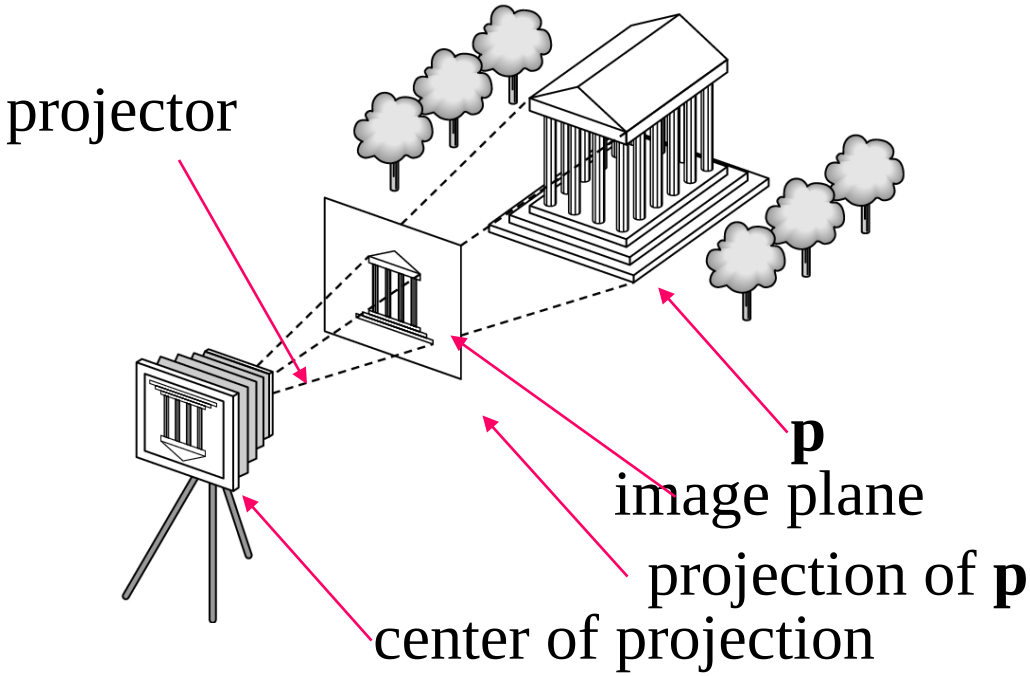
Synthetic Camera Model



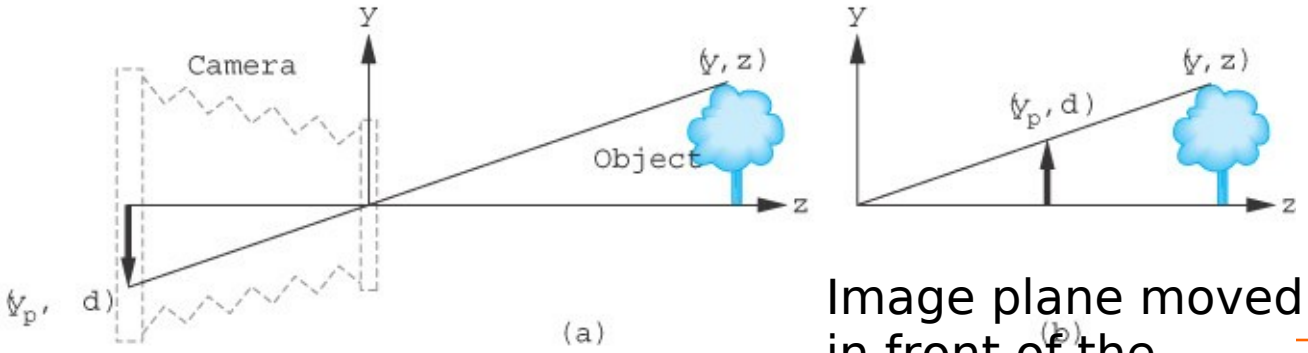
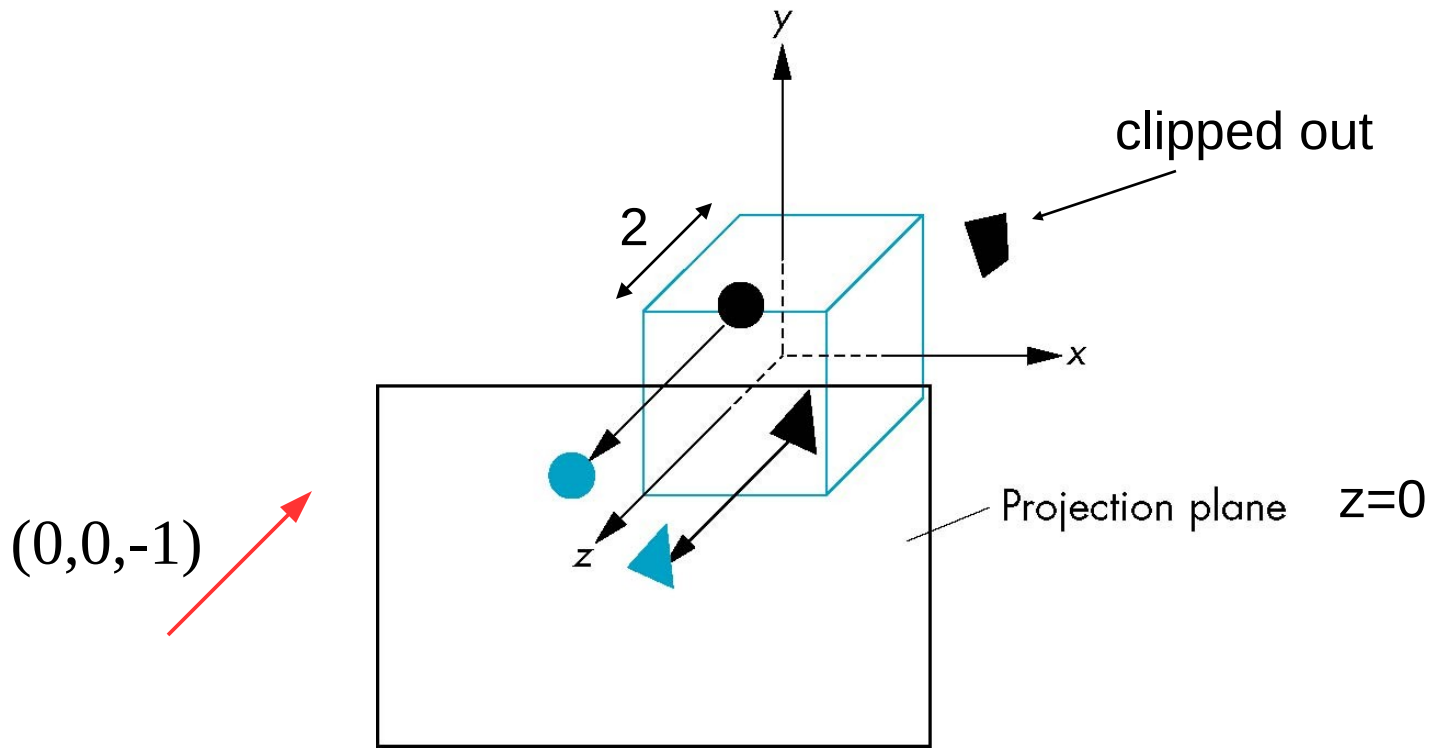


