COMP3271 Computer Graphics

Viewing

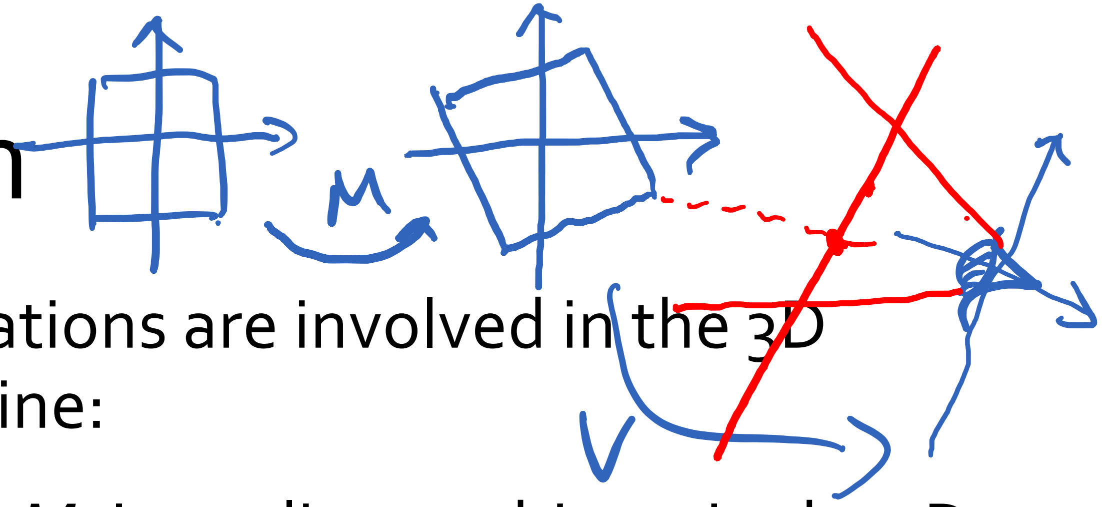
2019-20

Objectives

Understand the viewing process

Derive the projection matrices used for standard OpenGL projections

Transformation



Three kinds of transformations are involved in the 3D graphics processing pipeline:

- model transformation *M*: It applies to objects in the 3D world coordinate system (the object space);
- view transformation *V*: It maps objects from the 3D world coordinate system to the 3D eye coordinate system, with the origin at the eye-point (viewpoint);
- view projection *P*: It maps objects from the 3D eye-coordinate system to the 2D view plane.

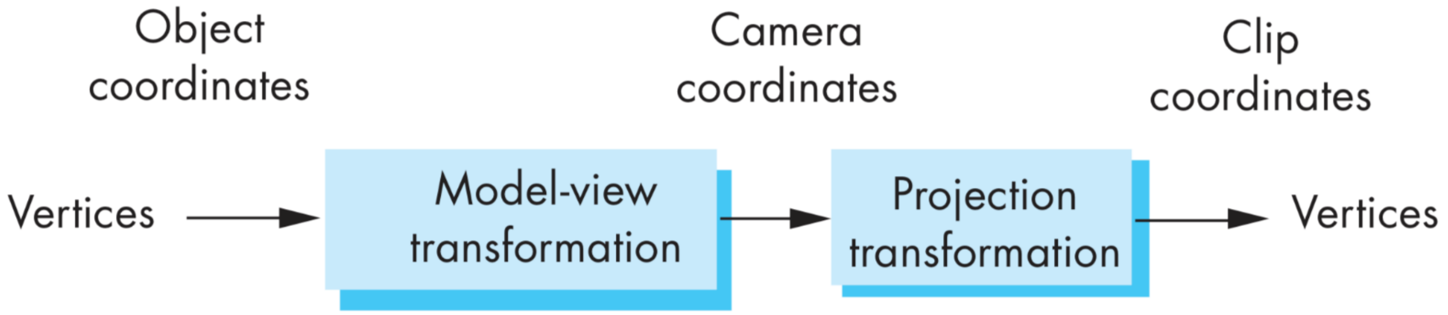
A vertex will be transformed by the concatenation of these transformations before appearing on screen

$$X_3' = P_{3\times4}\boxed{V_{4\times4}M_{4\times4}}X_4.$$

Viewing

There are two main steps in the viewing process:

