

Announcements

Assignment 1 is due Thursday at midnight

Turnin space:

`/afs/andrew/scs/cs/15-462/turnin`

You should be able to create files there but not read or remove

Use a directory name of the format: assignment1_jkh

TA hours Tuesday 3-5 (Joel—cluster) and Thursday 3-5 (Michael—Wean 8121)

Or send any of us email

Questions on Assignment 1?

Written Assignment #1 out this afternoon

Wrap-up on Transformations

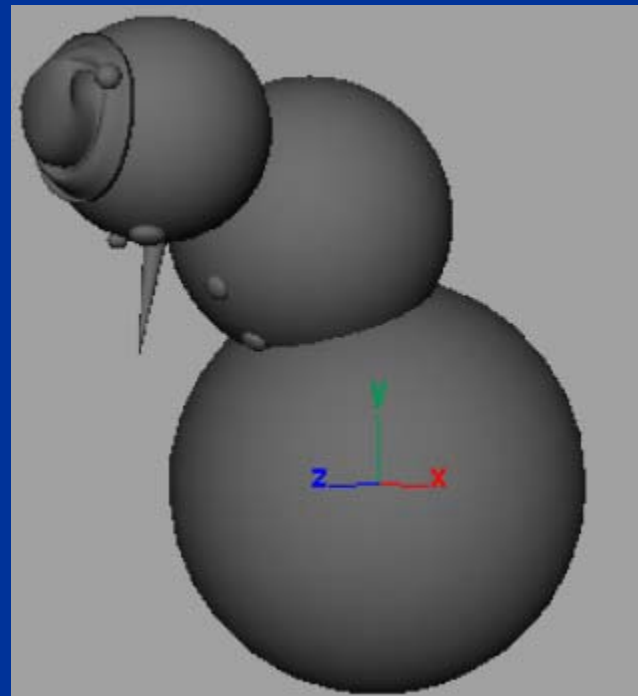
3D Viewing

Perspective projection
Viewing transformation

Shirley Chapter 7

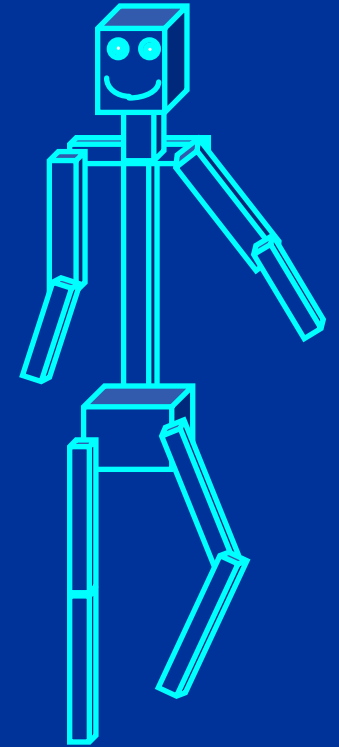
Example: Modeling

Modeling with primitive shapes



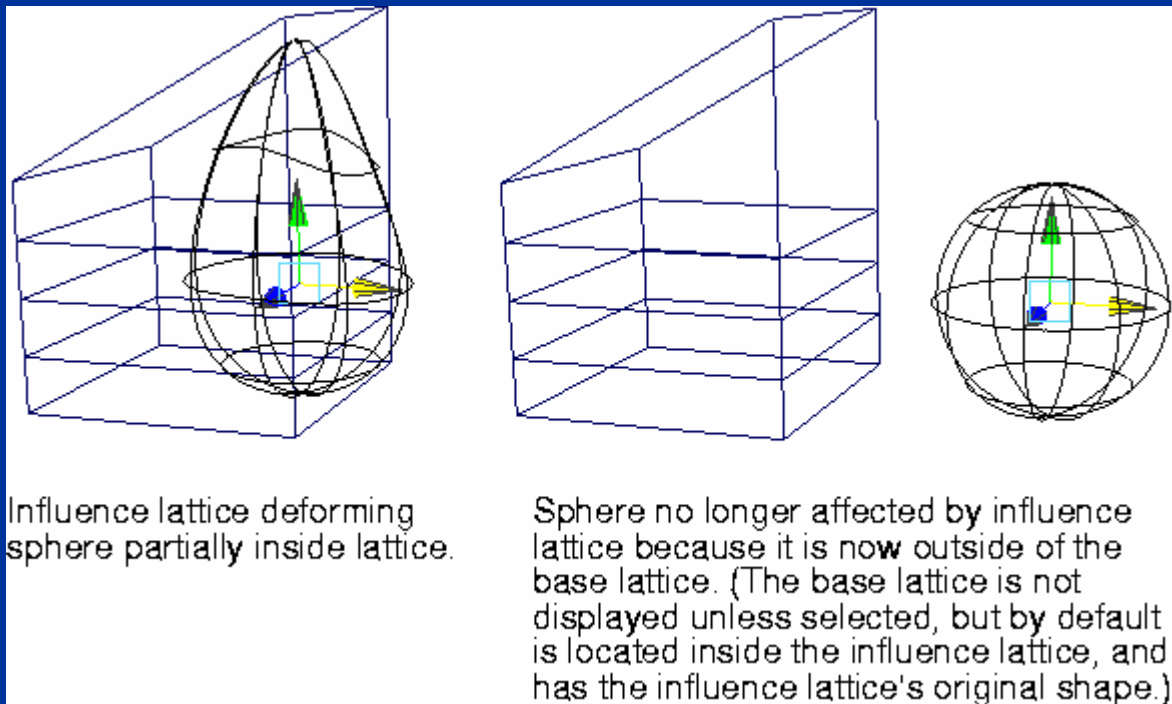
Example: Animation

Setting up a scene and animating

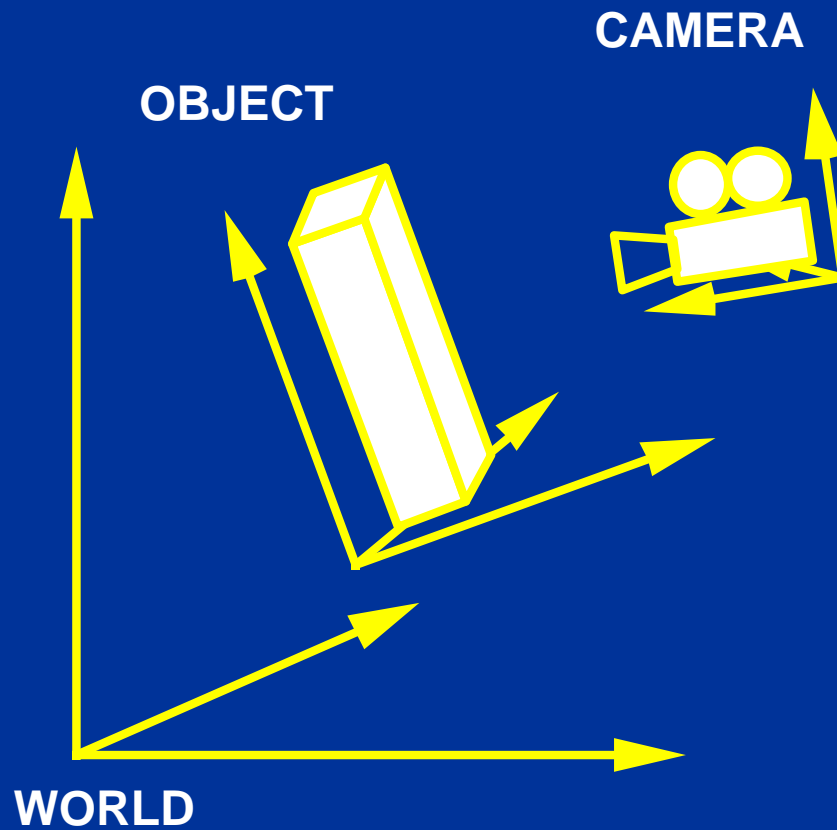


Example: Modeling Deformations

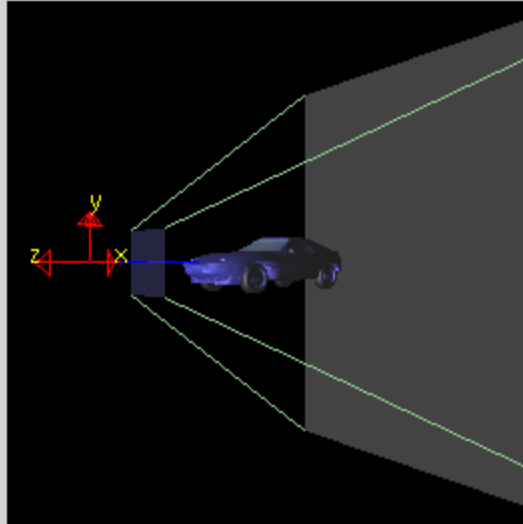
Incorporating deformations into a modeling system



Example: Viewing Transformation



World-space view



Screen-space view



Command manipulation window

```
glTranslatef( 0.00 , 0.00 , 0.00 );  
glRotatef( 0.0 , 0.00 , 1.00 , 0.00 );  
glScalef( 1.00 , 1.00 , 1.00 );  
glBegin( ... );  
...
```

Click on the arguments and move the mouse to modify values.

Question about rotating about an arbitrary axis



chalkboard

Implementing Transformation Sequences

- Calculate the matrices and cumulatively multiply them into a global *Current Transformation Matrix*
- Postmultiplication is more convenient in hierarchies -- multiplication is computed in the opposite order of function application
