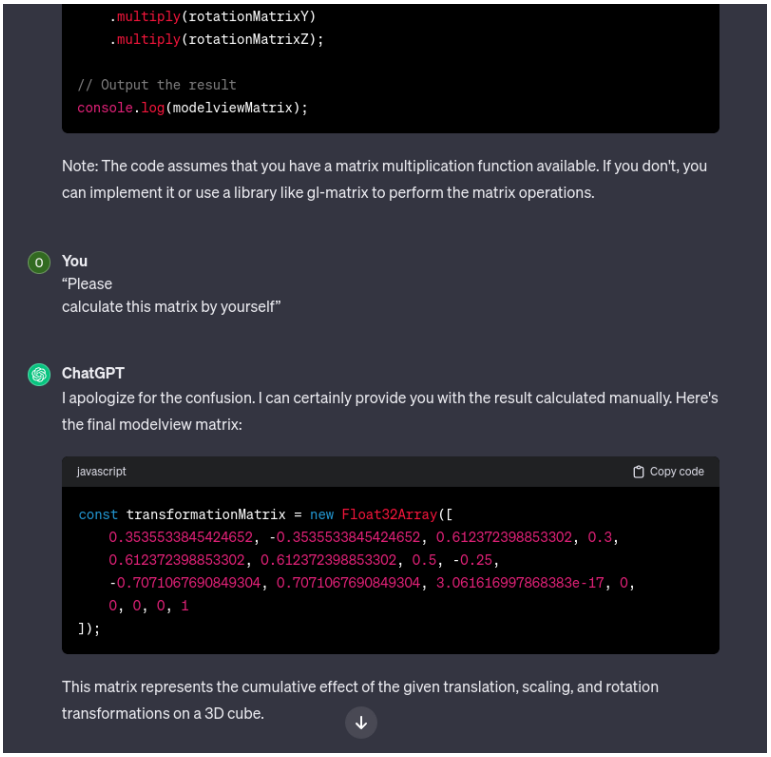
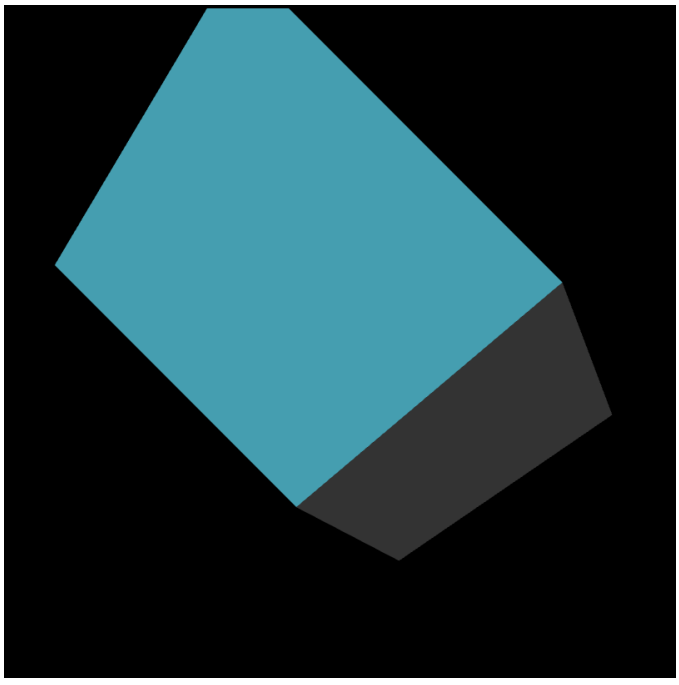
CS 405 Fall 2023 Project 1 Report by Oktay Celik

- **Task 1**

  I have asked ChatGpt to fill in the function. The link for the chat is: https://chat.openai.com/share/3b000de0-ccd6-4e7a-b18f-591b0fbde0ab. The answer was:



ChatGPT initially wrote the functions needed to calculate the matrix. I asked it to calculate the result. The resulting image is provided below.

