COMPUTER GRAPHICS

CSCI 173

California State University, Fresno

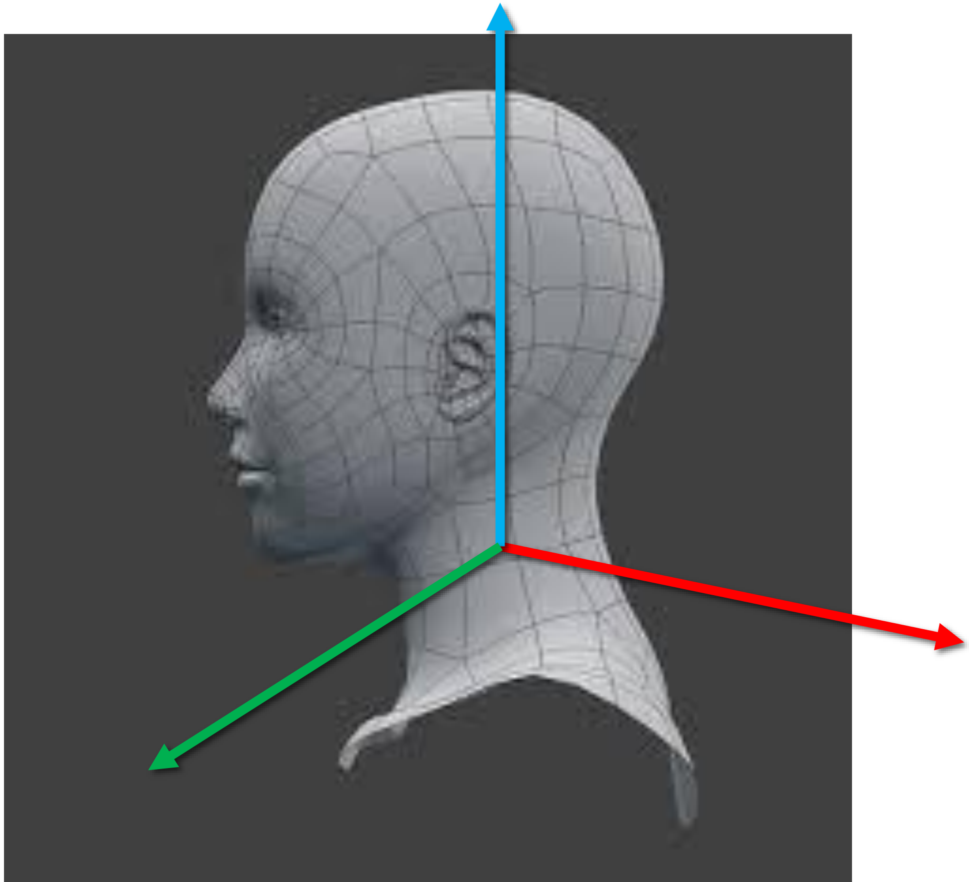
MODELVIEW Matrix Modes

- glMatrixMode(GL_MODELVIEW);
- glMatrixMode(GL_PROJECTION);

- glLoadIdentity() - **glLoadIdentity** replaces the current matrix with the identity matrix
- The Model, View and Projection matrices