- The calculation of the transformation matrix, M ,
 - initialize M to the identity
 - in reverse order compute a basic transformation matrix, T
 - post-multiply T into the global matrix M , $M \leftarrow MT$
- Example - to rotate by θ around $[x,y]$:

```
glLoadIdentity()      /* initialize M to identity mat.*/  
glTranslatef(x, y, 0) /* LAST:  undo translation */  
glRotatef(theta,0,0,1) /* rotate about z axis */  
glTranslatef(-x, -y, 0) /* FIRST: move [x,y] to origin. */
```

- Remember the last T calculated is the first applied to the points
 - calculate the matrices in reverse order

Column Vector Convention

- The convention in the slides
 - transformation is by matrix times vector, Mv
 - textbook uses this convention, 90% of the world too

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- The composite function $A(B(C(D(x))))$ is the matrix-vector product $ABCDx$

Beware: Row Vector Convention

- The transpose is also possible

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} m_{11} & m_{21} & m_{31} \\ m_{12} & m_{22} & m_{32} \\ m_{13} & m_{23} & m_{33} \end{bmatrix}$$

- How does this change things?
 - all transformation matrices must be transposed
 - ABCDx transposed is $x^T D^T C^T B^T A^T$
 - pre- and post-multiply are reversed
- OpenGL uses transposed matrices!
 - You'll only notice this if you DIRECTLY pass matrices as arguments to OpenGL subroutines, e.g. `glLoadMatrix`.
 - Most routines take only scalars or vectors as arguments.

