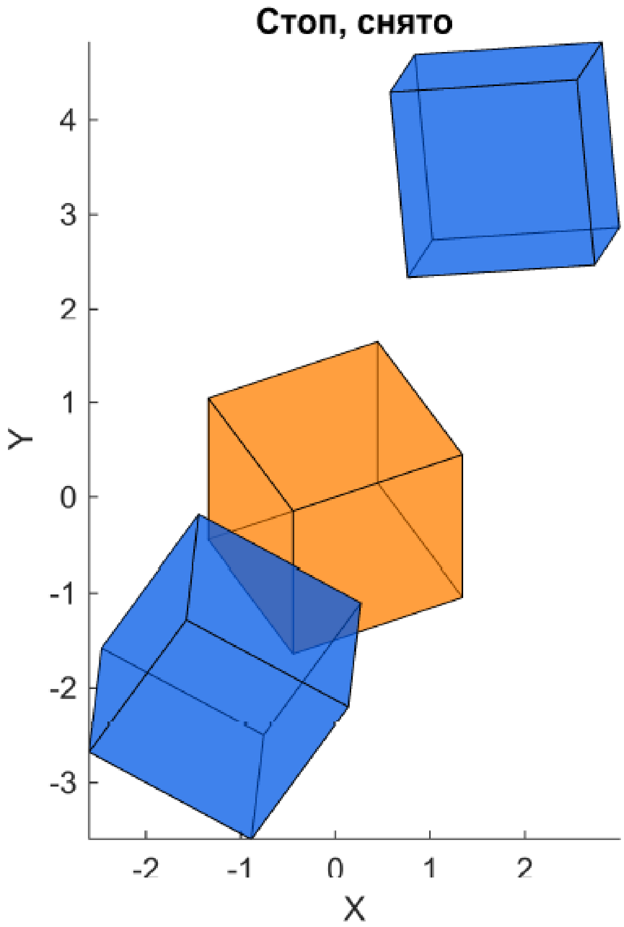
Опять же, повторюсь, можно было сделать и вовсе без сложения в явном виде.
Суть в том, что какая точка окажется в начале координат, вокруг такой вращать и будем.

Такие красивые кубы, что хочется сфотографировать

Реализуем же камеру. *Тут и на следующее в черновике была килотонна объяснений, но, экономя время читателя, не стану в финальной версии их приводить, тем более, они есть тут и на русском* Код матлаба см. по ссылке

Смотреть будем из точки 5, 5, 2.5 в 000.

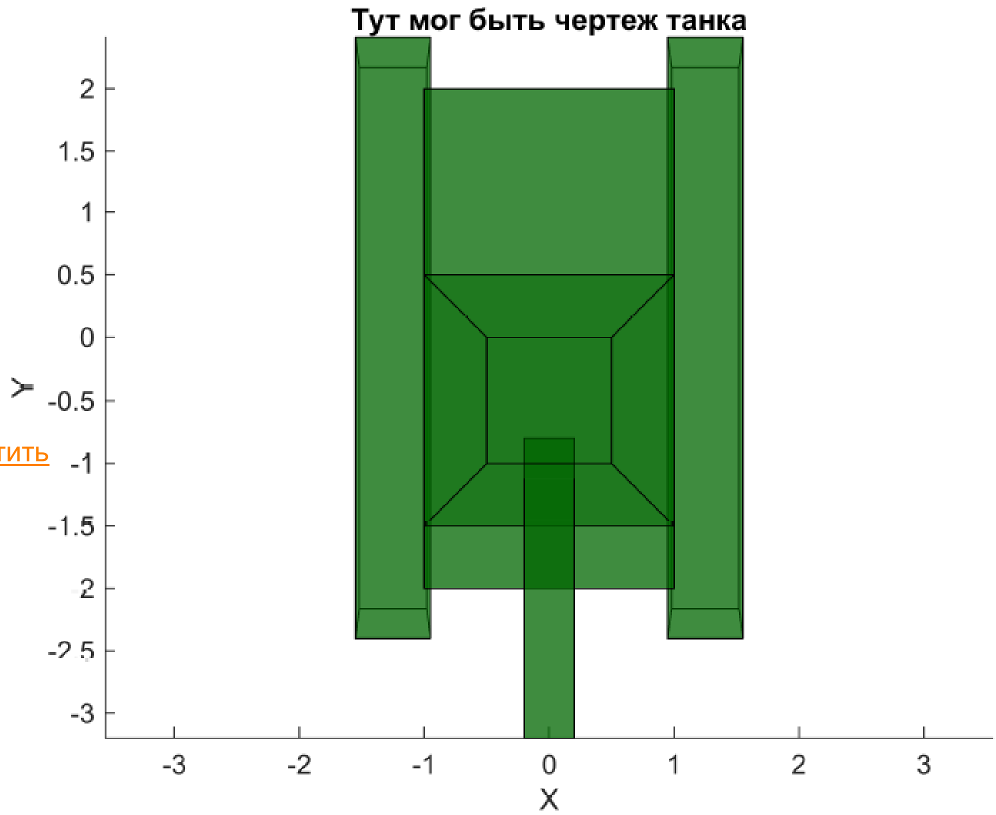




Затем спозиционировал камеру на 6 вверх над центром координат, смотрю вниз, а там танк:

