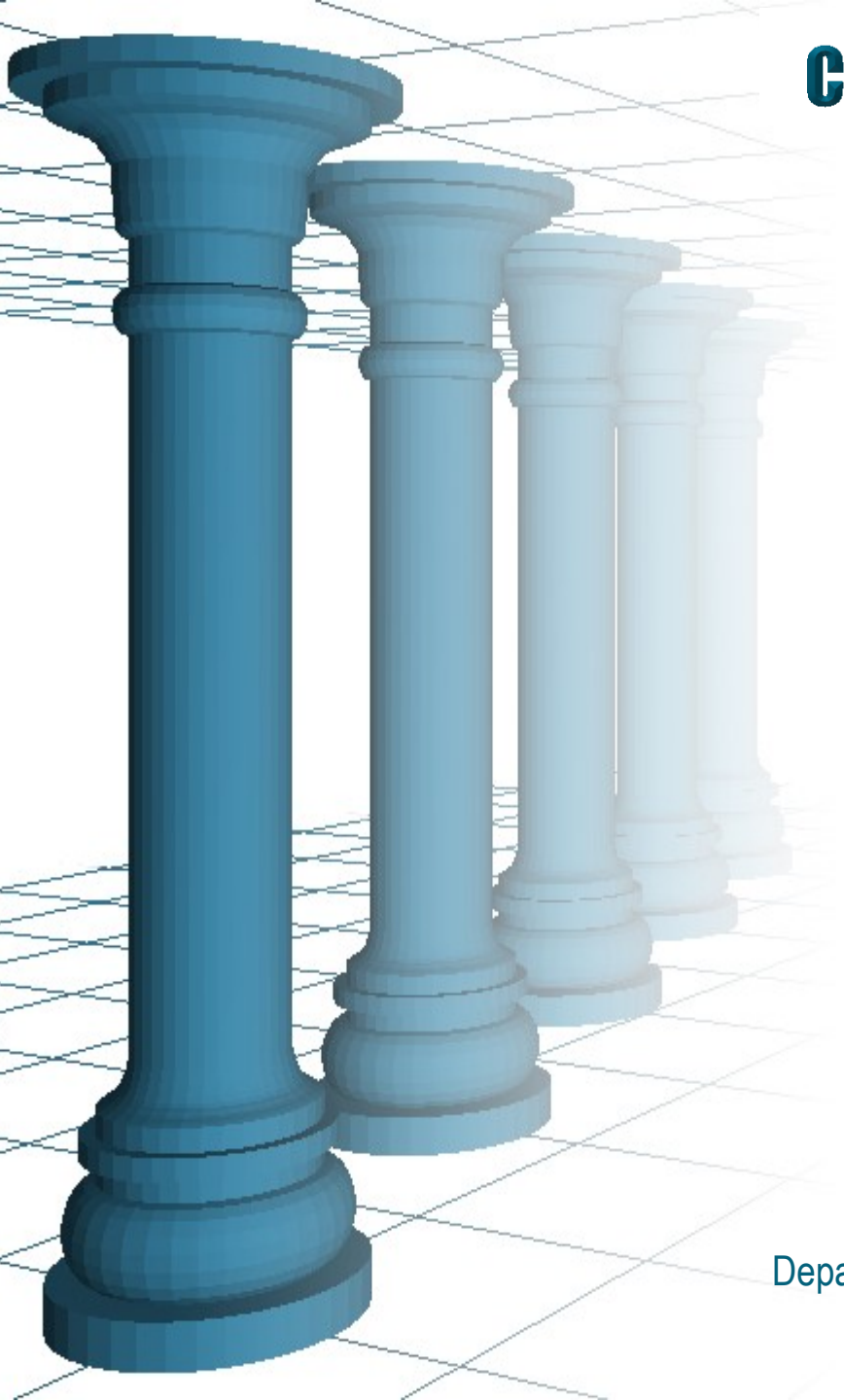


COSC363 Computer Graphics

GLM

Maths for OpenGL

R. Mukundan (mukundan@canterbury.ac.nz)
Department of Computer Science and Software Engineering
University of Canterbury, New Zealand.

GLM Library

The **OpenGL Mathematics** (GLM) library is a convenient C++ library for developing graphics applications.

- Header only library

```
#include <glm/glm.hpp>
#include <glm/gtc/matrix_transform.hpp>
```

- Several functions and variables use similar naming convention as OpenGL and GLSL

```
glm::vec4  point(2, -3, 4, 1);
glm::mat4  viewMat = glm::lookAt(eye, look, up);
glm::mat4  rotnMat = glm::rotate(inMat, angle, axis);
```

- Particularly useful for matrix operations, lighting computations, **ray tracing** and **OpenGL-4 shader development**.

