What is GLM?

GLM is a set of C++ classes and functions to fill in the programming gaps in writing the basic vector and matrix mathematics for OpenGL applications. However, even though it was written for OpenGL, it works fine with Vulkan (with one small exception which can be worked around.

Even though GLM looks like a library, it actually isn’t – it is all specified in *.hpp header files so that it gets compiled in with your source code.

You can find it at: [http://glm.g-truc.net/0.9.8.5/](http://glm.g-truc.net/0.9.8.5/)

You invoke GLM like this:

```cpp
#define    GLM_FORCE_RADIANS
#include <glm/glm.hpp>
#include <glm/gtc/matrix_transform.hpp>
#include  <glm/gtc/matrix_inverse.hpp>
```

If GLM is not installed in a system place, put it somewhere you can get access to. Later on, these notes will show you how to use it from there.

The Most Useful GLM Variables, Operations, and Functions

GLM recommends that you use the “glm::” syntax and avoid “using namespace” syntax because they have not made any effort to create unique function names

// constructor:
```cpp
glm::mat4(); // identity matrix
glm::vec4();
```

// multiplications:
```cpp
glm::mat4 * glm::mat4; // promote vec3 to a vec4 via a constructor
glm::mat4 * glm::vec4( glm::vec3 );

// emulating OpenGL transformations with concatenation:
```cpp
glm::mat4 glm::rotate( glm::mat4 const & m, float angle, glm::vec3 const & axis );
glm::mat4 glm::scale( glm::mat4 const & m, glm::vec3 const & factors );
glm::mat4 glm::translate( glm::mat4 const & m, glm::vec3 const & translation );
```
// viewing volume (assign, not concatenate):

glm::mat4 glm::ortho( float left, float right, float bottom, float top, float near, float far );

 glm::mat4 glm::ortho( float left, float right, float bottom, float top );

 glm::mat4 glm::frustum( float left, float right, float bottom, float top, float near, float far );

 glm::mat4 glm::perspective( float fovy, float aspect, float near, float far );

// viewing (assign, not concatenate):

 glm::mat4 glm::lookAt( glm::vec3 const & eye, glm::vec3 const & look, glm::vec3 const & up );

---

Installing GLM into your own space

I like to just put the whole thing under my Visual Studio project folder so I can zip up a complete project and give it to someone else.

---

Here’s what that GLM folder looks like

---

Telling Visual Studio about where the GLM folder is

1.  

2.  

---

The Most Useful GLM Variables, Operations, and Functions
A period, indicating that the project folder should also be searched when a
#include <xxx>
is encountered. If you put it somewhere else, enter that full or relative path instead.

Telling Visual Studio about where the GLM folder is

Your Sample2017.zip File Contains GLM Already

GLM in the Vulkan sample.cpp Program

How Does this Matrix Stuff Really Work?

This is called a “Linear Transformation” because all of the coordinates are raised to the 1st power, that is, there are no x^2, x^3, etc. terms.

Or, in matrix form:

\[
\begin{align*}
x' &= Ax + By + Cz + D \\
y' &= Ex + Fy + Gz + H \\
z' &= Ix + Jy + Kz + L
\end{align*}
\]
Transformation Matrices

Translation
\[
\begin{bmatrix}
1 & 0 & 0 & 0 & x' \\
0 & 1 & 0 & 0 & y' \\
0 & 0 & 1 & 0 & z' \\
0 & 0 & 0 & 1 & 1
\end{bmatrix}
\]

Rotation about X
\[
\begin{bmatrix}
1 & 0 & 0 & 0 & x' \\
0 & 1 & 0 & 0 & y' \\
0 & 0 & 1 & 0 & z' \\
0 & 0 & 0 & 1 & 1
\end{bmatrix}
\]

Rotation about Y
\[
\begin{bmatrix}
1 & 0 & 0 & 0 & x' \\
0 & 1 & 0 & 0 & y' \\
0 & 0 & 1 & 0 & z' \\
0 & 0 & 0 & 1 & 1
\end{bmatrix}
\]

Rotation about Z
\[
\begin{bmatrix}
1 & 0 & 0 & 0 & x' \\
0 & 1 & 0 & 0 & y' \\
0 & 0 & 1 & 0 & z' \\
0 & 0 & 0 & 1 & 1
\end{bmatrix}
\]

Scaling
\[
\begin{bmatrix}
S_x & 0 & 0 & 0 & x' \\
0 & S_y & 0 & 0 & y' \\
0 & 0 & S_z & 0 & z' \\
0 & 0 & 0 & 1 & 1
\end{bmatrix}
\]

The Rotation Matrix for an Angle (θ) about an Arbitrary Axis (Ax, Ay, Az)
\[
\begin{bmatrix}
A_x A_x + \cos(θ)(1 - A_x A_x) & A_x A_y - \cos(θ)A_y A_z - \sin(θ)A_z & A_x A_z + \cos(θ)A_z A_y + \sin(θ)A_y \\
A_x A_y + \cos(θ)A_y A_z + \sin(θ)A_z & A_y A_y + \cos(θ)(1 - A_y A_y) & A_y A_z - \cos(θ)A_z A_y - \sin(θ)A_y \\
A_x A_z - \cos(θ)A_z A_y + \sin(θ)A_y & A_y A_z + \cos(θ)A_z A_y - \sin(θ)A_y & A_z A_z + \cos(θ)(1 - A_z A_z)
\end{bmatrix}
\]

For this to be correct, A must be a unit vector.

How it Really Works :-)

\[
\begin{bmatrix}
\cos \theta & \sin \theta & 0 \\
-\sin \theta & \cos \theta & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
\alpha_1 \\
\alpha_2 \\
\alpha_3
\end{bmatrix}
= \begin{bmatrix}
\alpha_1' \\
\alpha_2' \\
\alpha_3'
\end{bmatrix}
\]

http://xkcd.com

Compound Transformations

Q: Our rotation matrices only work around the origin? What if we want to rotate about an arbitrary point (A,B)?

A: We create more than one matrix.

Write it
\[
\begin{bmatrix}
x' \\
y' \\
z'
\end{bmatrix} = \begin{bmatrix}
T_{+A,B}^{-1} & R_\theta & T_{-A,B}
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z
\end{bmatrix}
\]

Say it
\[
\begin{bmatrix}
x' \\
y' \\
z'
\end{bmatrix} = \begin{bmatrix}
3 & 2 & 1 \\
1 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z
\end{bmatrix}
\]
Matrix Multiplication is not Commutative

Matrix Multiplication is Associative

One Matrix to Rule Them All

Why Isn't The Normal Matrix just the same as the Model Matrix?