3D Viewing

Canonical View Volume
Orthographic Projection
Perspective Projection

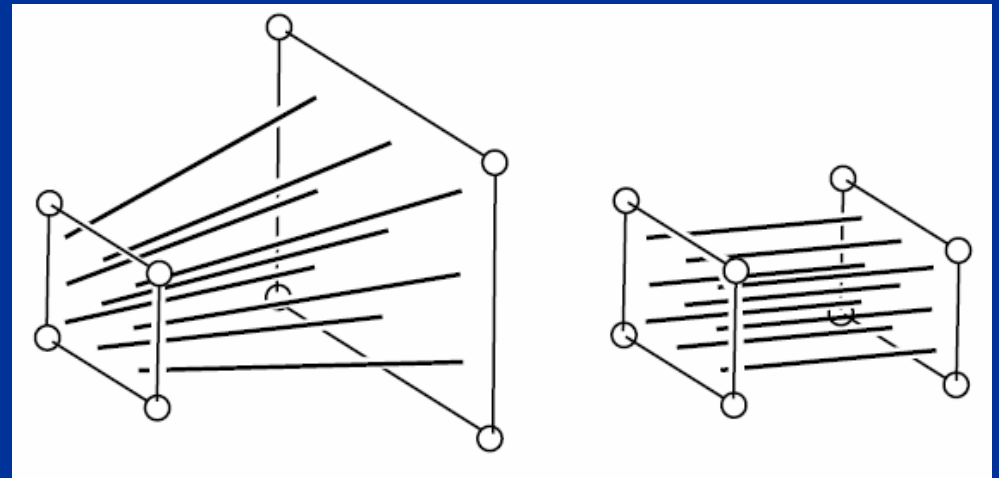
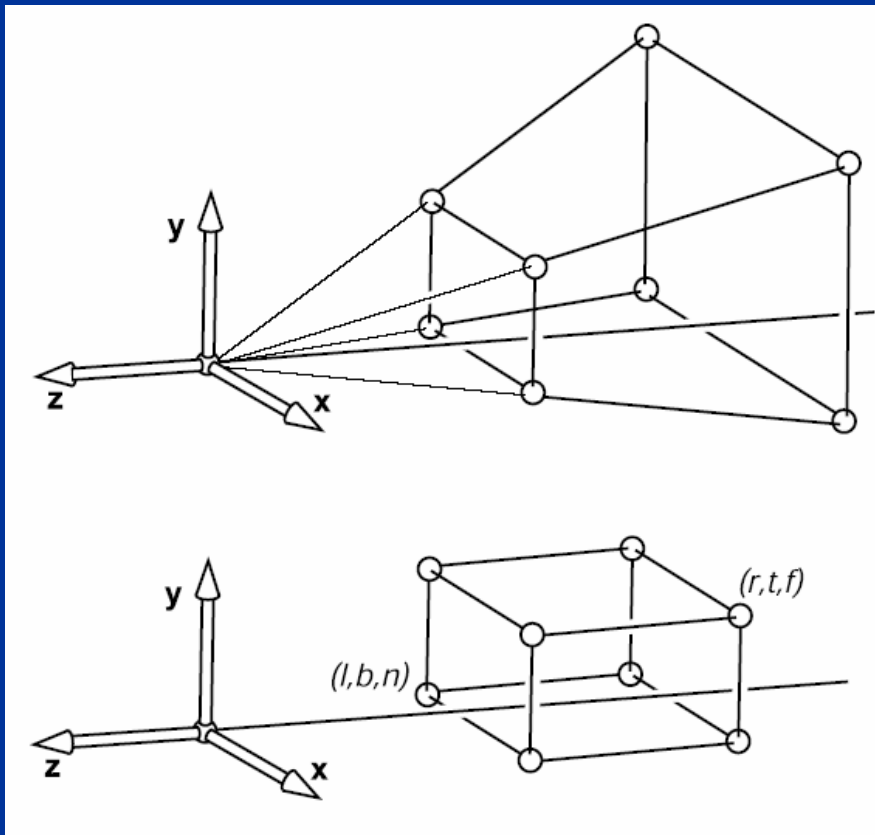
Shirley Chapter 7

Getting Geometry on the Screen

Given geometry positioned in the world coordinate system, how do we get it onto the display?

- Transform to camera coordinate system
- Transform (warp) into canonical view volume
- Clip
- Project to display coordinates
- Rasterize

Perspective and Orthographic Projection

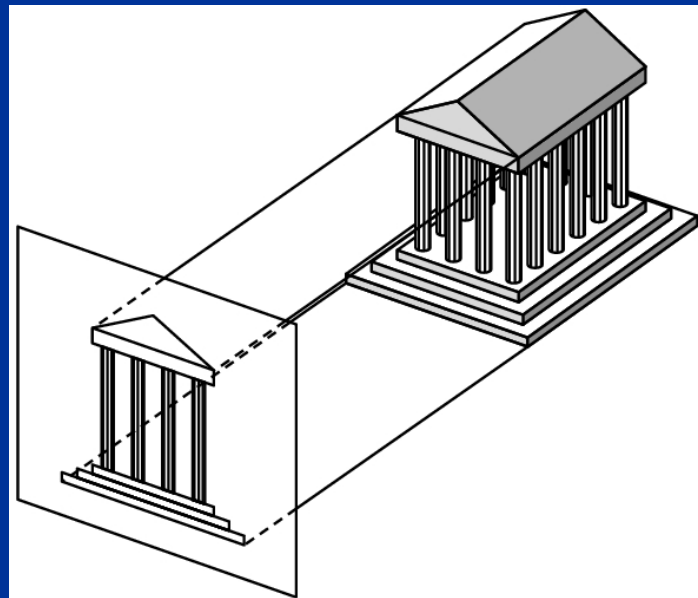


Orthographic Projection

the focal point is at infinity, the rays are parallel, and orthogonal to the image plane

good model for telephoto lens. No perspective effects.

