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/

You invoke GLM like this:

```
#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

- // viewing volume (assign, not concatenate):
  - glm: mat4 glm::ortho( 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 );
Here's what that GLM folder looks like

Telling Visual Studio about where the GLM folder is

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.

GLM in the Vulkan sample.cpp Program

Your Sample2017.zip File Contains GLM Already

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:

$x' = Ax + By + Cz + D

$y' = Ex + Fy + Gz + H

$z' = Ix + Jy + Kz + L

<table>
<thead>
<tr>
<th>$x'$ producing row</th>
<th>$y'$ producing row</th>
<th>$z'$ producing row</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>$y$</td>
<td>$z$</td>
</tr>
<tr>
<td>$y$</td>
<td>$I$ $J$ $K$</td>
<td>$I$ $J$ $K$</td>
</tr>
<tr>
<td>$z$</td>
<td>$L$ $M$ $N$</td>
<td>$L$ $M$ $N$</td>
</tr>
</tbody>
</table>

*Note: $A$, $B$, $C$, $D$, $E$, $F$, $G$, $H$, $I$, $J$, $K$, $L$, $M$, $N$ are constants.*
### Transformation Matrices

<table>
<thead>
<tr>
<th>Translation</th>
<th>Rotation about X</th>
<th>Scaling</th>
</tr>
</thead>
</table>
| \[
\begin{bmatrix}
 x' \\
 y' \\
 z'
\end{bmatrix} = \begin{bmatrix}
 1 & 0 & 0 & T_x \\
 0 & 1 & 0 & T_y \\
 0 & 0 & 1 & T_z \\
 0 & 0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
 x \\
 y \\
 z \\
 1
\end{bmatrix}
\] |
| \[
\begin{bmatrix}
 x' \\
 y' \\
 z'
\end{bmatrix} = \begin{bmatrix}
 1 & 0 & 0 & 0 \\
 0 & 1 & 0 & 0 \\
 0 & 0 & 1 & 0 \\
 0 & 0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
 x \\
 y \\
 z \\
 1
\end{bmatrix}
\] |

Rotation about Y and Z:

\[
\begin{bmatrix}
 x' \\
 y' \\
 z'
\end{bmatrix} = \begin{bmatrix}
 \cos \theta & 0 & \sin \theta & 0 \\
 0 & 1 & 0 & 0 \\
 -\sin \theta & 0 & \cos \theta & 0 \\
 0 & 0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
 x \\
 y \\
 z \\
 1
\end{bmatrix}
\]

\[
\begin{bmatrix}
 x' \\
 y' \\
 z'
\end{bmatrix} = \begin{bmatrix}
 \cos \theta & -\sin \theta & 0 & 0 \\
 \sin \theta & \cos \theta & 0 & 0 \\
 0 & 0 & 1 & 0 \\
 0 & 0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
 x \\
 y \\
 z \\
 1
\end{bmatrix}
\]

### The Rotation Matrix for an Angle ($\theta$) about an Arbitrary Axis ($A_x, A_y, A_z$)

\[
M = \begin{bmatrix}
 1 - A_x^2 (1 - \cos \theta) & -A_y (1 - \cos \theta) & -A_z (1 - \cos \theta) & A_x \sin \theta \\
 -A_y (1 - \cos \theta) & 1 - A_y^2 (1 - \cos \theta) & -A_z (1 - \cos \theta) & A_y \sin \theta \\
 -A_z (1 - \cos \theta) & A_z (1 - \cos \theta) & 1 - A_z^2 (1 - \cos \theta) & A_z \sin \theta \\
 A_x \sin \theta & A_y \sin \theta & A_z \sin \theta & 1
\end{bmatrix}
\]

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

### Compound Transformations

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

A: We create more than one matrix.

```
\begin{bmatrix}
 x' \\
 y' \\
 z'
\end{bmatrix} = \begin{bmatrix}
 T_{x-A_z} \\
 T_{y-A_y} \\
 T_{z-A_z}
\end{bmatrix} \begin{bmatrix}
 R_x \\
 R_y \\
 R_z
\end{bmatrix} \begin{bmatrix}
 x \\
 y \\
 z \\
 1
\end{bmatrix}
```

### Matrix Multiplication is Not Commutative

```
\begin{bmatrix}
 x' \\
 y' \\
 z'
\end{bmatrix} = \begin{bmatrix}
[T_{x-A_z}] & [R_x] & [T_{y-A_y}] & [R_y] & [T_{z-A_z}] & [R_z]
\end{bmatrix} \begin{bmatrix}
 x \\
 y \\
 z \\
 1
\end{bmatrix}
```

### Matrix Multiplication is Associative

```
\begin{bmatrix}
 x' \\
 y' \\
 z'
\end{bmatrix} = ([T_{x-A_z}][R_x][T_{y-A_y}][R_y][T_{z-A_z}][R_z]) \begin{bmatrix}
 x \\
 y \\
 z \\
 1
\end{bmatrix}
```

One matrix — the Current Transformation Matrix, or CTM
One Matrix to Rule Them All

\[
\begin{pmatrix}
    x' \\
    y' \\
    z' \\
    1
\end{pmatrix} = \begin{bmatrix}
    R & T_x & T_y & T_z \\
    0 & 1 & 0 & 0 \\
    0 & 0 & 1 & 0 \\
    0 & 0 & 0 & 1
\end{bmatrix}
\begin{pmatrix}
    x \\
    y \\
    z \\
    1
\end{pmatrix}
\]

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

It is, if the Model Matrix is all rotations and uniform scalings, but if it has non-uniform scalings, then it is not. These diagrams show you why.

Wrong!

Right!

Original object and normal

Wrong!

Right!

Normal object and normal

Original object and normal

Wrong!

Right!