

The OpenGL Mathematics (GLM) Library



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/)



GLM.pptx

mjb - July 29, 2021

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.

GLM isn't really a *library* – 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>
```

Or, you can #include only the specific GLM .hpp files you need.

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.



mjb - July 29, 2021

Why are we even talking about this?

3

The OpenGL overlords have “deprecated” some of the OpenGL functions we have been using to perform transformations. In the desktop world, it means that the use of such functions is **discouraged**. In Vulkan and in the mobile world of OpenGL-ES, it means those functions are **gone**. You might as well become familiar with how to live without them. So, instead of saying:

```
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 );
```

for OpenGL, you would now 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) );
glMultMatrixf( glm::value_ptr( modelview ) );
```



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

mjb – July 29, 2021

The Most Useful GLM Variables, Operations, and Functions

4

// constructor:

```
glm::mat4( 1. ); // identity matrix
glm::vec4( );
glm::vec3( );
```

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

// multiplications – the * operator has been overloaded:

```
glm::mat4 * glm::mat4
glm::mat4 * glm::vec4
glm::mat4 * glm::vec4( glm::vec3, 1. ) // 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 );
```



mjb – July 29, 2021

// 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 );
```

// loading matrices into opengl:

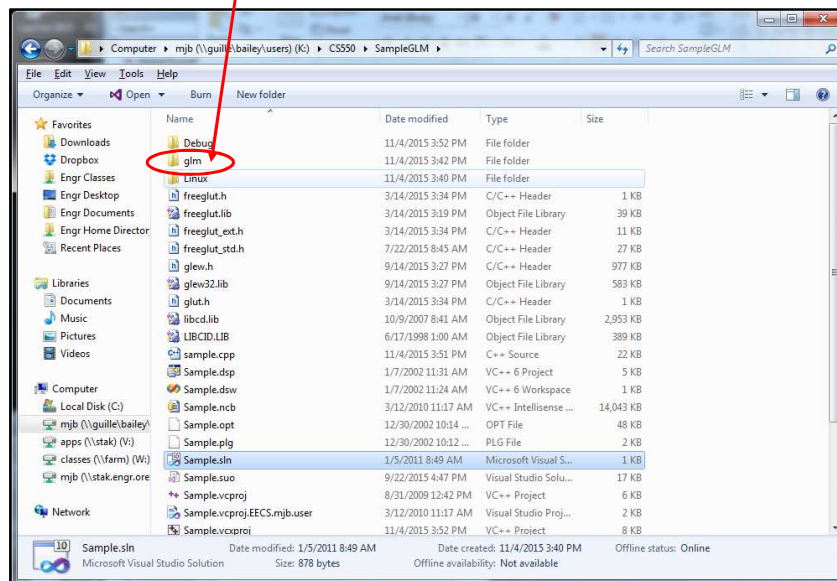
```
glLoadMatrix( glm::value_ptr( glm::mat4 ) );
```

```
glUniformMatrix4fv( Location, 1, GL_FALSE, glm::value_ptr( glm::mat4 ) );
```



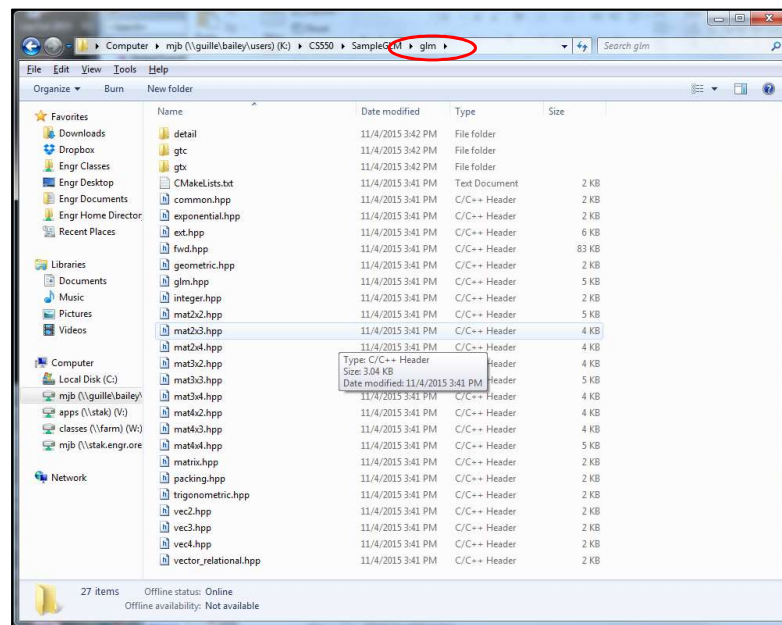
Installing GLM into your own space

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

7



Telling Linux about where the GLM folder is

8

