




**GLM**



**Oregon State University**  
Mike Bailey  
mjb@cs.oregonstate.edu



This work is licensed under a [Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License](https://creativecommons.org/licenses/by-nc-nd/4.0/)



Oregon State University  
Computer Graphics

GLM.pptx
mjb - September 17, 2018

### 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.



Oregon State University  
Computer Graphics

mjb - September 17, 2018

### Why are we even talking about this?

All of the things that we have talked about being *deprecated* in OpenGL are *really deprecated* in Vulkan -- built-in pipeline transformations, begin-end, fixed-function, etc. So, where you might have said in OpenGL:

```
gluLookAt( 0., 0., 3., 0., 0., 0., 0., 1., 0. );
glRotatef( (GLfloat)Yrot, 0., 1., 0. );
glRotatef( (GLfloat)Xrot, 1., 0., 0. );
glScalef( (GLfloat)Scale, (GLfloat)Scale, (GLfloat)Scale );
```


} Depreciated  
OpenGL

you would now have to say:

```
glm::mat4 modelview;
glm::vec3 eye(0.,0.,3.);
glm::vec3 look(0.,0.,0.);
glm::vec3 up(0.,1.,0.);
modelview = glm::lookAt( eye, look, up );
modelview = glm::rotate( modelview, D2R*Yrot, glm::vec3(0.,1.,0. ) );
modelview = glm::rotate( modelview, D2R*Xrot, glm::vec3(1.,0.,0. ) );
modelview = glm::scale( modelview, glm::vec3(Scale,Scale,Scale) );
```

} GLM

Exactly the same concept, but a different expression of it. Read on for details ...



Oregon State University  
Computer Graphics

mjb - September 17, 2018

### The Most Useful GLM Variables, Operations, and Functions

```
// constructor:
glm::mat4( );
glm::vec4( ); // identity matrix
glm::vec3( );
```

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

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

**// emulating OpenGL transformations with concatenation:**

```
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 );
```

Cont

### The Most Useful GLM Variables, Operations, and Functions

5


**// 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 );
```

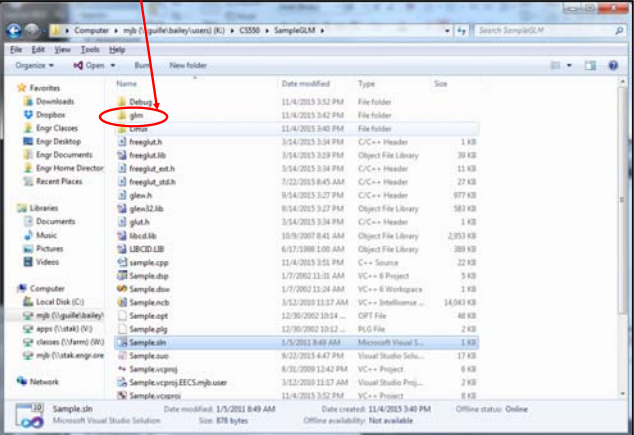



mjb - September 17, 2018

### Installing GLM into your own space

6

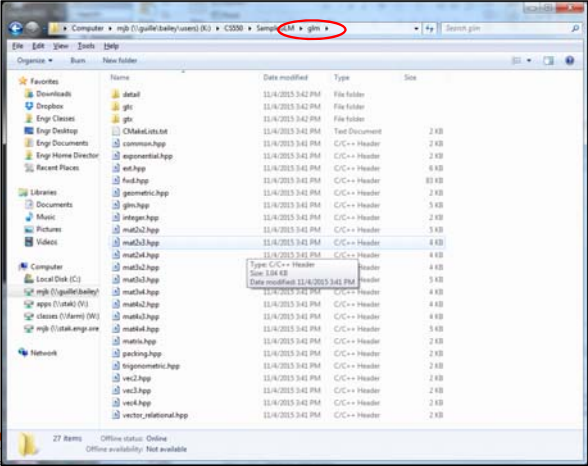

I like to just put the whole thing under my Visual Studio project folder so I can zip up a complete project and give it to someone else.

mjb - September 17, 2018

### Here's what that GLM folder looks like

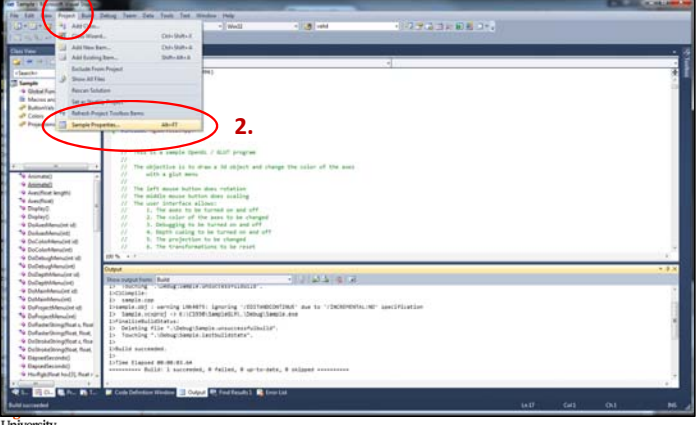

7

mjb - September 17, 2018

### Telling Visual Studio about where the GLM folder is

8

mjb - September 17, 2018

### 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