Image formed on the back of the camera

Image plane moved in front of the camera

Orthogonal Projection

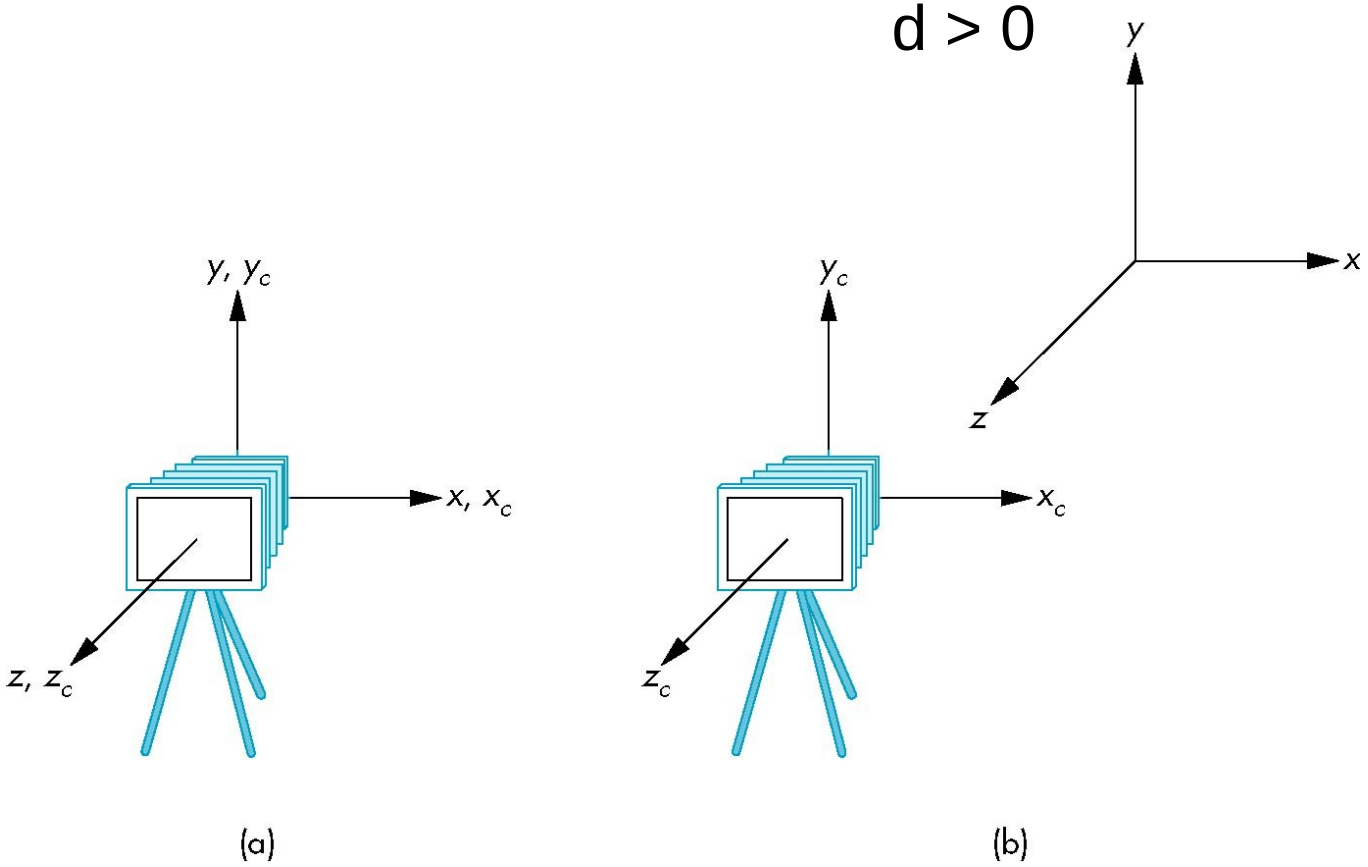
Default projection is orthogonal



Moving Camera back

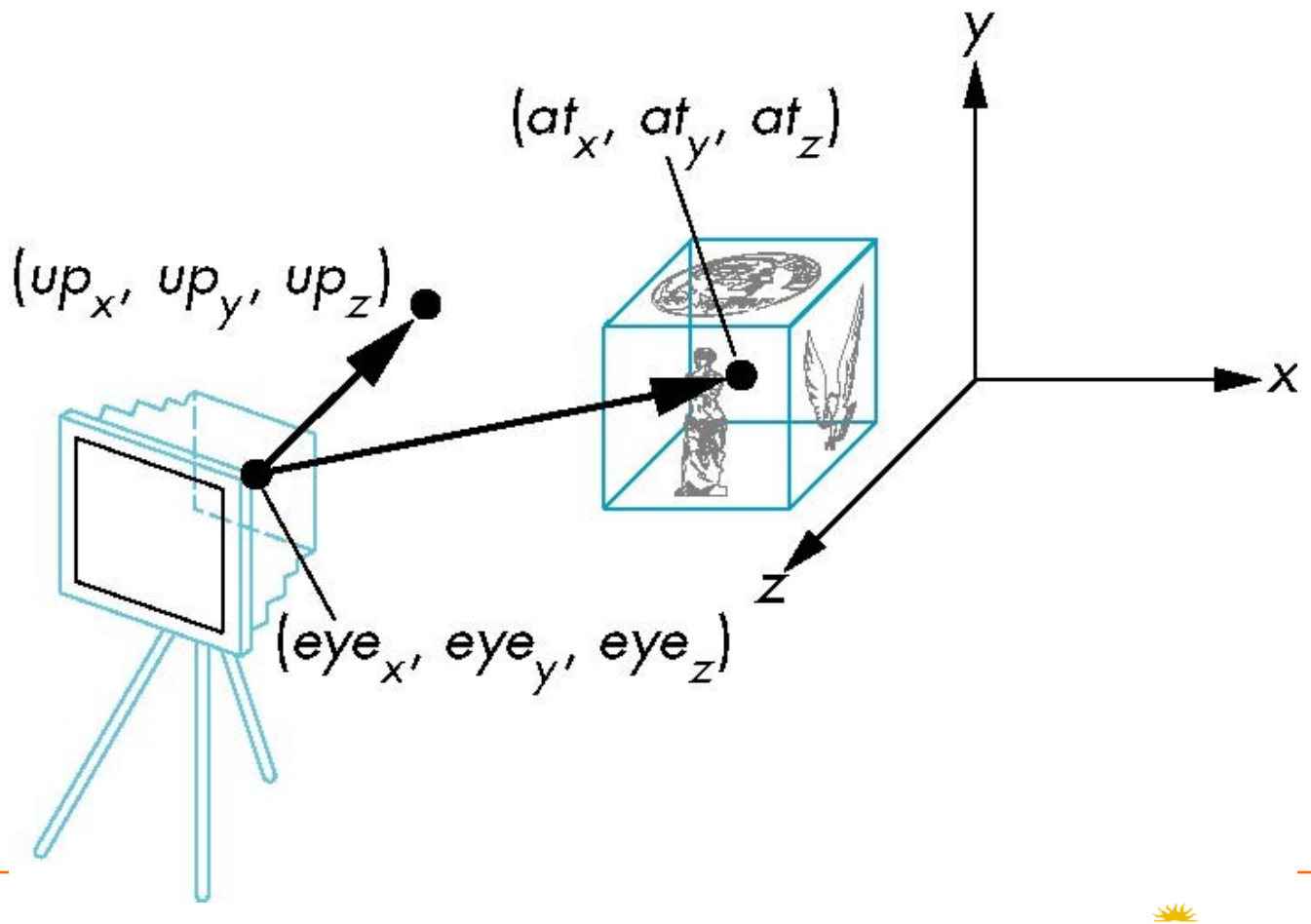
frames after translation by –d

d > 0

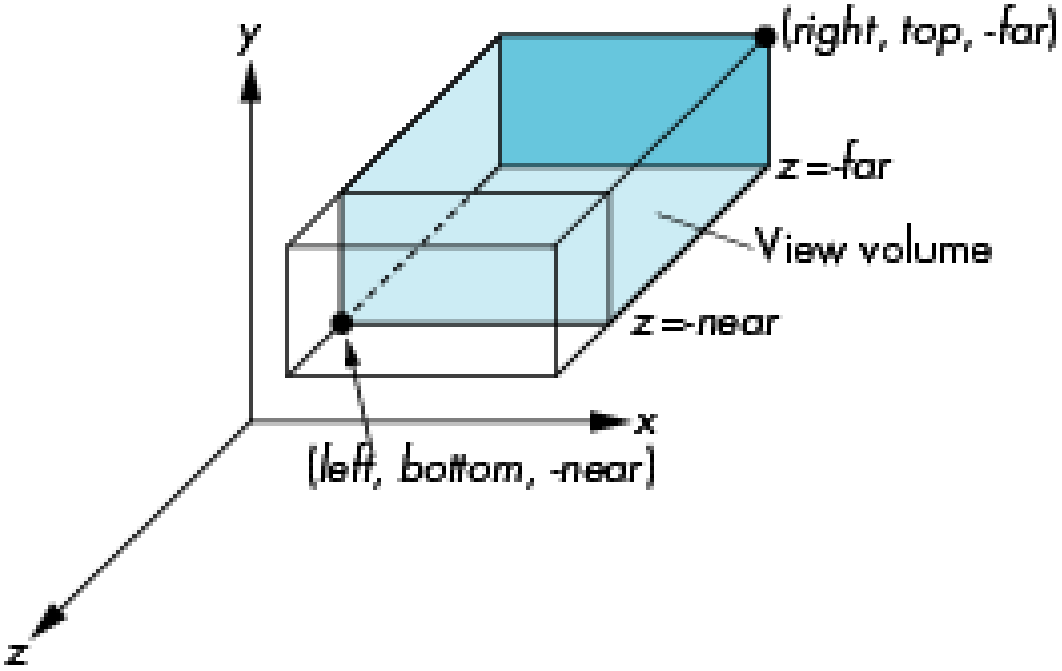


LookAt

```
glLookAt(eyex, eyey, eyez, atx, aty, atz, upx, upy,
   upz)
```



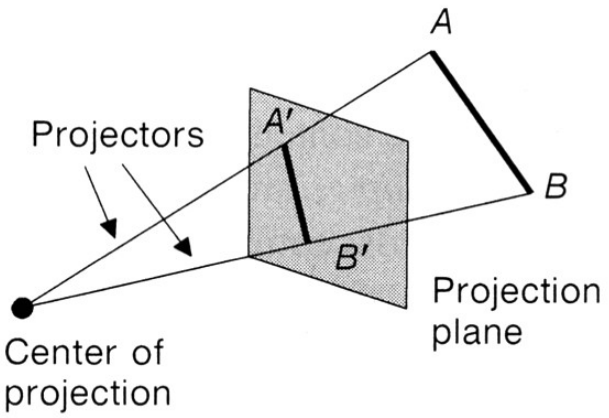
Orthogonal Viewing

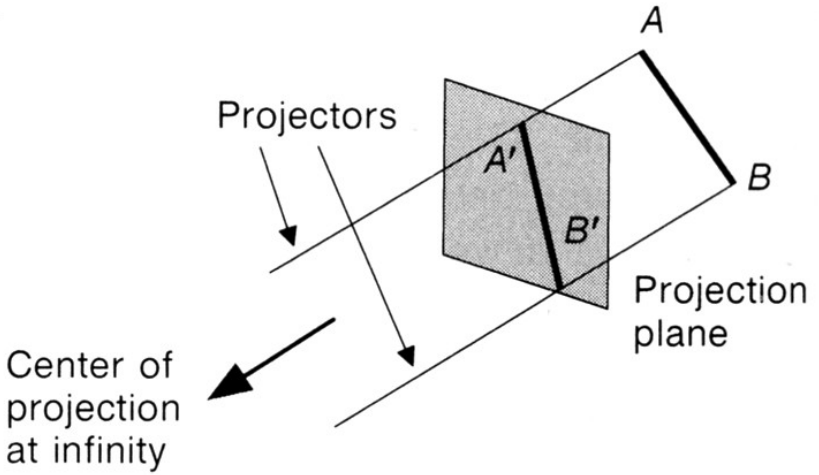


