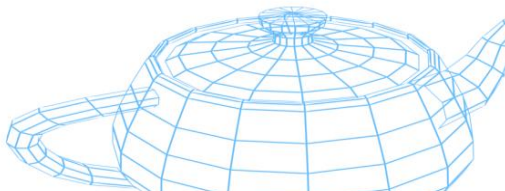
SUPSI

Computer Graphics

Mathematics for Computer Graphics – Quick summary

Achille Peternier, lecturer



Transformations

and their glm methods

$$\textbf{Translation} = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
glm::translate

$$\textbf{Scaling} = \begin{bmatrix} x & 0 & 0 & 0 \\ 0 & y & 0 & 0 \\ 0 & 0 & z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
glm::scale

$$\textbf{Rotation}_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\propto & -\sin\propto & 0 \\ 0 & \sin\propto & \cos\propto & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
glm::rotate

$$\textbf{Rotation}_y = \begin{bmatrix} \cos\propto & 0 & \sin\propto & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\propto & 0 & \cos\propto & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
glm::rotate

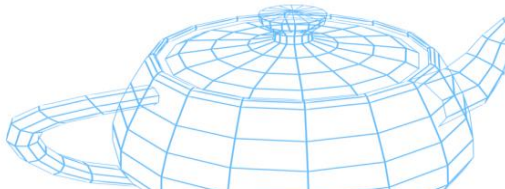
$$\textbf{Rotation}_z = \begin{bmatrix} \cos\propto & -\sin\propto & 0 & 0 \\ \sin\propto & \cos\propto & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
glm::rotate

Vertices as column vectors:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$
w

Transformation concatenation: $\mathbf{v}_n = \mathbf{T}_3\mathbf{T}_2\mathbf{T}_1\mathbf{v}_p$

(using post-multiplication and column vectors)



Projections

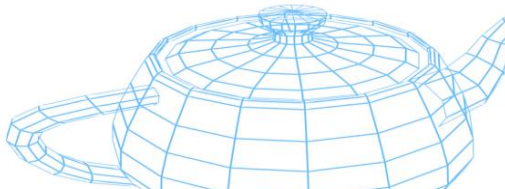
Orthographic =
glm::ortho

$$\begin{bmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{right+left}{right-left} \\ 0 & \frac{2}{top-bottom} & 0 & -\frac{top+bottom}{top-bottom} \\ 0 & 0 & \frac{-2}{far-near} & -\frac{far+near}{far-near} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

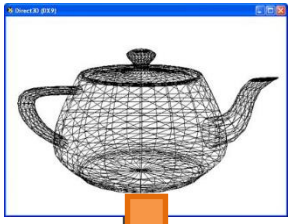
Perspective =
glm::perspective

$$\begin{bmatrix} \frac{f}{aspect} & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & \frac{far+near}{near-far} & \frac{2 \times far \times near}{near-far} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

fieldOfView	vertical (y) view angle
f	cotangent(*fieldOfView*/2)
aspect	aspect ratio (4:3, 16:9, etc.)



Coordinate spaces



World Transformations

Lighting

Projection Transformations

Clipping

Rasterization

Object coordinates glMatrixMode(GL_MODELVIEW)
World coordinates
Eye coordinates

Clip coordinates glMatrixMode(GL_PROJECTION)
Division by *w*

Normalized device coordinates
Screen coordinates

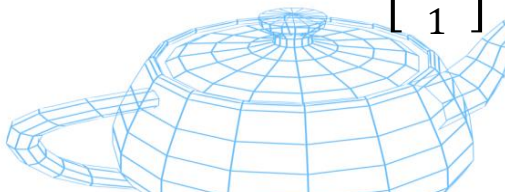
user-specified

OpenGL*

E.g., from object to clip coordinates:

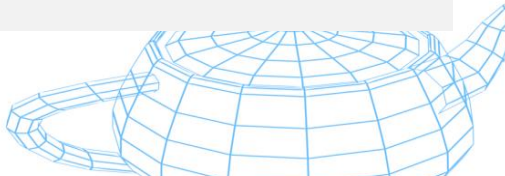
$$\begin{bmatrix} clip_x \\ clip_y \\ clip_z \\ clip_w \end{bmatrix} = \text{projMat} * \text{cameraMat}^{-1} * \text{transMat} * \text{rotMat} * \text{scaleMat} * \begin{bmatrix} obj_x \\ obj_y \\ obj_z \\ 1 \end{bmatrix}$$

** Screen sizes are specified via glViewport()*



GLM


```
// Main include:
#include <glm/glm.hpp>

// Extension for deprecated OpenGL matrix transformations:
#include <glm/gtc/matrix_transform.hpp>

glm::scale(glm::mat4(1.0f), glm::vec3(0.5f));
glm::rotate(glm::mat4(1.0f), glm::radians(90.0f), glm::vec3(0.0f, 0.0f, 1.0f));
glm::translate(glm::mat4(1.0f), glm::vec3(10.0f, 0.0f, 0.0f));
glm::perspective(glm::radians(45.0f), 1.0f, 1.0f, 100.0f);

// Extension for printing variables via to_string(var):
#include <glm/ext.hpp>

// Extension for using common math constants:
#include <glm/gtc/constants.hpp>

// Shortcuts:
glm::mat4();      // 4x4 identity matrix
glm::mat4(1.0f); // 4x4 identity matrix
glm::vec3(0.5f); // same as glm::vec3(0.5f, 0.5f, 0.5f)
```



GLM

Matrices in the documentation:

$$\begin{bmatrix} a & e & i & m \\ b & f & j & n \\ c & g & k & o \\ d & h & l & p \end{bmatrix}$$

Matrices in GLM:

```
glm::mat4 mat( a, b, c, d,
               e, f, g, h,
               i, j, k, l,
               m, n, o, p );
```

Example:

Translation $= \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$

```
glm::mat4 translation(1, 0, 0, 0,
                      0, 1, 0, 0,
                      0, 0, 1, 0,
                      x, y, z, 1);
// or:
glm::mat4 translation =
glm::translate(glm::mat4(),
               vec3(x, y, z));
```