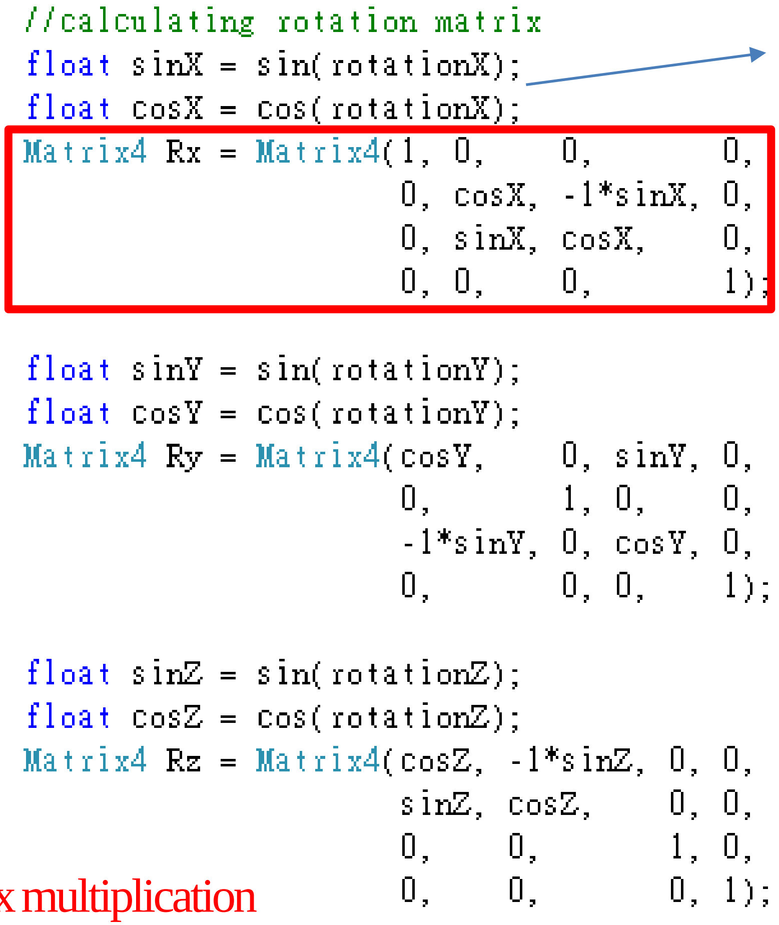
Matrix class

(provided by Matrices.h and .cpp)

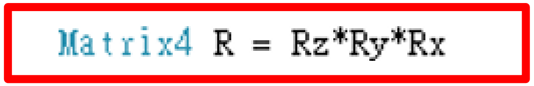


```
//calculating rotation matrix
float sinX = sin(rotationX);
float cosX = cos(rotationX);
Matrix4 Rx = Matrix4(1, 0,     0,        0,
                     0, cosX, -1*sinX, 0,
                     0, sinX, cosX,    0,
                     0, 0,     0,        1);

float sinY = sin(rotationY);
float cosY = cos(rotationY);
Matrix4 Ry = Matrix4(cosY,     0, sinY, 0,
                     0,        1, 0,    0,
                     -1*sinY, 0, cosY, 0,
                     0,        0, 0,    1);

float sinZ = sin(rotationZ);
float cosZ = cos(rotationZ);
Matrix4 Rz = Matrix4(cosZ, -1*sinZ, 0, 0,
                     sinZ, cosZ,    0, 0,
                     0,    0,       1, 0,
                     0,    0,       0, 1);
```

#include <cmath>

Matrix initialization

Matrix multiplication



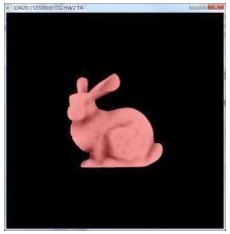
```
Matrix4 R = Rz*Ry*Rx
```

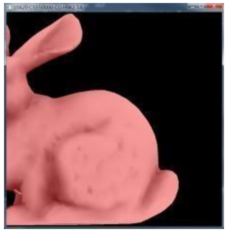
Geometrical Transformation

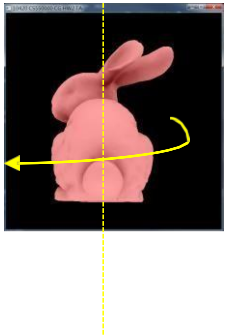
- Manipulate 3D models
  - Translation, scaling,rotation

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

P **V** **M**







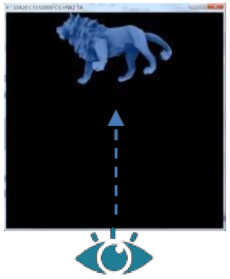
- Normalize the model with transformation matrix!