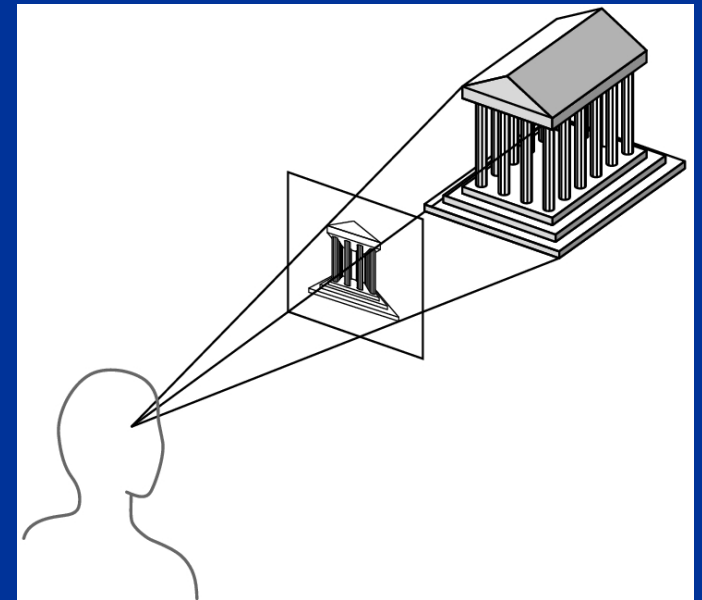
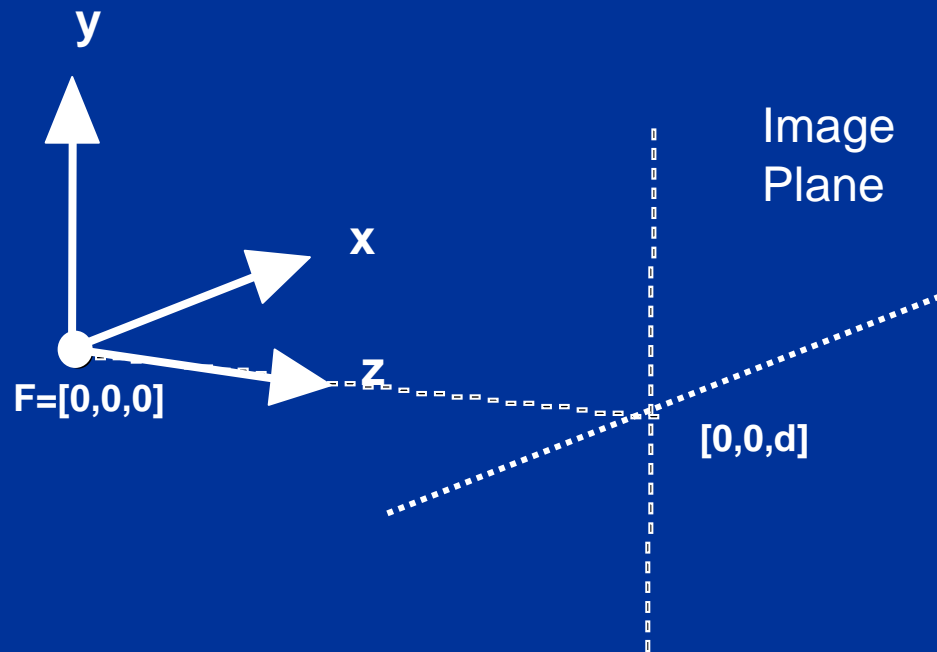
when xy -plane is the image plane $(x,y,z) \rightarrow (x,y,0)$
front orthographic view



Simple Perspective Camera

Canonical case:

- camera looks along the z-axis
- focal point is the origin
- image plane is parallel to the xy-plane at distance d
- (We call d the focal length, mainly for historical reasons)



Viewing and Projection

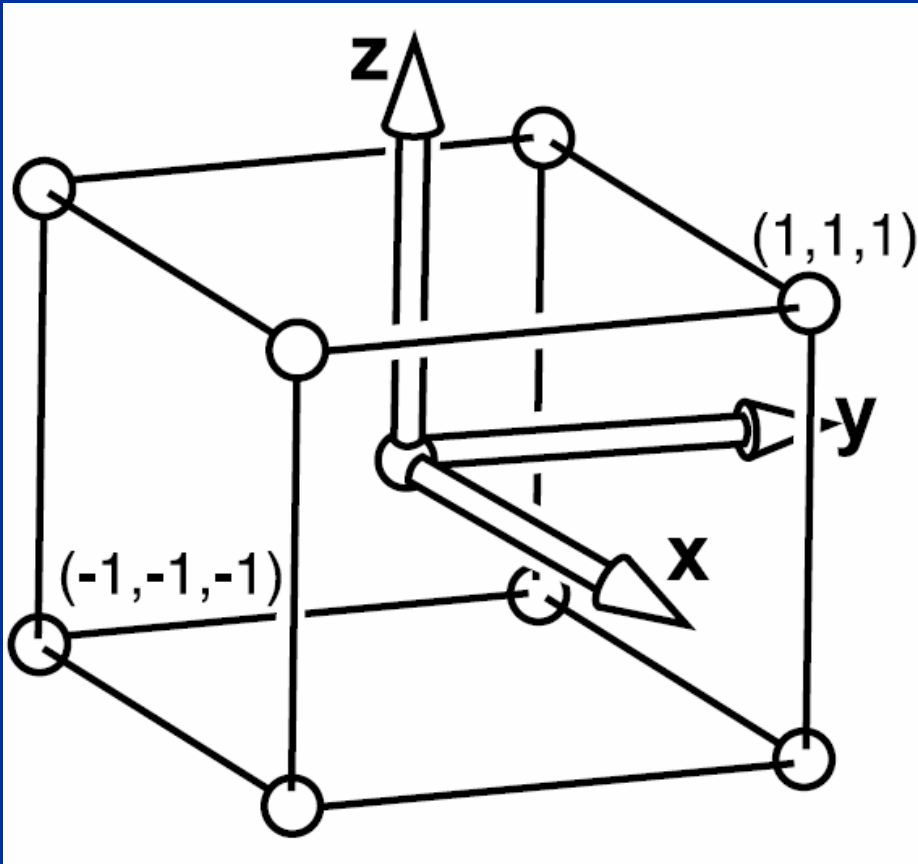
- Our eyes collapse 3-D world to 2-D retinal image (brain then has to reconstruct 3D)
- In CG, this process occurs by *projection*
- Projection has two parts:
 - *Viewing transformations*: camera position and direction
 - *Perspective/orthographic transformation*: reduces 3-D to 2-D
- Use homogeneous transformations (of course...)

