

Vulkan.

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

Oregon State University Computer Graphics

GLM.hpp

mjb – December 21, 2022

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.

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
```

OpenGL treats all angles as given in degrees. This line forces GLM to treat all angles as given in radians. I recommend this so that all angles you create in all programming will be in radians.

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 – December 21, 2022

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:

```
glMatrixMode( GL_MODELVIEW );
glLoadIdentity();
gluLookAt( 0., 0., 3., 0., 0., 0., 1., 0., 0. );
glRotatef( GLfloat)Rot, 0., 1., 0. );
glRotatef( GLfloat)Xrot, 1., 0., 0. );
glScalef( (GLfloat)Scale, (GLfloat)Scale, (GLfloat)Scale );
```

you would now say:

```
glm::mat4 modelview = glm::mat4( 1. ); // identity
glm::vec3 eye(0.,0.,3.); // eye
glm::vec3 look(0.,0.,0.); // look
glm::vec3 up(0.,1.,0.); // up
modelview = glm::lookAt( eye, look, up ); // [x,y,z] = [v]*[x,y,z]
modelview = glm::rotate( modelview, D2R*Yrot, glm::vec3(0.,1.,0.) ); // [x,y,z] = [v]*[y]*[x,y,z]
modelview = glm::rotate( modelview, D2R*Xrot, glm::vec3(1.,0.,0.) ); // [x,y,z] = [v]*[y]*[x]*[x,y,z]
modelview = glm::scale( modelview, glm::vec3(Scale,Scale,Scale) ); // [x,y,z] = [v]*[y]*[x]*[s]*[x,y,z]
```

This is exactly the same concept as OpenGL, but a different expression of it. Read on for details ...

Oregon State University Computer Graphics

mjb – December 21, 2022

The Most Useful GLM Variables, Operations, and Functions

// constructor:

```
glm::mat4( 1. ); // identity matrix
glm::vec4(); // identity vector
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, 1. ) // promote a 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 );
Con| glm::mat4 glm::translate( glm::mat4 const & m, glm::vec3 const & translation );
```

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::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 );
```

Oregon State University Computer Graphics

mjb – December 21, 2022

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.

A screenshot of a Windows File Explorer window titled "SampleGLM". It shows a file structure with several files and folders related to the GLM library. A red circle highlights the "glm" folder. The files listed include:

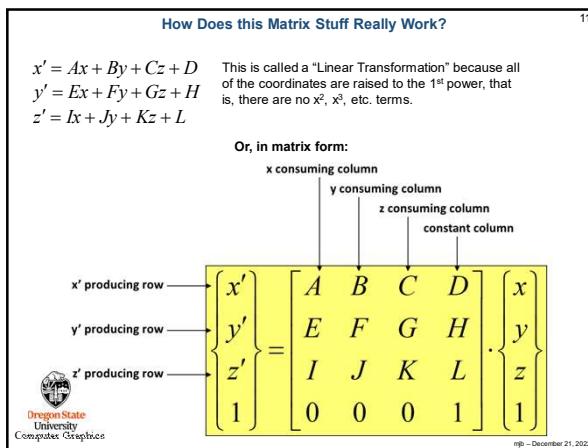
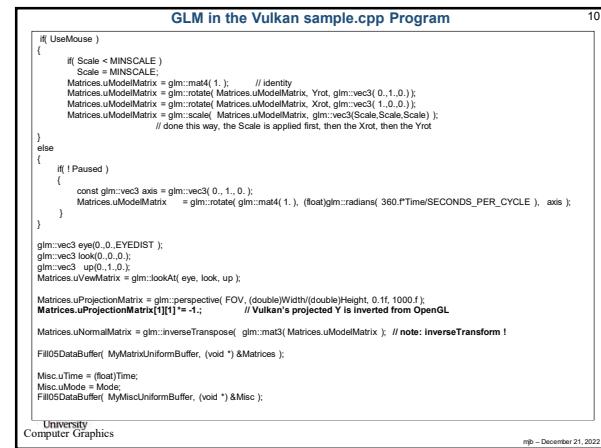
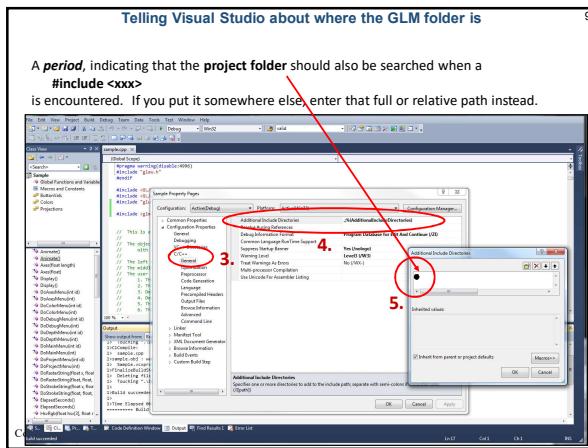
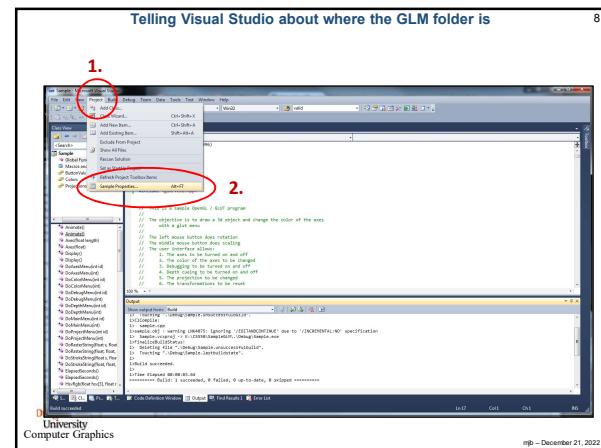
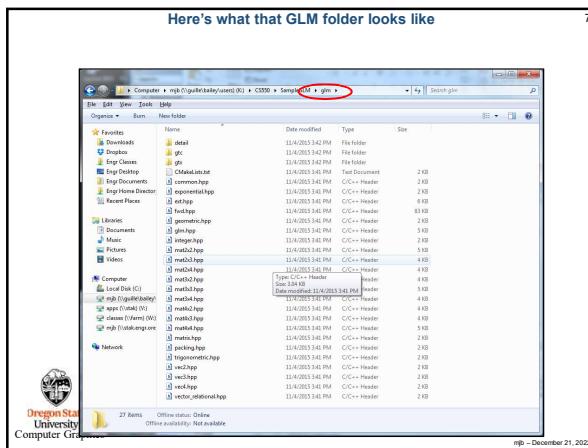
- glm.h (File, Date modified: 11/20/2011 3:22 PM, Size: 870 bytes)
- glm.hpp (File, Date modified: 11/20/2011 3:22 PM, Size: 870 bytes)
- glm.hpp (Object File Library, Date modified: 11/20/2011 3:22 PM, Size: 1 KB)
- glm.hpp (C/C++ Header, Date modified: 11/20/2011 3:22 PM, Size: 1 KB)
- glm.hpp (C/C++ Header, Date modified: 3/14/2012 3:30 PM, Size: 27 KB)
- glm.hpp (C/C++ Header, Date modified: 7/20/2012 11:55 AM, Size: 1 KB)
- glm.hpp (C/C++ Header, Date modified: 8/14/2012 3:27 PM, Size: 107 KB)
- glm.hpp (Object File Library, Date modified: 8/14/2012 3:27 PM, Size: 503 KB)
- glm.hpp (C/C++ Header, Date modified: 3/14/2012 3:30 PM, Size: 1 KB)
- glm.hpp (Object File Library, Date modified: 6/17/2012 8:41 AM, Size: 389 KB)
- glm.hpp (C/C++ Header, Date modified: 11/20/2011 3:22 PM, Size: 2 KB)
- glm.hpp (C/C++ Header, Date modified: 1/7/2009 11:15 AM, Size: 5 KB)
- glm.hpp (C/C++ Header, Date modified: 3/14/2012 3:30 PM, Size: 1 KB)
- glm.hpp (C/C++ Header, Date modified: 3/12/2012 11:17 AM, Size: 14,043 KB)
- glm.hpp (OPT File, Date modified: 12/30/2002 10:14 AM, Size: 48 KB)
- glm.hpp (PCH File, Date modified: 12/30/2002 10:14 AM, Size: 2 KB)
- glm.hpp (Microsoft Shared Object, Date modified: 8/12/2012 8:49 AM, Size: 1 KB)
- glm.hpp (Project File, Date modified: 8/12/2012 12:42 PM, Size: 17 KB)
- glm.hpp (Project File, Date modified: 3/12/2010 11:17 AM, Size: 2 KB)
- glm.hpp (Project File, Date modified: 11/4/2012 1:58 PM, Size: 8 KB)