```
g++ ... -I. ...
```

“minus-capital-eye-period” means “also look for the < > includes in this folder”

Instead of the period, you can list a full or relative pathname.

Telling Visual Studio about where the GLM folder is

9

1.

2.

The screenshot shows the Visual Studio interface. The 'Project' menu is open, and the 'Sample Properties...' option is highlighted. A red circle labeled '1.' is around the 'Project' menu, and another red circle labeled '2.' is around the 'Sample Properties...' option. The background shows a C++ source file with GLM-related code and an output window showing build logs.

Build succeeded

Ln17 Col1 Ch1 INS

mjb - July 29, 2021



Telling Visual Studio about where the GLM folder is

10

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.

3.

4.

5.

The screenshot shows the Visual Studio interface with the 'Sample Properties' dialog box open. The 'C/C++' section is selected, and the 'Additional Include Directories' field is highlighted with a red circle labeled '3.'. The 'Additional Include Directories' dialog box is also open, showing the 'Additional Include Directories' field with a red circle labeled '5.'. A red arrow points from the text above to the 'Additional Include Directories' field in the 'Sample Properties' dialog box, which is labeled '4.'. The background shows a C++ source file with GLM-related code and an output window showing build logs.

Build succeeded

Ln17 Col1 Ch1 INS

mjb - July 29, 2021



Using Transformations, OpenGL-style, like in the sample.cpp Program

11

```
glMatrixMode( GL_PROJECTION );
glLoadIdentity( );
if( WhichProjection == ORTHO )
    glOrtho( -3., 3., -3., 3., 0.1, 1000. );
else
    gluPerspective( 90., 1., 0.1, 1000. );

// place the objects into the scene:
glMatrixMode( GL_MODELVIEW );
glLoadIdentity( );

// set the eye position, look-at position, and up-vector:
gluLookAt( 0., 0., 3., 0., 0., 0., 0., 1., 0. );

// rotate the scene:
glRotatef( (GLfloat)Yrot, 0., 1., 0. );
glRotatef( (GLfloat)Xrot, 1., 0., 0. );

// uniformly scale the scene:
if( Scale < MINSCALE )
    Scale = MINSCALE;
glScalef( (GLfloat)Scale, (GLfloat)Scale, (GLfloat)Scale );
```



mjb - July 29, 2021

Using Transformations, GLM-style, I

12

```
#include <glm/vec3.hpp>
#include <glm/mat4x4.hpp>
#include <glm/gtc/matrix_transform.hpp>
#include <glm/gtc/type_ptr.hpp>

...

// convert degrees to radians:
const float D2R = M_PI/180.f;           // 0.01745...

...

glMatrixMode( GL_PROJECTION );
glLoadIdentity( );
glm::mat4 projection;

if( WhichProjection == ORTHO )
    projection = glm::ortho( -3., 3., -3., 3., 0.1, 1000. );
else
    projection = glm::perspective( D2R*90., 1., 0.1, 1000. );

// apply the projection matrix:
glMultMatrixf( glm::value_ptr( projection ) );
```



mjb - July 29, 2021

Using Transformations, GLM-style, II

13

```
// place the objects into the scene:
glMatrixMode( GL_MODELVIEW );
glLoadIdentity( );

// set the eye position, look-at position, and up-vector:
glm::vec3 eye(0.,0.,3.);
glm::vec3 look(0.,0.,0.);
glm::vec3 up(0.,1.,0.);
glm::mat4 modelview = glm::lookAt( eye, look, up );

// rotate the scene (warning -- unlike OpenGL's glRotatef,
// GLM's rotate method takes angles in *radians*):
modelview = glm::rotate( modelview, D2R*Yrot, glm::vec3(0.,1.,0.) );
modelview = glm::rotate( modelview, D2R*Xrot, glm::vec3(1.,0.,0.) );

// uniformly scale the scene:
if( Scale < MINSCALE )
    Scale = MINSCALE;
modelview = glm::scale( modelview, glm::vec3(Scale,Scale,Scale) );

// apply the modelview matrix:
glMultMatrixf( glm::value_ptr( modelview ) );
```



mjb - July 29, 2021

Passing GLM Matrices into a Vertex Shader

14

In the shader:

```
uniform mat4 projectionMatrix;
uniform mat4 viewMatrix;
uniform mat4 modelMatrix;

mat4 PVM = projectionMatrix * viewMatrix * modelMatrix;
gl_Position = PVM * gl_Vertex;
```

In the C/C++ program:

```
glm::mat4 projection = glm::perspective( D2R*90., 1., 0.1, 1000. );

glm::vec3 eye(0.,0.,3.);
glm::vec3 look(0.,0.,0.);
glm::vec3 up(0.,1.,0.);
glm::mat4 view = glm::lookAt( eye, look, up );

glm::mat4 model( 1. ); // identity
model = glm::rotate( model, D2R*Yrot, glm::vec3(0.,1.,0.) );
model = glm::rotate( model, D2R*Xrot, glm::vec3(1.,0.,0.) );

Pattern->Use( );
Pattern->SetUniformVariable( "projectionMatrix", projection );
Pattern->SetUniformVariable( "viewMatrix", view );
Pattern->SetUniformVariable( "modelMatrix", model );
```



mjb - July 29, 2021

```
glm::mat4 projection = glm::perspective( D2R*90., 1., 0.1, 1000. );  
projection[1][1] *= -1.; // Vulkan's projected Y is inverted from OpenGL's  
  
glm::vec3 eye(0.,0.,3.);  
glm::vec3 look(0.,0.,0.);  
glm::vec3 up(0.,1.,0.);  
glm::mat4 view = glm::lookAt( eye, look, up );  
  
glm::mat4 model( 1. ); // identity  
model = glm::rotate( model, D2R*Yrot, glm::vec3(0.,1.,0.) );  
model = glm::rotate( model, D2R*Xrot, glm::vec3(1.,0.,0.) );
```