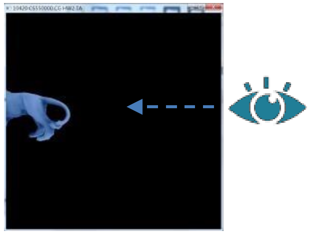
Viewing Transformation

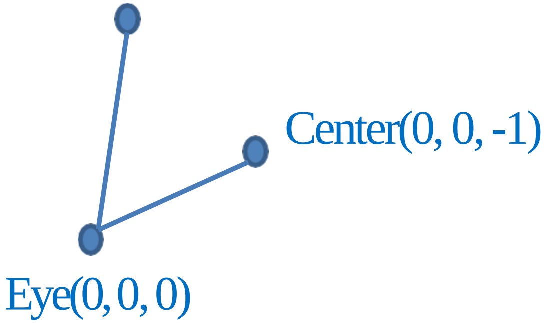
- Display 3D models from different view.
  - Eye position, center position, up position

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

P **V** **M**

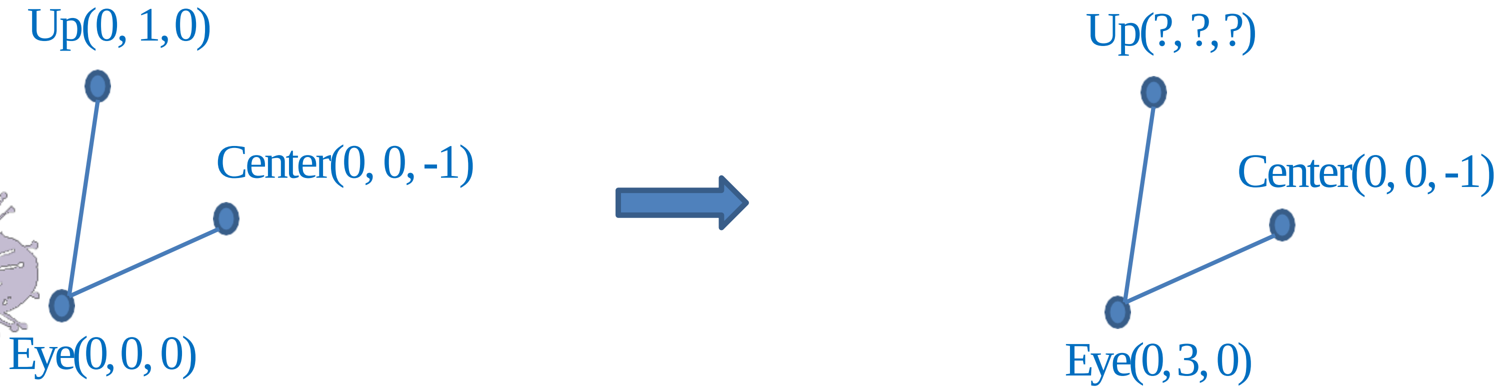






Relationship between Eye, Center, Up

- When change eye(center) position, we have to adjust up position to get a proper result, here is an example, if we move eye from (0,0,0) to (0,3,0)



Up(0, 1, 0)

Center(0, 0, -1)

Eye(0, 0, 0)

Up(?, ?, ?)

Center(0, 0, -1)

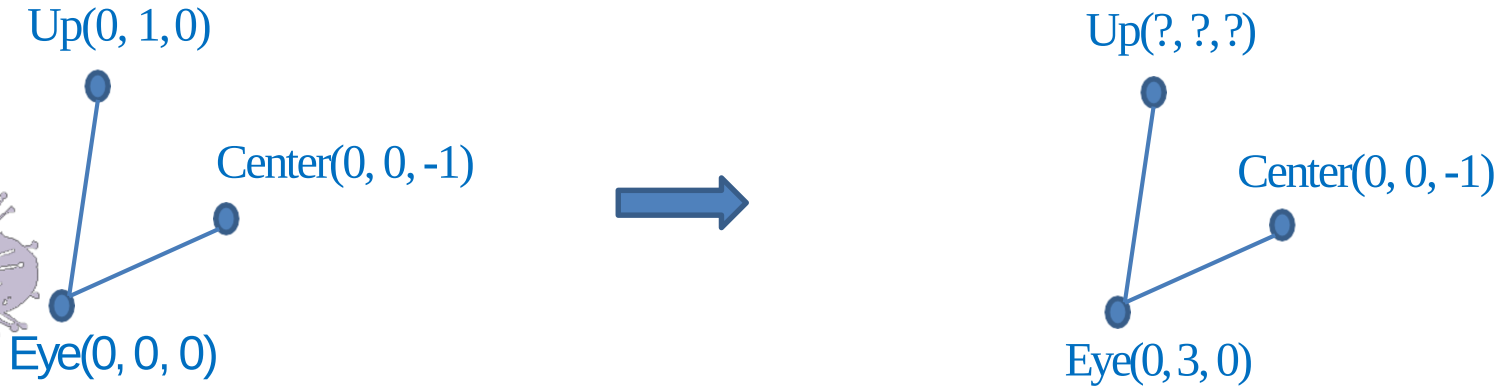
Eye(0, 3, 0)

