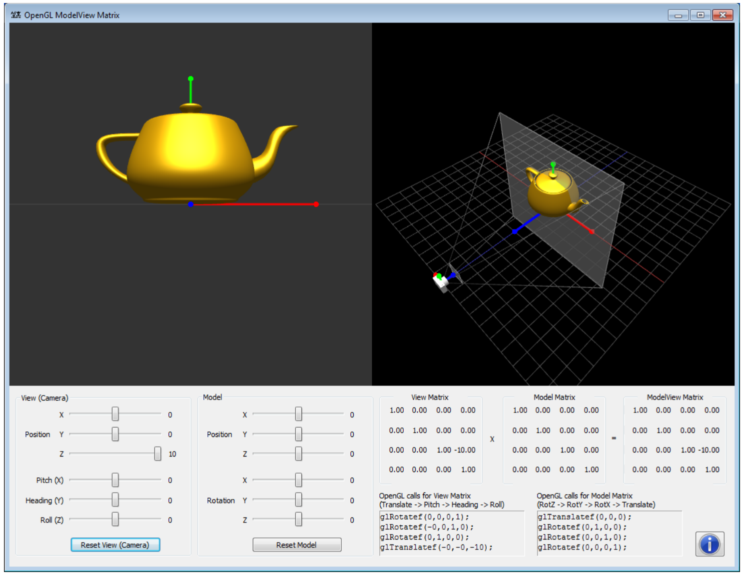
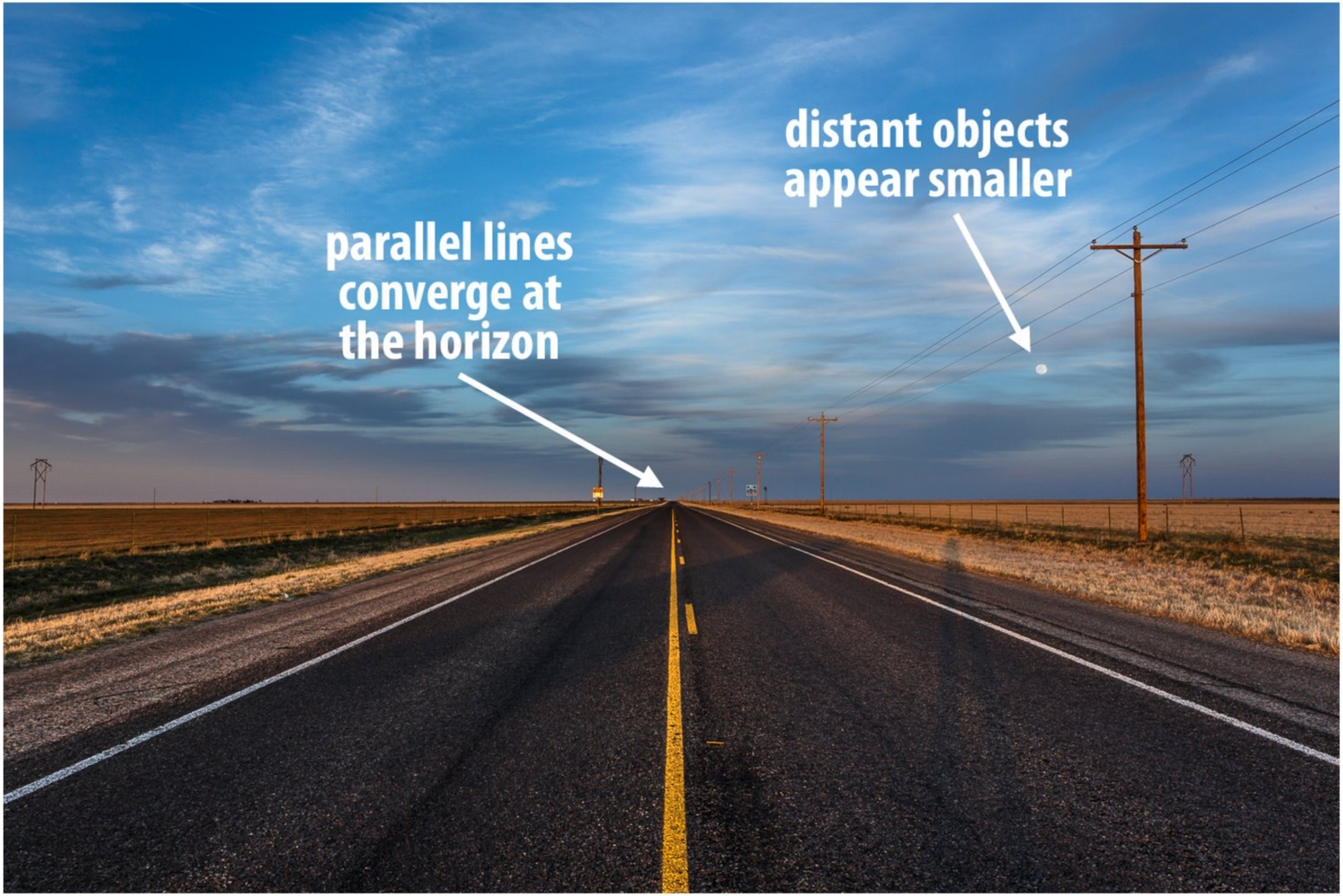
The Camera - CSE160

- History of Projection
- View Transform
- Projection Transform
- Clipping and Screen Transform
- Graphics vs Real Cameras
- Administrative
- Q&A



History of projection

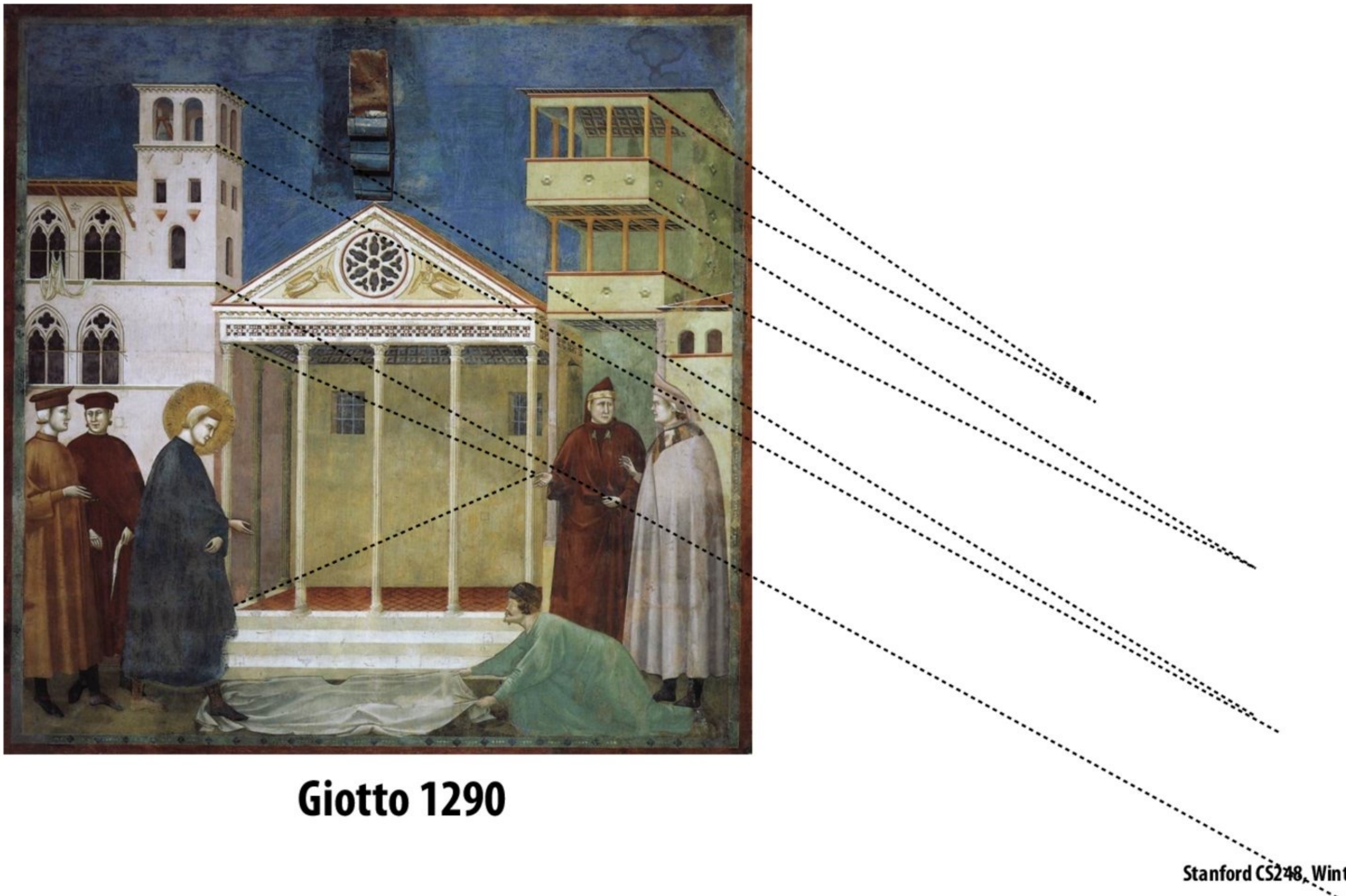
Perspective projection



Early painting: incorrect perspective

Carolingian painting from the 8-9th century

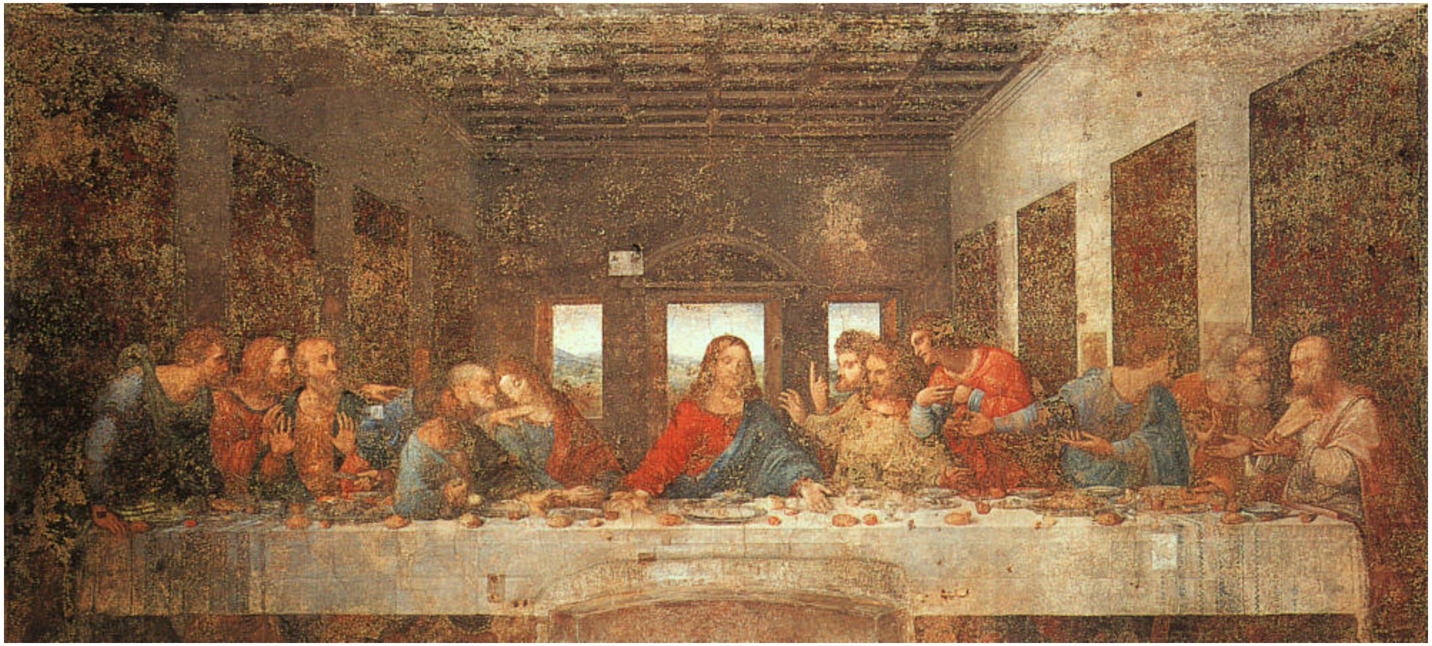
Perspective in art



Giotto 1290

History of projection

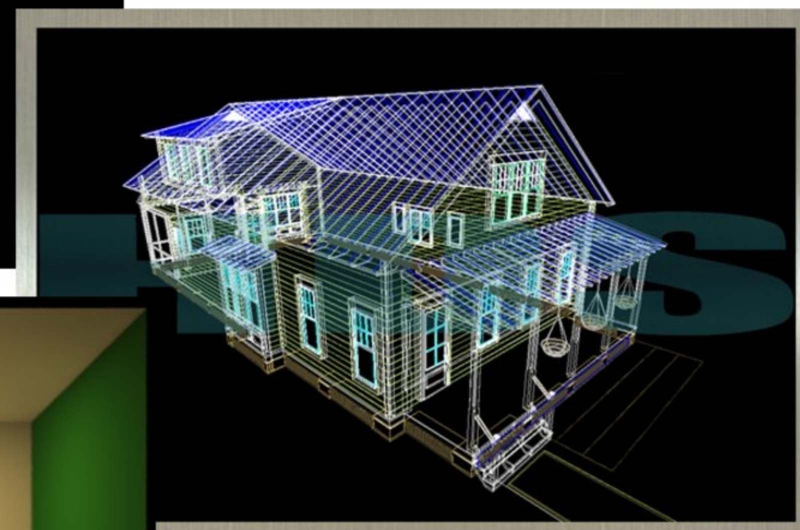
- Later Renaissance: perspective formalized precisely

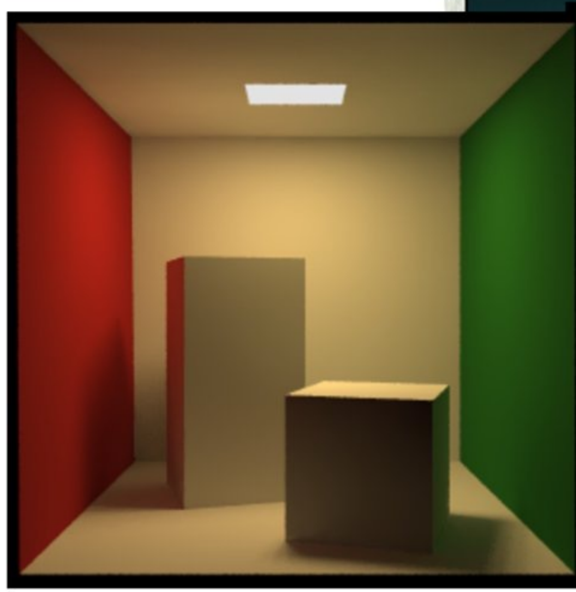


da Vinci c. 1498

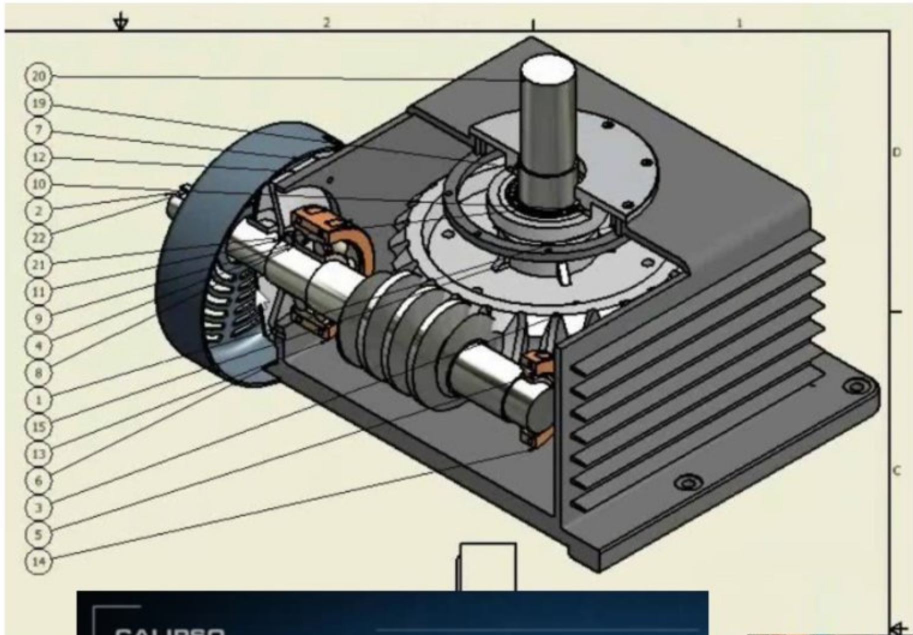
Later… rejection of proper perspective projection