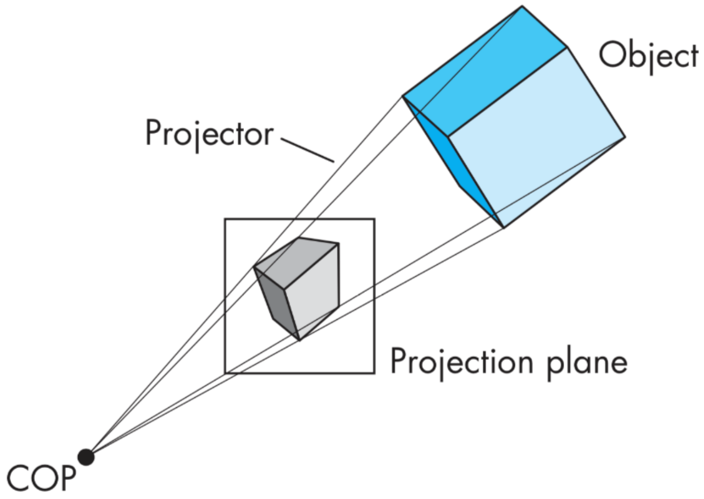
- Position and orient the camera
  - Setting the model-view matrix
  - Vertices in object coordinates will be transformed to eye or camera coordinates
- Selecting a lens
  - Setting the projection matrix
  - Normalize to a canonical view volume



Viewing

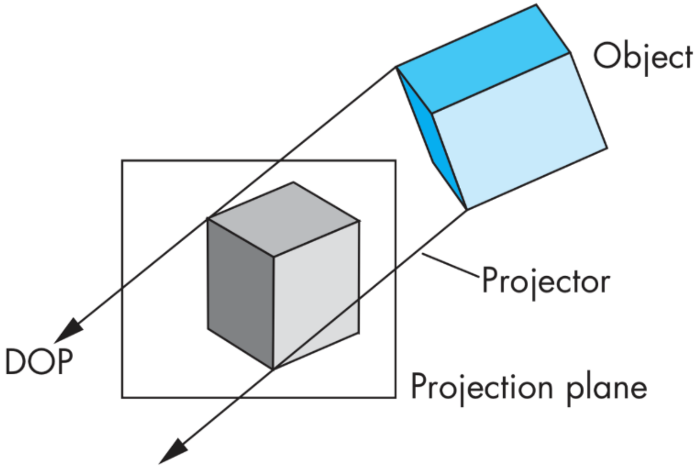
Projection determines how objects appear on screen

Perspective projection



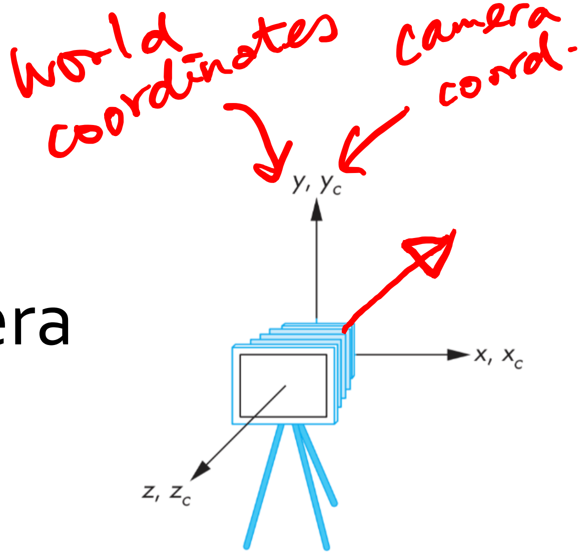
COP: Center of Projection
Original of the camera frame

