# Project 1

ChatGPT Chat Link: <https://chat.openai.com/share/11f3f87f-7b5f-49b6-b102-9186e6bea19c>

# Task 1

In 2 prompts, I managed to obtain this from ChatGPT:

const transformationMatrix = new Float32Array([

0.3535533845424652, -0.612372398853302, 0.7071067690849304, 0,

0.6123724579811096, 0.3535534143447876, 0.7071067690849304, -0.125,

-0.7071067690849304, -0.7071067690849304, 2.9802322387695312e-8, 0,

0, 0, 0, 1

]);

index.html:

ekran görüntüsü, siyah, grafik, beyaz içeren bir resim

Açıklama otomatik olarak oluşturuldu

# Task 2

Here is my function:

function getModelViewMatrix() {

// start with identity matrix

let modelViewMatrix = createIdentityMatrix();

// translation

const translationMatrix = createTranslationMatrix(0.3, -0.25, 0);

modelViewMatrix = multiplyMatrices(modelViewMatrix, translationMatrix);

// scaling

const scalingMatrix = createScaleMatrix(0.5, 0.5, 1);

modelViewMatrix = multiplyMatrices(modelViewMatrix, scalingMatrix);

// rotations

const rotationXMatrix = createRotationMatrix\_X((30 \* Math.PI) / 180);

modelViewMatrix = multiplyMatrices(modelViewMatrix, rotationXMatrix);

const rotationYMatrix = createRotationMatrix\_Y((45 \* Math.PI) / 180);

modelViewMatrix = multiplyMatrices(modelViewMatrix, rotationYMatrix);

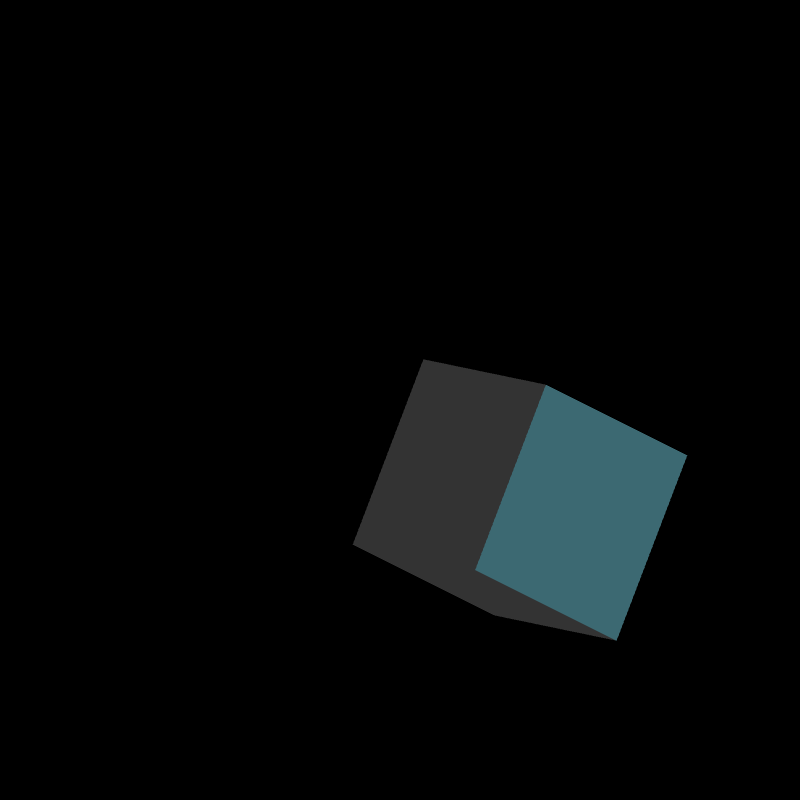
const rotationZMatrix = createRotationMatrix\_Z((60 \* Math.PI) / 180);

modelViewMatrix = multiplyMatrices(modelViewMatrix, rotationZMatrix);

return modelViewMatrix;

}

1. **Initialization:** I start with an identity matrix as the base for the 3D model's transformations.
2. **Translation:** The model is moved by (0.3, -0.25, 0) units, adjusting its position.
3. **Scaling:** The model is scaled by factors of 0.5 along the x and y axes, maintaining its shape.
4. **Rotations:** I apply rotations of 30°, 45°, and 60° around the x, y, and z axes, respectively.
5. **Return:** The modified matrix is returned, representing the transformed state of the cube.



index.html:

# Task 3

In many prompts, I managed to obtain this from ChatGPT:

function getPeriodicMovement(startTime) {

const animationDuration = 10; // total animation duration in seconds

const currentTime = ((Date.now() - startTime) % (animationDuration \* 1000)) / 1000;

const startMatrix = createIdentityMatrix();

const endMatrix = getModelViewMatrix()

const isForward = currentTime < 5;

// Linear interpolation between start and end matrices

const interpolatedMatrix = new Float32Array(16);

for (let i = 0; i < 16; i++) {

const startValue = isForward ? startMatrix[i] : endMatrix[i];

const endValue = isForward ? endMatrix[i] : startMatrix[i];

interpolatedMatrix[i] = startValue + (currentTime % 5) \* (endValue - startValue) / 5;

}

return interpolatedMatrix;

}

1. **Dynamic Elapsed Time:** The **currentTime** is updated dynamically within the function, ensuring continuous progression of animation time based on the elapsed time since the start.
2. **Forward-Backward Animation:** A boolean variable (**isForward**) is introduced to determine whether the animation is currently moving forward or backward, allowing for smooth transitions between the start and end matrices.
3. **Linear Interpolation:** The linear interpolation is adjusted to loop within a 5-second interval, creating a seamless transition between the start and end matrices. The animation smoothly progresses from start to end over 5 seconds and then reverses over the next 5 seconds, forming a continuous loop.

index.html shows a succesfull animation as requested.