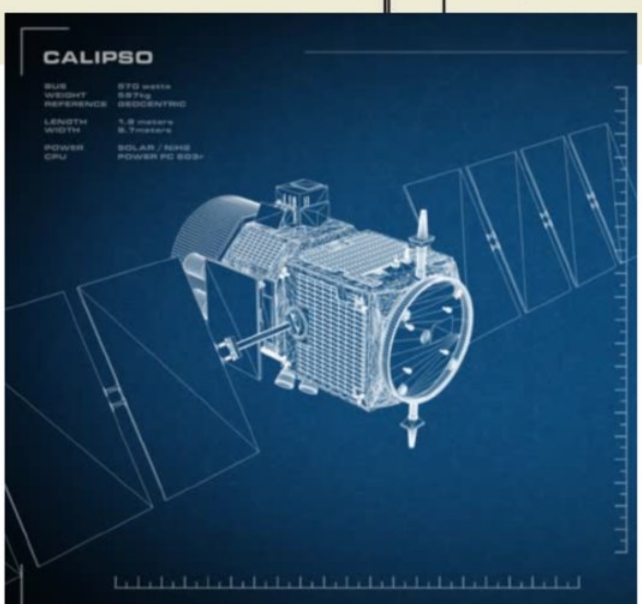
Correct perspective in computer graphics





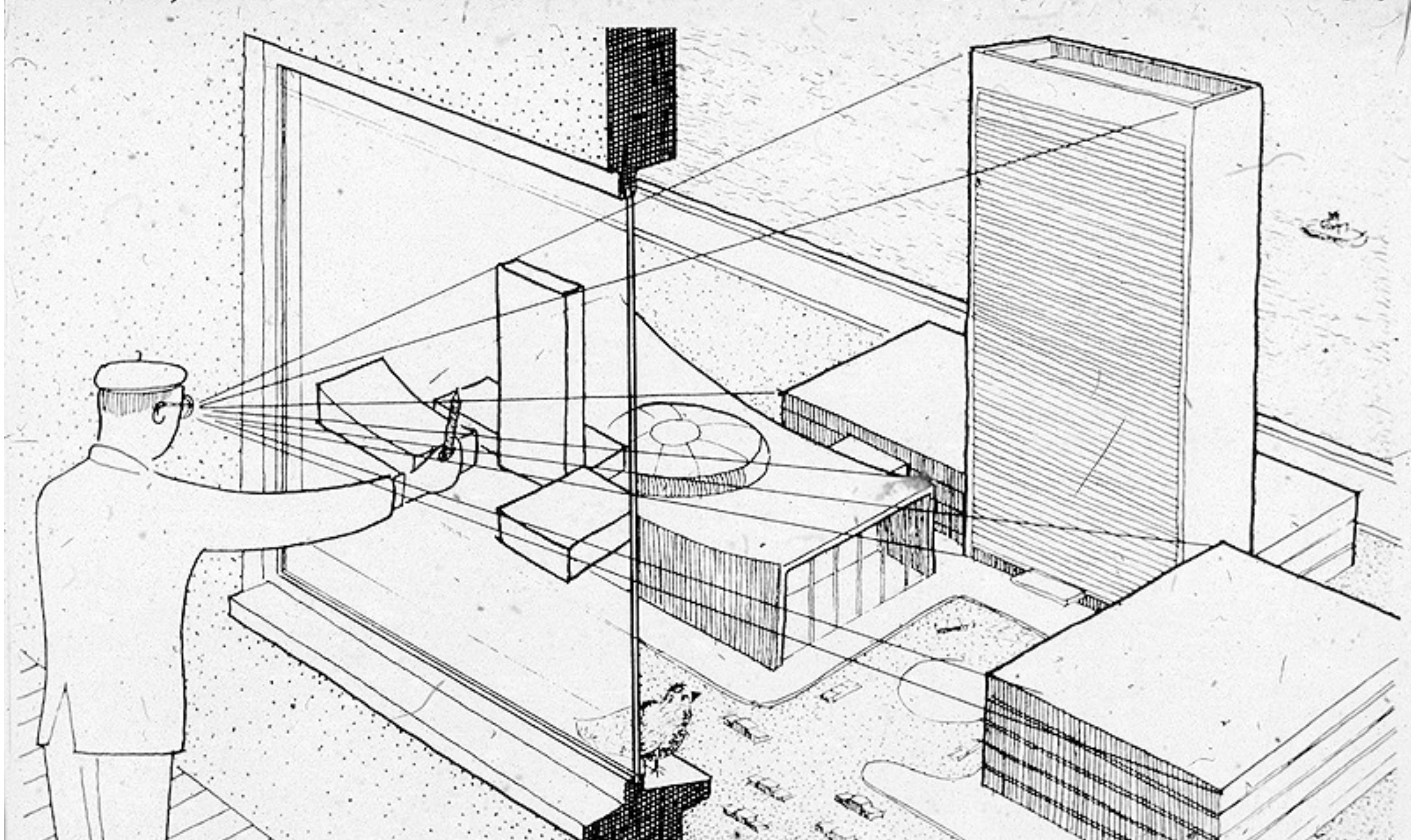
Rejection of perspective in computer graphics





Computer graphics works like this

Basis Of Perspective – Lines Of Sight Through A Picture Plane [19]

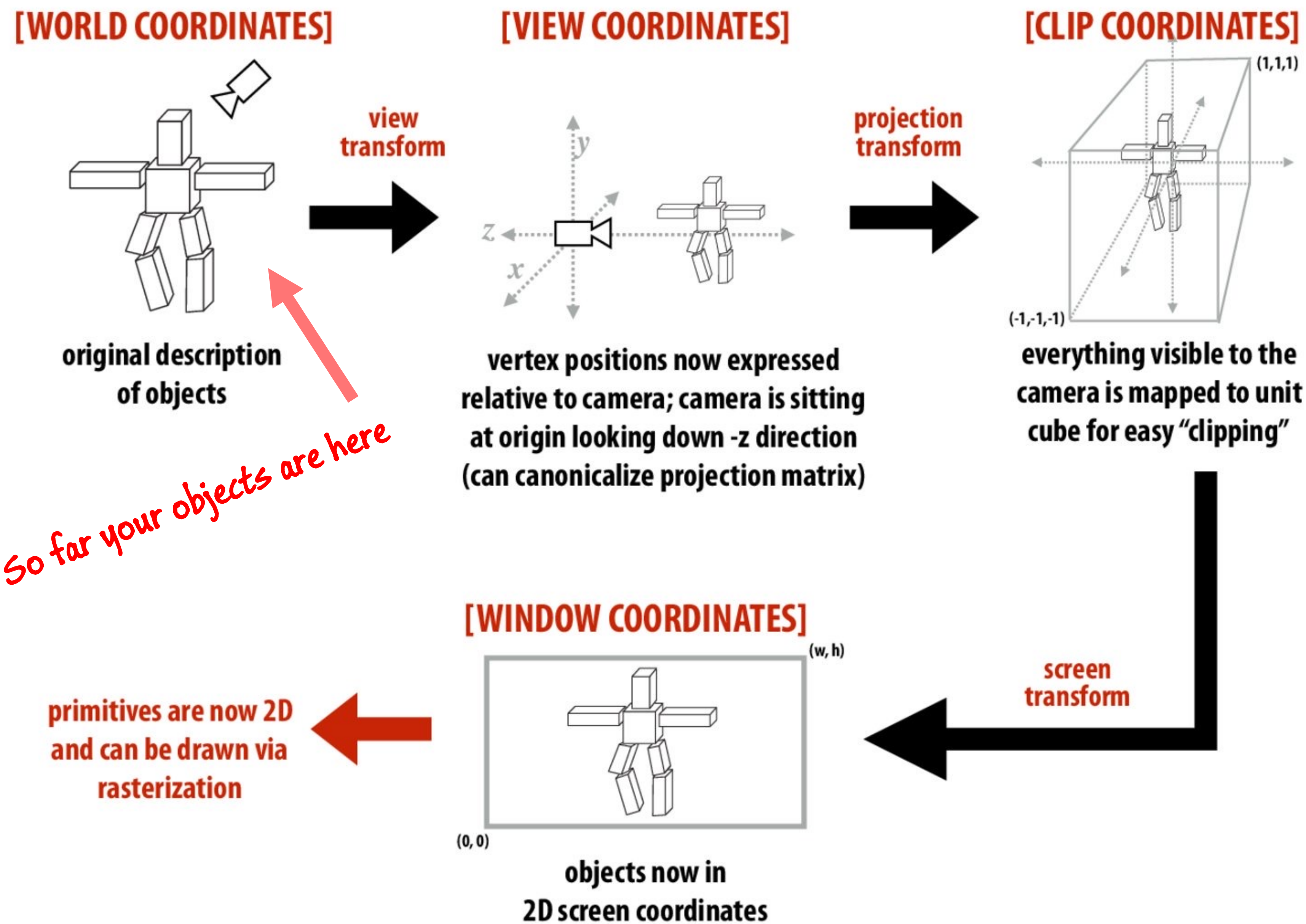


The concept of the picture plane may be better understood by looking through a window or other transparent plane from a fixed viewpoint. Your lines of sight, the multitude of straight lines leading from your eye to the subject, will all intersect this plane. Therefore, if you were to reach out with a grease pencil and draw the image of the subject on this plane you would be "tracing out" the infinite number of points of intersection of sight rays and plane. The result would be that you would have "transferred" a real three-dimensional object to a two-dimensional plane.

[CS 417 Spring 2002]

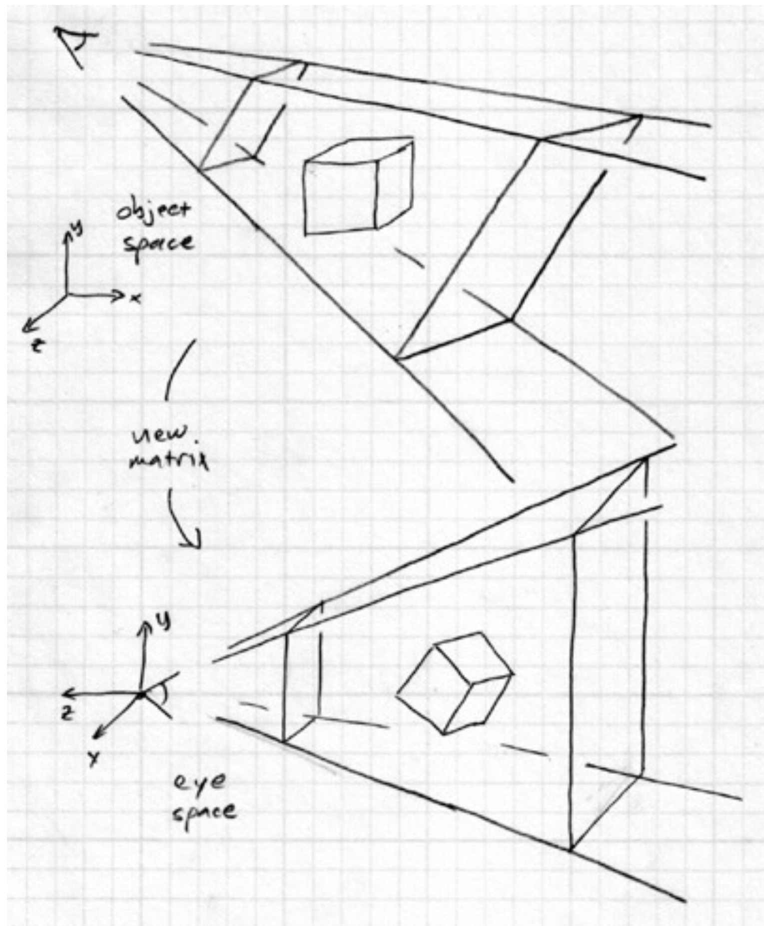
View Transform

Transformations: from objects to the screen



Jovan Popovic at MIT

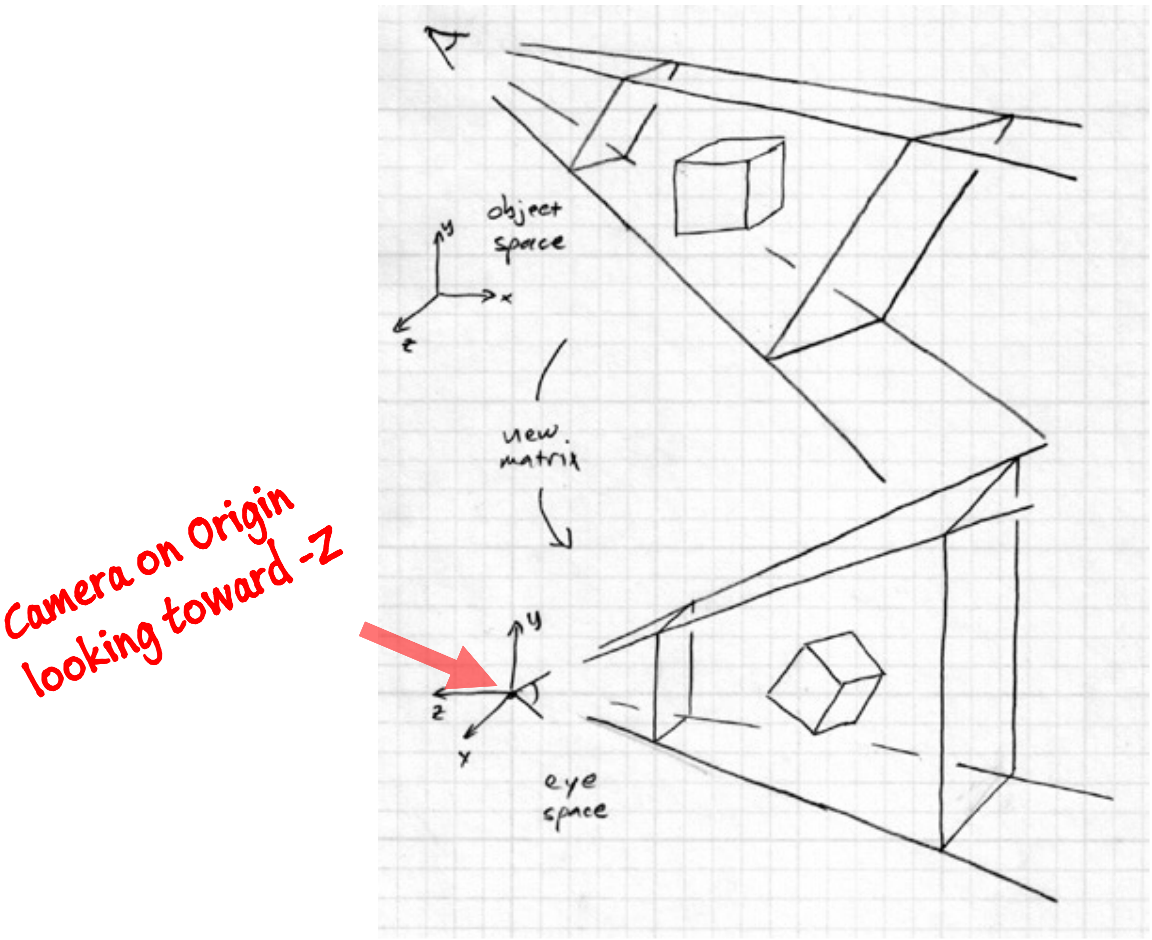
Viewing transformation



the view matrix rewrites all world coordinates in view coordinates (eye space)