Forward = center(0,0,-1) - eye(0,0,0) = (0,0,-1)

Forward = center(0,0,-1) - eye(0,3,0) = (0,-3,-1)

Relationship between Eye, Center, Up

- Because the forward vector and up vector must be perpendicular, now forward vector changed, we need to compute new upvector



Up(0, 1, 0)

Center(0, 0, -1)

Eye(0, 0, 0)

Up(?, ?, ?)

Center(0, 0, -1)

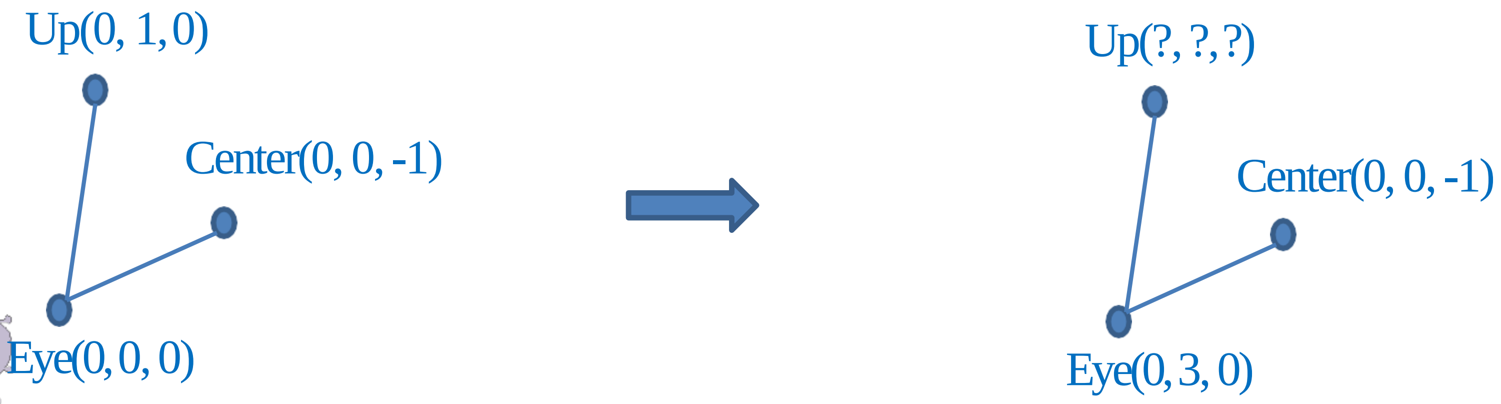
Eye(0, 3, 0)

Forward = center(0,0,-1) - eye(0,0,0) = (0,0,-1)

Forward = center(0,0,-1) - eye(0,3,0) = (0,-3,-1)

Relationship between Eye, Center, Up

- In this case, we can compute right vector by the cross product of forward and up vector(old one)



Up(0, 1, 0)

Center(0, 0, -1)

Eye(0, 0, 0)

Up(?, ?, ?)

Center(0, 0, -1)

Eye(0, 3, 0)

Forward = center(0,0,-1) - eye(0,0,0) = (0,0,-1)

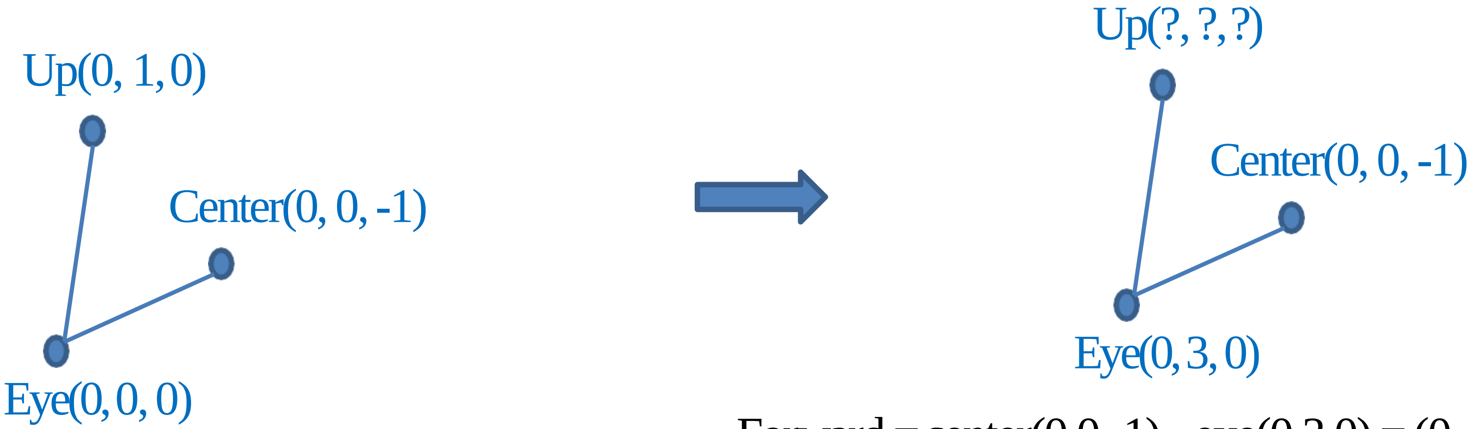
Right = forward(0,0,-1) X up(0,1,0) = (1,0,0)