near and **far** measured from camera

Projection

- *Projection*: from 3D objects to 2D image
  - Perspective projections: all projectors meet at the center of projection
  - Parallel (orthogonal) projection: projectors are parallel, center of projection is replaced by a direction of projection

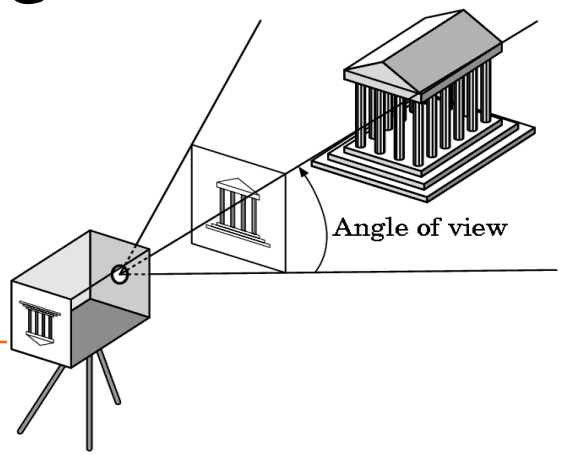


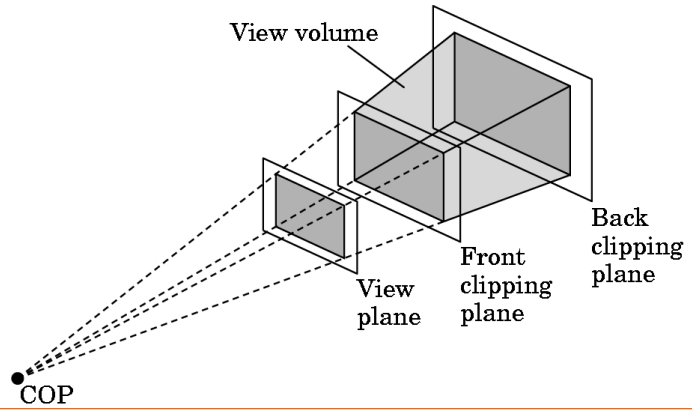


Clipping

Just as a real camera cannot “see” the whole world, the virtual camera can only see part of the world or object space