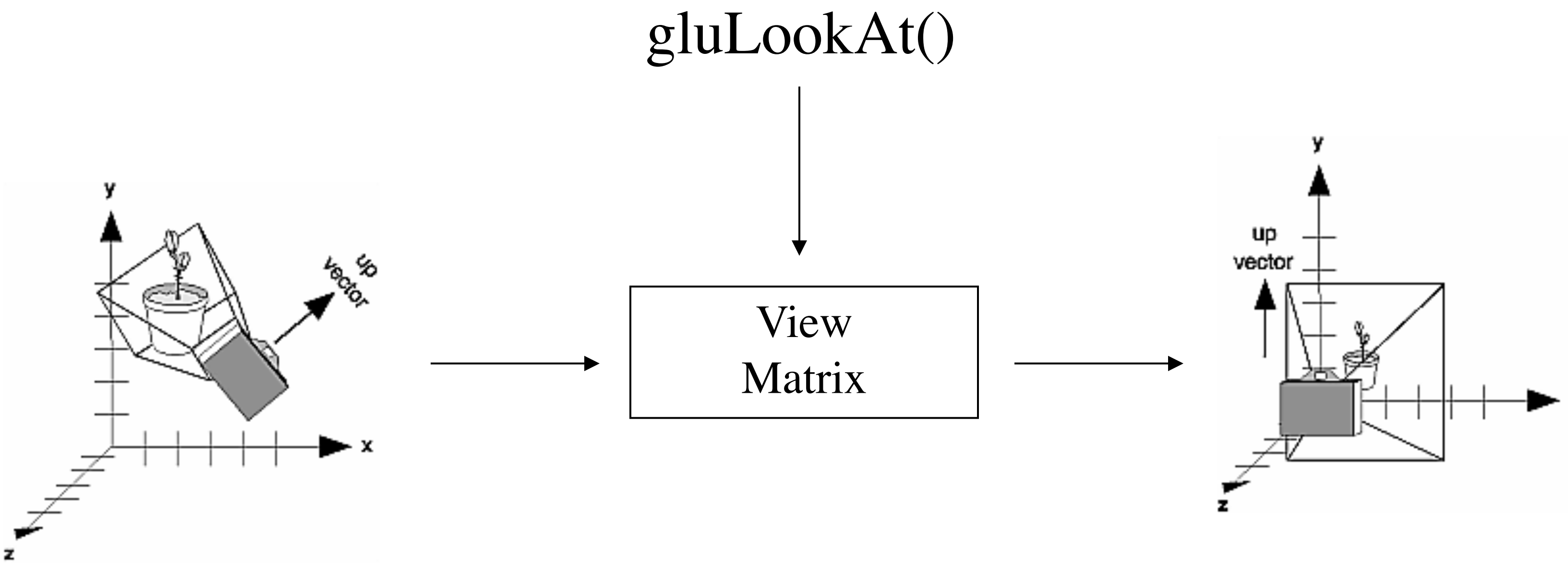
[Cornell]

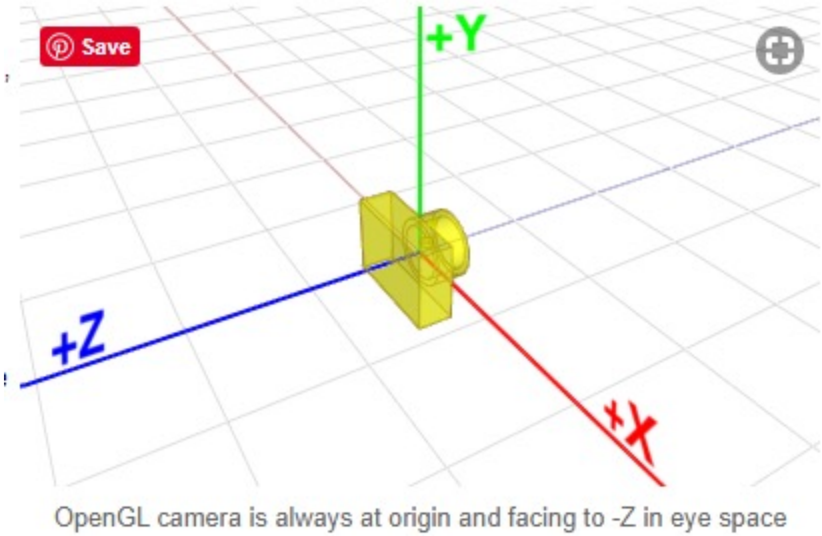
Viewing transformation

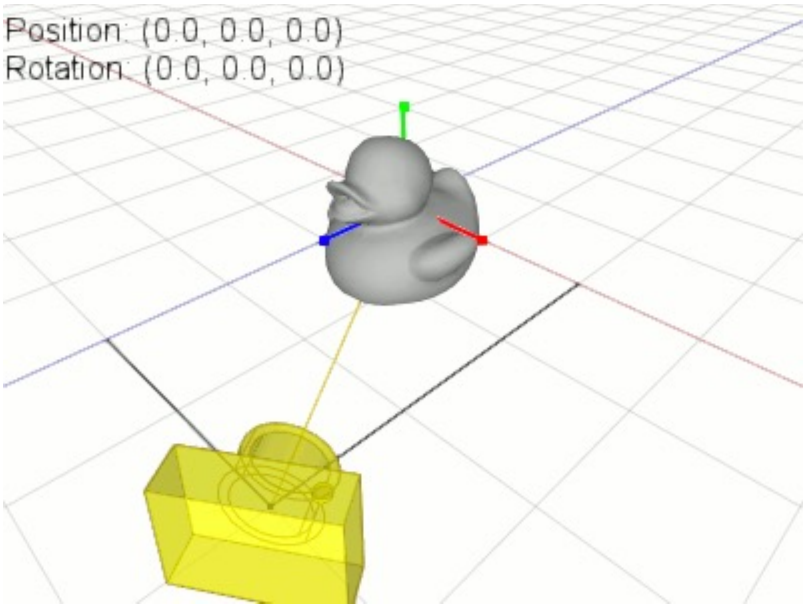


the view matrix rewrites all world coordinates in view coordinates (eye space)

[Cornell]


gluLookAt()
View
Matrix
y
x
z
up
vector
y
up
vector
x
z



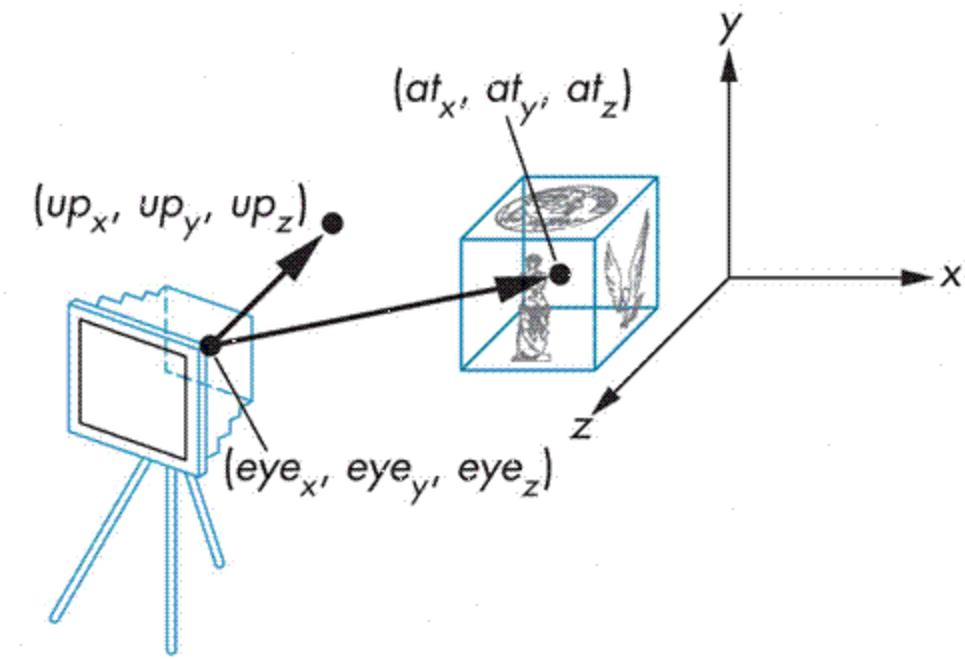


(this is animated GIF)

http://www.songho.ca/opengl/gl_camera.html

void gluLookAt(
GLdouble *eyeX* , GLdouble *eyeY* , GLdouble *eyeZ* ,
GLdouble *centerX* , GLdouble *centerY* , GLdouble *centerZ* ,
GLdouble *upX* , GLdouble *upY* , GLdouble *upZ*
);

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(
    0.0, 0.0, 5.0,
    0.0, 0.0, 0.0,
    0.0, 1.0, 0.0);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(50.0, 1.0, 3.0, 7.0);
```



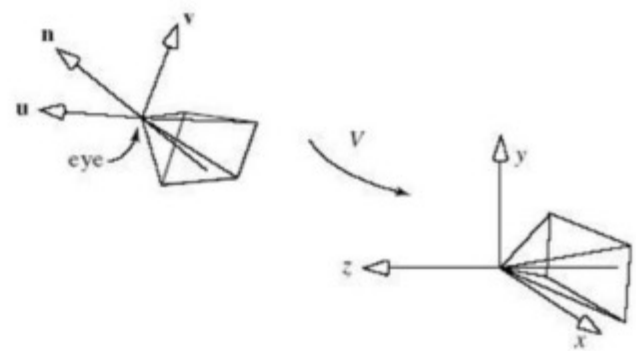
What does gluLookAt() do?

- gluLookAt(eyex, eyey, eyez, atx, aty, atz, upx, upy, upz) is equivalent to

  glMultMatrixf(M); // post-multiply M with current model-view matrix

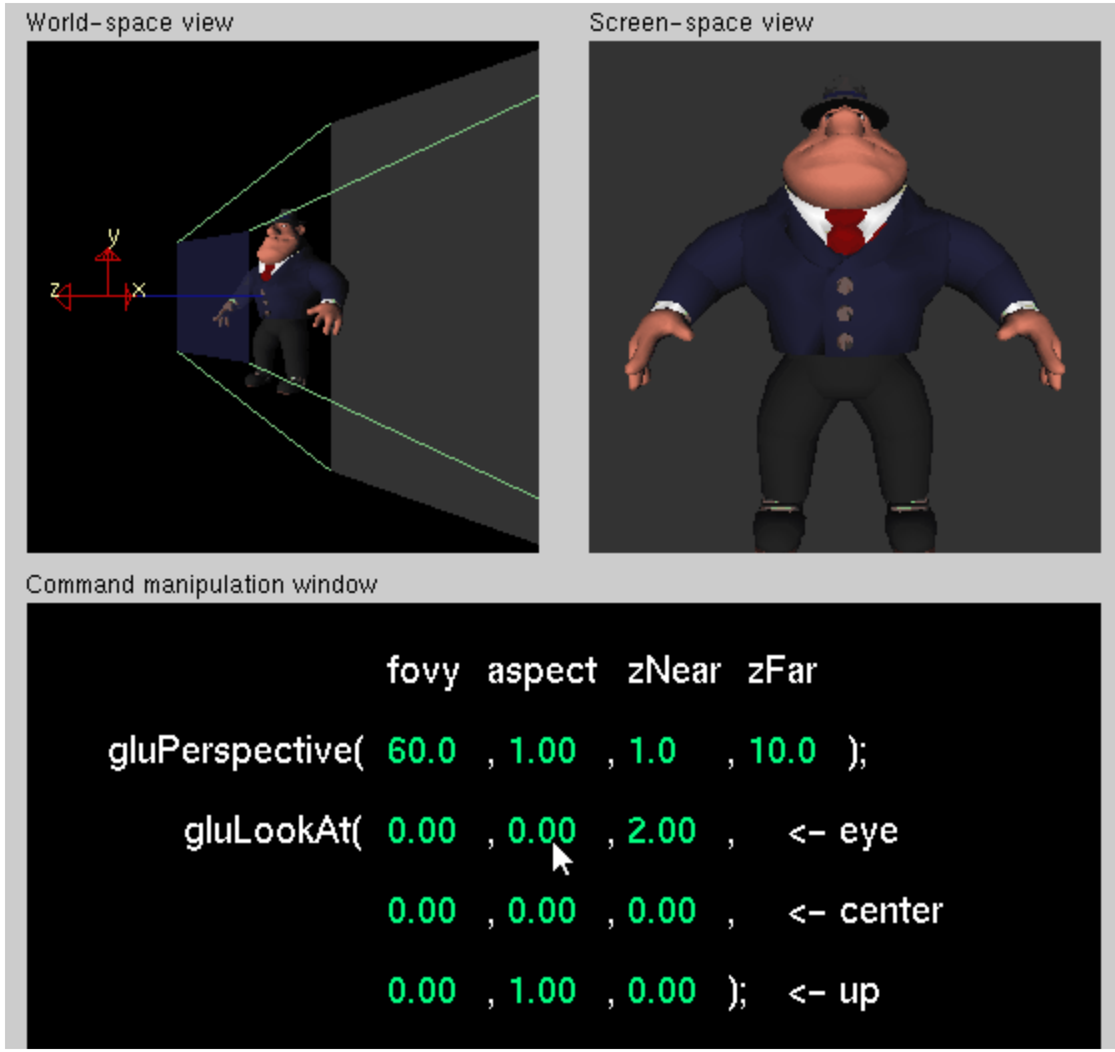
  glTranslated(-eyex, -eyey, -eyez);

Where M = $\begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

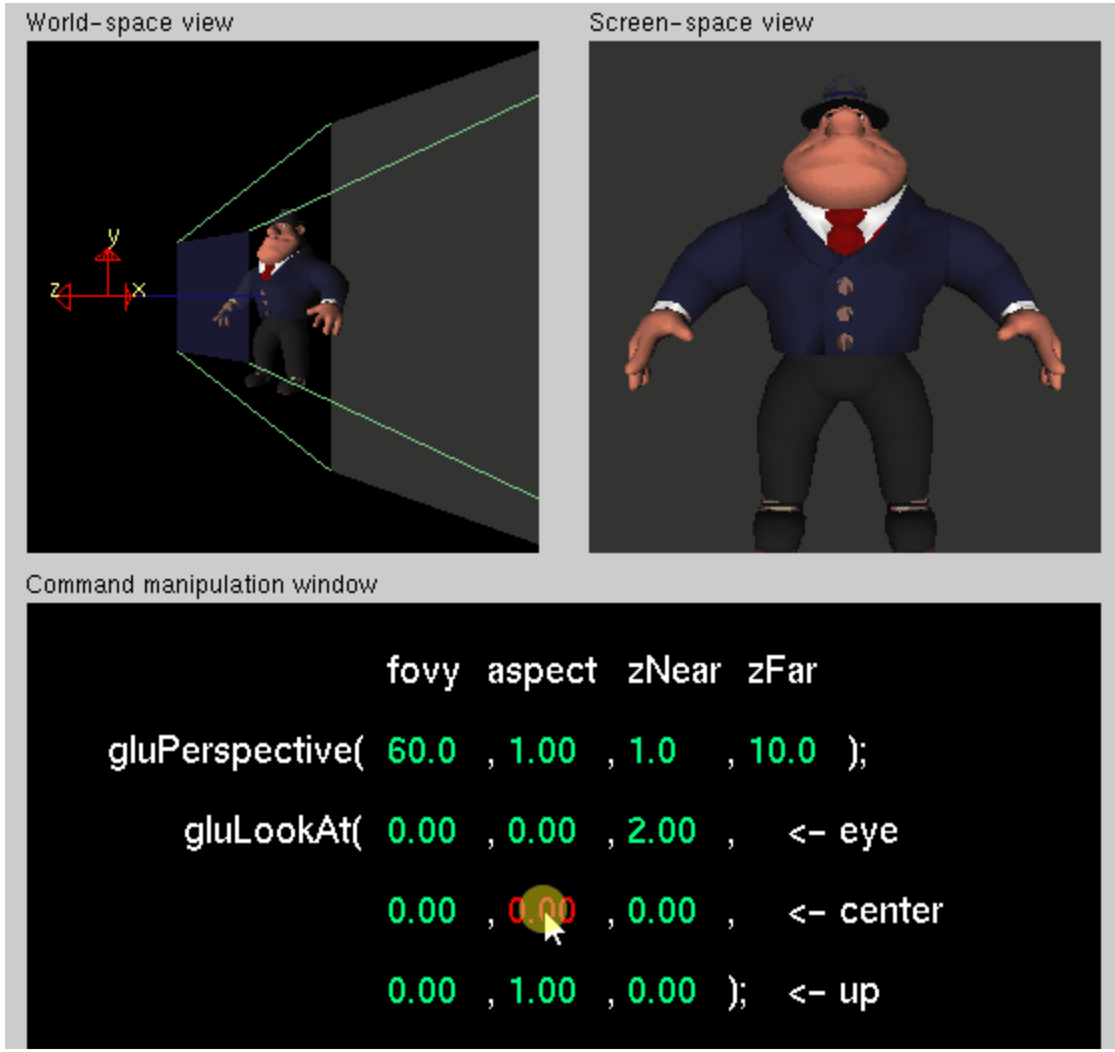


u, n, v are unit vectors.

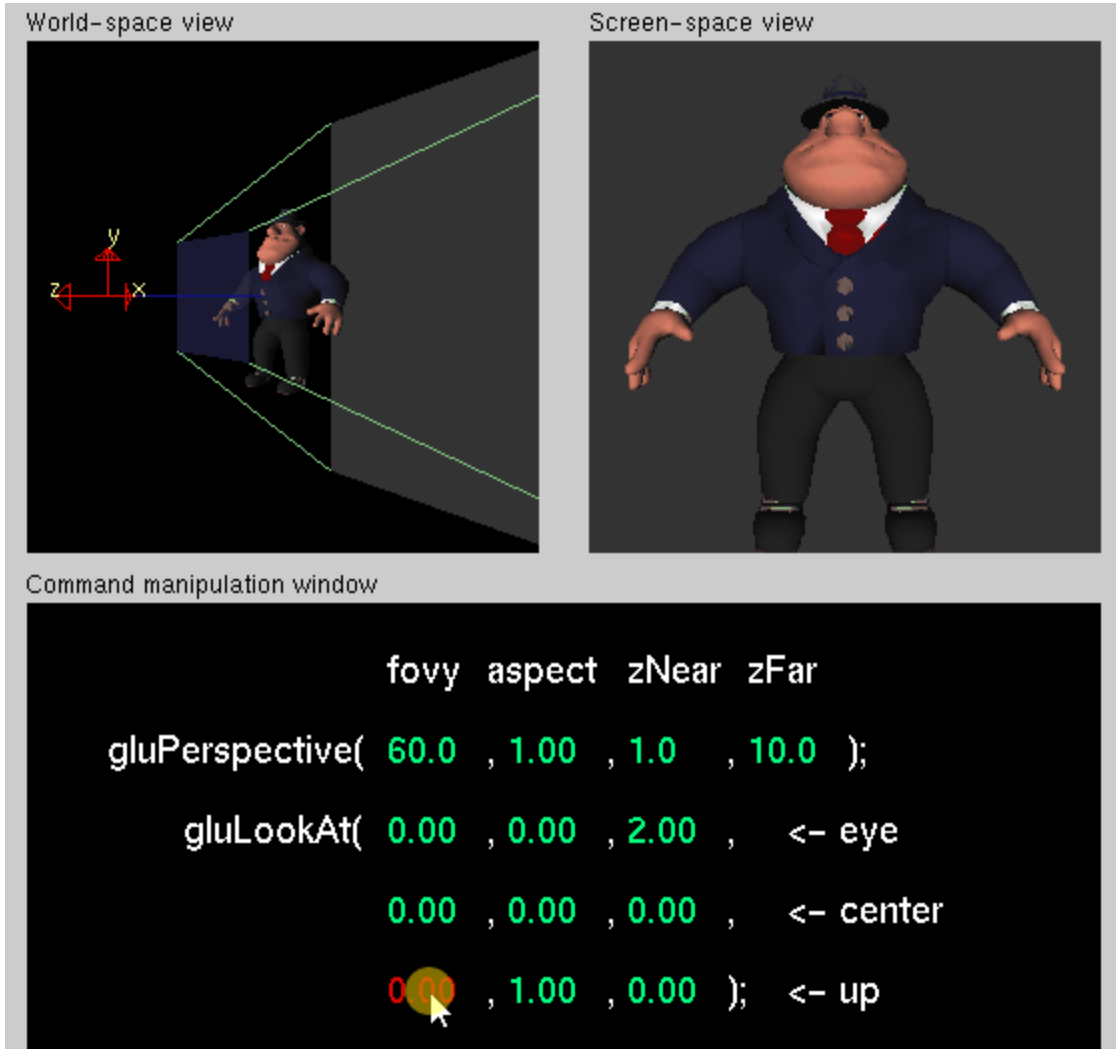
LookAt(eye, at, up) – Changing EYE



LookAt(eye, at, up) – Changing AT

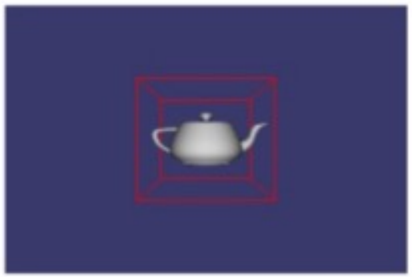


LookAt(eye, at, up) – Changing UP



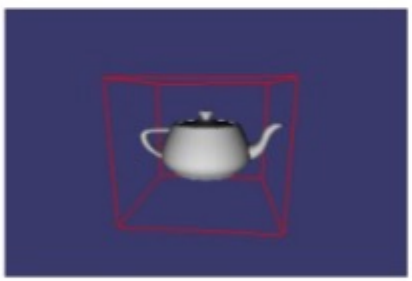
The above examples were animated GIF, so here are some static ones in case viewing slides in PDF

"Look At" Examples



