CS405 HomeWork 1 Final Report

The goal of this project is to explore how artificial intelligence, specifically ChatGPT, can be integrated into a computer graphics pipeline to calculate and apply 3D transformations, and create smooth object animations.

Github repository link: <https://github.com/egegunes3/CS405_HW1>

**Task1:** This part was about giving chatgpt a predefined prompt and asking it to perform the operations that were told to be done in the prompt sent.

<https://chatgpt.com/share/67113208-c4d4-8008-a5db-35d8f1f8d12f>

\* @TASK1 Calculate the model view matrix by using the chatGPT

\*/

function getChatGPTModelViewMatrix() {

const transformationMatrix = new Float32Array([

0.17677669, -0.28661165, 0.7391989, 0.3,

0.30618623, 0.36959946, 0.2803301, -0.25,

-0.35355338, 0.17677669, 0.61237246, 0.0,

0.0, 0.0, 0.0, 1.0

]);

return getTransposeMatrix(transformationMatrix);

}

A blue and black cube

Description automatically generated

(3D Object of Task 1)

**Task 2:** Our task in this task is to do the parts that we had chat gpt do in the previous task by ourselves. We create the desired three-dimensional object by performing certain translation, rotation and scaling operations by ourselves.

function getModelViewMatrix() {

// Step 1: In this part we are creating the identity matrix

let modelViewMatrix = createIdentityMatrix();

// Step 2: In this part we are defining the scale matrix

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

// Step 3: In this part we are defining the rotation matrices with the given angles

const angleX= 30 \* (Math.PI / 180); // Convertion of 30 degrees

const angleY = 45 \* (Math.PI / 180); // Convertion 45 degrees

const angleZ = 60 \* (Math.PI / 180); // Convertion 60 degrees

const rotationMatrixX = createRotationMatrix\_X(angleX); // 30 degrees

const rotationMatrixY = createRotationMatrix\_Y(angleY); // 45 degrees

const rotationMatrixZ = createRotationMatrix\_Z(angleZ); // 60 degrees

// Step 4: In this part we are combining the rotation matrices by multiplication (Z \* Y \* X)

const multipliedMatrix = multiplyMatrices(rotationMatrixZ, multiplyMatrices(rotationMatrixY, rotationMatrixX));

// Step 5: In this part we are applying the scale transformation

modelViewMatrix = multiplyMatrices(scalingMatrix, modelViewMatrix);

// Step 6: In this part we are applying the rotation transformation

modelViewMatrix = multiplyMatrices(multipliedMatrix, modelViewMatrix);

// Step 5: In this part we are applying the scale transformation

modelViewMatrix = multiplyMatrices(scalingMatrix, modelViewMatrix);

// Step 7: In this part we are defining the translation matrix

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

// Step 8: In this part we are applying the translation transformation

modelViewMatrix = multiplyMatrices(translationMatrix, modelViewMatrix);

// Return the modelViewMatrix

return new Float32Array(modelViewMatrix);

}

**A blue and black rectangular object

Description automatically generated**

(3D Object of Task 2)

**Differences Between Task1 and Task2:** I interpret this difference as the application of multiple transformations (translation, scaling, rotation). In 3D transformations, the order of the transformations (translation, scaling, rotation) being different or the matrix multiplication being done incorrectly affects the result. I think that chatgpt is doing the order of the transformations in a different order and while doing these, it is also making a mistake in matrix multiplication, and this explains this object that is close to the cube shape but has crooked edges.

**Task3:** In this task, we want chatgpt to animate the matrix we made in task2.

<https://chatgpt.com/share/67113091-55e8-8008-b25f-e516c35ea73f>

function getPeriodicMovement(startTime) {

// Duration of one full cycle (10 seconds)

const cycleDuration = 10000;

// Calculate the elapsed time since the animation started

const elapsedTime = Date.now() - startTime;

// Normalize time to a range between 0 and 1 (representing progress in the cycle)

const t = (elapsedTime % cycleDuration) / cycleDuration;

// Interpolation factor: Moves from 0 to 1 in the first 5 seconds, and 1 to 0 in the next 5 seconds

const interpolationFactor = t < 0.5 ? (t \* 2) : (2 - t \* 2);

// Initial position [0, 0, 0]

const initialPosition = [0, 0, 0];

// Target position [0.3, -0.25, 0]

const targetPosition = [0.3, -0.25, 0];

// Interpolated position (linear interpolation between initial and target)

const interpolatedPosition = [

initialPosition[0] + interpolationFactor \* (targetPosition[0] - initialPosition[0]),

initialPosition[1] + interpolationFactor \* (targetPosition[1] - initialPosition[1]),

initialPosition[2] + interpolationFactor \* (targetPosition[2] - initialPosition[2])

];

// Create translation matrix based on interpolated position

const translationMatrix = createTranslationMatrix(

interpolatedPosition[0],

interpolatedPosition[1],

interpolatedPosition[2]

);

// Apply the translation matrix to the model view matrix from Task 2

let modelViewMatrix = getModelViewMatrix();

modelViewMatrix = multiplyMatrices(translationMatrix, modelViewMatrix);

return new Float32Array(modelViewMatrix);

}