```

if( !UseMouse )
{
    if( Scale < MINSCALE )
        Scale = MINSCALE;
    Matrices.uModelMatrix = glm::mat4(); // identity
    Matrices.uModelMatrix = glm::scale( Matrices.uModelMatrix, glm::vec3( Scale, Scale, Scale ) );
    Matrices.uModelMatrix = glm::rotate( Matrices.uModelMatrix, Yrot, glm::vec3( 0., 1., 0. ) );
    Matrices.uModelMatrix = glm::rotate( Matrices.uModelMatrix, Xrot, glm::vec3( 1., 0., 0. ) );
    // done this way, the Xrot is applied first, then the Yrot, then the Scale
}
else
{
    if( !Paused )
    {
        const glm::vec3 axis = glm::vec3( 0., 1., 0. );
        Matrices.uModelMatrix = glm::rotate( glm::mat4(), (float)glm::radians( 360.f*Time/SECONDS_PER_CYCLE ), axis );
    }
}

Matrices.uProjectionMatrix = glm::perspective( FOV, (double)Width/(double)Height, 0.1, 1000. );
Matrices.uProjectionMatrix[1][1] *= -1. // Vulkan's projected Y is inverted from OpenGL

Matrices.uNormalMatrix = glm::inverseTranspose( glm::mat3( Matrices.uModelMatrix ) );
Matrices.uProjectionMatrix = glm::perspective( FOV, (double)Width/(double)Height, 0.1, 1000. );
Matrices.uProjectionMatrix[1][1] *= -1.;

Matrices.uNormalMatrix = glm::inverseTranspose( glm::mat3( Matrices.uModelMatrix ) );
Fill05DataBuffer( MyMiscUniformBuffer, (void *) &Matrices );

Misc.uTime = (float)Time;
Misc.uMode = Mode;
Fill05DataBuffer( MyMiscUniformBuffer, (void *) &Misc );
    