Orthogonal projection



DOP: Direction of Projection
same as COP at infinity

The OpenGL Camera



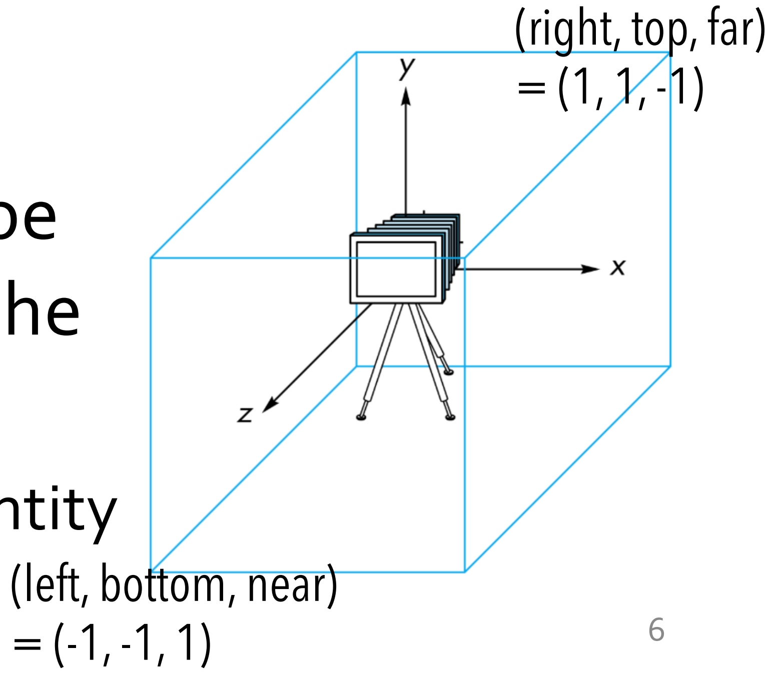
In OpenGL, initially the object and camera frames are the same

- Default model-view matrix is an identity

The camera is located at origin and points in the negative z direction

OpenGL also specifies a default canonical view volume that is a cube with sides of length 2 centered at the origin

- Default projection matrix is an identity (i.e., orthogonal projection)



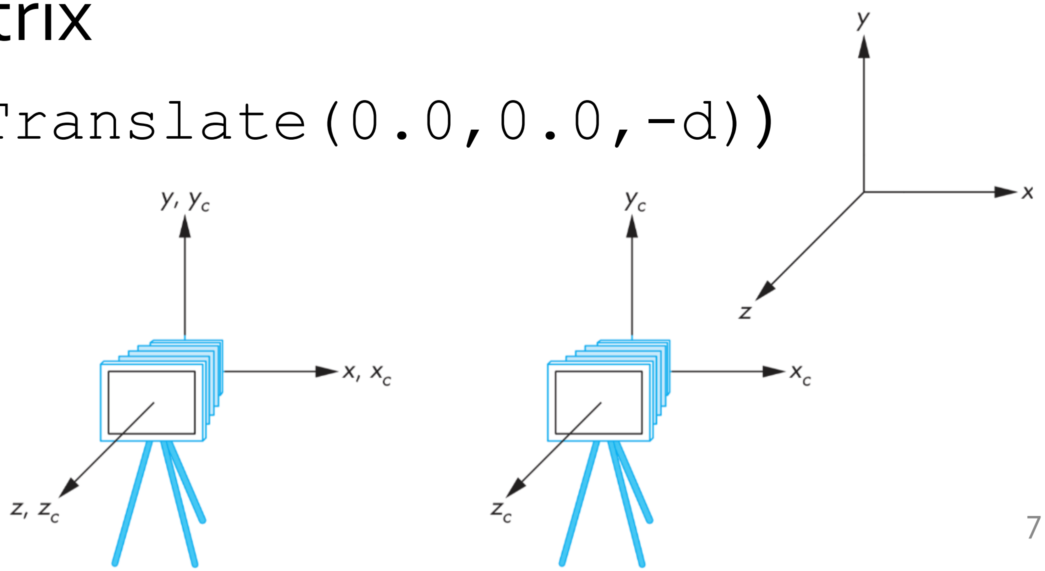
Moving the Camera Frame

Consider

- Moving the camera in the positive z direction
  - Translate the camera frame
- Moving the objects in the negative z direction
  - Translate the world frame