Viewing and Projection

Build this up in stages

- Canonical view volume to screen
- Orthographic projection to canonical view volume
- Perspective projection to orthographic space

Canonical View Volume

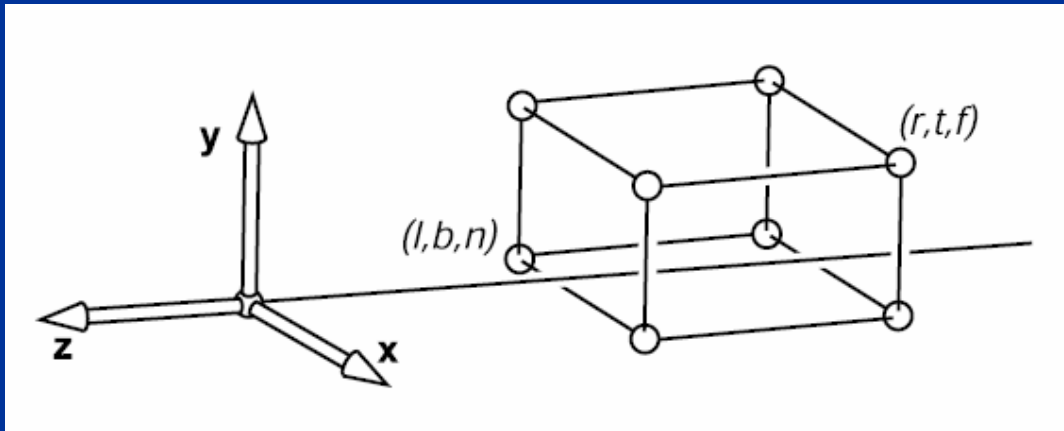


Why this shape?

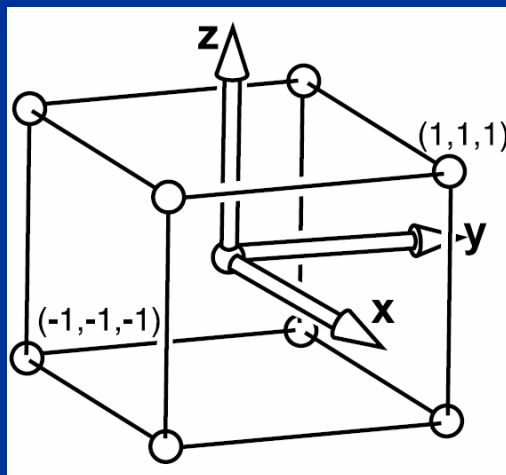
- Easy to clip to
- Trivial to project from 3D to 2D image plane

chalkboard

Orthographic Projection



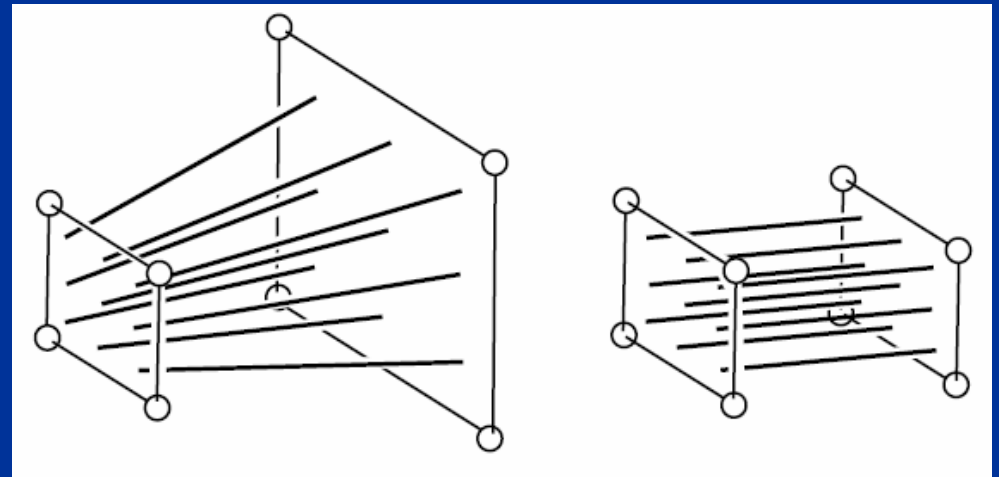
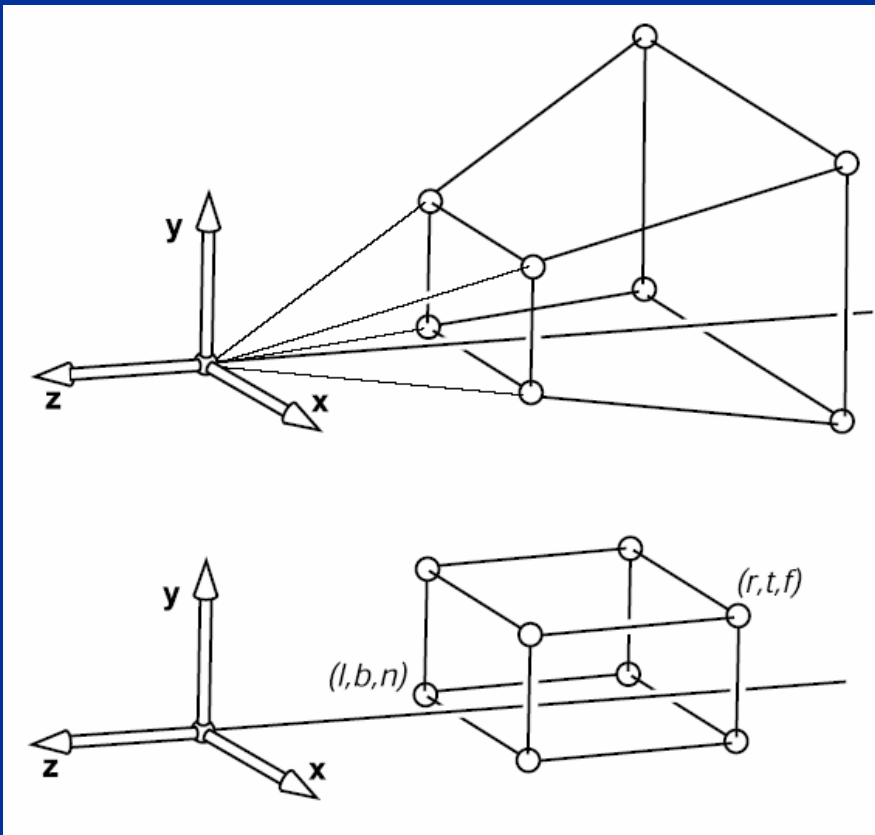
X=l left plane
X=r right plane
Y=b bottom plane
Y=t top plane
Z=n near plane
Z=f far plane