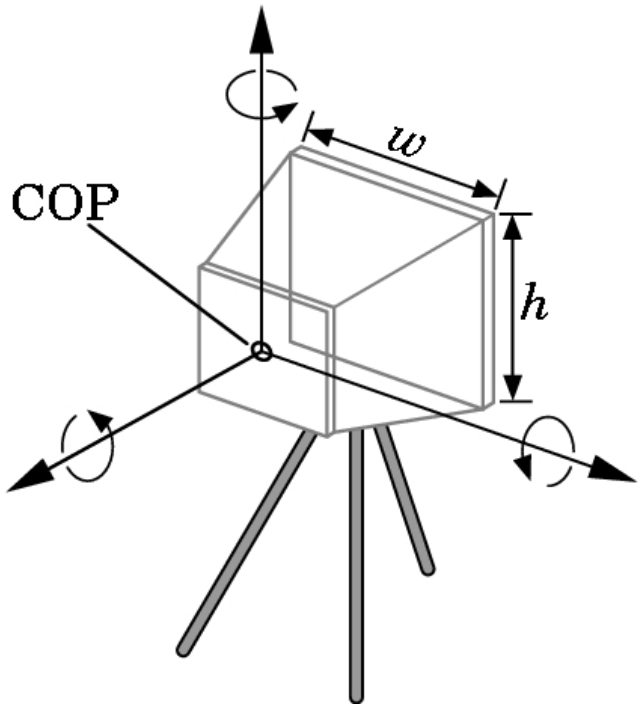
- Objects that are not within this view volume are said to be *clipped* out of the scene





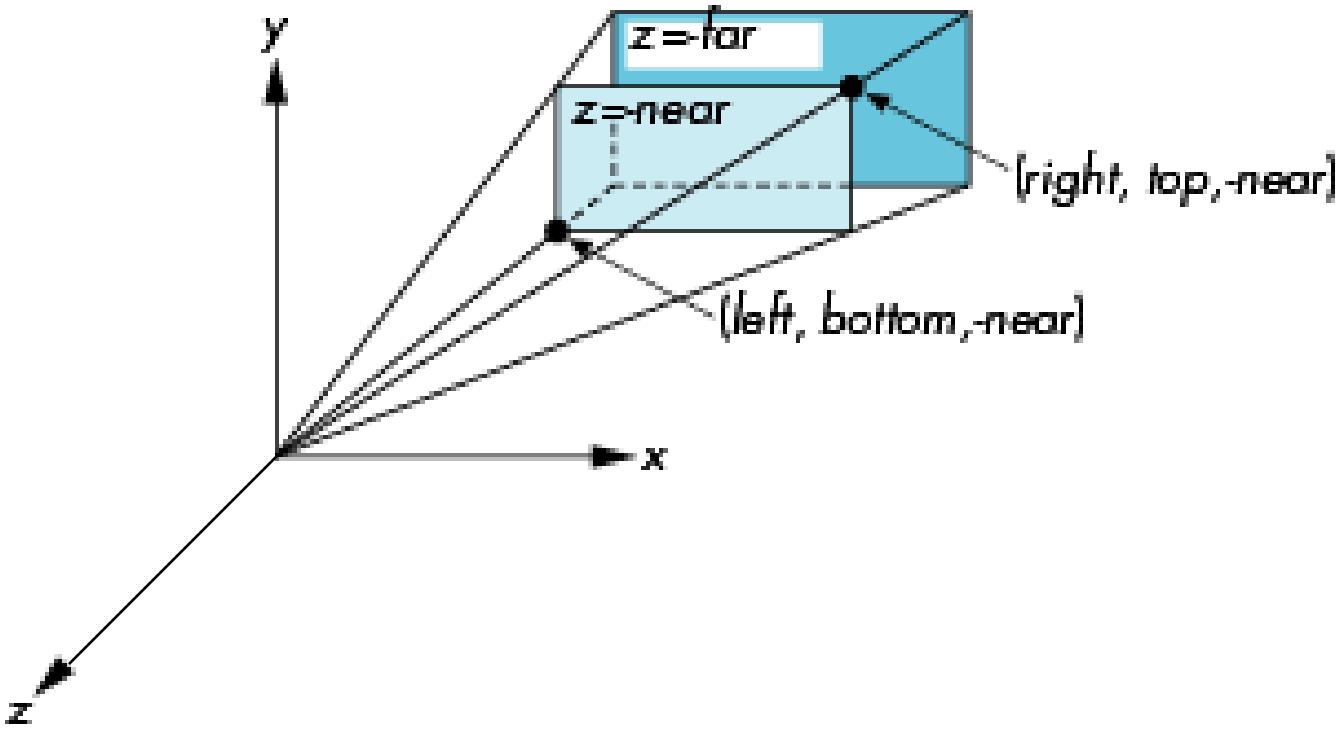
Camera Specification

- Six degrees of freedom
  - Position of center of lens (COP)
  - Orientation
- Lens – focal length
- Film size (h,w)
- Orientation of film plane