Vector Type **glm::vec3**

glm::**vec3** is the most commonly used vector type for storing points, vectors and color values.

Examples:

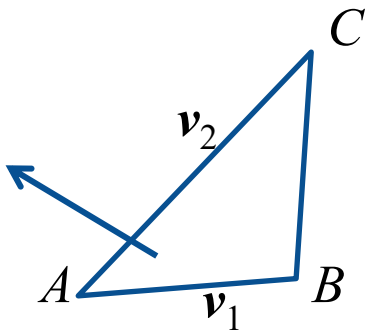
```
//Declarations
glm::vec3  lightPos(-15, 100, 10);
glm::vec3  backgroundCol(0, 0, 1);
glm::vec3  normalVec(0.8, -0.6, 0);
glm::vec3  ambientCol(0.2);        // =(0.2, 0.2, 0.2)

//Assignment of values
normalVec = glm::vec3(0, 0, 1);
backgroundCol = glm::vec3(0);      // =(0, 0, 0)
glm::vec4 aVec(8, 3, 5, 2);
glm::vec3 newVec = aVec.xyz;       // =(8, 3, 5)
```

Vector Operations

```
glm::vec3  aVec(1, 2, 9);
glm::vec3  bVec(5, 2, 0);
glm::vec3  sumVec = aVec + bVec;     // =(6, 4, 9)
glm::vec3  prodVec = aVec * bVec;    // =(5, 4, 0)
glm::vec3  scaledVec = 10 * aVec;    // =(10, 20, 90)
//Length and Normalization
float len = glm::length(aVec);       //=9.273
bVec = glm::normalize(bVec);         //unit vector
aVec = glm::normalize(aVec);
//Dot and Cross Products
float dotProd = glm::dot(aVec, bVec);
glm::vec3 crossProd = glm::cross(aVec, bVec);
aVec = glm::normalize(aVec);
//Vectors from Points
glm::vec3  vertex(0, 2, 1);
glm::vec3  light(5, 9, 6);
glm::vec3  lightVec = light - vertex;
```

Vector Operations




```
#include <glm/gtx/string_cast.hpp>

glm::vec3  ptA(1, 0, 1);
glm::vec3  ptB(5, 2, 3);
glm::vec3  ptC(5, 4, 0);

//Computing Normal Vector
glm::vec3  norm;
norm = glm::cross(ptB - ptA, ptC - ptA);
norm = glm::normalize(norm);

//Printing
cout << "Normal Vec= " << glm::to_string(norm) << endl;
```

Lighting Calculations Using GLM

```
glm::vec3  lgtA, lgtD, lgtS; //Light's Amb, Diff, Spec
glm::vec3  matA, matD, matS; //Material Amb, Diff, Spec
glm::vec3  lgtPos, verPos; // Light and vertex position
glm::vec3  normVec, viewVec;   //Normal, view vectors
float  shin;   // shininess

//All the above are assumed to have proper values.

glm::vec3  lightVec = lgtPos - verPos;   //Light vec
lightVec = glm::normalize(lightVec);
float lDotn = glm::dot(lightVec, normVec);
glm::vec3 reflVec = glm::reflect(-lightVec, normVec);
float rDotv = glm::dot(reflVec, viewVec);
glm::vec3  color = (lgtA*matA) +
                   (lgtD*matD)*lDotn +
                   (lgtS*matS)* pow(rDotv, shin);
```

4x4 Matrices

```
glm::mat4 idMat(1);          //Identity matrix

glm::mat4 zeroMat(0);        //Null matrix

glm::mat4 aMat(5);           //Diagonal matrix

//Matrix construction using vectors
glm::vec4  v1(0.5, 0, 8, -2);   //First row
glm::vec4  v2(3, 10, 1, 0);
glm::vec4  v3(0.1, 6, -4.2, 9);
glm::vec4  v4(0, 0, 0, 1);
glm::mat4 t = glm::mat4(v1, v2, v3, v4);
```

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 5 \end{bmatrix}$$

$$\begin{bmatrix} 0.5 & 0 & 8 & -2 \\ 3 & 10 & 1 & 0 \\ 0.1 & 6 & -4.2 & 9 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matrix Operations

```
glm::mat4 trMat;                          //A matrix
glm::vec4 shift(5, -3, 7, 1);             //A vector
//Get a translation matrix for the shift distance
trMat = glm::translate(glm::mat4(1), shift);

trMatInv = glm::inverse(trMat);   //matrix inverse
trMatTrs = glm::transpose(trMat);  //transpose

glm::vec4 pt(4, 7, -1, 1);     //A point
glm::vec4  ptT = trMat * pt;  //Transformed point
```