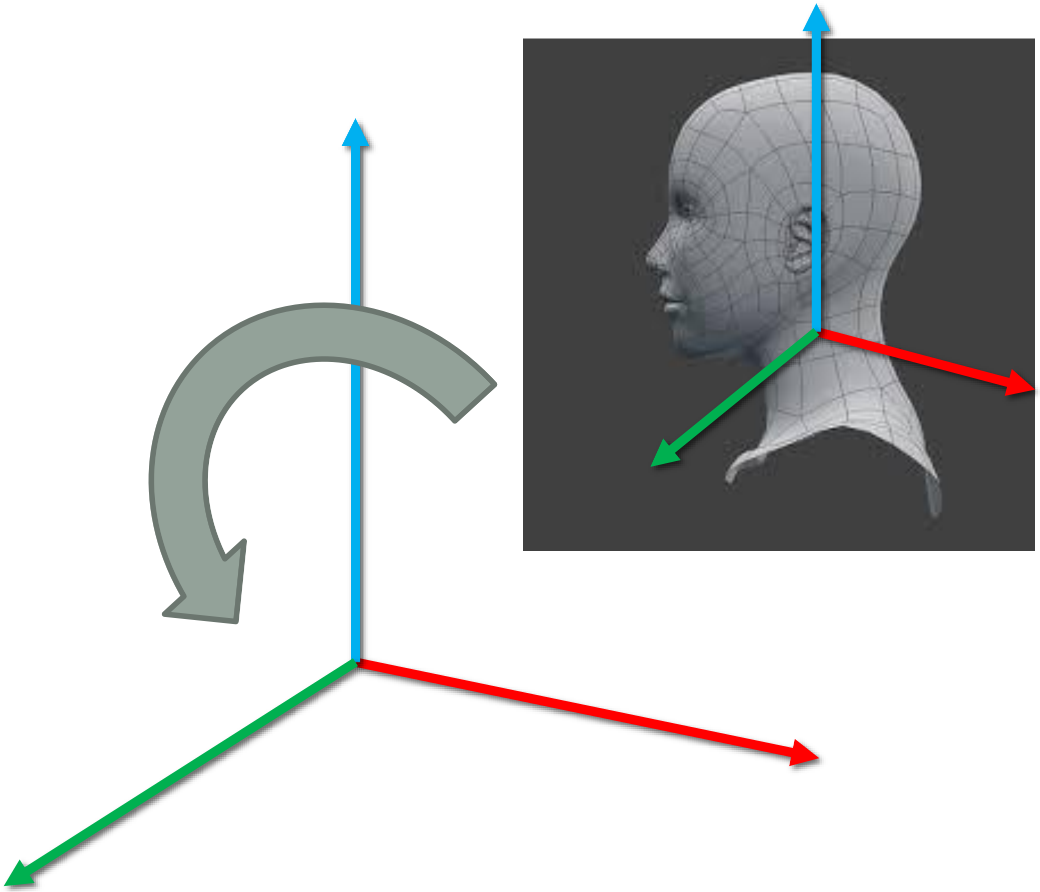
The Model Matrix

- The X,Y,Z coordinates of a mesh are defined relative to the object's center



The Model Matrix

- Objects must move on the stage relative to one pivot point



