# LWJGL Coding exercise – 2D Transformations

Download the week 3 prac project from GitHub classroom:

<https://classroom.github.com/a/-Pqv2bnk>

REMEMBER TO FETCH AND PULL THE MOST RECENT VERSION OF THE COMP3170 LWJGL LIBRARY

The package contains four files:

* Week3.java – the main program, implementing IWindowListener
* Plane.java – code to model and draw a simple plane (shown below)
* vertex.glsl – a basic vertex shader with vertex colours
* fragment.glsl – a basic fragment shader with vertex colours

If you run the code as is, it should draw the plane without any transformation:

Shape

Description automatically generated

## Step 1 – Complete the matrix code

The Plane class defines three static methods:

* translationMatrix()
* rotationMatrix()
* scaleMatrix()

Each of these methods writes the specified kind of matrix form into a pre-allocated destination matrix. You will want to use these methods to create your model matrix.

The code for translationMatrix has been provided. Complete the other two methods.

## Step 2 – Add a model matrix

In **vertex.glsl**: Add code to the vertex shader to allow a **model matrix** to be passed as a uniform. Use this code to transform the vertex before writing it to gl\_Position.

In **Plane.java**: Create a model matrix and pass it to the shader using setUniform when drawing the plane.

How would you set the matrix to achieve the following output?

a)Shape

Description automatically generatedb)Shape

Description automatically generated

Vector2f offset = **new** Vector2f(0.0f, 0.0f); *translationMatrix*(offset.x, offset.y, modelMatrix);

**float** rotation = *TAU*; *rotationMatrix*(rotation, modelMatrix);

Vector2f scale = **new** Vector2f(0.0f, 0.0f); *scaleMatrix*(scale.x, scale.y, modelMatrix);

Vector2f offset = **new** Vector2f(0.0f, 0.0f); *translationMatrix*(offset.x, offset.y, modelMatrix);

**float** rotation = *TAU*/-4; *rotationMatrix*(rotation, modelMatrix);

Vector2f scale = **new** Vector2f(0.0f, 0.0f); *scaleMatrix*(scale.x, scale.y, modelMatrix);

c)Shape

Description automatically generatedd)Graphical user interface, application

Description automatically generated

Vector2f offset = **new** Vector2f(-0.65f, 0.65f); *translationMatrix*(offset.x, offset.y, modelMatrix);

**float** rotation = *TAU*/8; *rotationMatrix*(rotation, modelMatrix);

Vector2f scale = **new** Vector2f(0.5f, 0.5f); *scaleMatrix*(scale.x, scale.y, modelMatrix);

Vector2f offset = **new** Vector2f(0.5f, -0.5f); *translationMatrix*(offset.x, offset.y, modelMatrix);

**float** rotation = *TAU*/-4; *rotationMatrix*(rotation, modelMatrix);

Vector2f scale = **new** Vector2f(0.5f, 0.5f); *scaleMatrix*(scale.x, scale.y, modelMatrix);

## Step 3 – Add animaton

Following the example in lectures, for adding and using deltaTime, update the code to make the ship move along a circular path like this:

A picture containing icon

Description automatically generatedA picture containing graphical user interface

Description automatically generatedIcon

Description automatically generated with low confidenceIcon

Description automatically generated with medium confidence

Hints:

* Initialise the plane with its starting position, angle and size.
* On every frame:
  + Make the plane rotate on the spot
  + Add forward motion in the plane’s local coordinate space.