```
gluLookAt(0,0,14,     // eye (x,y,z)
          0,0,0,      // at (x,y,z)
          0,1,0);     // up (x,y,z)
```

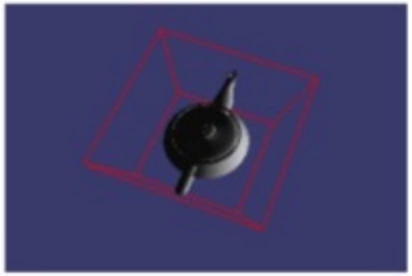
Same as the glTranslatef(0,0,-14) as expected



```
gluLookAt(1,2.5,11,   // eye (x,y,z)
          0,0,0,      // at (x,y,z)
          0,1,0);     // up (x,y,z)
```

Similar to original, but just a little off angle due to slightly perturbed eye vector

"Look At" Major Eye Changes



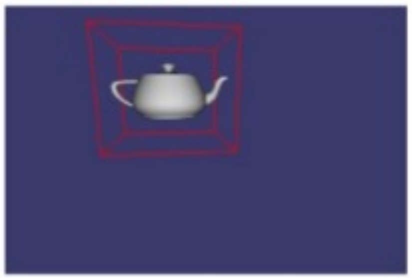
```
gluLookAt(-2.5,11,1,  // eye (x,y,z)
          0,0,0,      // at (x,y,z)
          0,1,0);     // up (x,y,z)
```

Eye is "above" the scene

```
gluLookAt(-2.5,-11,1, // eye (x,y,z)
          0,0,0,      // at (x,y,z)
          0,1,0);     // up (x,y,z)
```

Eye is "below" the scene

"Look At" Changes to AT and UP