```

Computer Graphics mjb - September 17, 2018

### Your Sample2017.zip File Contains GLM Already

Name	Date modified	Type	Size
vs	12/27/2017 9:45 AM	File folder	
Debug	1/3/2018 12:33 PM	File folder	
glm	1/3/2018 9:44 AM	File folder	
gltf2.h	12/26/2017 10:48 ...	C/C++ Header	149 KB
glfw3.lib	8/18/2016 5:06 AM	Object File Library	240 KB
glslangValidator	12/31/2017 5:24 PM	File	1,817 KB
glslangValidator.exe	6/15/2017 12:33 PM	Application	1,633 KB
glslangValidator.help	10/6/2017 2:31 PM	HELP File	6 KB
Makefile	12/31/2017 5:57 PM	File	1 KB
puppy.bmp	1/1/2018 9:37 AM	BMP File	3,075 KB
puppy.jpg	1/1/2018 9:38 AM	JPG File	455 KB
sample.cpp	1/3/2018 12:33 PM	C++ Source	103 KB
Sample.sln	12/27/2017 9:45 AM	Microsoft Visual S...	2 KB
Sample.vcproj	12/27/2017 10:17 ...	VC++ Project	7 KB
Sample.vcproj.filters	12/27/2017 9:47 AM	VC++ Project Filte...	1 KB
sample-frag.frag	1/1/2018 10:50 AM	FRAG File	1 KB
sample-frag.spv	1/1/2018 10:50 AM	SPV File	1 KB
sample-vert.spv	1/3/2018 9:42 AM	SPV File	3 KB
sample-vert.vert	1/3/2018 9:42 AM	VERT File	1 KB
SampleVertData.cpp	12/26/2017 10:27 ...	C++ Source	4 KB
vk_icd.h	12/26/2017 10:42 ...	C/C++ Header	5 KB
vk_layer.h	12/26/2017 10:42 ...	C/C++ Header	5 KB
vk_layer_dispatch_table.h	12/26/2017 10:42 ...	C/C++ Header	20 KB
vk_platform.h	12/26/2017 10:42 ...	C/C++ Header	4 KB
vk_sdk_platform.h	12/26/2017 10:42 ...	C/C++ Header	2 KB
vulkan.h	12/26/2017 10:42 ...	C/C++ Header	274 KB
vulkan.hpp	12/26/2017 10:39 ...	C/C++ Header	1,101 KB
vulkan.lib	6/15/2017 12:38 PM	Object File Library	41 KB
VulkanDebug.txt	1/3/2018 12:34 PM	Text Document	217 KB

Computer Graphics mjb - September 17, 2018

### How Does this Matrix Stuff Really Work?

$$\begin{aligned}
 x' &= Ax + By + Cz + D \\
 y' &= Ex + Fy + Gz + H \\
 z' &= Ix + Jy + Kz + L
 \end{aligned}$$

This is called a "Linear Transformation" because all of the coordinates are raised to the 1<sup>st</sup> power, that is, there are no  $x^2$ ,  $x^3$ , etc. terms.

Or, in matrix form:

$$\begin{matrix}
 \text{x consuming column} \\
 \text{y consuming column} \\
 \text{z consuming column} \\
 \text{constant column}
 \end{matrix}
 \begin{matrix}
 \text{x' producing row} \\
 \text{y' producing row} \\
 \text{z' producing row}
 \end{matrix}
 \begin{bmatrix}
 x' \\
 y' \\
 z' \\
 1
 \end{bmatrix}
 =
 \begin{bmatrix}
 A & B & C & D \\
 E & F & G & H \\
 I & J & K & L \\
 0 & 0 & 0 & 1
 \end{bmatrix}
 \cdot
 \begin{bmatrix}
 x \\
 y \\
 z \\
 1
 \end{bmatrix}$$

Computer Graphics mjb - September 17, 2018

### Transformation Matrices

**Translation**

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

**Rotation about X**

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

**Scaling**


$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

**Rotation about Y**

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

**Rotation about Z**

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




mjb - September 17, 2018

### How it Really Works :-)

$$\begin{bmatrix} \cos 90^\circ & \sin 90^\circ \\ -\sin 90^\circ & \cos 90^\circ \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

<http://xkcd.com>




mjb - September 17, 2018

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

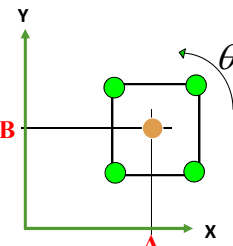
$$[M] = \begin{bmatrix} A_x A_x + \cos \theta (1 - A_x A_x) & A_x A_y - \cos \theta (A_x A_y) - \sin \theta A_z & A_x A_z - \cos \theta (A_x A_z) + \sin \theta A_y \\ A_y A_x - \cos \theta (A_y A_x) + \sin \theta A_z & A_y A_y + \cos \theta (1 - A_y A_y) & A_y A_z - \cos \theta (A_y A_z) - \sin \theta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

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



mjb - September 17, 2018

### 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

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \boxed{3} \\ [T_{+A,+B}] \cdot \begin{bmatrix} \boxed{2} \\ [R_\theta] \cdot \begin{bmatrix} \boxed{1} \\ [T_{-A,-B}] \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Say it



mjb - September 17, 2018

### Matrix Multiplication is not Commutative

**Rotate, then translate**

**Translate, then rotate**

Oregon State University Computer Graphics

mjb - September 17, 2018

### Matrix Multiplication is Associative

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \left( [T_{+A,+B}] \cdot [R_\theta] \cdot [T_{-A,-B}] \right) \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \underbrace{\left( [T_{+A,+B}] \cdot [R_\theta] \cdot [T_{-A,-B}] \right)}_{\text{One matrix - the Current Transformation Matrix, or CTM}} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Oregon State University Computer Graphics

mjb - September 17, 2018

### One Matrix to Rule Them All

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \left( [T_{+A,+B}] \cdot [R_\theta] \cdot [T_{-A,-B}] \right) \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

```

glm::mat4 Model = glm::mat4( );
Model = glm::translate(Model, glm::vec3(-A, -B, 0. ) );
Model = glm::rotate(Model, thetaRadians, glm::vec3( Ax, Ay, Az ) );
Model = glm::translate(Model, glm::vec3( A, B, 0. ) );

glm::vec3 eye(0.,0.,EYEDIST);
glm::vec3 look(0.,0.,0.); glm::vec3 up(0.,1.,0.);
glm::mat4 View = glm::lookAt( eye, look, up );

glm::mat4 Projection = glm::perspective( FOV, (double)Width/(double)Height, 0.1, 1000. );
Projection[1][1] *= -1.;

...
glm::mat3 Matrix = Projection * View * Model;
glm::mat3 NormalMatrix = glm::inverseTranspose( glm::mat3(Matrix) );
    
```

Oregon State University Computer Graphics

mjb - September 17, 2018

### 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.

Original object and normal

**Wrong!**

glm::mat3 NormalMatrix = glm::mat3(Model);

Original object and normal

**Right!**

glm::mat3 NormalMatrix = glm::inverseTranspose( glm::mat3(Model) );

Oregon State University Computer Graphics

mjb - September 17, 2018