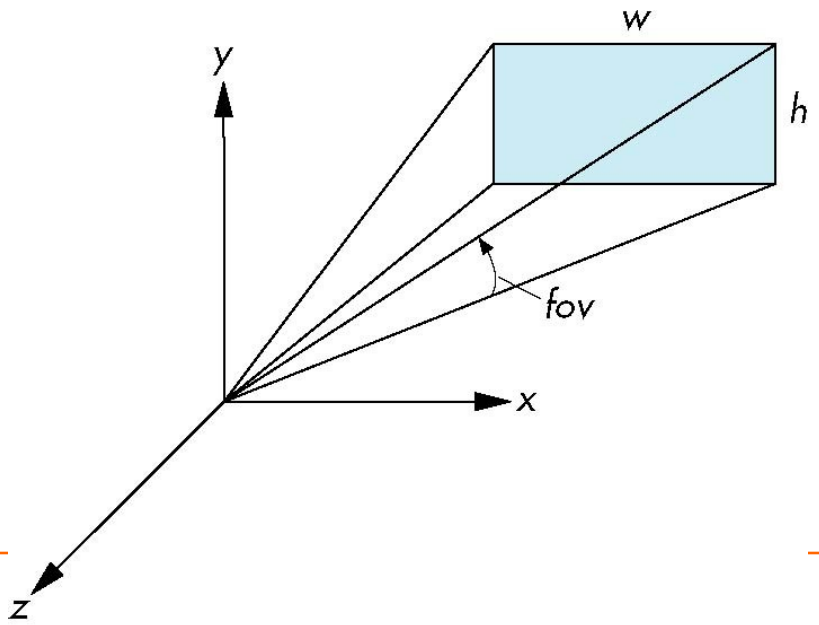


Perspective

Frustum: a truncated pyramid



Using Field of View

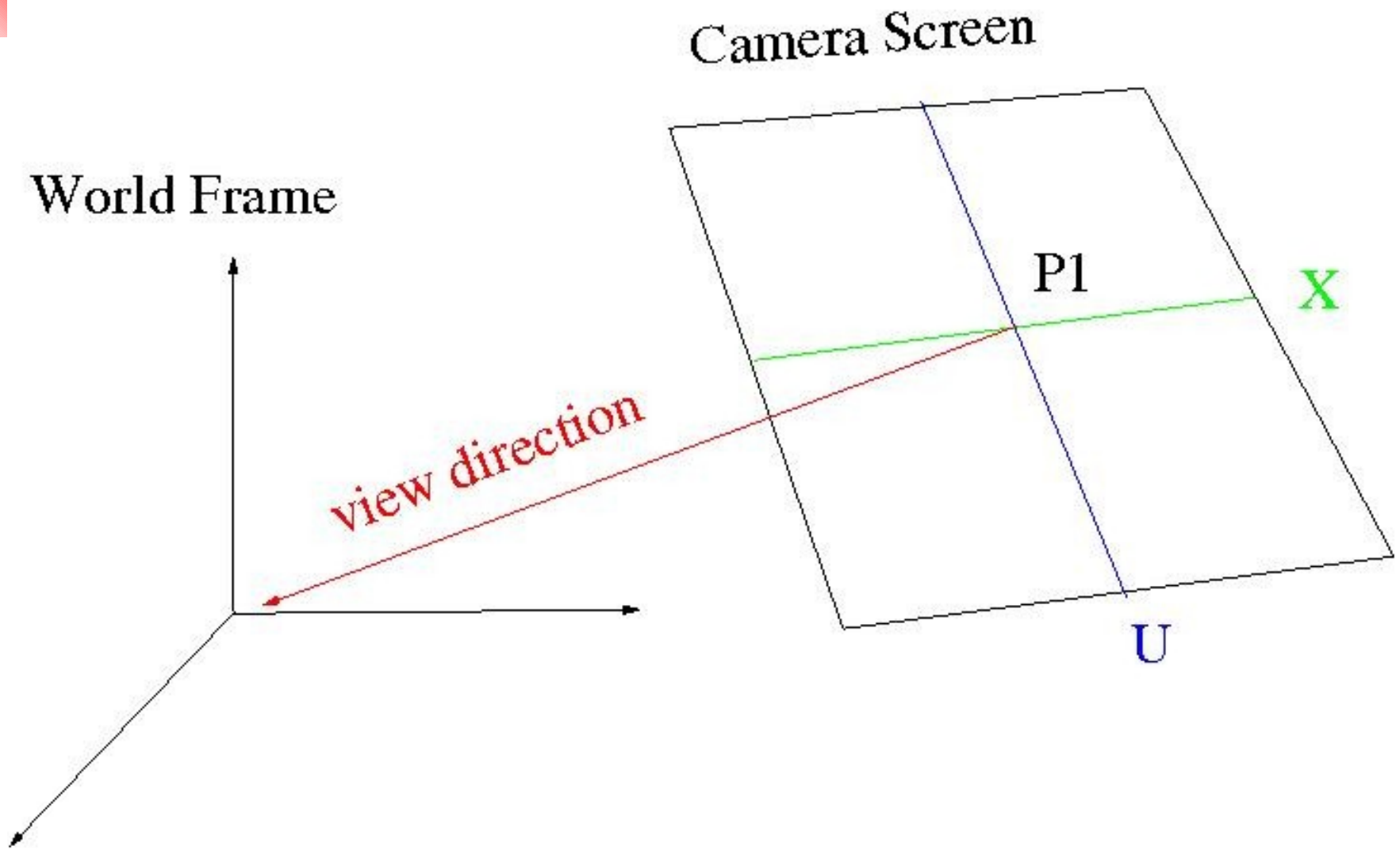


front plane

fovy – angle in up direction

aspect = w/h

Moving the Camera

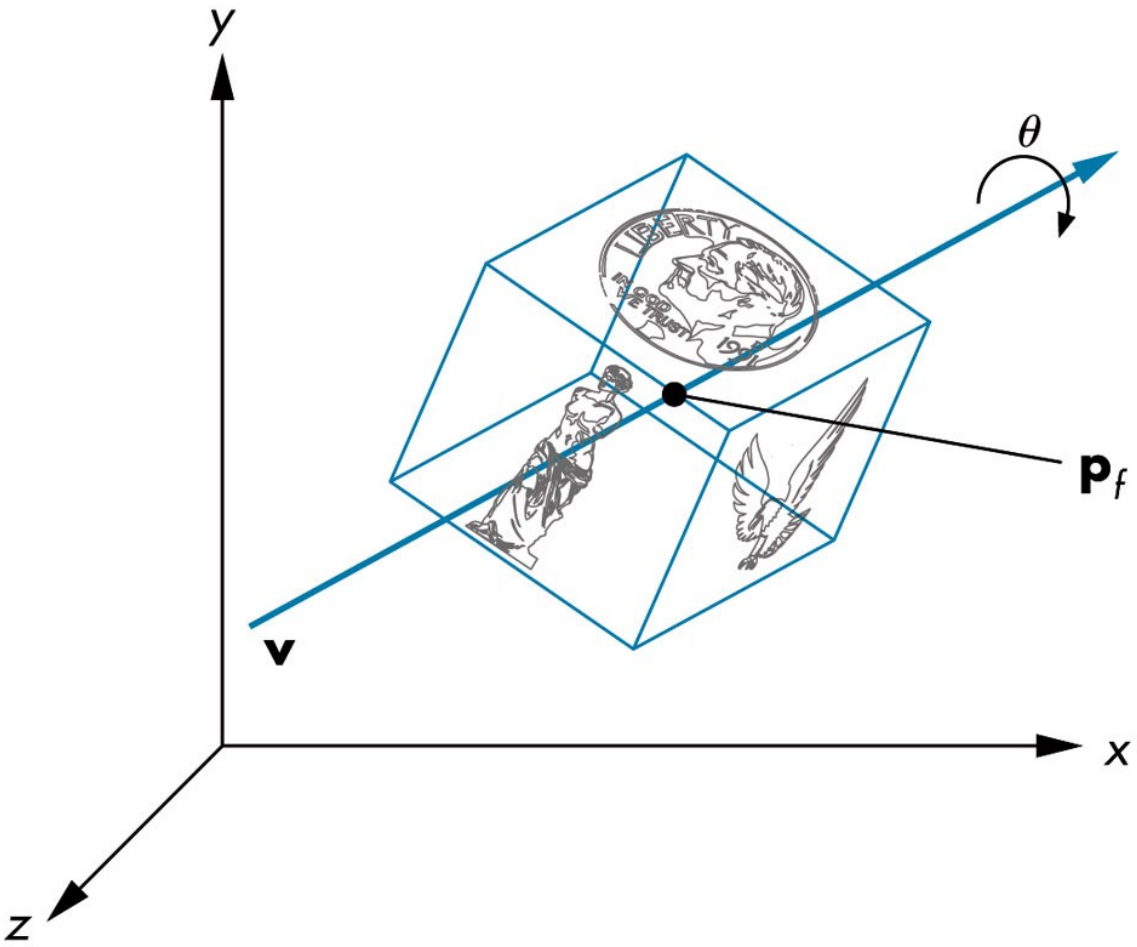


Transformations

Objectives

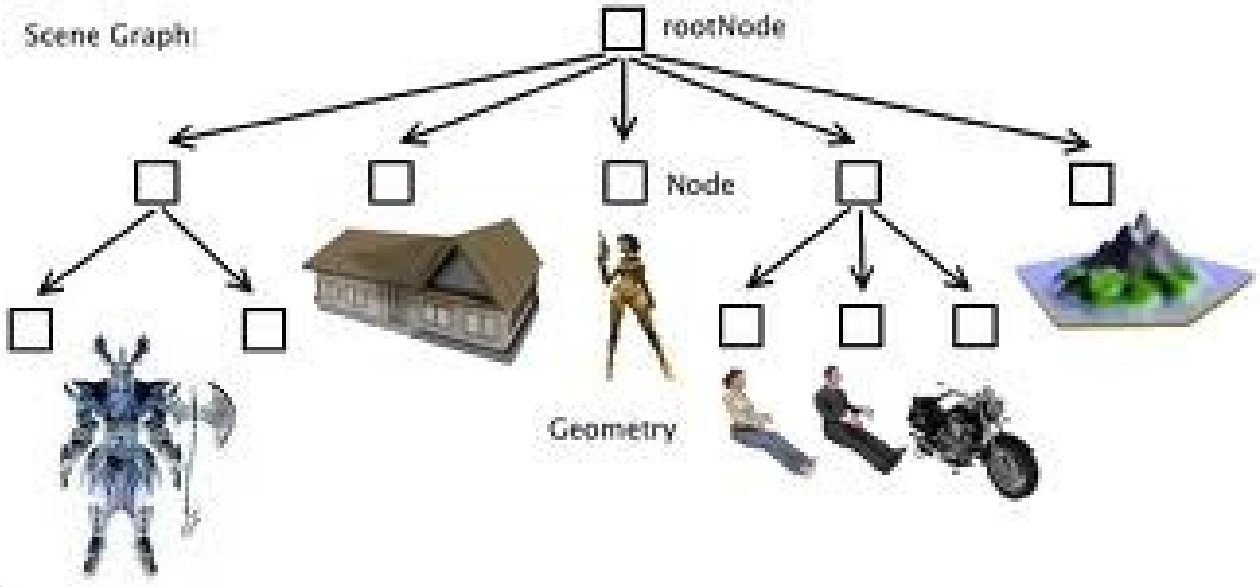
- Introduce standard transformations
  - Rotation
  - Translation