We must connect Model Space (all vertices defined relatively to the center of the model)

To

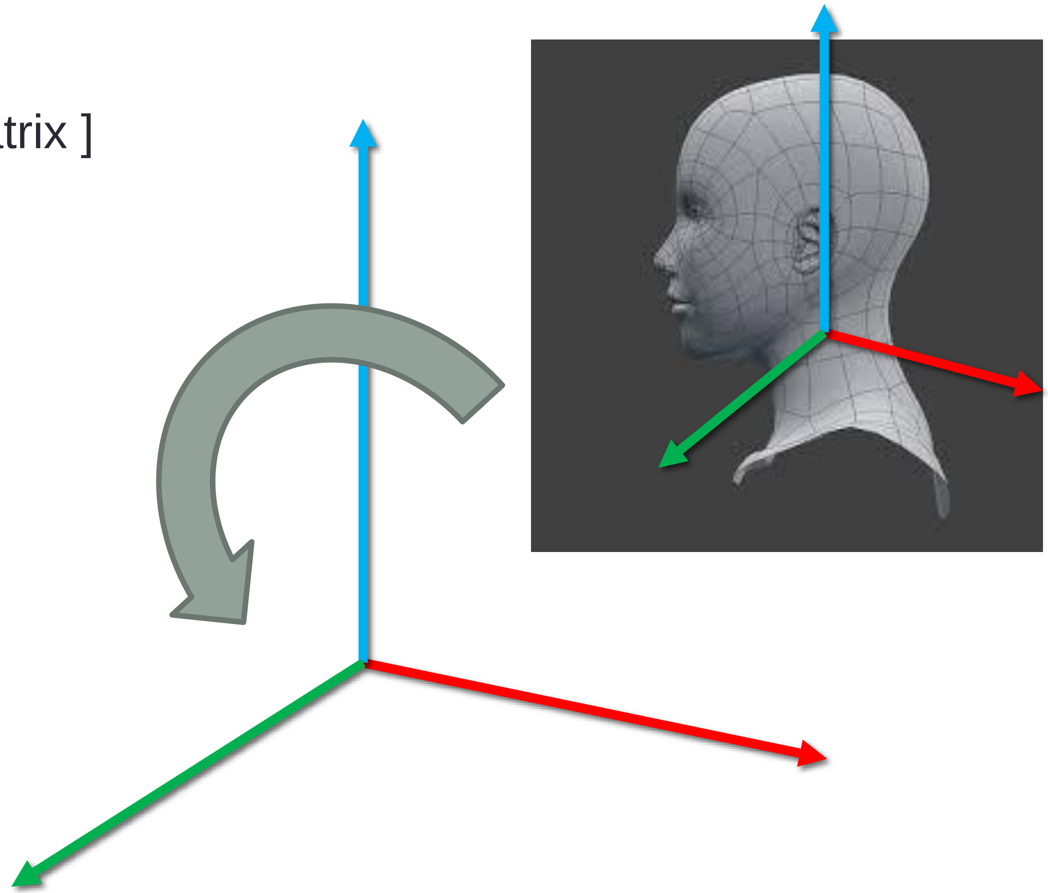
World Space (all vertices defined relatively to the center of the world).