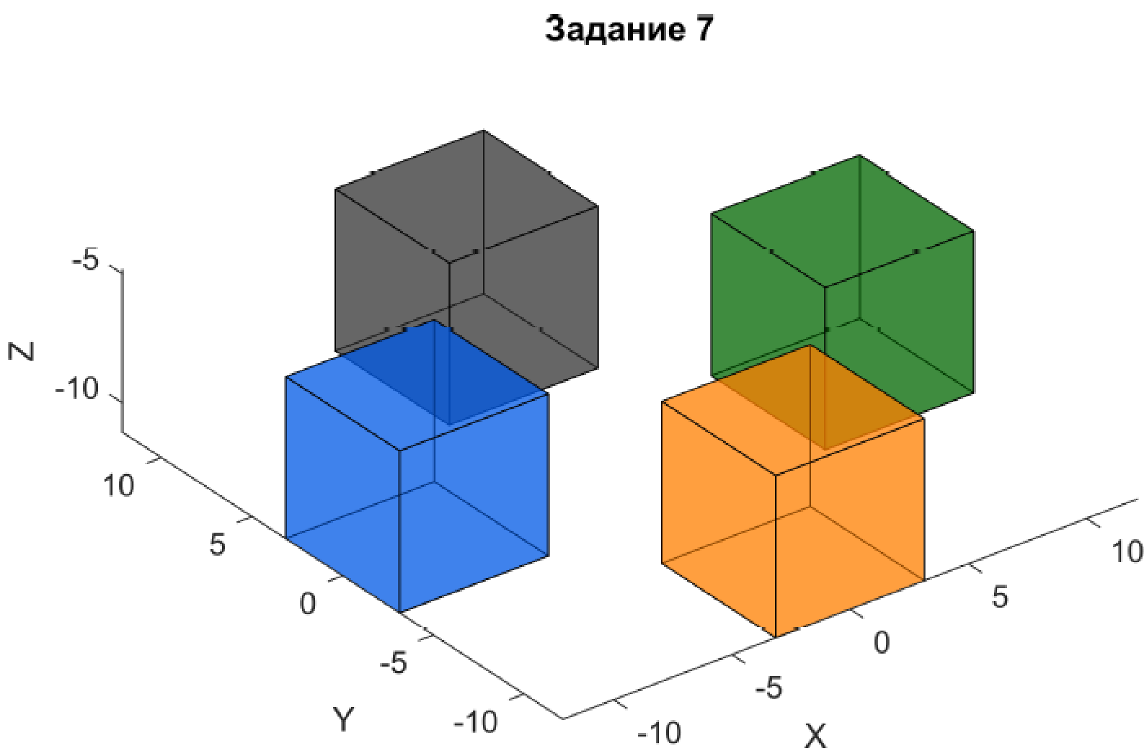
Замечу, определитель матрицы, которую я использовал, равен единице

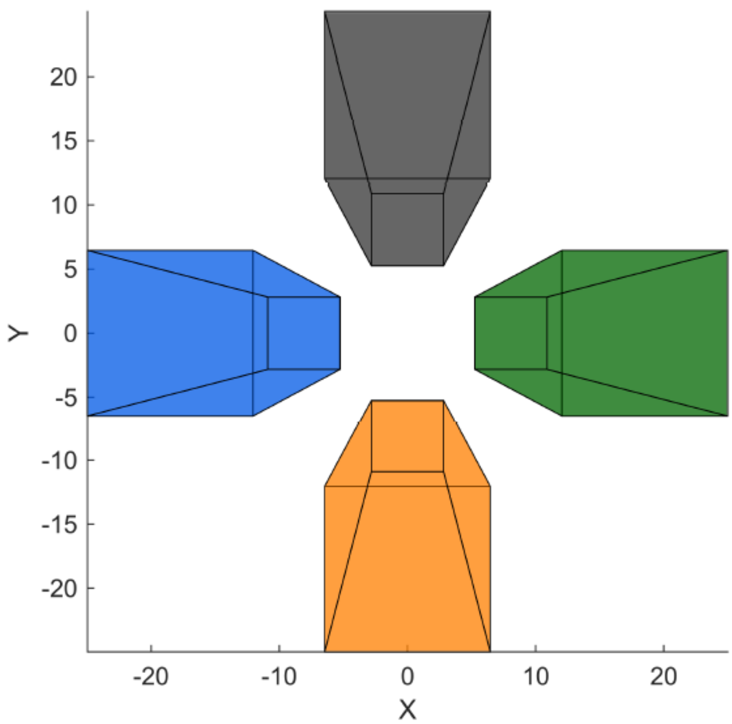
*В интерактивной версии можно покрутить



Но нам не хватает явных перспектив

Путь поле зрения, оно же field of view, будет 90 градусов. Код матлаба см. по ссылке.Объяснения, снова более подробные и тоже на русском, в статье. Тогда:





$$S = \frac{1}{\tan\left(\frac{fov}{2} * \frac{\pi}{180}\right)}$$

$$A_7 = \begin{bmatrix} S & 0 & 0 & 0 \\ 0 & S & 0 & 0 \\ 0 & 0 & -\frac{f}{(f-n)} & -1 \\ 0 & 0 & -\frac{f*n}{(f-n)} & 0 \end{bmatrix}$$