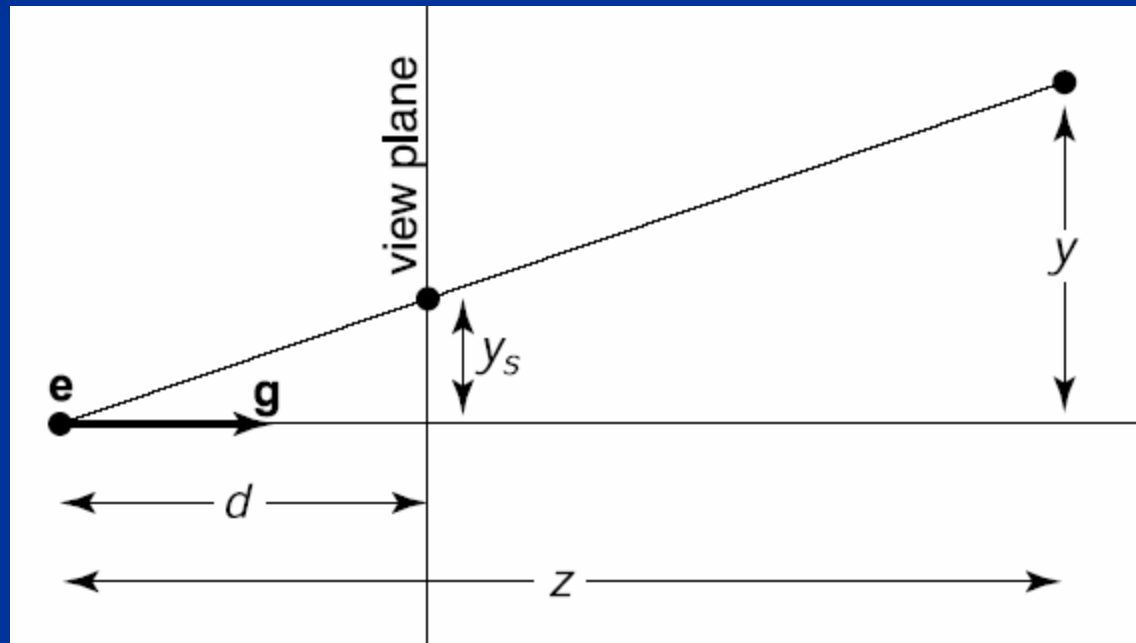
Why near plane? Prevent points behind the camera being seen
Why far plane? Allows z to be scaled to a limited fixed-point value (z-buffering)