```
gluLookAt(0,0,14,     // eye (x,y,z)
          2,-3,0,     // at (x,y,z)
          0,1,0);     // up (x,y,z)
```

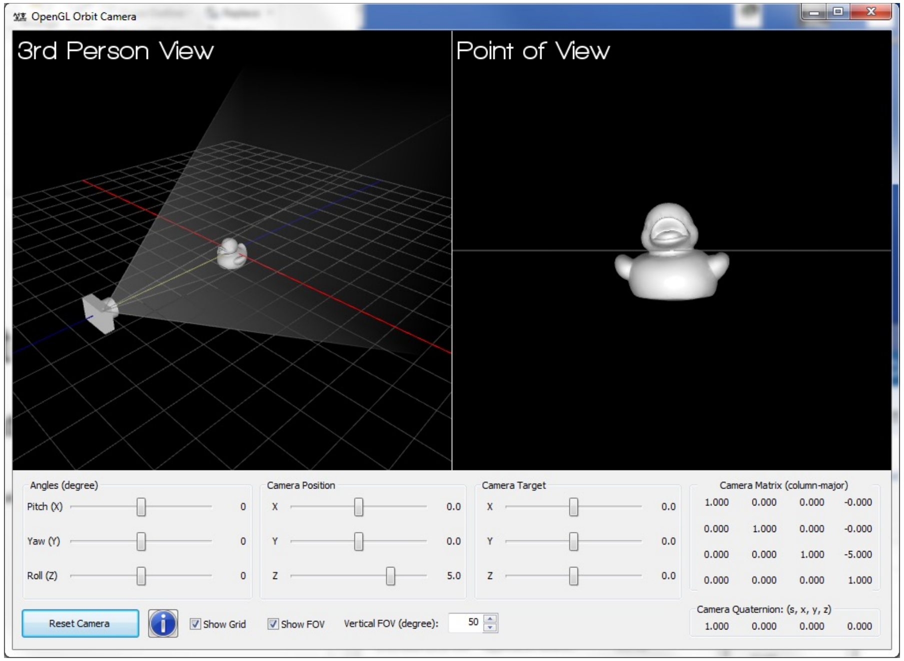
Original eye position, but "at" position shifted

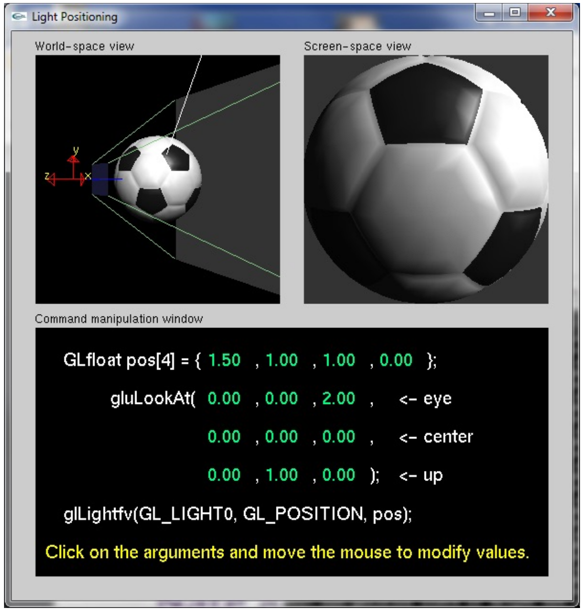
```
gluLookAt(0,0,14,     // eye (x,y,z)
          0,0,0,      // at (x,y,z)
          1,1,0);     // up (x,y,z)
```

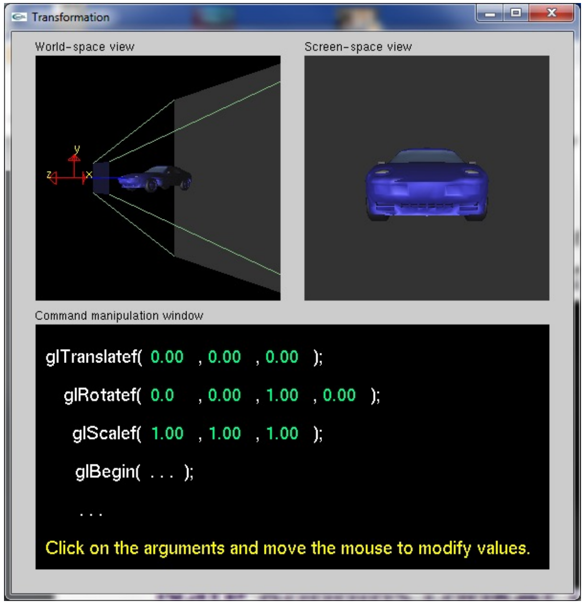
Eye is "below" the scene

http://www.songho.ca/opengl/gl_camera.html

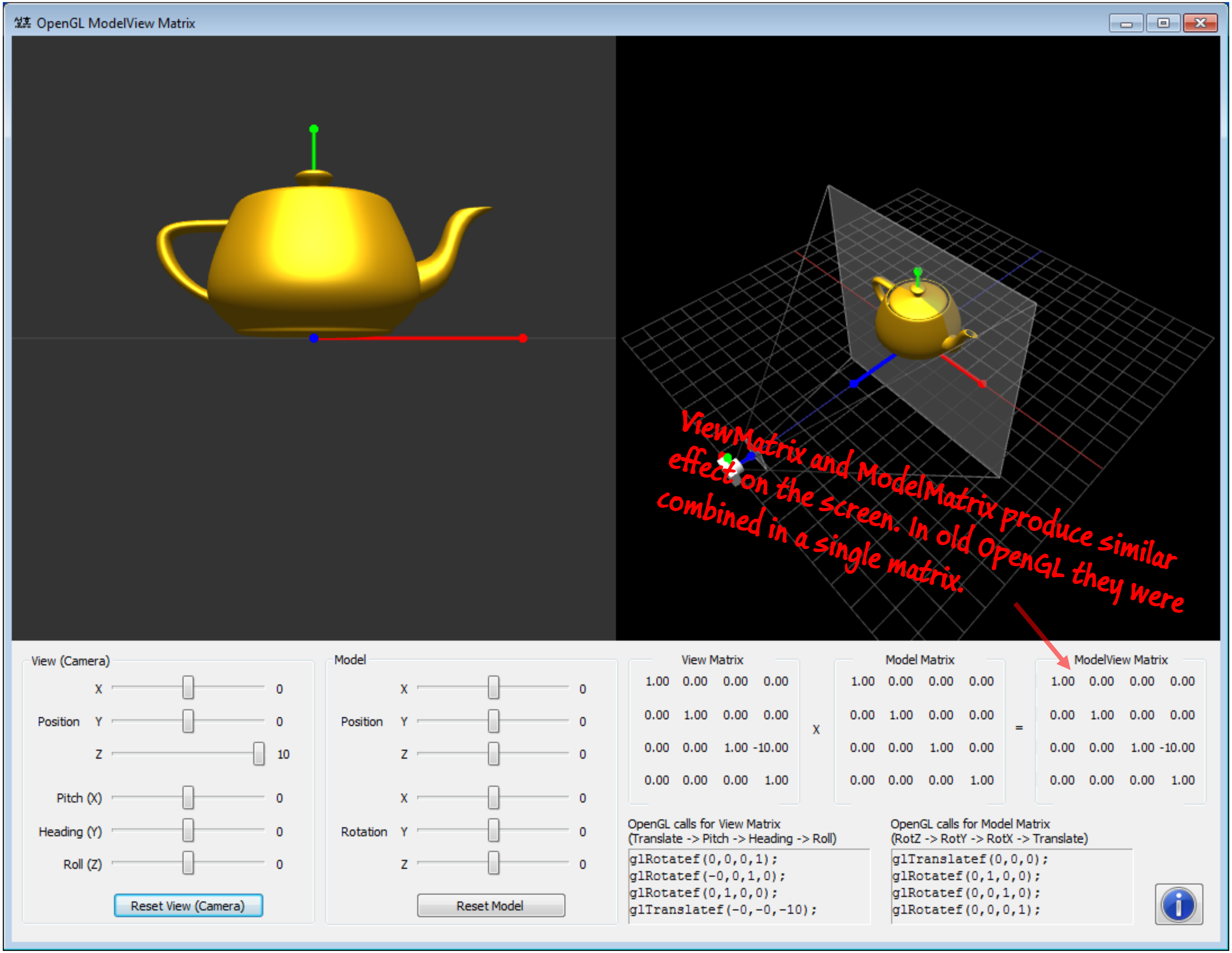
Some great interactive tools if you want to play with them

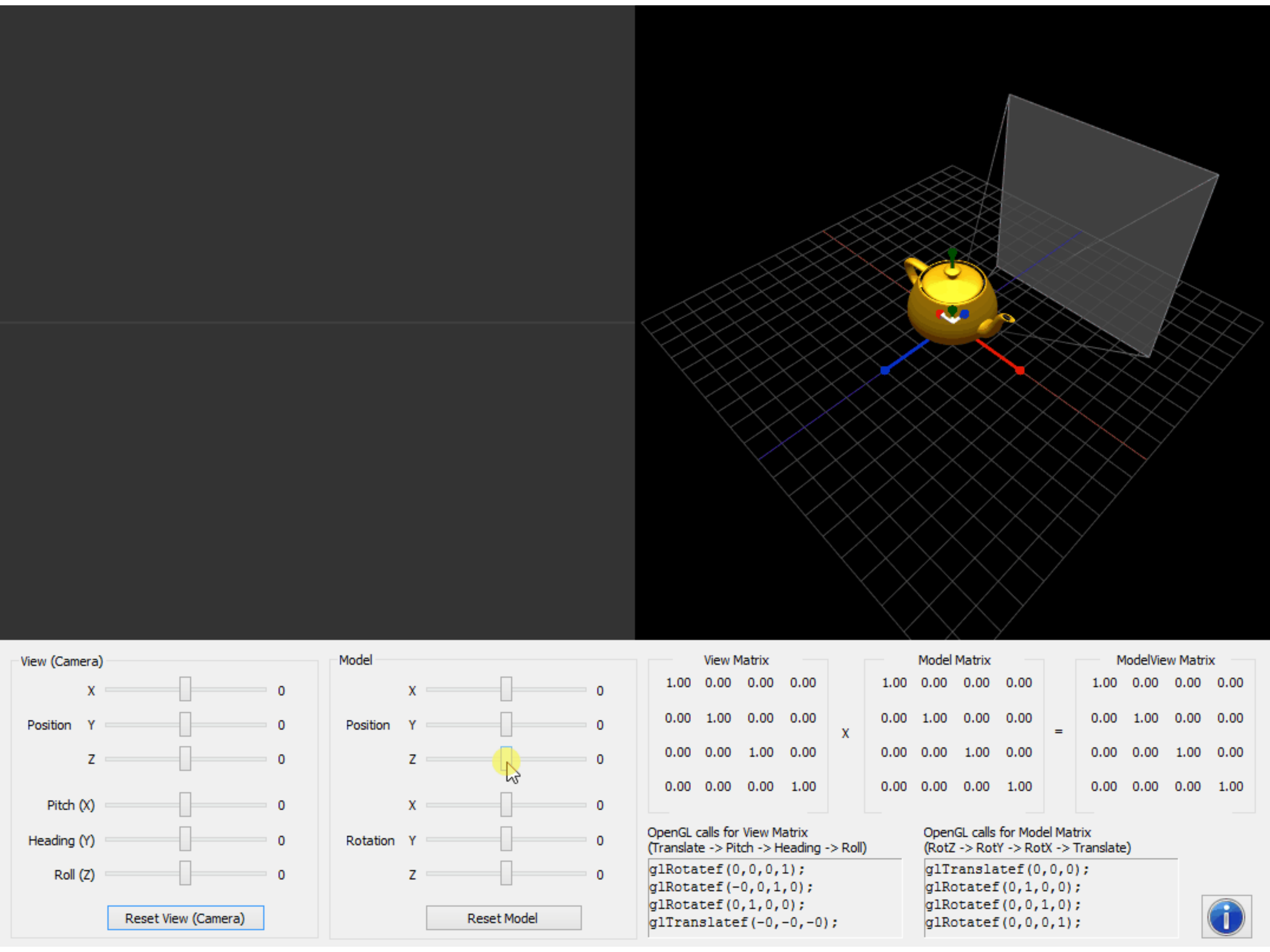






https://user.xmission.com/~nate/tutors.html


OpenGL ModelView Matrix
ViewMatrix and ModelMatrix produce similar effect on the screen. In old OpenGL they were combined in a single matrix.
View (Camera)
Position X 0
Y 0
Z 10
Pitch (X) 0
Heading (Y) 0
Roll (Z) 0
Reset View (Camera)
Model
Position X 0
Y 0
Z 0
Rotation X 0
Y 0
Z 0
Reset Model
View Matrix
1.00 0.00 0.00 0.00
0.00 1.00 0.00 0.00
0.00 0.00 1.00 -10.00
0.00 0.00 0.00 1.00
X
Model Matrix
1.00 0.00 0.00 0.00
0.00 1.00 0.00 0.00
0.00 0.00 1.00 0.00
0.00 0.00 0.00 1.00
=
ModelView Matrix
1.00 0.00 0.00 0.00
0.00 1.00 0.00 0.00
0.00 0.00 1.00 -10.00
0.00 0.00 0.00 1.00
OpenGL calls for View Matrix
(Translate -> Pitch -> Heading -> Roll)
glRotatef(0,0,0,1);
glRotatef(-0,0,1,0);
glRotatef(0,1,0,0);
glTranslatef(-0,-0,-10);
OpenGL calls for Model Matrix
(RotZ -> RotY -> RotX -> Translate)
glTranslatef(0,0,0);
glRotatef(0,1,0,0);
glRotatef(0,0,1,0);
glRotatef(0,0,0,1);


View (Camera)
Position
X
Y
Z
0
Pitch (X)
Heading (Y)
Roll (Z)
Reset View (Camera)
Model
Position
Rotation
Reset Model