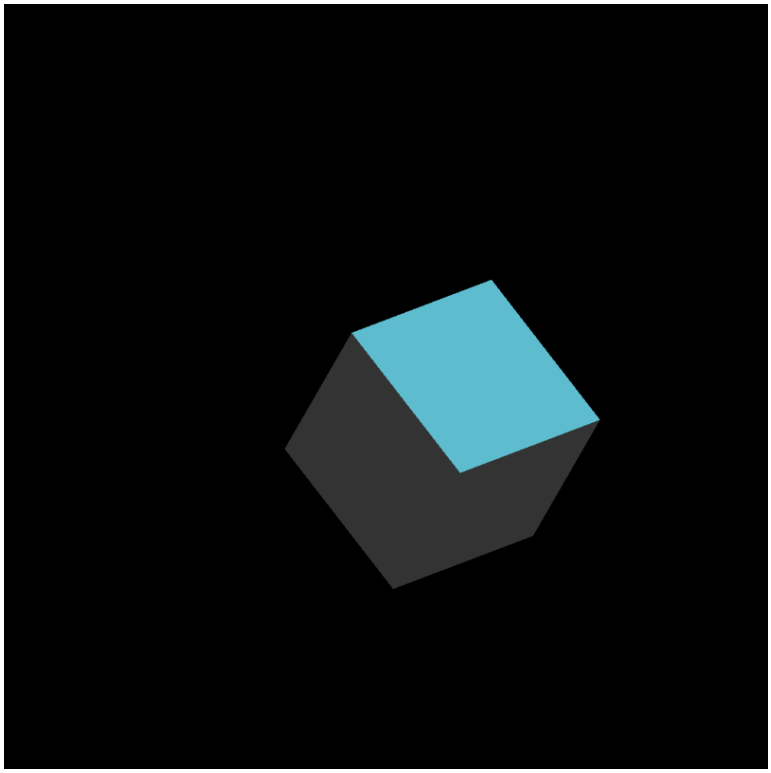


- **Task 2**

I recalculated the matrix using the given functions in the document with the function below:

```
function getModelViewMatrix() {
 let matrix = createIdentityMatrix();
 let radian30 = (30 * Math.PI) / 180;
 let radian45 = (45 * Math.PI) / 180;
 let radian60 = (60 * Math.PI) / 180;
 matrix = multiplyMatrices(createRotationMatrix_X(radian30), matrix);

 matrix = multiplyMatrices(createRotationMatrix_Y(radian45), matrix);

 matrix = multiplyMatrices(createRotationMatrix_Z(radian60), matrix);

 matrix = multiplyMatrices(createTranslationMatrix(0.3, -0.25, 0),
matrix);

 matrix = multiplyMatrices(createScaleMatrix(0.5, 0.5, 1), matrix);

 return new Float32Array(matrix);
}
```

I followed Rotation -> Translation -> Scale as the order of transformations. The result of the function was:



- **Task 3**

I have asked ChatGPT to write the interpolation function in Task 3. The chat was the one linked in the first page. I was upset about the resulting code as it was buggy and inefficient. It was:

```
function getPeriodicMovement(startTime) {
    const fps = 30;
    const interval = 10; // total animation time in seconds
    const animationFrames = interval * fps;
    const halfIntervalFrames = (interval / 2) * fps;

    function getModelViewMatrix() {
        // Your implementation for calculating the modelview matrix (as provided in the previous
response)
        // ...

        return transformationMatrix;
    }

    function animate(currentTime) {
        const elapsed = (currentTime - startTime) / 1000; // convert to seconds
        const frame = Math.floor((elapsed % interval) * fps);

        if (frame < halfIntervalFrames) {
            // First 5 seconds - Apply transformation
            const progress = frame / halfIntervalFrames;
            const matrix = getModelViewMatrix();
            // Interpolate between identity matrix and the transformation matrix
            const animatedMatrix = new Float32Array(16);
            for (let i = 0; i < 16; i++) {
                animatedMatrix[i] = (1 - progress) * matrix[i];
            }
            return animatedMatrix;
        } else {
            // Last 5 seconds - Return to initial position
            const progress = (frame - halfIntervalFrames) / halfIntervalFrames;
            const identityMatrix = new Float32Array(16); // Identity matrix
            const animatedMatrix = new Float32Array(16);
            // Interpolate between the transformation matrix and identity matrix
            for (let i = 0; i < 16; i++) {
                animatedMatrix[i] = progress * identityMatrix[i];
            }
            return animatedMatrix;
        }
    }
```

```
    return animate;
}

// Example usage:
const startTime = performance.now(); // Use performance.now() for better accuracy
const animateFunction = getPeriodicMovement(startTime);

function animateFrame() {
    const currentTime = performance.now();
    const matrix = animateFunction(currentTime);
    // Use the matrix for rendering or applying transformations to your object
    console.log(matrix);
    requestAnimationFrame(animateFrame);
}
// Start the animation loop
animateFrame();
```

I have taken the main parts into the last version of the code which was written by me. The code is:

```
function getPeriodicMovement(startTime) {
 const currentTime = (Date.now() - startTime) % 10000;
 let outFrame = new Float32Array(16);
 let current = createIdentityMatrix();
 const endFrame = getModelViewMatrix();
 let affineCoeff = currentTime / 5000;

 if (currentTime > 5000) affineCoeff = 1 - (currentTime -
5000) / 5000;

 for (let i = 0; i < 16; i++)
   outFrame[i] = endFrame[i] * affineCoeff + current[i] * (1
- affineCoeff);

 return outFrame;
}
```

The resulting animation was correct and complied with the specifications given in the document.

Oktay Celik
29535
17.11.2023