chalkboard

Perspective Projection

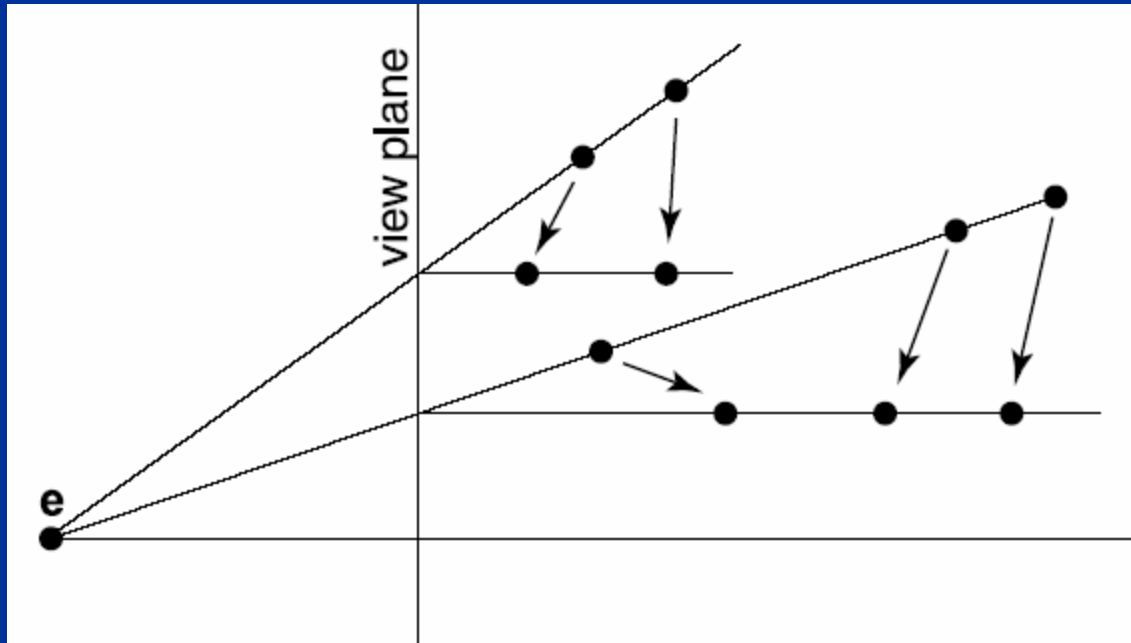


Perspective Projection of a Point



$$y_s = \frac{d}{z} y$$

Perspective Projection



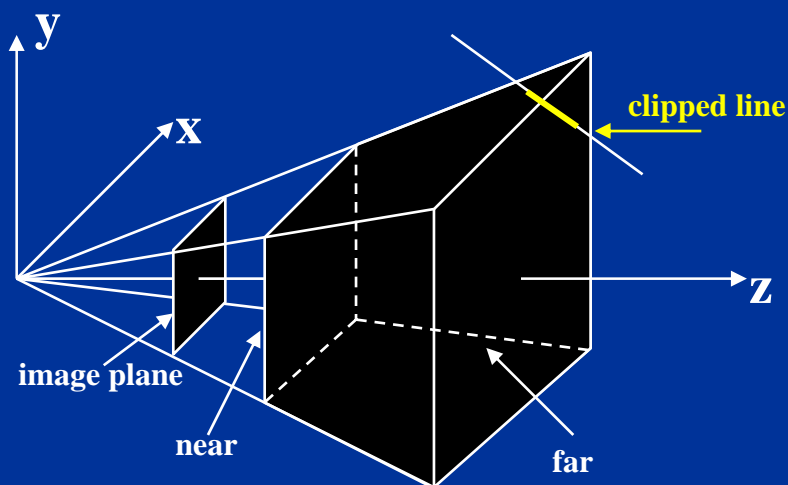
chalkboard

Warping a perspective projection into an orthographic one
Lines for the two projections intersect at the view plane
How can we put this in matrix form?

Need to divide by z —haven't seen a divide in our matrices so far...
Requires our w from last time (or h in the book)

Clipping

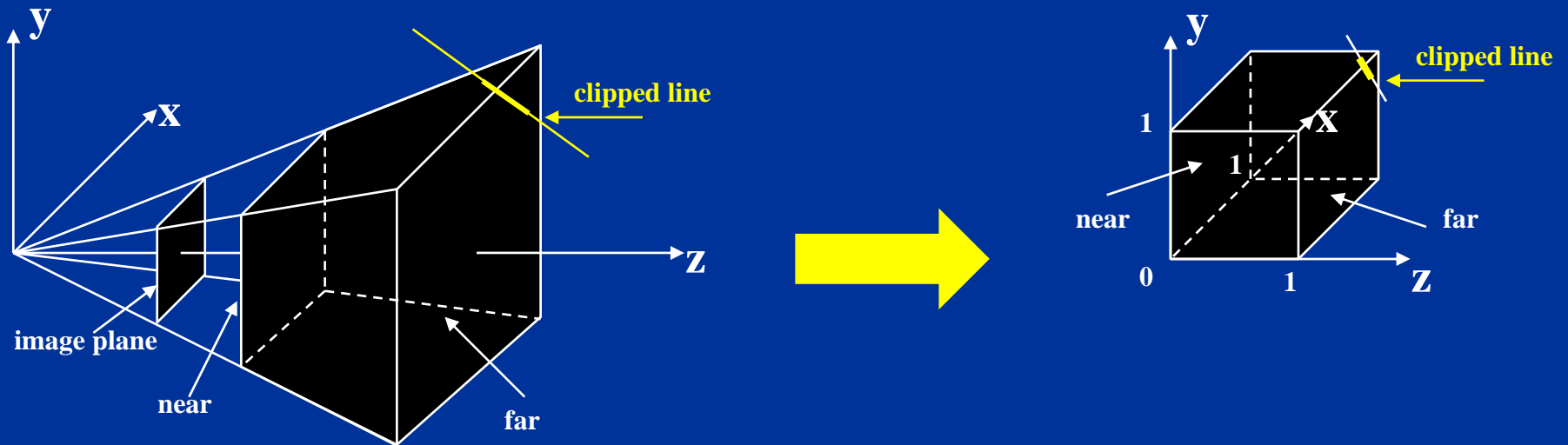
Something is missing between projection and viewing...
Before projecting, we need to eliminate the portion of scene that is outside the viewing frustum



Need to clip objects to the frustum (truncated pyramid)
Now in a canonical position but it still seems kind of tricky...

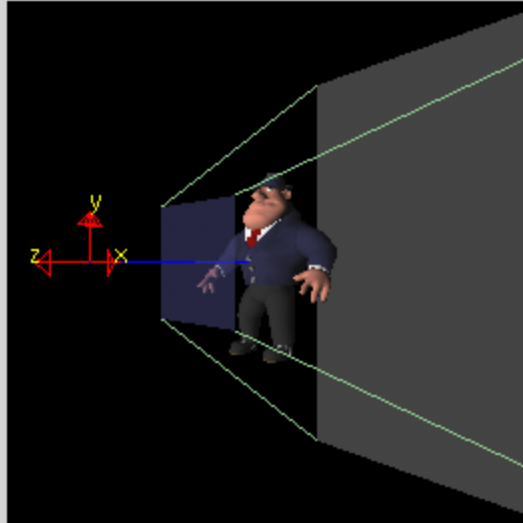
Normalizing the Viewing Frustum

Solution: transform frustum to a cube before clipping



Converts perspective frustum to orthographic frustum
Yet another homogeneous transform!