View Matrix
1.00 0.00 0.00 0.00
0.00 1.00 0.00 0.00
0.00 0.00 1.00 0.00
0.00 0.00 0.00 1.00
X
Model Matrix
1.00 0.00 0.00 0.00
0.00 1.00 0.00 0.00
0.00 0.00 1.00 0.00
0.00 0.00 0.00 1.00
=
ModelView Matrix
1.00 0.00 0.00 0.00
0.00 1.00 0.00 0.00
0.00 0.00 1.00 0.00
0.00 0.00 0.00 1.00
OpenGL calls for View Matrix
(Translate -> Pitch -> Heading -> Roll)
glRotatef(0,0,0,1);
glRotatef(-0,0,1,0);
glRotatef(0,1,0,0);
glTranslatef(-0,-0,-0);
OpenGL calls for Model Matrix
(RotZ -> RotY -> RotX -> Translate)
glTranslatef(0,0,0);
glRotatef(0,1,0,0);
glRotatef(0,0,1,0);
glRotatef(0,0,0,1);

Participation Survey

Participation May 5

Form description

This form is automatically collecting email addresses for UC Santa Cruz users. Change settings

I was in class May 5

- Yes
- No

Roughly how long did you spend on HW3 (Color+Texture)

- 0-1 hours
- 1-2 hours
- 2-4 hours
- 4+ hours

There are videos from Lucas introducing Labs

Multiple choice

- I didn't watch it, I just started the assignment
- I watched it, but its NOT helpful
- I watched it, and it IS helpful
- Other...
- Add option

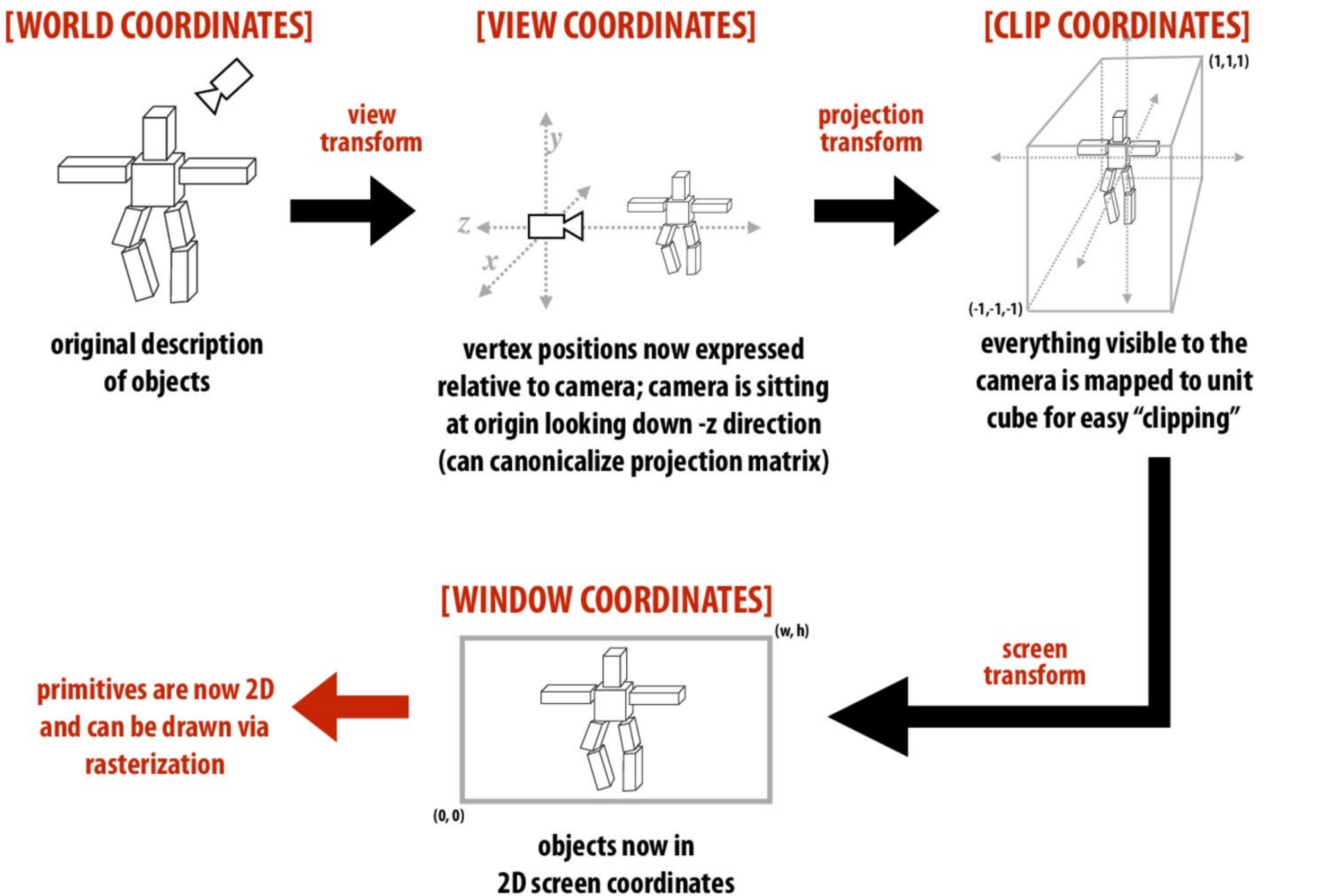
Required

There are videos from James introducing

- I didn't watch them, I just started the assignment

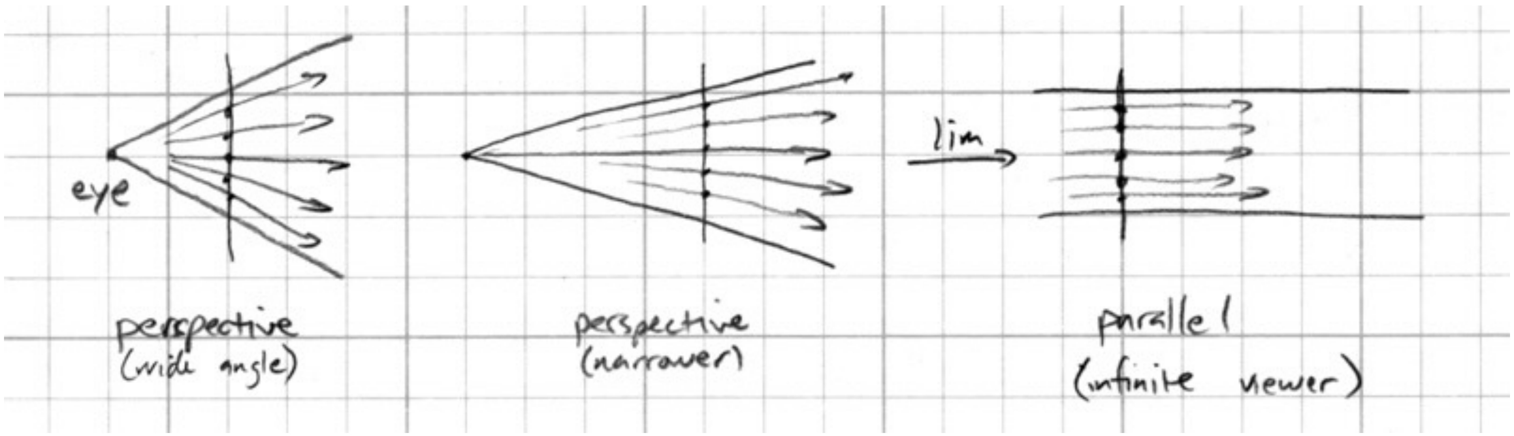
Projection Transform

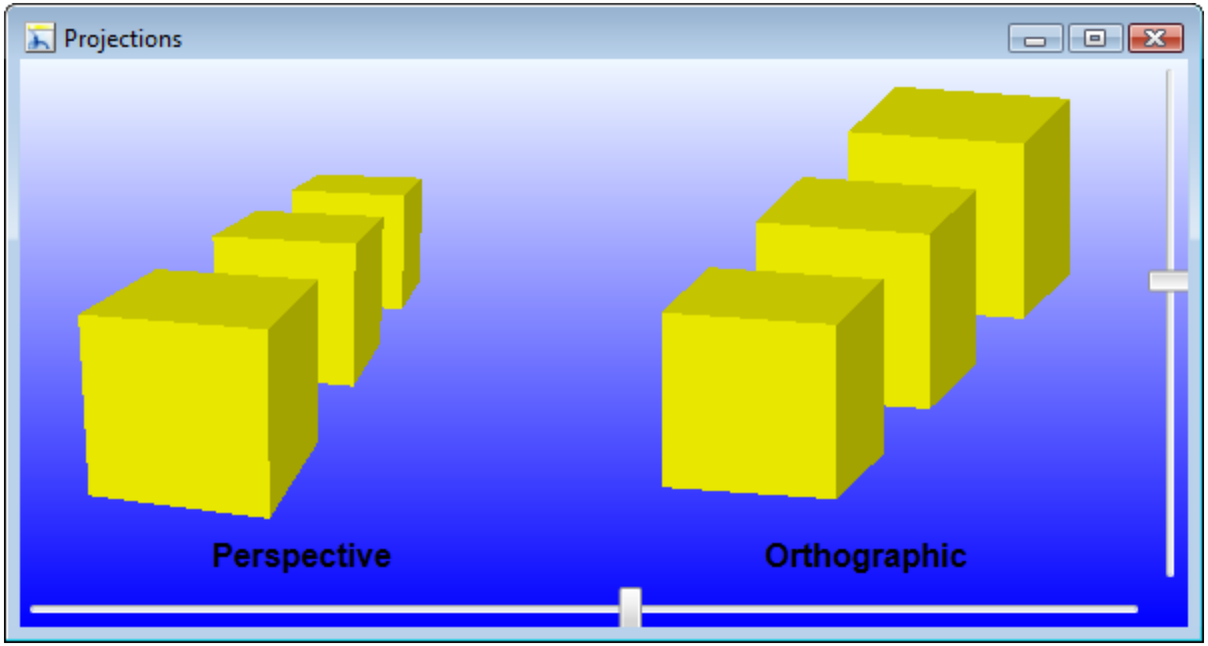
Transformations: from objects to the screen



Parallel projection

- Viewing rays are parallel rather than diverging
  - like a perspective camera that's far away

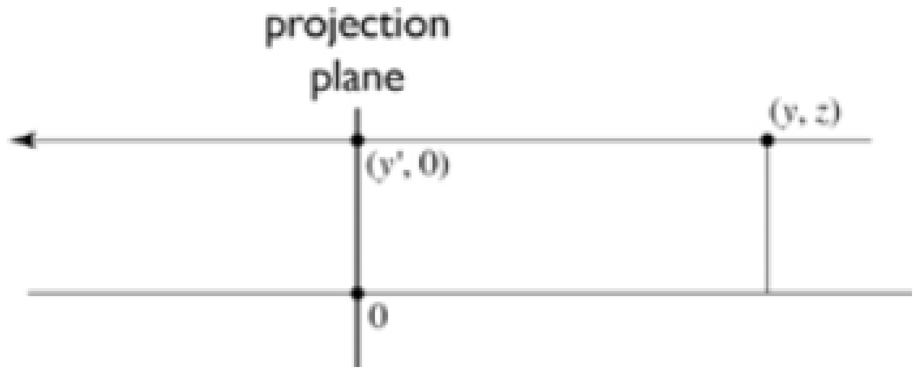




perspective projection

orthographic projection

Parallel projection: orthographic



to implement orthographic, just toss out z:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

