**Model Rendering – Supporting document**

Order of TRS matrix application

The order that TRS (Translate, Rotation and Scale) matrices are applied in code is essential to how the program draws them in the console window due to how each matrix changes the theoretical shape that is to be displayed. When drawing shapes in a console window, they come from the origin (0,0,0) and are then altered in the order that the code applies them.

For example; If we were to use a translate matrix to draw a cube at the world origin first and then rotated said matrix, the shape would rotate around it’s Centrepoint. However, if we were to rotate first and then draw the cube with a translate matrix, it would be set to rotate around the world origin instead as if it were orbiting a planet.

World, View and Projection matrices

World (or Model), View and Projection matrices are used to control separate parts of the drawn model/world inside the screen. Generally, they derive from the world origin, with each matrix transferring into one another. The model coordinates change through the model matrix into the world coordinates then through the view matrix into the camera coordinates and then through the projection matrix to the homogeneous coordinates.

When being applied to functions, the world matrix is in charge of where in the world the model is positioned and transformed, the view matrix determines the position of the camera and the projection matrix determines whether the camera has an orthographic view or a perspective view on the world.

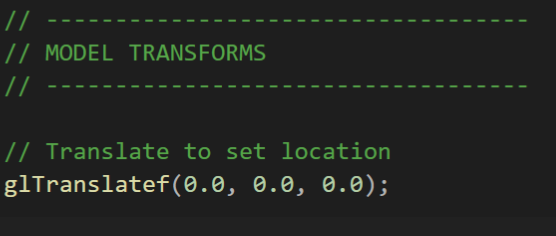
Construction of TRS matrices

When constructing TRS matrices, there is a set way to set them up, these ways differ between the translate, rotation and scale matrices.

Translate matrices are applied to model matrices and shown on paper like this:

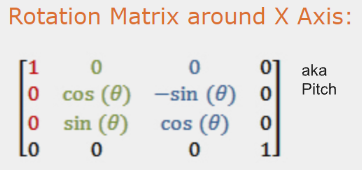
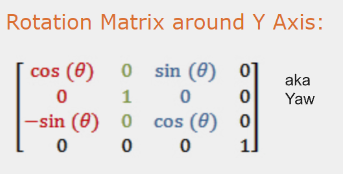


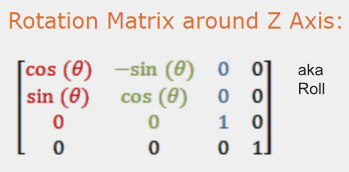
This is actually an identity matrix and has a translate matrix applied to it where “Translation.x” “Translation.y” and “Translation.z” are positioned. When setting this up in code, you type it as such:



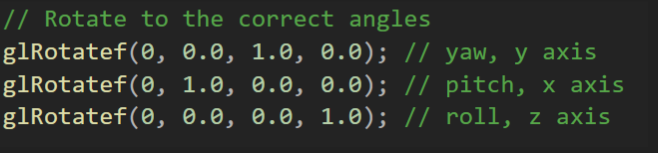
This matrix takes the x, y and z coordinates and uses matrix multiplication to change the model’s position in space.

Rotation matrices work around three types of rotation; Yaw along the Y axis, Pitch along the X axis and Roll along the Z axis. Below are how Yaw, Pitch and Roll are applied to model matrices.

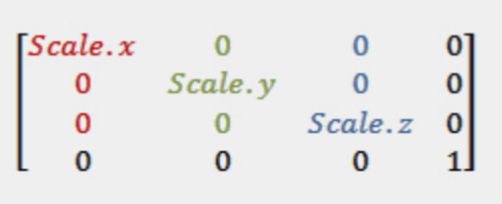




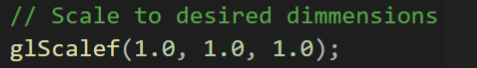
Rotation matrices use Euler angles to rotate around an axis and use degrees to measure their rotation. When applying this in code, it should be applied in this exact order or it will possibly break other parts of the program:



Lastly, Scale matrices are applied to Model matrices to change their size in space, they are written like this:



The “Scale.x”, “Scale.y” and “Scale.Z” alter the coordinates of the model’s limits, making them appear larger on the screen. These coordinates may be scaled separately, creating strange shapes other than just larger versions of the model they are applied to. In code, scale matrices are shown like this:



Disclaimer: All of the code examples shown in this document were applied using OpenGL. If applied to other libraries, they may not work as intended.