Both of these views are equivalent and are determined by the model-view matrix

- Want a translation (`Translate(0.0,0.0,-d)`)
- `d > 0`



Moving the Camera Frame

We can move the camera to any desired position by a sequence of rotations and translations

Example: side view

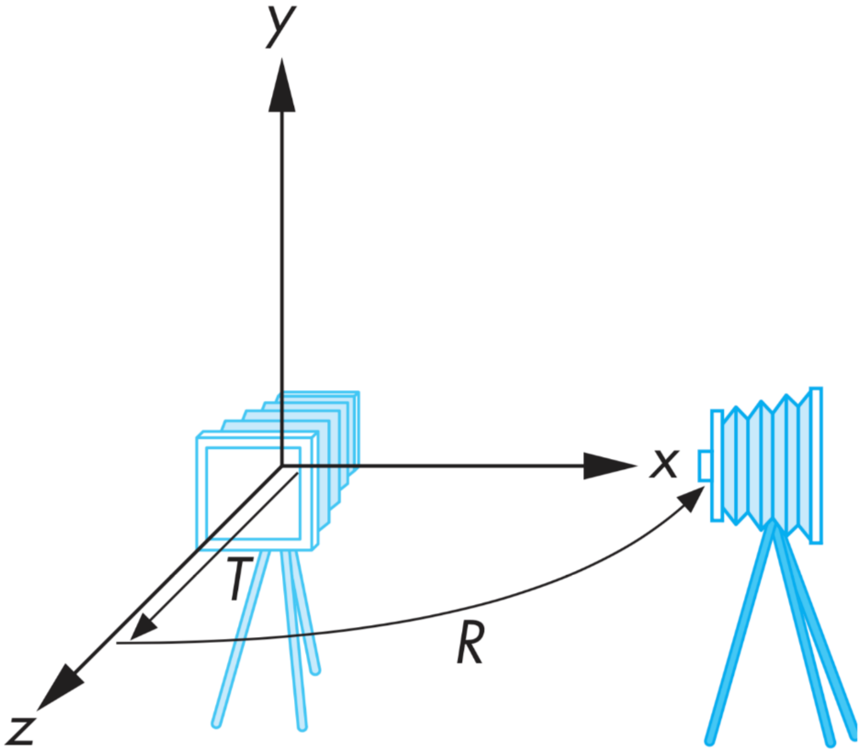
- Rotate the camera
- Move it away from origin
- Model-view matrix C = TR

Move Camera :

$$R_y(\theta) \cdot T_z(d)$$

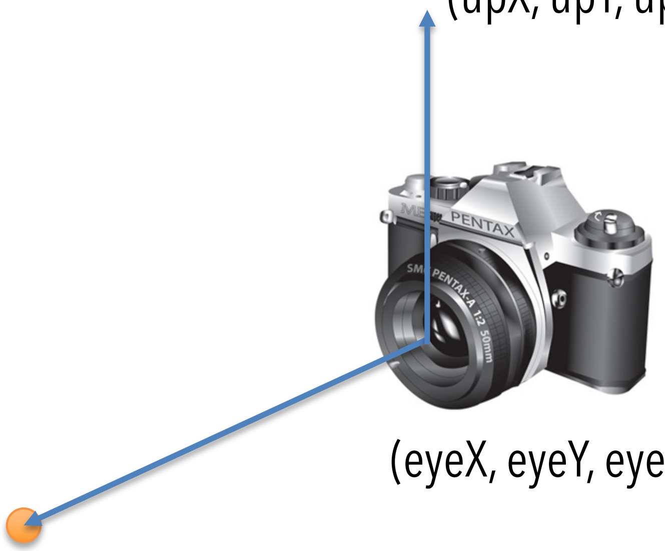
Move World :

$$T_z(-d) \cdot R_y(-\theta)$$



OpenGL API

```
LookAt(eyeX,eyeY,eyeZ,
 centerX,centerY,centerZ,upX,upY,upZ);
```



(upX, upY, upZ): a vector

(eyeX, eyeY, eyeZ): a point

(centerX, centerY, centerZ): a point

Note that this is a transformation that applies to the ModelView matrix