More on this later: Left, right, bottom, top, near, far

Orthographic projection:

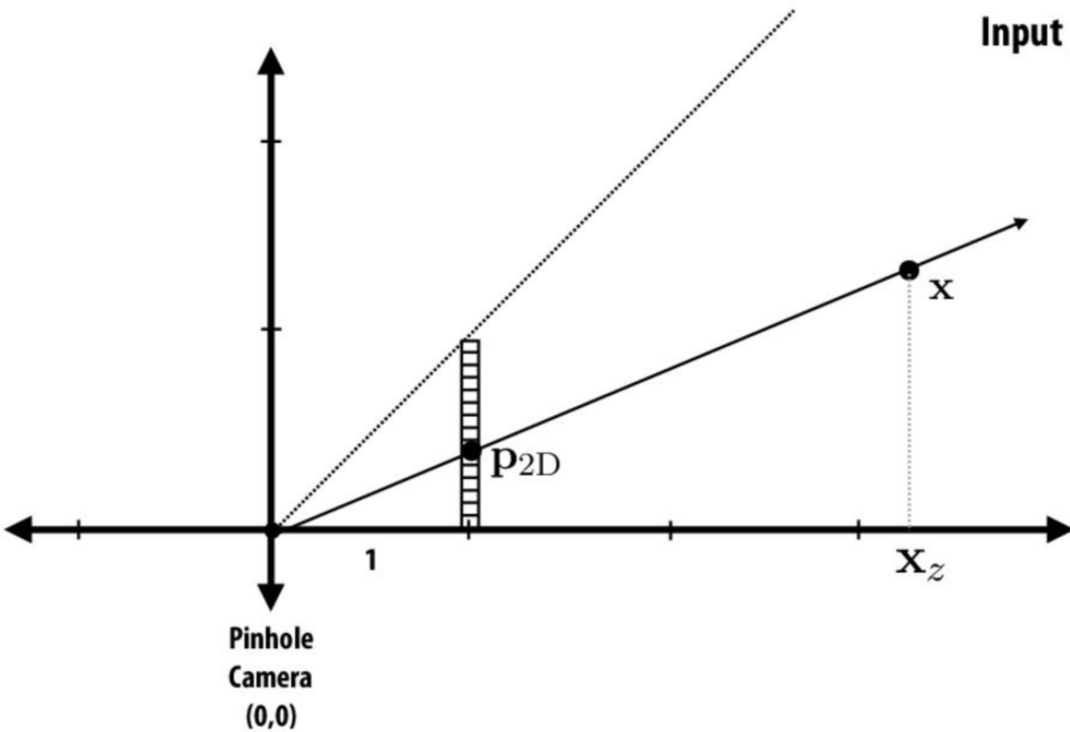
glOrtho(l, r, b, t, n, F); (n and F always > 0)

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & -\frac{2}{F-n} & -\frac{F+n}{F-n} \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- transforms to interval -1..+1

Basic perspective projection

Input point in 3D-H: $\mathbf{x} = \begin{bmatrix}\mathbf{x}_x & \mathbf{x}_y & \mathbf{x}_z & 1\end{bmatrix}^T$



$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Assumption:
Pinhole camera at (0,0) looking down z

Perspective vs. orthographic projection

- **Most basic version of perspective matrix:**

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ z \end{bmatrix} \longmapsto \begin{bmatrix} x/z \\ y/z \\ 1 \\ 1 \end{bmatrix}$$

objects shrink in distance

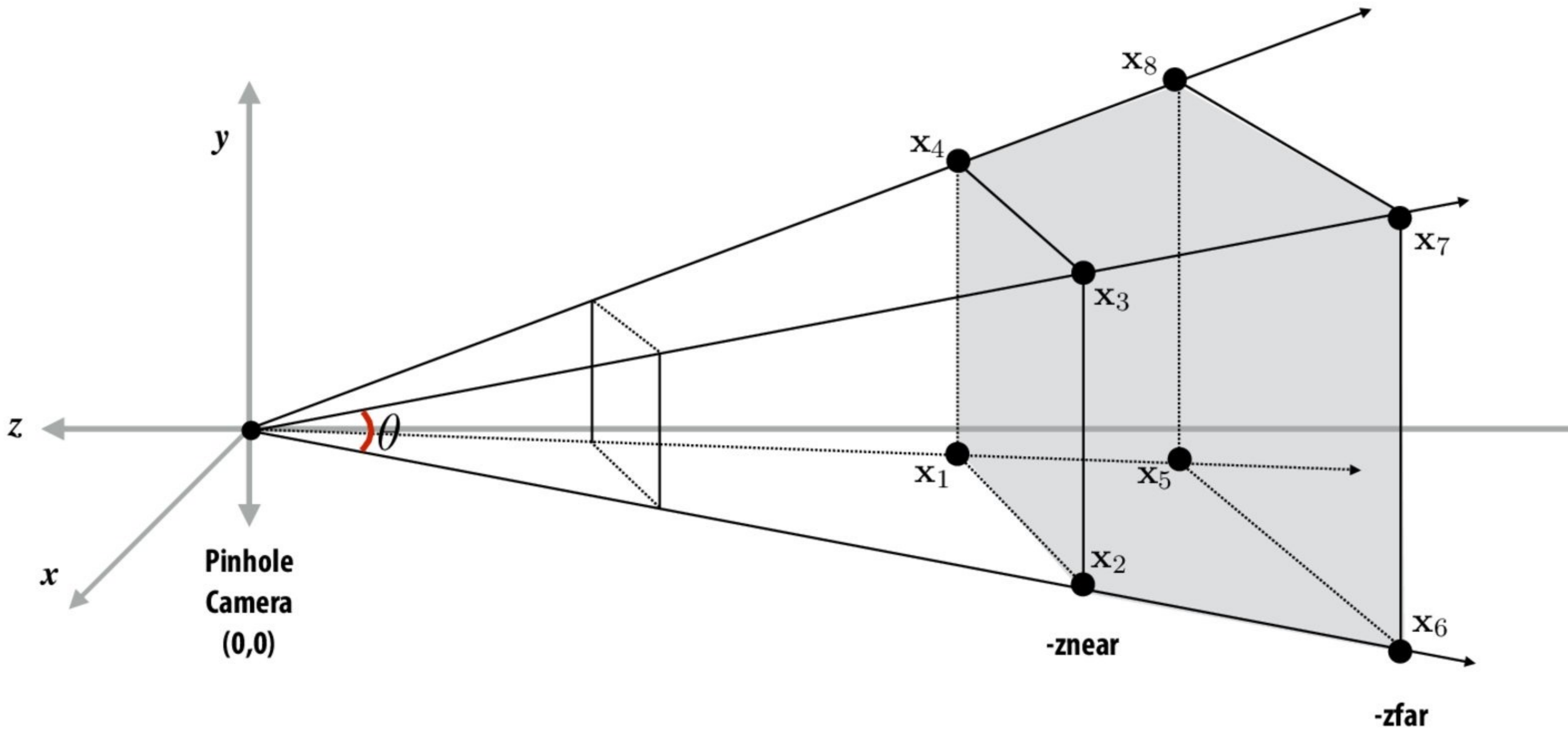
- **Most basic version of orthographic matrix:**

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \longmapsto \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

objects stay the same size

View frustum

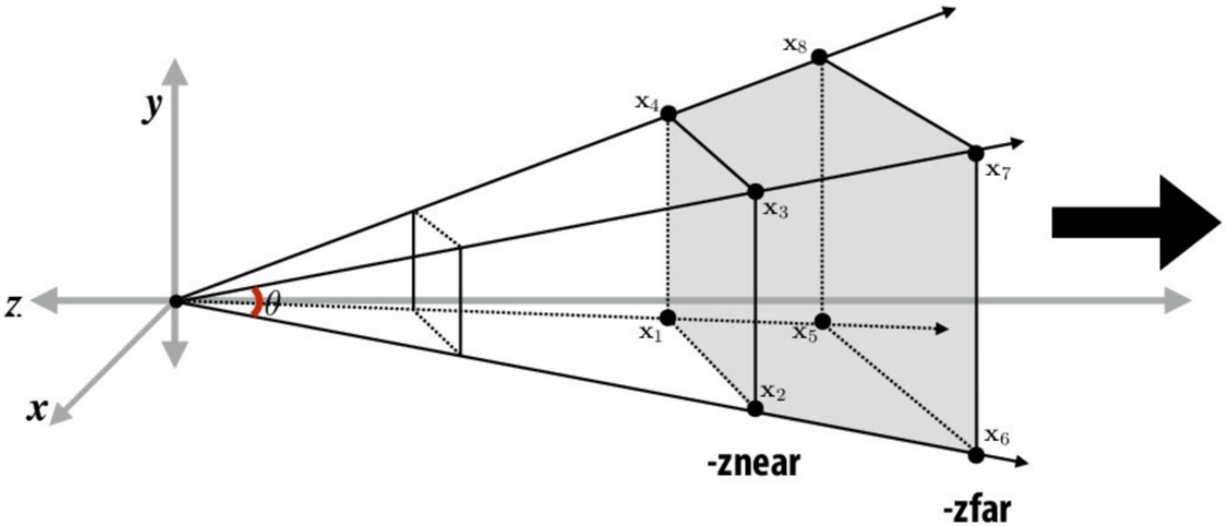
View frustum is the region of space the camera can see:

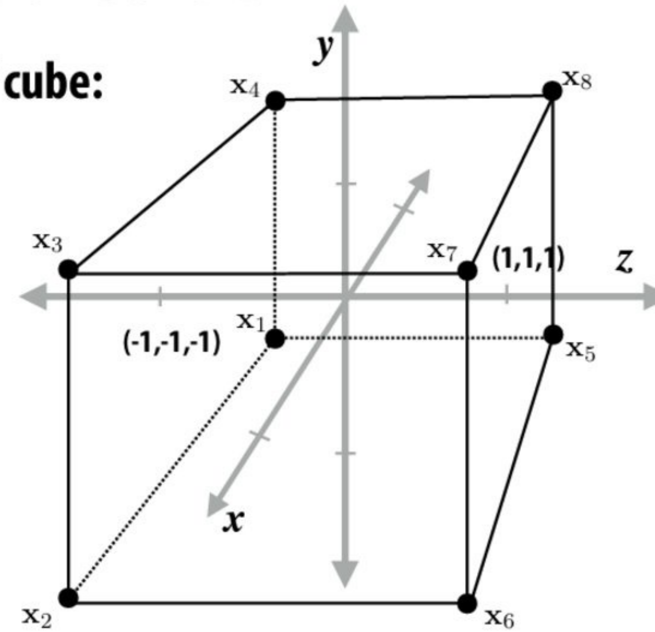


- **Top/bottom/left/right planes correspond to sides of screen**
- **Near/far planes correspond to closest/furthest thing we want to draw**

Mapping frustum to normalized cube

Before moving to 2D, map corners of view frustum to corners of cube:





View frustum corresponding to pinhole camera
(perspective projection transform transforms this volume to normalized cube)