World-space view



Screen-space view



Command manipulation window

```
fovy aspect zNear zFar
gluPerspective( 60.0 , 1.00 , 1.0 , 10.0 );
gluLookAt( 0.00 , 0.00 , 2.00 , <- eye
           0.00 , 0.00 , 0.00 , <- center
           0.00 , 1.00 , 0.00 ); <- up
```

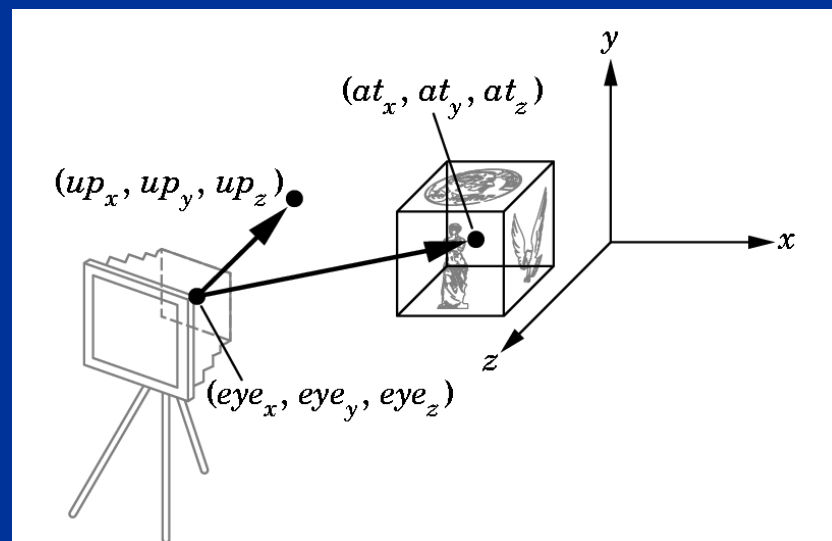
Click on the arguments and move the mouse to modify values.

Camera Control Values

- All we need is a single translation and angle-axis rotation (orientation), but...
- Good animation requires good camera control--we need better control knobs
- Translation knob - move to the *lookfrom* point
- Orientation can be specified in several ways:
 - specify camera rotations
 - specify a *lookat* point (solve for camera rotations)

A Popular View Specification Approach

- Focal length, image size/shape and clipping planes are in the perspective transformation
- In addition:
 - *lookfrom*: where the focal point (camera) is
 - *lookat*: the world point to be centered in the image
- Also specify camera orientation about the *lookat-lookfrom* axis



Implementation

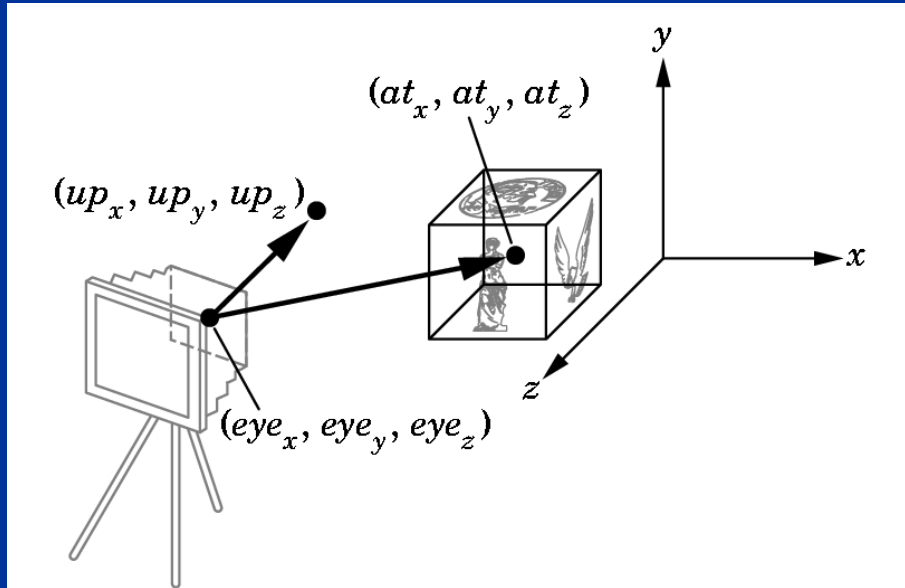
Implementing the *lookat/lookfrom/vup* viewing scheme

- (1) Translate by *-lookfrom*, bring focal point to origin
- (2) Rotate *lookat-lookfrom* to the z-axis with matrix R:
 - » $v = (\textit{lookat-lookfrom})$ (normalized) and $z = [0,0,1]$
 - » rotation axis: $a = (v \times z) / |v \times z|$
 - » rotation angle: $\cos\theta = v \cdot z$ and $\sin\theta = |v \times z|$

`glRotate(θ , a_x , a_y , a_z)`

- (3) Rotate about z-axis to get *vup* parallel to the y-axis

The Whole Picture



LOOKFROM:

Where the camera is

LOOKAT:

A point that should be centered in the image

VUP:

A vector that will be pointing straight up in the image

FOV:

Field-of-view angle.

d:

focal length

WORLD COORDINATES

Announcements

Assignment 1 is due Thursday at midnight

Turnin space:

`/afs/andrew/scs/cs/15-462/turnin`

You should be able to create files there but not read or remove

Use a directory name of the format: `assignment1_jkh`

TA hours Tuesday 3-5 (Joel—cluster) and Thursday 3-5 (Michael—Wean 8121)

Or send any of us email

Questions on Assignment 1?

Written Assignment #1 out this afternoon