  - Scaling
  - Shear
- Derive homogeneous coordinate transformation matrices
- Learn to build arbitrary transformation matrices from simple transformations

3D Rotations.

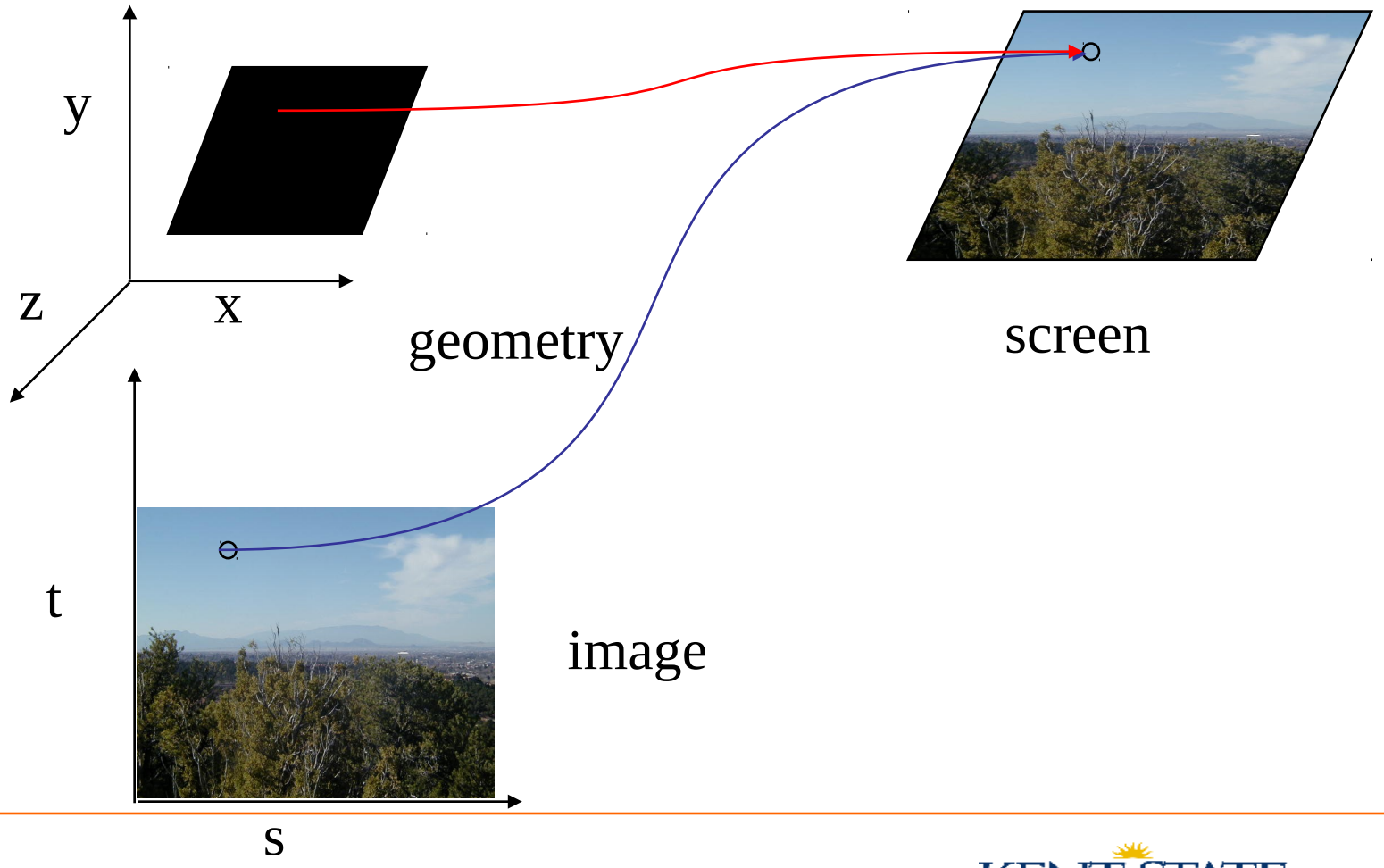


Scene Graph

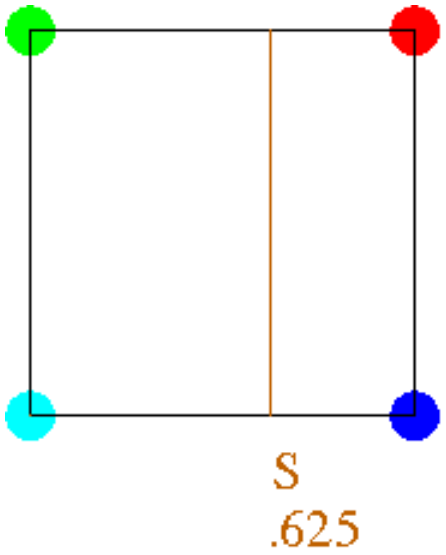
- In unity each object has a transformation and can include other objects and meshes. Together these produce the scene graph for the scene.