The Model matrix

- Model Coordinates

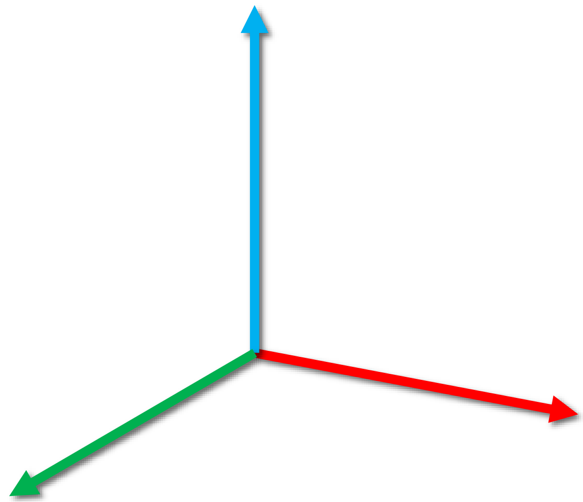
[Model Matrix]

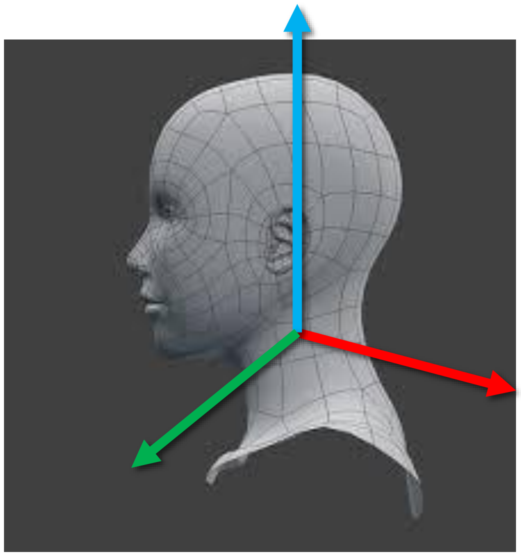
- World Coordinates



The View matrix

- *The ship stays where it is and the engines move the universe around it.*
- 





It you want to view a mountain from another angle, you can either

1. Move the camera
2. Move the mountain

The View matrix

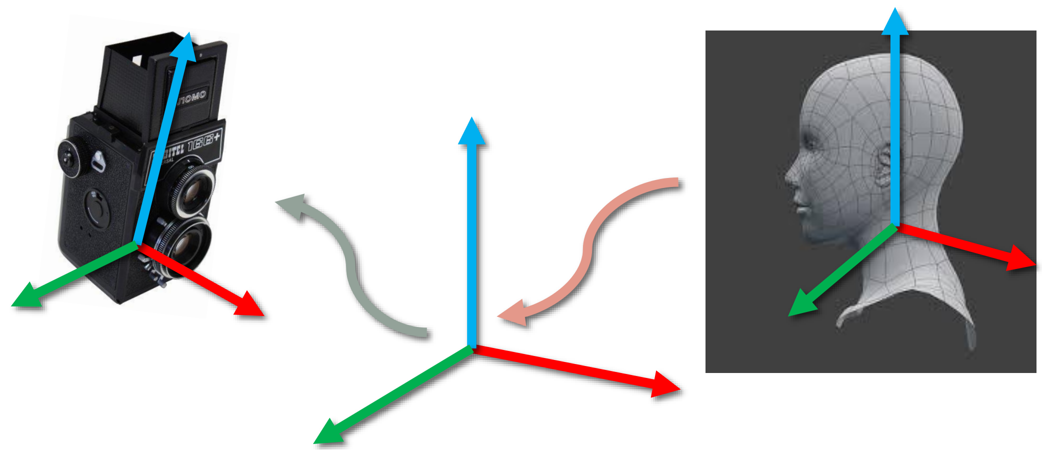
- So initially your camera is at the origin of the World Space. In order to move the world, you simply introduce another matrix
- Model Coordinates

  [Model Matrix]

- World Coordinates

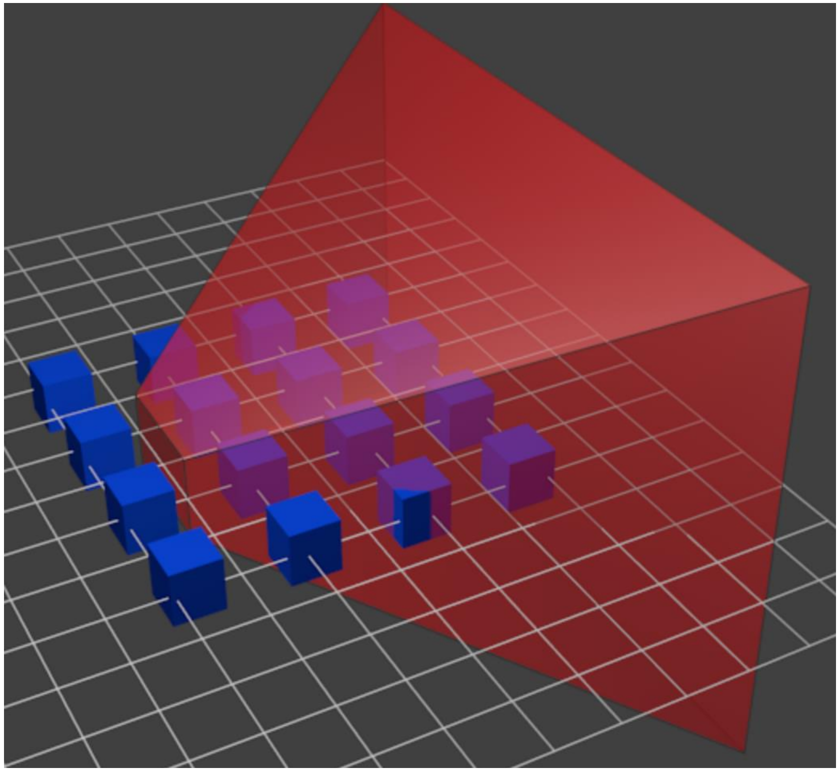
  [Model Matrix]

- Camera Coordinates



The Projection matrix

- In order to represent realistic depth we may apply perspective projection



The Projection matrix

- Model Coordinates

[Model Matrix]

