CS405 Project 1 Report

Akif Işıtan

Task 1)

Pasted the transformation prompt into ChatGPT. Resulting image is provided in Figure 1.

Chat URL: https://chat.openai.com/share/c4897530-e102-458d-b697-2df571d7e739

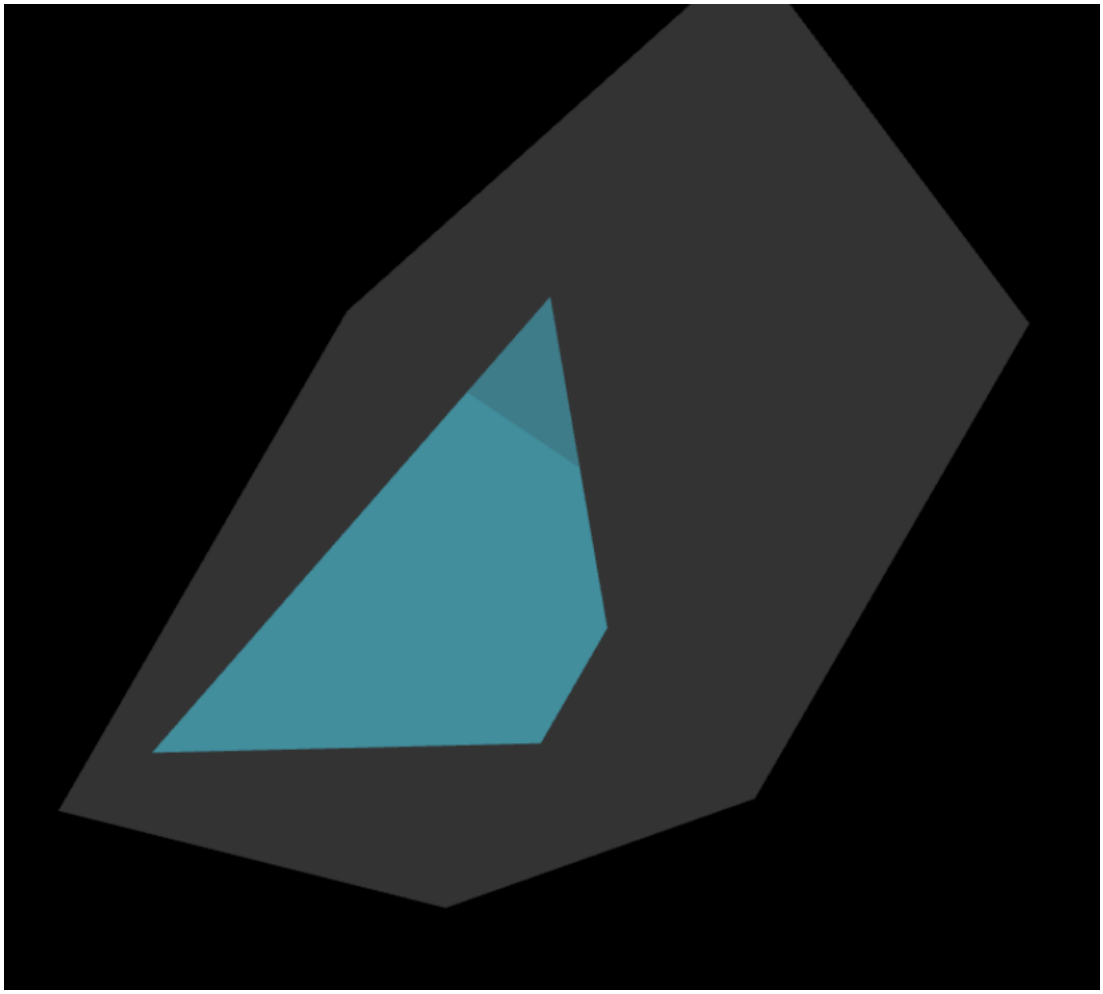

Figure 1. Cube after applying transformations according to ChatGPT calculation

Task 2)

The transformation matrix was calculated using the provided helper given functions with JavaScript. The order of multiplying the matrices is reverse of the application of transformations. The transformations were applied in the following order:

1) Translation: 0.3 units in x-axis and -0.25 units in y-axis

2) Scaling: 0.5 by x-axis and 0.5 by y-axis

3) Rotation: 30 degrees on x-axis, 45 degrees on y-axis and 60 degrees on z-axis

The angles must be changed into radians for the helper function, which can be done by the following formula:

Angle in Radians = Angle * (π / 180)

Transformation Matrix = RotateZ(60 * (π / 180)) x Rotate Y(45 * (π / 180)) x

RotateX(30 * (π / 180)) x Scale(0.5, 0.5, 1) x Translate(0.3, -0.25, 0)

The resulting cube can be seen in Figure 2.

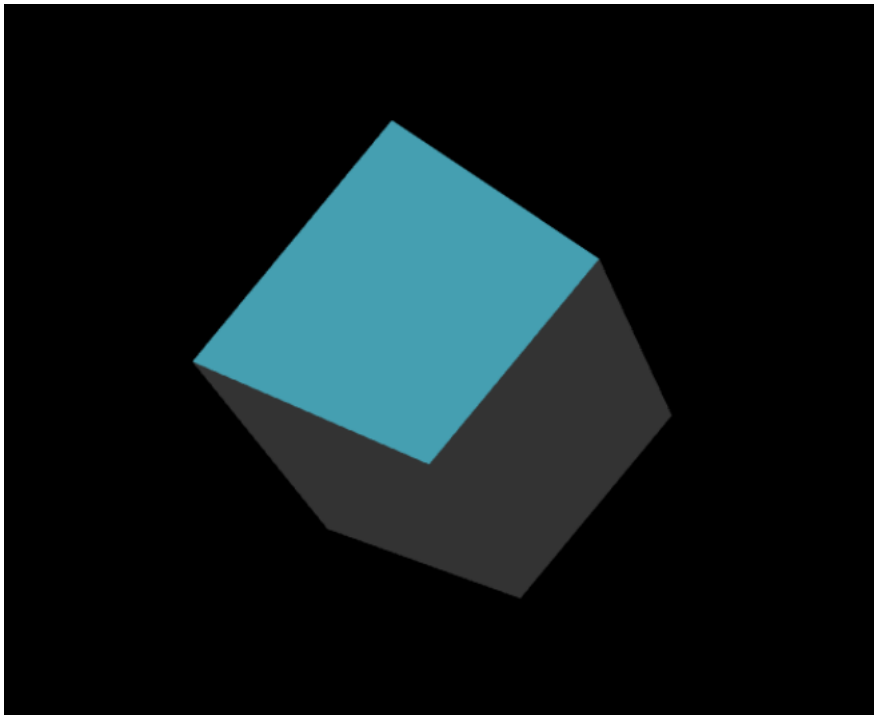

Figure 2. Cube after applying transformations

The model view matrices calculated are different because ChatGPT 3.5 is just a language model, it has no way of making the necessary calculations that would result in the matrix. It only makes guesses, which can be seen from the different answers it gives each time it is asked to recalculate.

Task 3)

Gave a detailed prompt to ChatGPT. Here is the prompt:

*I want you to write the code to animate a transformation calculated with a period of 10 seconds. The first 5 seconds the cube should transform from its initial position to the target position, the next 5 seconds the cube should transform from the target position to the initial position. The target position is the result of the cube after the transformations applied below (in the getModelViewMatrix), meanwhile the initial position is the transformationless cube. Here is the function used to create the model view matrix: ``` function getModelViewMatrix() { // calculate the model view matrix by using the transformation // methods and return the modelView matrix in this method const translate = createTranslationMatrix(0.3, -0.25, 0); const scale = createScaleMatrix(0.5, 0.5, 1); const rotateX = createRotationMatrix_X(30 * (Math.PI / 180)) const rotateY = createRotationMatrix_Y(45 * (Math.PI / 180)) const rotateZ = createRotationMatrix_Z(60 * (Math.PI / 180)) // TransformationMatrix = RotateZ x RotateY x RotateX x Scale x Translate const firstMul = multiplyMatrices(rotateZ, rotateY); const secondMul = multiplyMatrices(firstMul, rotateX); const thirdMul = multiplyMatrices(secondMul, scale); const fourthMul = multiplyMatrices(thirdMul, translate); const transformationMatrix = new Float32Array(fourthMul); return getTransposeMatrix(transformationMatrix) } Here is the snippet to demonstrate the usage of the modelViewMatrix: ``` const startTime = Date.now(); let modelViewMatrix = createIdentityMatrix(); function render() { // Clear the canvas gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT); modelViewMatrix = getPeriodicMovement(startTime); // Pass the model-view and projection matrices to the shader gl.uniformMatrix4fv(modelViewMatrixLocation, false, modelViewMatrix); gl.uniformMatrix4fv(projectionMatrixLocation, false, projectionMatrix); let normalMatrix = glMatrix.mat4.create(); normalMatrix = glMatrix.mat4.transpose(normalMatrix, modelViewMatrix); normalMatrix = glMatrix.mat4.invert(normalMatrix, normalMatrix); gl.uniformMatrix4fv(normalMatrixLocation, false, normalMatrix); // Draw the cube gl.drawArrays(gl.TRIANGLES, 0, vertexData.length / 6); requestAnimationFrame(render); } requestAnimationFrame(render); // Start the render loop ``` make sure to write the code in javascript by implementing the getPeriodicMovement(startTime) function.*

The resulting code was tested and confirmed to work. Here is a brief explanation of what the code does according to ChatGPT.

The task involves animating a cube's transformation over a 10-second period, transitioning from its initial position to a target position in the first 5 seconds and then back to the initial position in the next 5 seconds. This is achieved by implementing the getPeriodicMovement(startTime) function. It tracks time using Date.now(), calculates progress within the animation period, and performs linear interpolation between the initial identity matrix and the target transformation matrix (obtained from getModelViewMatrix()) based on the progress. The resulting matrix is integrated into the WebGL rendering loop, updating the cube's transformation matrix as time progresses for smooth animation.

Chat URL: https://chat.openai.com/share/825afa04-e893-4a9a-a785-0e2a963462c5

A frame of the resulting motion can be seen in Figure 3.

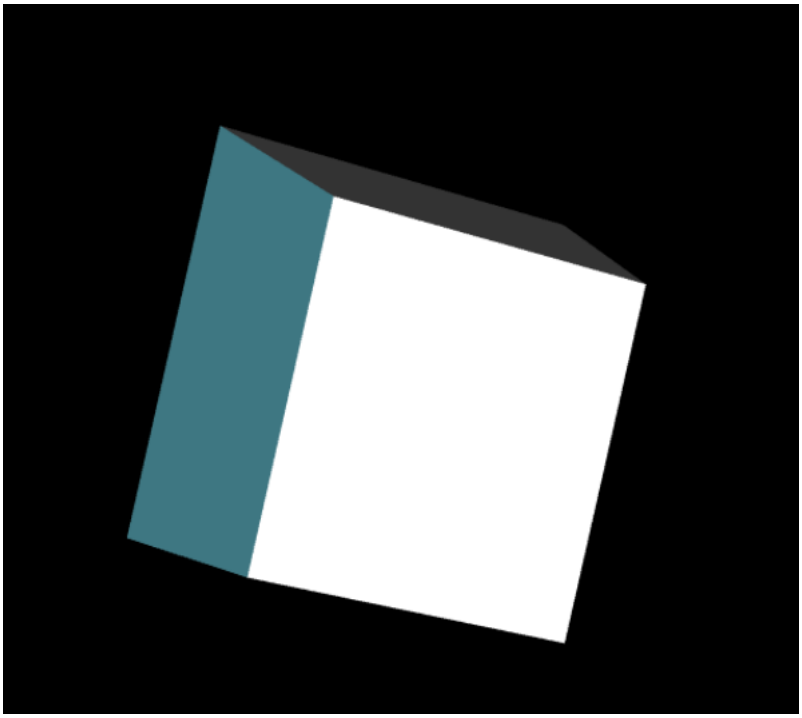

Figure 3. A frame of motion where cube transforms from initial position to target position