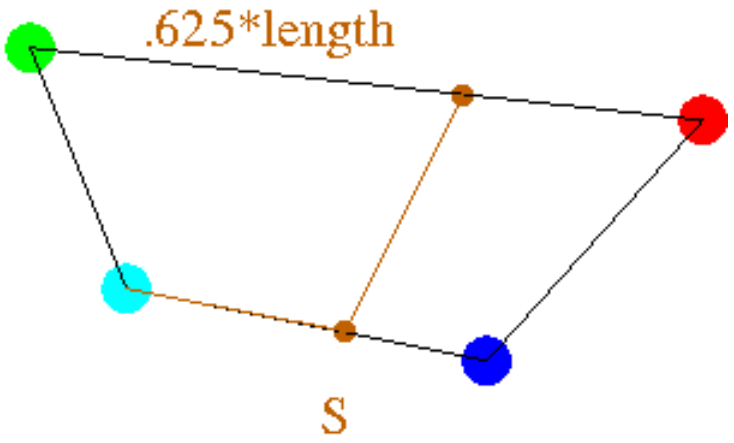


Texture Mapping



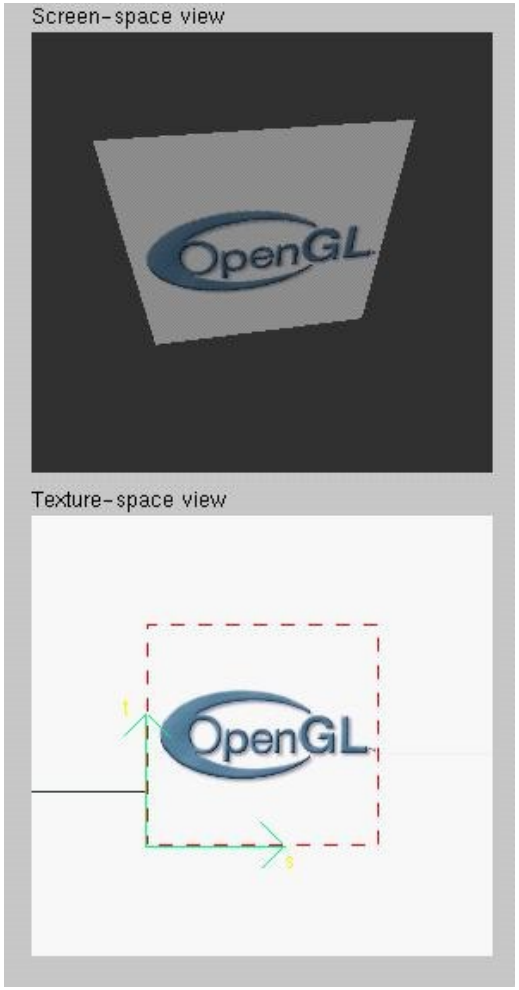
Texture Mapping





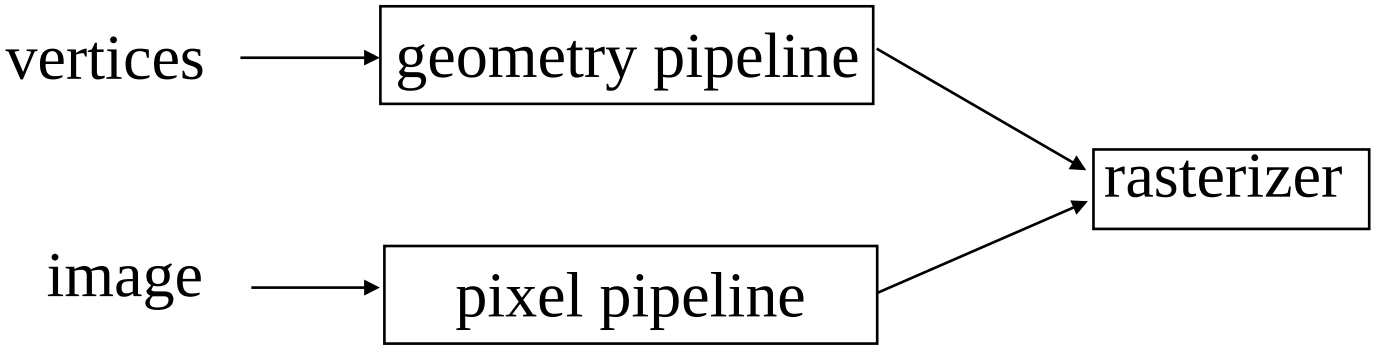
Texture Example

- The texture (below) is a 256 x 256 image that has been mapped to a rectangular polygon which is viewed in perspective



Texture Mapping and the OpenGL Pipeline

- Images and geometry flow through separate pipelines that join at the rasterizer
  - "complex" textures do not affect geometric complexity