Sample.sln Microsoft Visual Studio Solution Date modified: 1/20/2011 8:49 AM Size: 870 bytes Offline availability: Not available

Sample.vcxproj Microsoft Visual Studio Project Date modified: 1/20/2011 8:49 AM VC++ Project Offline status: Online

Sample.vcxproj Microsoft Visual Studio Project Date modified: 11/4/2012 1:58 PM VC++ Project Offline status: Online

mjb – December 21, 2022



Transformation Matrices

Translation

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

Rotation about X

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

Scaling

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

Rotation about Y

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

Rotation about Z

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

Oregon State University Computer Graphics

12

mb – December 21, 2022

How it Really Works :-)

$$\begin{bmatrix} \cos 90^\circ & \sin 90^\circ \\ -\sin 90^\circ & \cos 90^\circ \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = \boxed{\text{Rotated Vector}}$$

<http://xkcd.com>

Oregon State University Computer Graphics

mb – December 21, 2022

The Rotation Matrix for an Angle (θ) about an Arbitrary Axis (Ax, Ay, Az)

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

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

Oregon State University Computer Graphics

mb – December 21, 2022

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

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

Say it

Oregon State University Computer Graphics

mb – December 21, 2022

Matrix Multiplication is not Commutative

Rotate, then translate

Translate, then rotate

Oregon State University Computer Graphics

mb – December 21, 2022

Matrix Multiplication is Associative

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

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

Oregon State University Computer Graphics

mb – December 21, 2022

From the Data Buffer Noteset

Here's the vertex shader shader code to use the matrices:

```
layout( std140, set = 0, binding = 0 ) uniform sceneMatBuf
{
    mat4 uProjectionMatrix;
    mat4 uViewMatrix;
    mat4 uSceneMatrix;
} SceneMatrices;
```

```
layout( std140, set = 1, binding = 0 ) uniform objectMatBuf
{
    mat4 uModelMatrix;
    mat4 uNormalMatrix;
} ObjectMatrices;
```

```
vNormal = uNormalMatrix * aNormal;
gl_Position = uProjectMatrix * uViewMatrix * uSceneMatrix * uModelMatrix * aVertex;
```

"CTM"

Oregon State University Computer Graphics

mb – December 21, 2022

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

19

```
glm::mat4 Model = uViewMatrix*uSceneMatrix*uModelMatrix;
uNormalMatrix = glm::inverseTranspose( glm::mat3(Model) );
```

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

Original object and normal:

Wrong!

Right!

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

njb – December 21, 2022

Oregon State University Computer Graphics