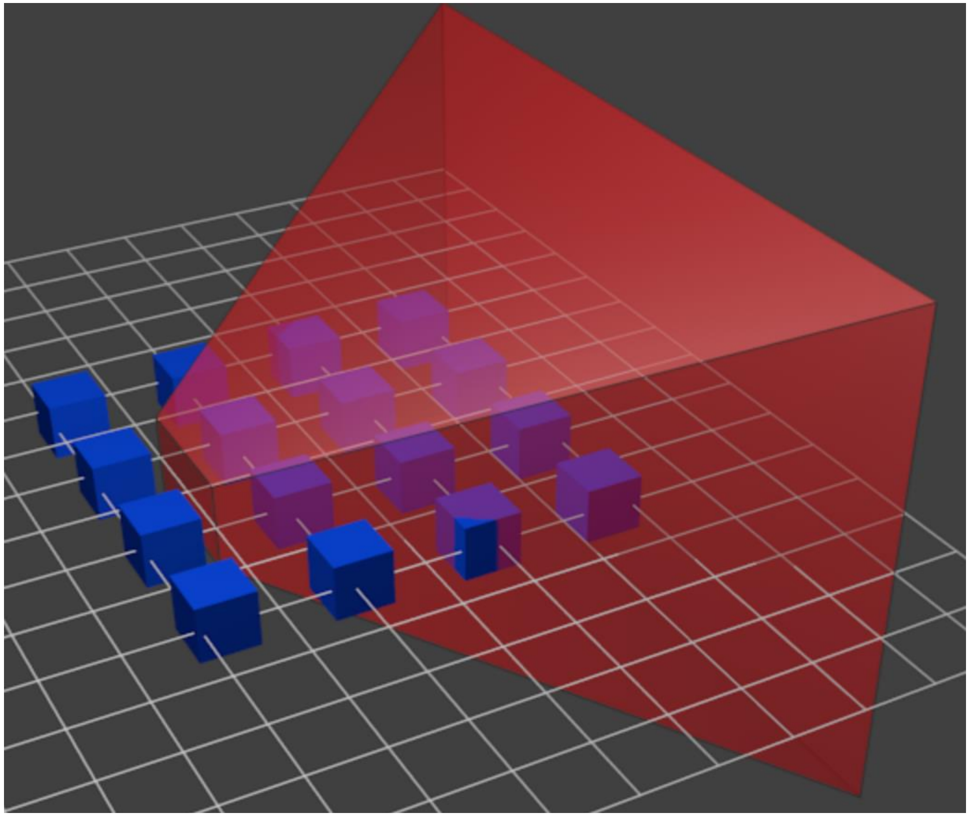
- World Coordinates

[View Matrix]