Specify Texture Image

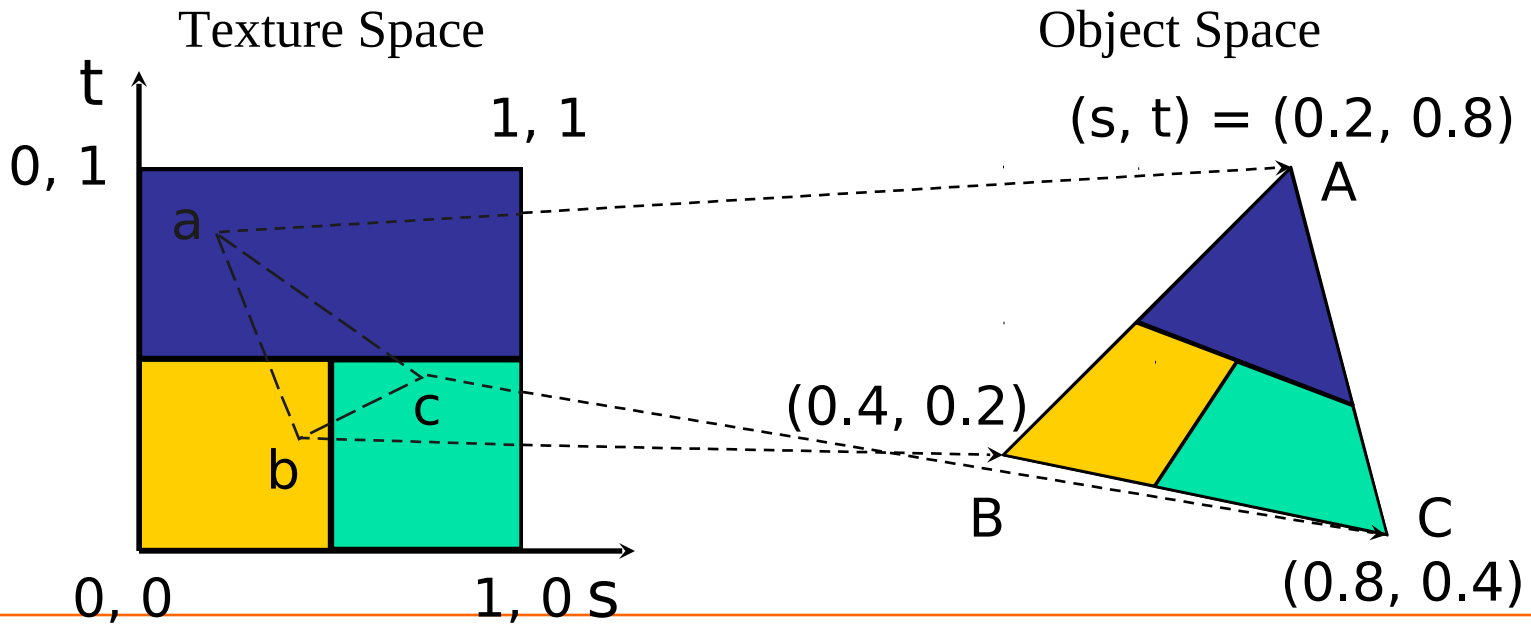
- Define a texture image from an array of *texels* (texture elements) in CPU memory

```
Glubyte my_texels[512][512];
```

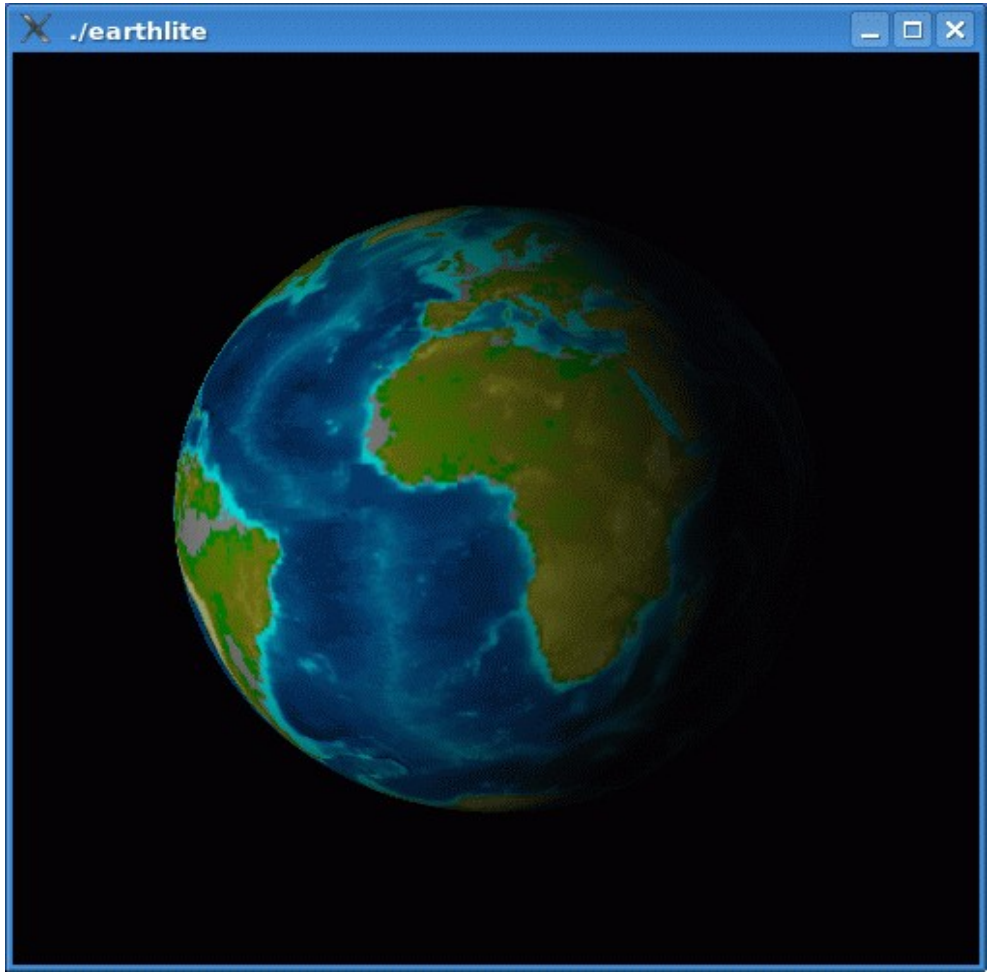
- Define as any other pixel map
  - Scan
  - Via application code
- Enable texture mapping
  - **`glEnable(GL_TEXTURE_2D)`**
  - OpenGL supports 1-4 dimensional texture maps

Mapping a Texture

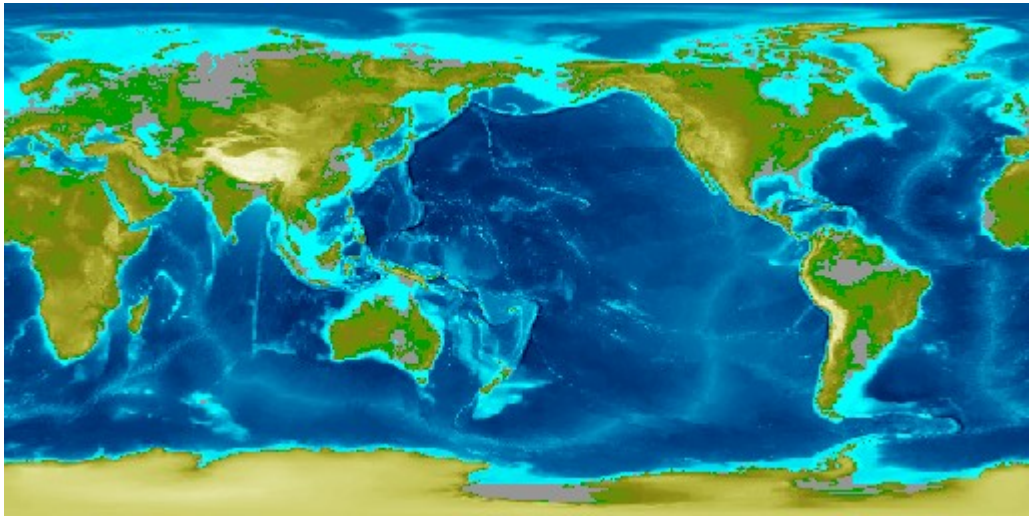
- Based on parametric texture coordinates



Example-Planet Earth



Planet Earth-Picture



Planet Earth-Wire Frame