# Report GD 3

# 2022/04/10

# Tumi Jourdan

# 2180153

## The Line between Game Design and Mathematics thins once again.

I just wanted to understand shaders and then cameras. Understanding has its price in work. Most of the project has been based on revising and using matrices. I would like to say that I had already covered matrices but that was 1.5 years ago in maths 2, although it did help.

The reason I wanted to learn about shaders and cameras is that it was a large gap in my knowledge. When watching tutorials, I would simply quit when I saw any coding relating to shaders and understood less when cameras were used for rendering. After looking into how shadows are rendered, I found that I could not fathom how the information was being passed between cameras/spaces and the calculations. I then decided that the best place to start is the basics of how the computer renders 3d objects. I had slightly touched on this last prototype but did not properly understand the math magic happening in the background.

To properly understand how a computer displays graphical information my lecturer suggested trying the edx course : <https://www.edx.org/course/computer-graphics-2>. The course starts with revising matrix operations, then manipulation of matrices and finally implementation of learned methods in Opengl and GLM.

A lot of the work that I did was learning mathematics, and so I will skip the parts that can easily be understood from my journal and rather document important mathematical and graphical concepts that I had learned during the online course.

### Rotation

In this course we used the Rodrigues’s Rotation Formula, in unity we use quaternion co-ordinates. The rotation is defined by the vector we are rotating, the axis upon which we are rotating and the degrees/radians of the rotation.   
In the case of moving the camera of the scene, we need to construct a unit vector from its forward and up vectors, this will be the axis. The direction the camera is viewing is the vector we are rotating.

### Translation

Translation is achieved by using homogenous co-ordinates. A translation matrix is constructed. It is made from an identity matrix with an extra row and column. The column dictates the translation and has the homogenous co-ordinate w ( usually W = 1). Then by multiplying it with a 4d vector which has (objectposVec3,W). This will appropriately move the matrix in 3d Space.

(note : in the lectures we cover when normals may need to change and how to change them Q(normal manipulation) = (m(object manipulation)^-1)^T)

### Developing A Co-ordinate frame.

This is very intriguing. I suppose this happens whenever a new camera is added to a scene, and a new view/camera space needs to be generated. The new co-ordinate frame needs to be orthonormal.

Orthonormality is that the unit vectors describing the 3 dimensional space must be orthogonal to one another and must be linearly independent.

* Orthogonal vecters are at right angles to one another.
* Linearly independent vectors mean that none of the vectors can be used to describe another.
  + This ensures that the vectors set up cover all of 3d space (e,g if they all lie on the same plane, they can only describe 2d space)

This construction of a co-ordinate system is done in OpenGl with the function glLookAt(). It creates a co-ordinate space, translates the camera to the origin. And now when the origin of the co ordinate system is rotated, so is the camera.

## HW1

Home work one is the second assignment of the course. HW0 is not being covered her as it was merely to set up appropriate programs and edit some shader/light values.   
Homework 1 is where we applied the knowledge that we had learned in the past lectures regarding matrices & transforms in 3d space. The steps of the assignment are as follows:

1. Set up a rotation function to rotate the camera
2. Generate a custom glLookAt function
3. Utilise the rotation function to rotate the camera around the jug.

I will not go into the specifics of the code as it can be found in my code for HW1. I will go over difficulties that I had found.

The major difficulty of this project was getting used to GLM. I am unfamiliar with it and finding appropriate functions consumed a lot of time.

A good 1 hour was spent on making debugging code to display the vectors and matrices. The helped me see what was going under the hood to fix some problems. For instance figuring out that my program had the camera facing in the +z direction instead of the -z direction.

When I arrived to using the rotation matrix to turn the camera I decided to use the formula :



This is so that I do not need to convert to and from matrix form in my calculations and remain in vertex.

In the rotate up and down function I hade to generate an axis that is the result of the cross product to the up and z axis (orthogonal to them). Then I could rotate the up and eye co ordinate frame accordingly. This code does rotate the camera but seems to move away. I assume that I am not updating the up axis appropriately, and can be solved with more coding or using the matrix Rodrigues Formula instead.

# Reflection

I understand more about the basic functions of a graphical engine and how it uses matrices to translate and display information. It helped even more with my understanding of spaces in unity. World space, object space and view space, I understand these systems a lot better now. Considering the rest of the course and HW2 will cover coding in OpenGl, I look forward to better understanding how to code in the graphics card/shaders.

For me personally these kind of rabbit holes are important because I struggle to use systems in good faith. This also helps me better scope my ideas to the limitations to graphics engines. Often I have ideas where the sky is the limit, but that is not true and understanding the engines can better inform any future ideas I have regarding video game mechanics.

A note is that I spent wat to much time on this. The course has an estimated time of 6 -12 weeks and I finished from what I can tell 2/3 rds of the course in 3 weeks. I have fallen behind on my other responsibilities and need to heed this experience while organising the next project.

Along this report I will also hand in lecture notes, design jurnal/maths hell, demonstration video, the project files, the transform.cpp file that I edited .

# Helpful links

[3D Rotations in General: Rodrigues Rotation Formula and Quaternion Exponentials - YouTube](https://www.youtube.com/watch?v=q-ESzg03mQc)

[Linear Independence - YouTube](https://www.youtube.com/watch?v=9kDpbZCK62Y)

[[VECTOR SPACES 1] Vector Basis, Span & Linear Independence - YouTube](https://www.youtube.com/watch?v=kNO9wQhLsEA)

[OpenGL Mathematics](http://glm.g-truc.net/0.9.4/api/index.html) [Euler (gimbal lock) Explained - YouTube](https://www.youtube.com/watch?v=zc8b2Jo7mno)

[manual.pdf](file:///D:\University\Year%204\WSOA3003A\EDX%20Graphics%20Programming%20course\manual.pdf)

[Euler (gimbal lock) Explained - YouTube](https://www.youtube.com/watch?v=zc8b2Jo7mno)