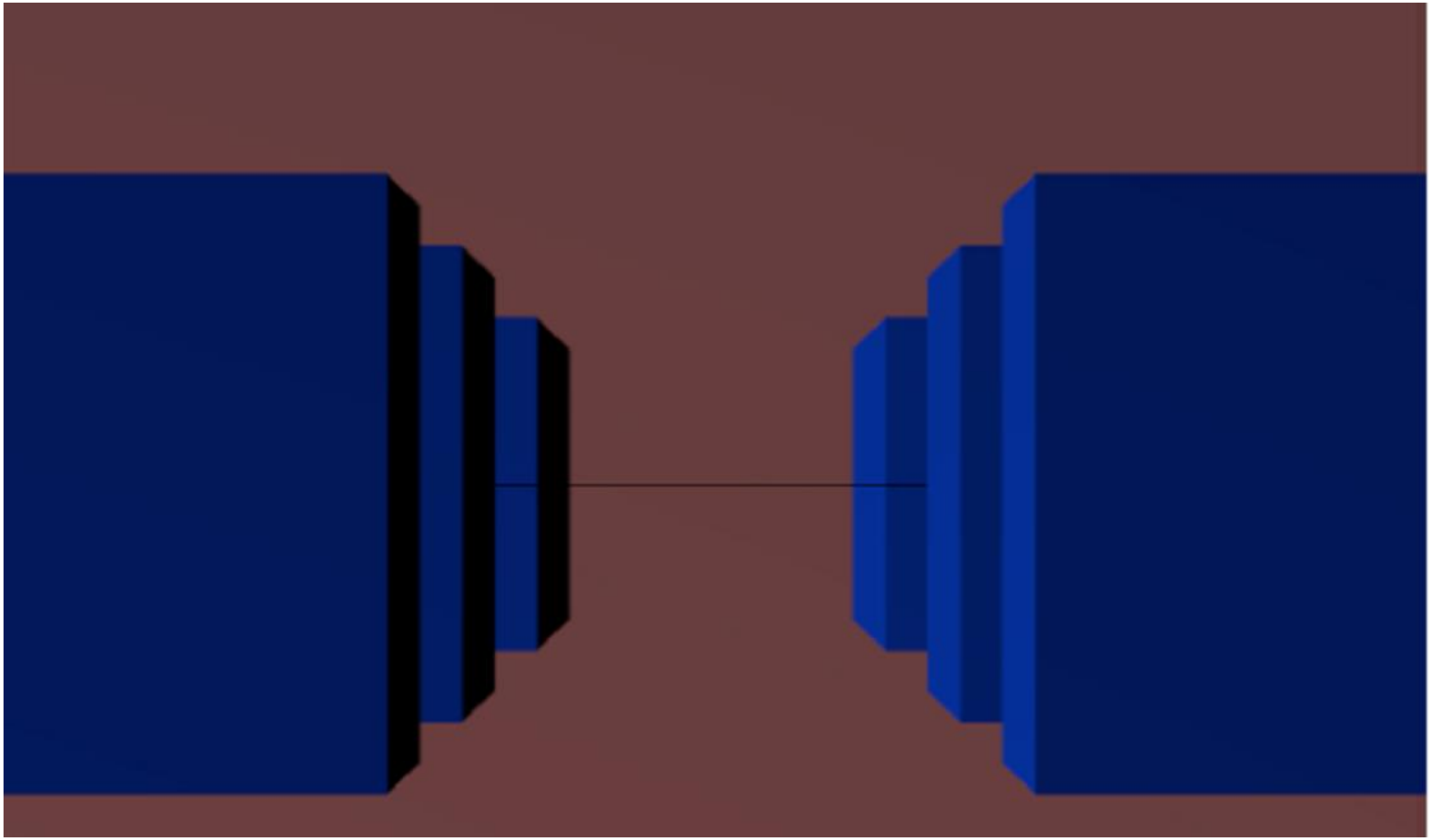
- Camera Coordinates

[Projection Matrix]

- Homogenous Coordinates



Final Results



Model Matrix Examples

Translation:

$$T = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation:

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, R_y = \begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, R_z = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Model Matrix Examples

Scaling:

$$S = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The view matrix Example

- view matrix that simulates a moving camera, usually named lookAt.
  - The *eye*, or the position of the viewer
  - The *center*, or the point where we the camera aims
  - The *up*, which defines the direction of the up for the viewer
  - defaults in OpenGL are
    - *eye* at (0, 0, -1);
    - *center* at (0, 0, 0);
    - *up* at *Oy* axis (0, 1, 0)

    - Results of the application will be $v' = V \cdot M \cdot v$

The Projection Matrix Examples

- The *orthographic* projection matrix:

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

- The perspective projection matrix is:

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

The Projection Matrix Examples

```
void glOrtho( GLdouble left,
              GLdouble right,
              GLdouble bottom,
              GLdouble top,
              GLdouble nearVal,
              GLdouble farVal);
```

orthographic matrix

```
void glFrustum( GLdouble left,
                GLdouble right,
                GLdouble bottom,
                GLdouble top,
                GLdouble nearVal,
                GLdouble farVal);
```

perspective matrix

The Projection Matrix Examples

```
gluPerspective( GLdouble fovy,
                GLdouble aspect,
                GLdouble zNear,
                GLdouble zFar);
```

set up a perspective projection matrix

$$top = near \cdot \tan\left(\frac{\pi}{180} \cdot FOV/2\right)$$

$$bottom = -top$$

$$right = top \cdot aspect$$

$$left = -right$$

Final output

$$v' = P \cdot V \cdot M \cdot v$$