Forward = center(0,0,-1) - eye(0,3,0) = (0,-3,-1)

Right = forward(0,-3,-1) X up(0,1,0) = (1,0,0)

Relationship between Eye, Center, Up

- Finally re-compute new up vector by the cross product of right vector and forward vector, and find new up position!



Forward = center(0,0,-1) - eye(0,0,0) = (0,0,-1)
Right = forward(0,0,-1) X up(0,1,0) = (1,0,0)
Up = right(1,0,0) X forward(0,0,-1) = (0,1,0)

Forward = center(0,0,-1) - eye(0,3,0) = (0,-3,-1)
Right = forward(0,-3,-1) X up(0,1,0) = (1,0,0)

Up vector(new) =
right(1,0,0) X forward(0,-3,-1) = (0,1,-3)
Up(position) =
eye(0,3,0) + up vector(0,1,-3) = (0, 4, -3)

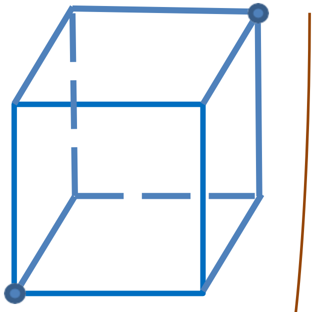
Projection Transformation

- Project 3D models on screen in different way.
  - Parallel(orthogonal), perspective projection

$$\underset{\mathbf{P}}{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}} \underset{\mathbf{V}}{\begin{bmatrix} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{bmatrix}} \underset{\mathbf{M}}{\begin{bmatrix} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{bmatrix}}$$

Default value in HW1 (parallel)

(xmax, ymax, zfar) = (1, 1, -1)



(xmin, ymin, znear) = (-1, -1,1)

Coordinates!

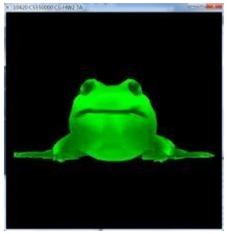
- Reference
  - Transformation p.81-p.95

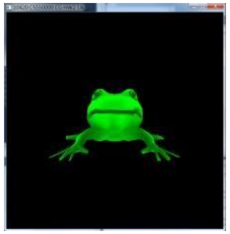
Projection Transformation

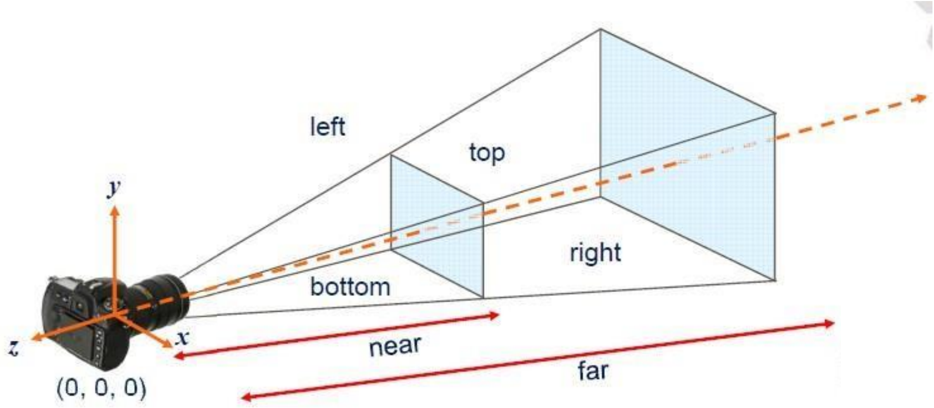
- Project 3D models on screen in different way.
  - Parallel(orthogonal), perspective

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

P **V** **M**






left
top
y
right
bottom
z
x
near
far
(0, 0, 0)