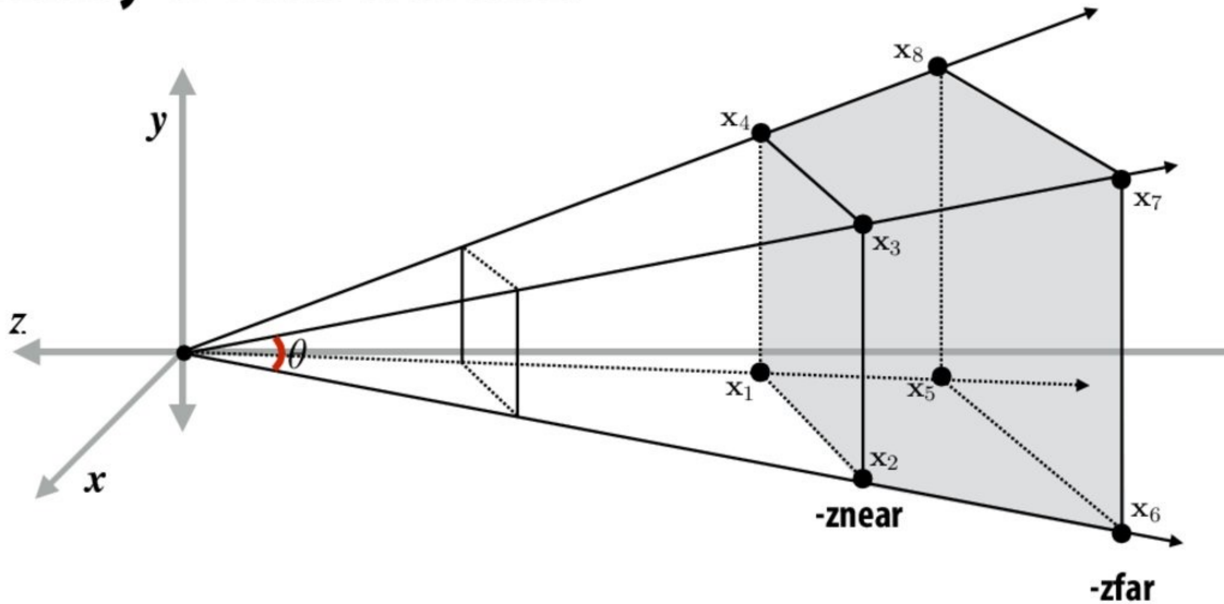
Why do we map frustum to unit cube?

1. Makes *clipping* much easier! (see next slide)
   - Can quickly discard geometry outside range [-1,1]
2. Represent all vertices in normalized cube in fixed point math

* Question: what does the frustum of an orthographic camera look like?

Matrix for perspective transform

Takes into account geometry of view frustum:

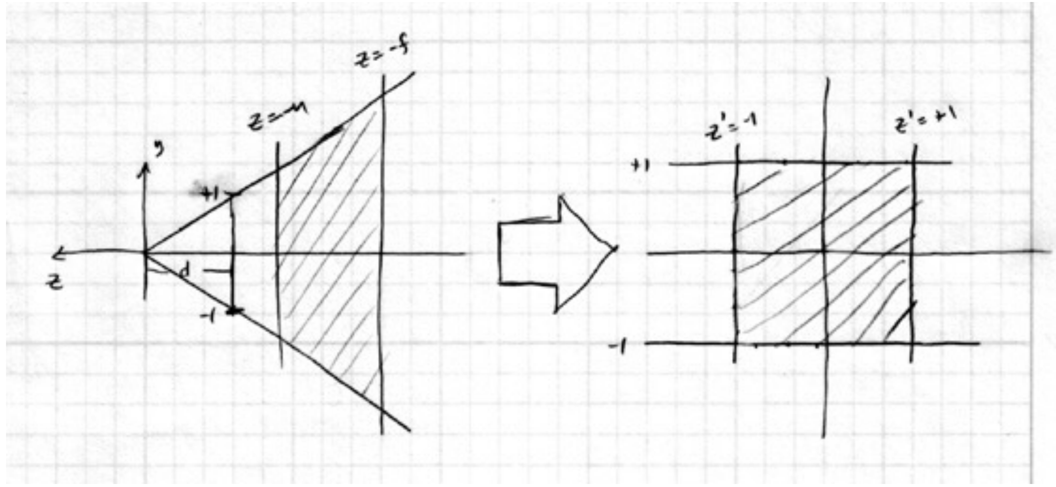


$$\begin{pmatrix} \frac{n}{r} & 0 & 0 & 0 \\ 0 & \frac{n}{t} & 0 & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

left (l), right (r), top (t), bottom (b), near (n), far (f)

(matrix at left is perspective projection for frustum that is symmetric about x,y axes: l=-r, t=-b)

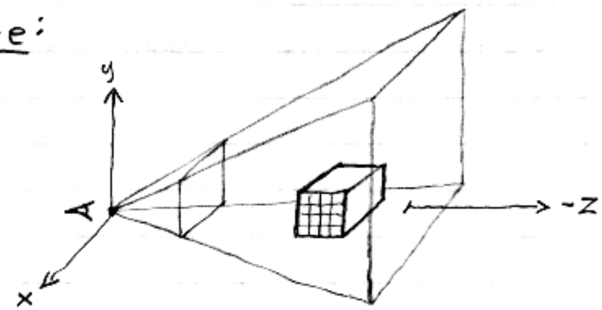
For a derivation: http://www.songho.ca/opengl/gl_projectionmatrix.html



[Marschner]

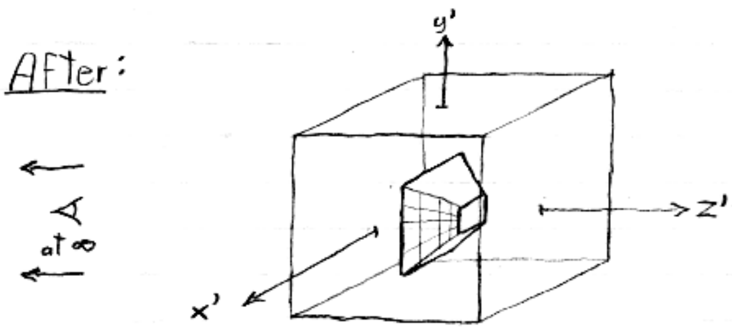
Perspective interpreted as a distortion of 3-space:

- Before:



[Levoy]

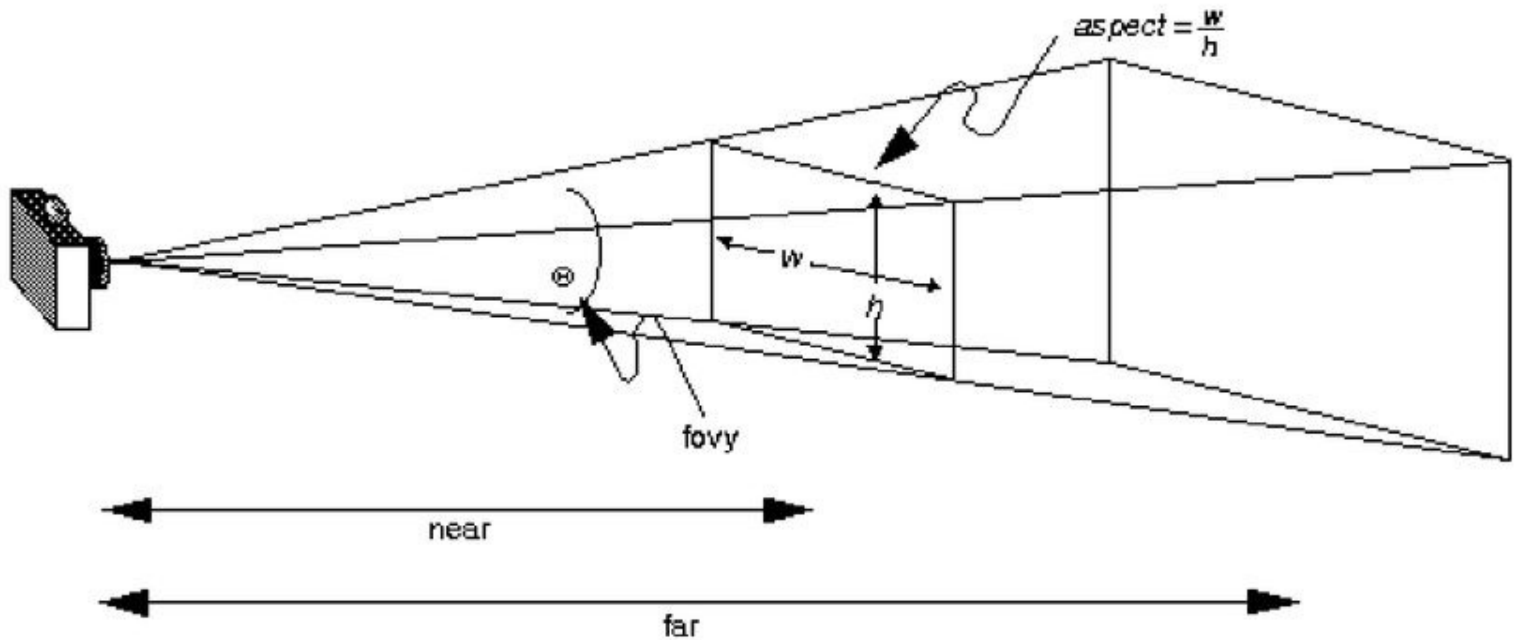
- After:



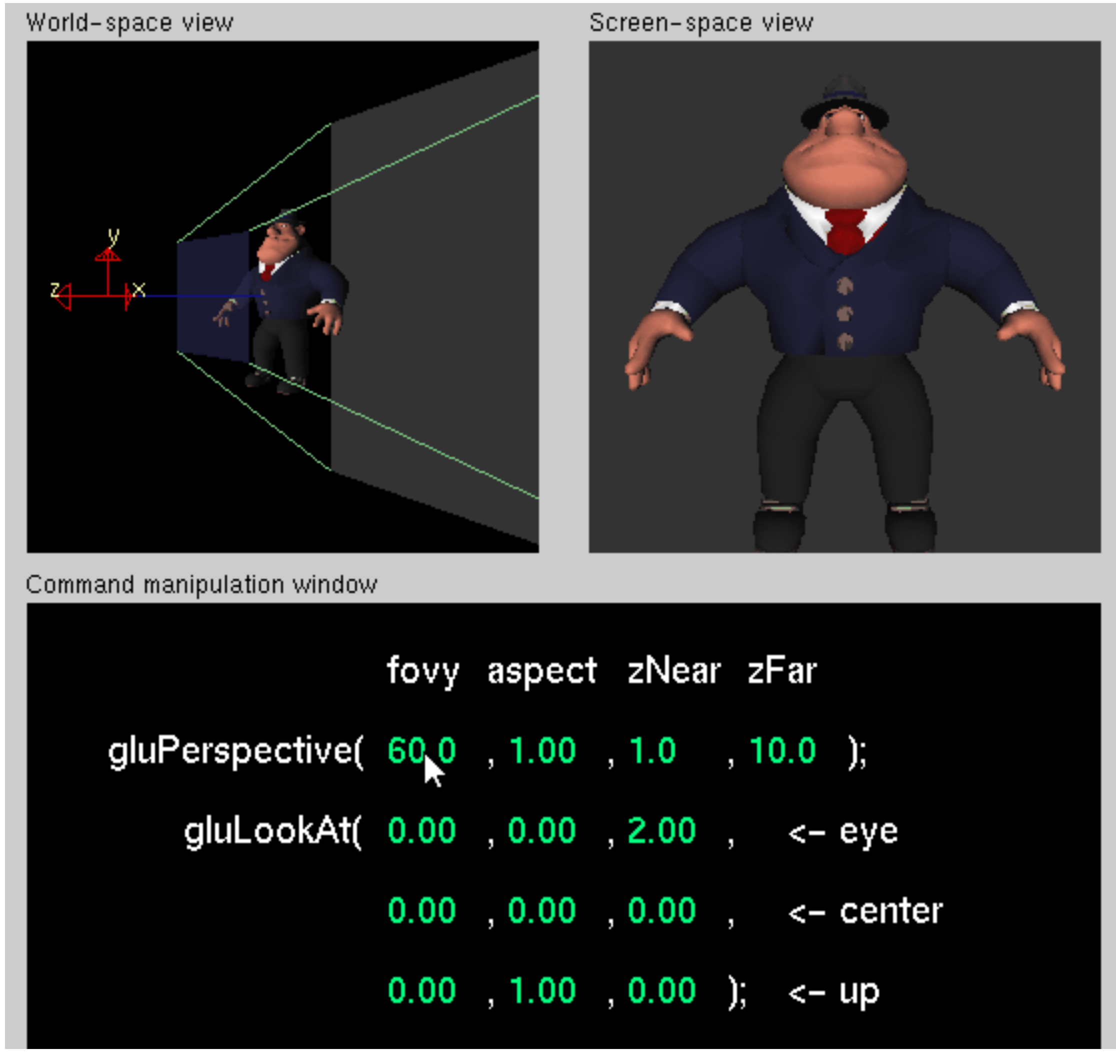
- Note compression of grid lines in x', y' and z'.

gluPerspective

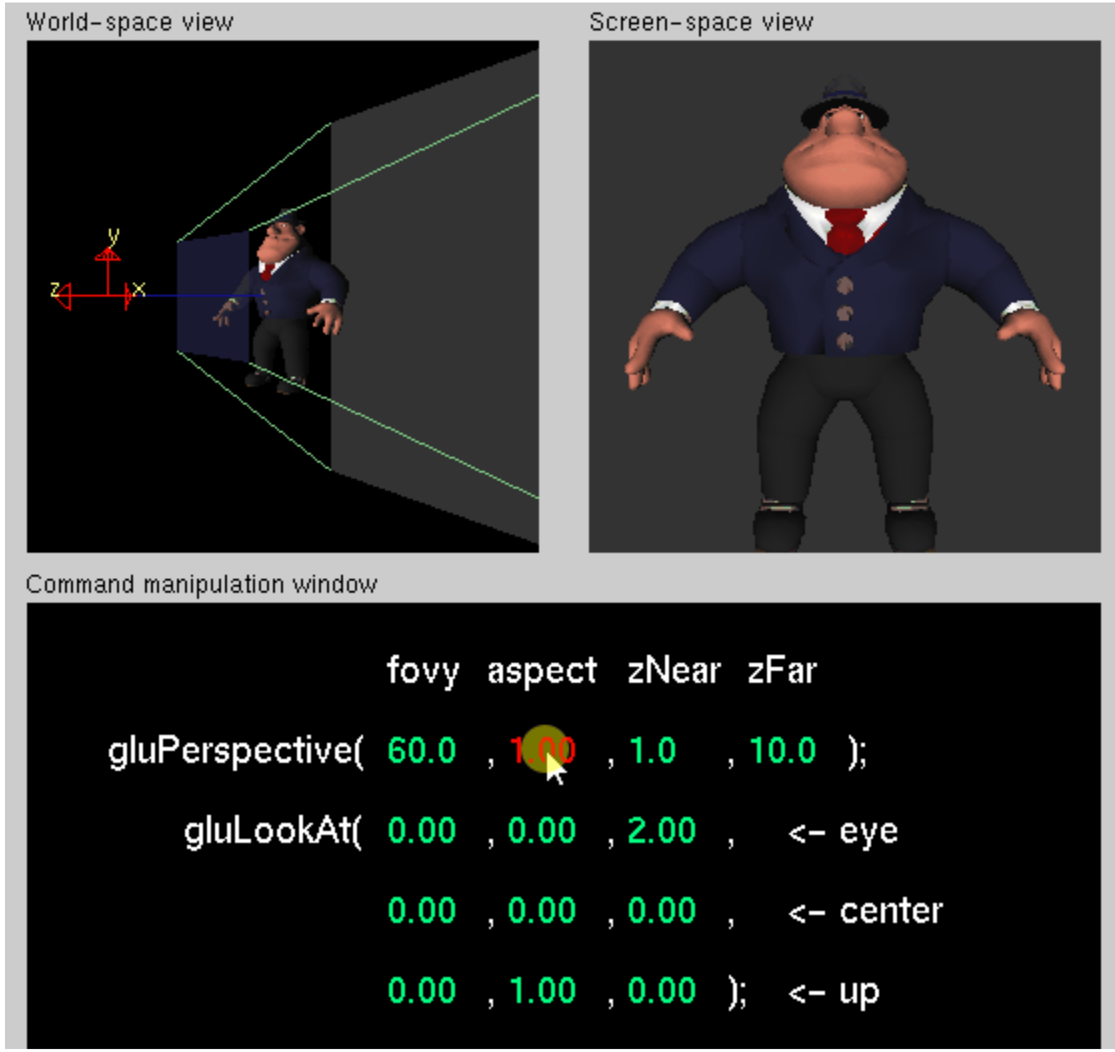
```
gluPerspective(double fovy, double aspect, double zNear, double zFar)
```



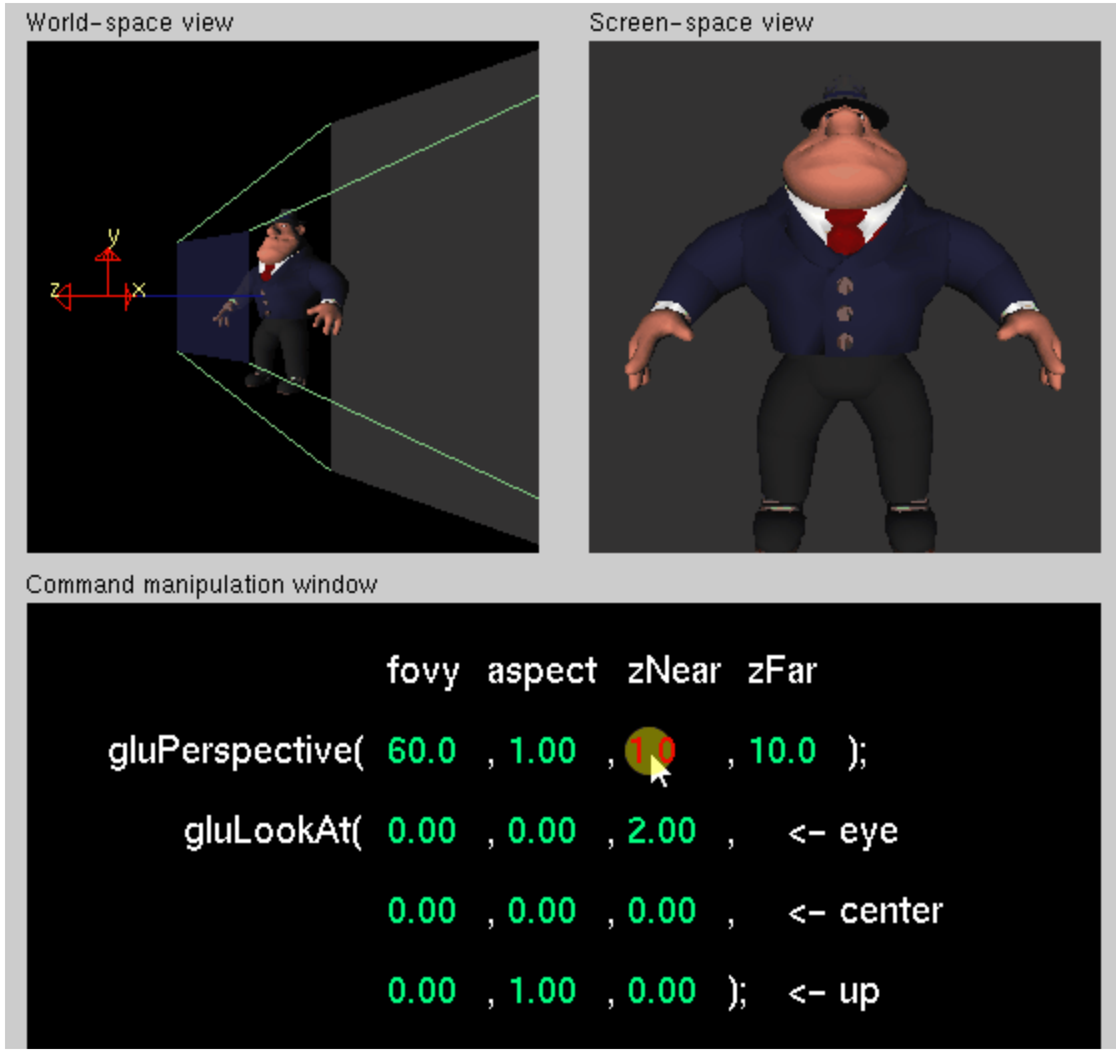
Perspective(fovy, aspect, zNear, zFar) – Changing FOVY



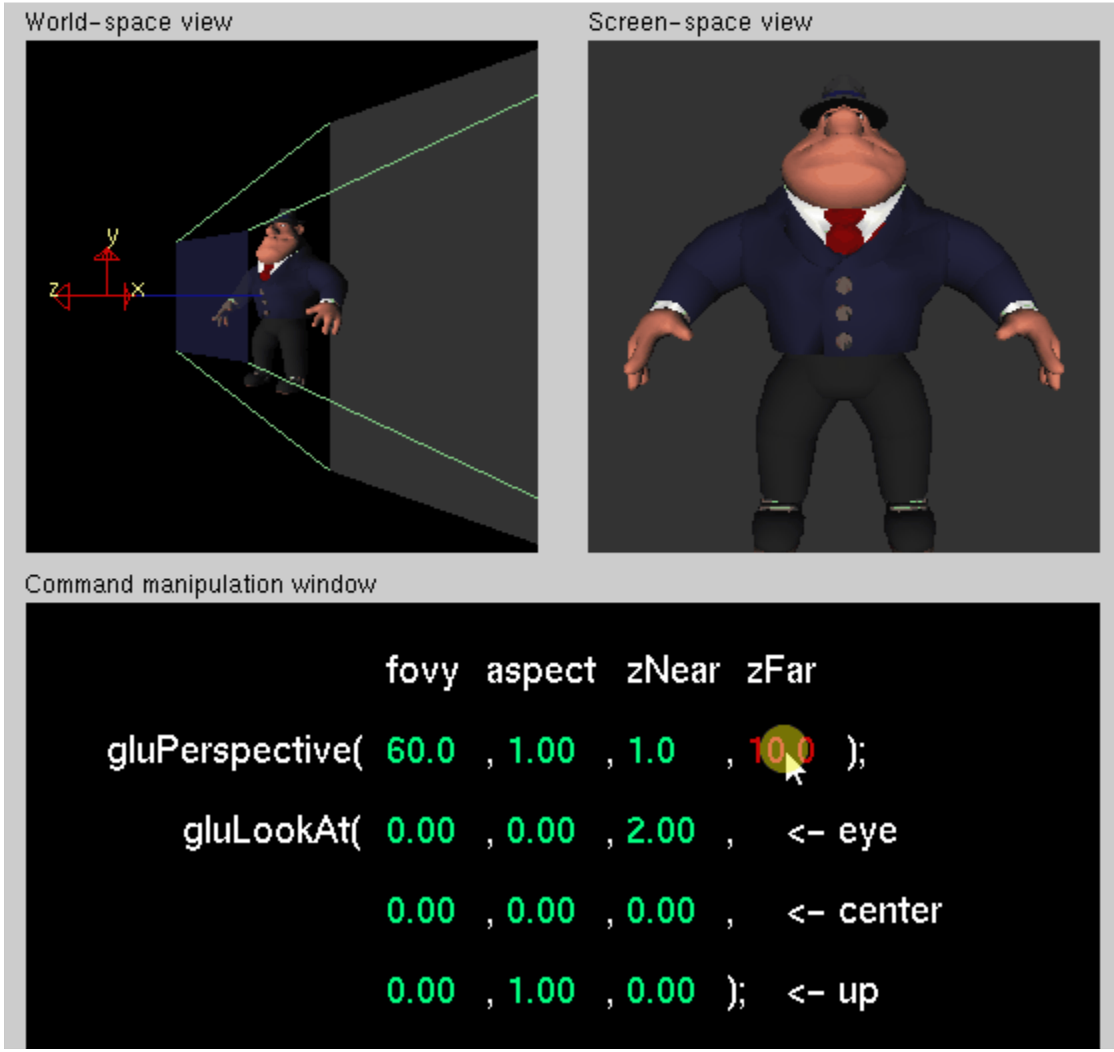
Perspective(fovy, aspect, zNear, zFar) – Changing ASPECT



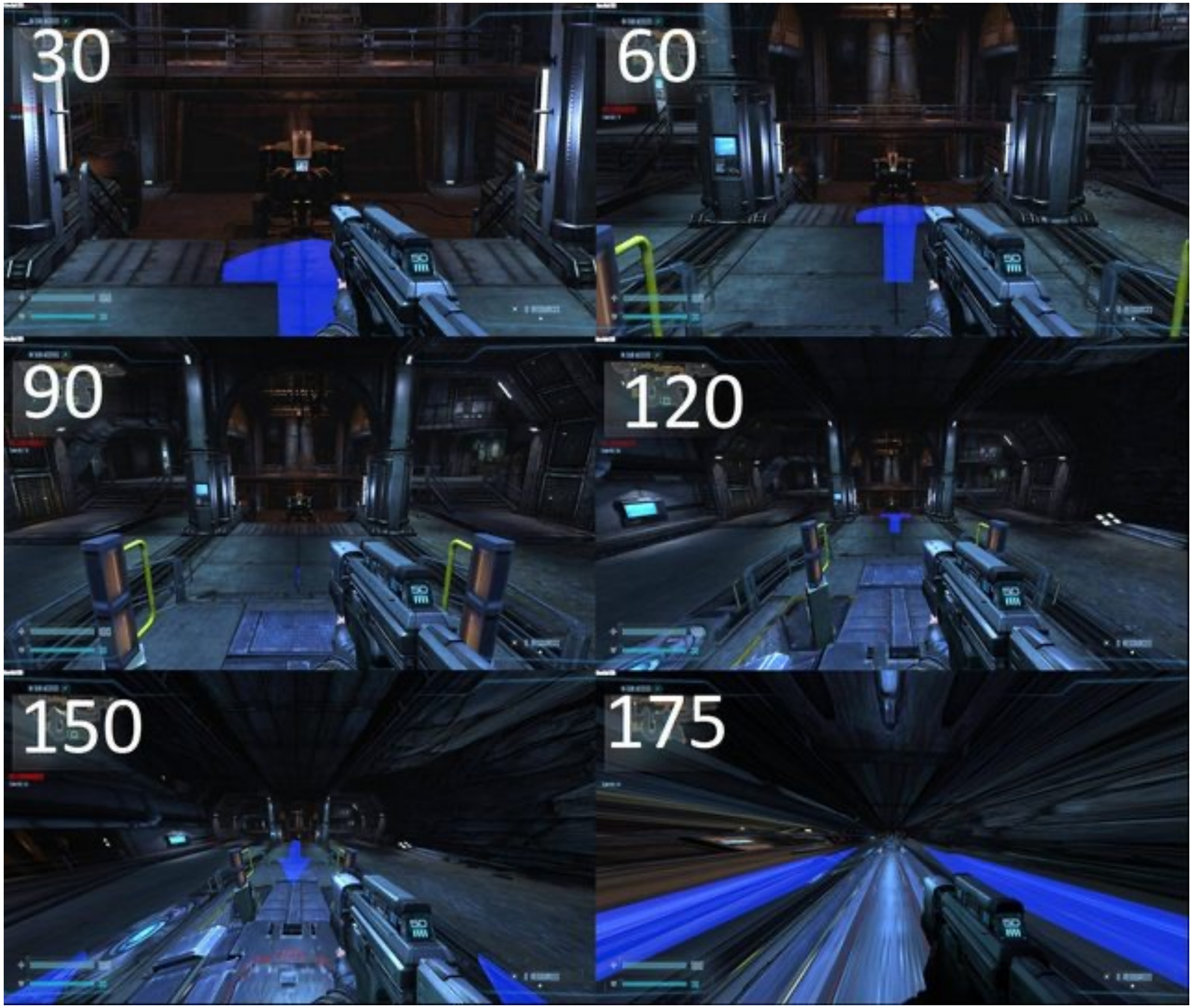
Perspective(fovy, aspect, zNear, zFar) – Changing NEAR



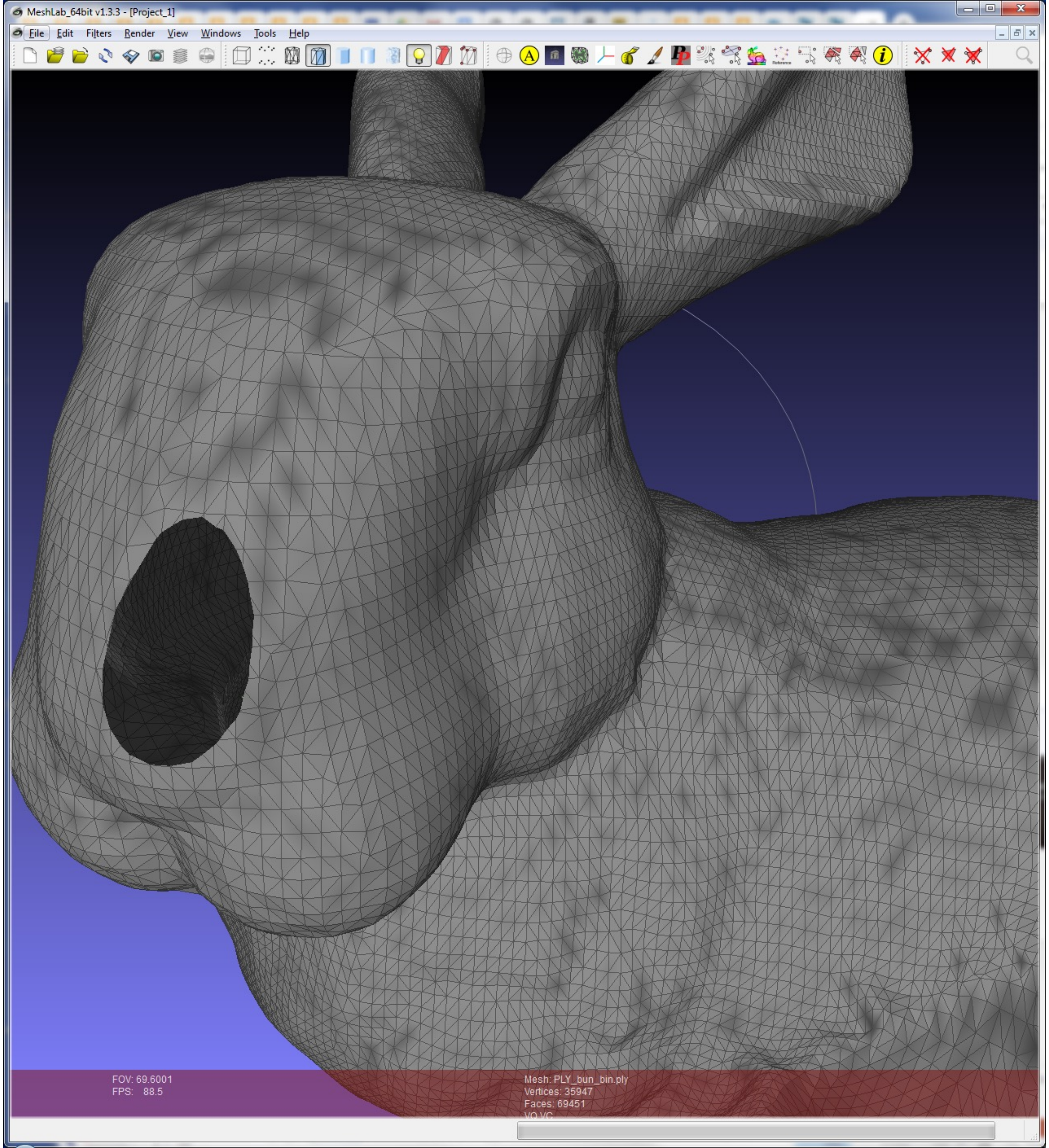
Perspective(fovy, aspect, zNear, zFar) – Changing FAR



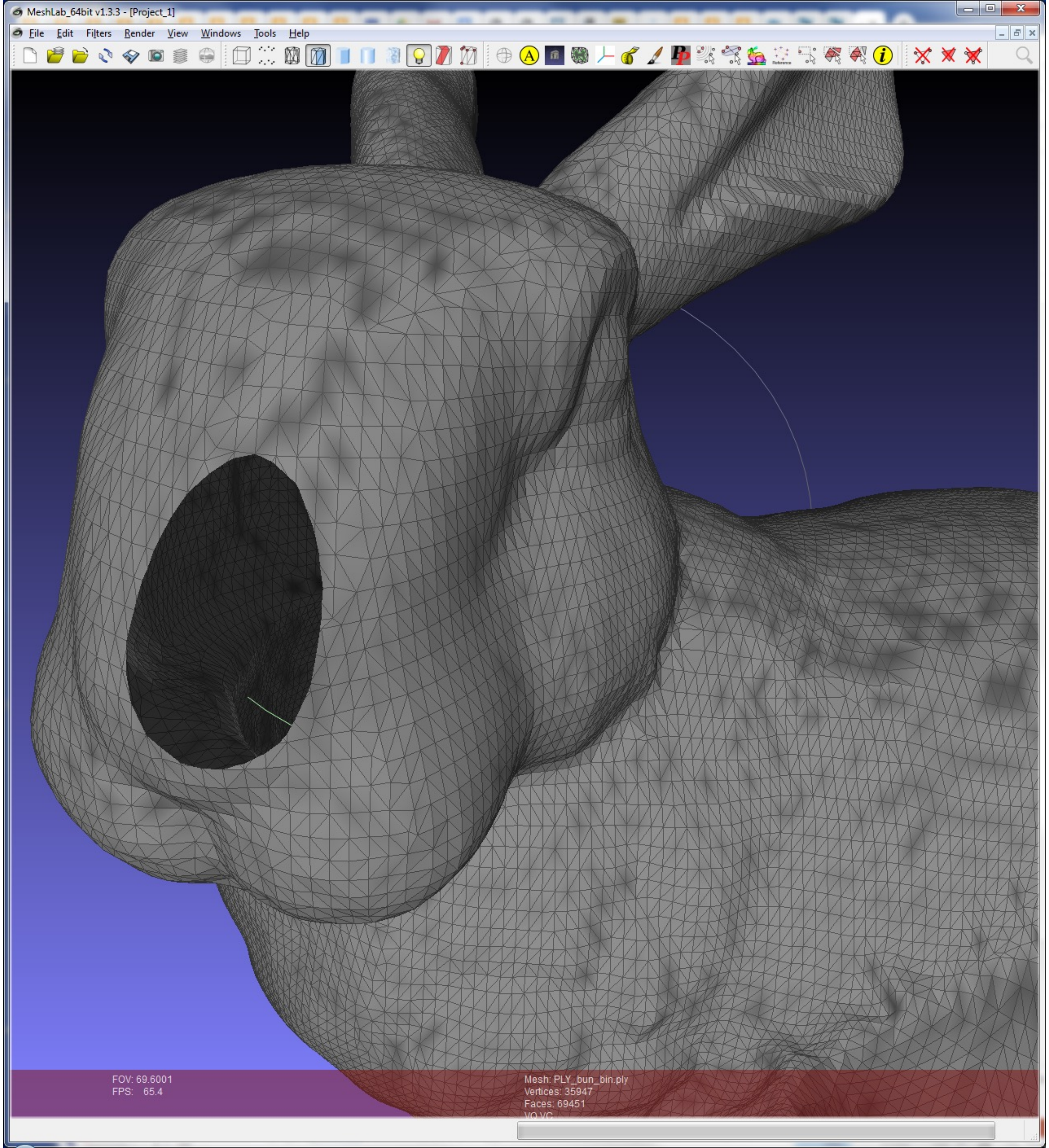
FOV



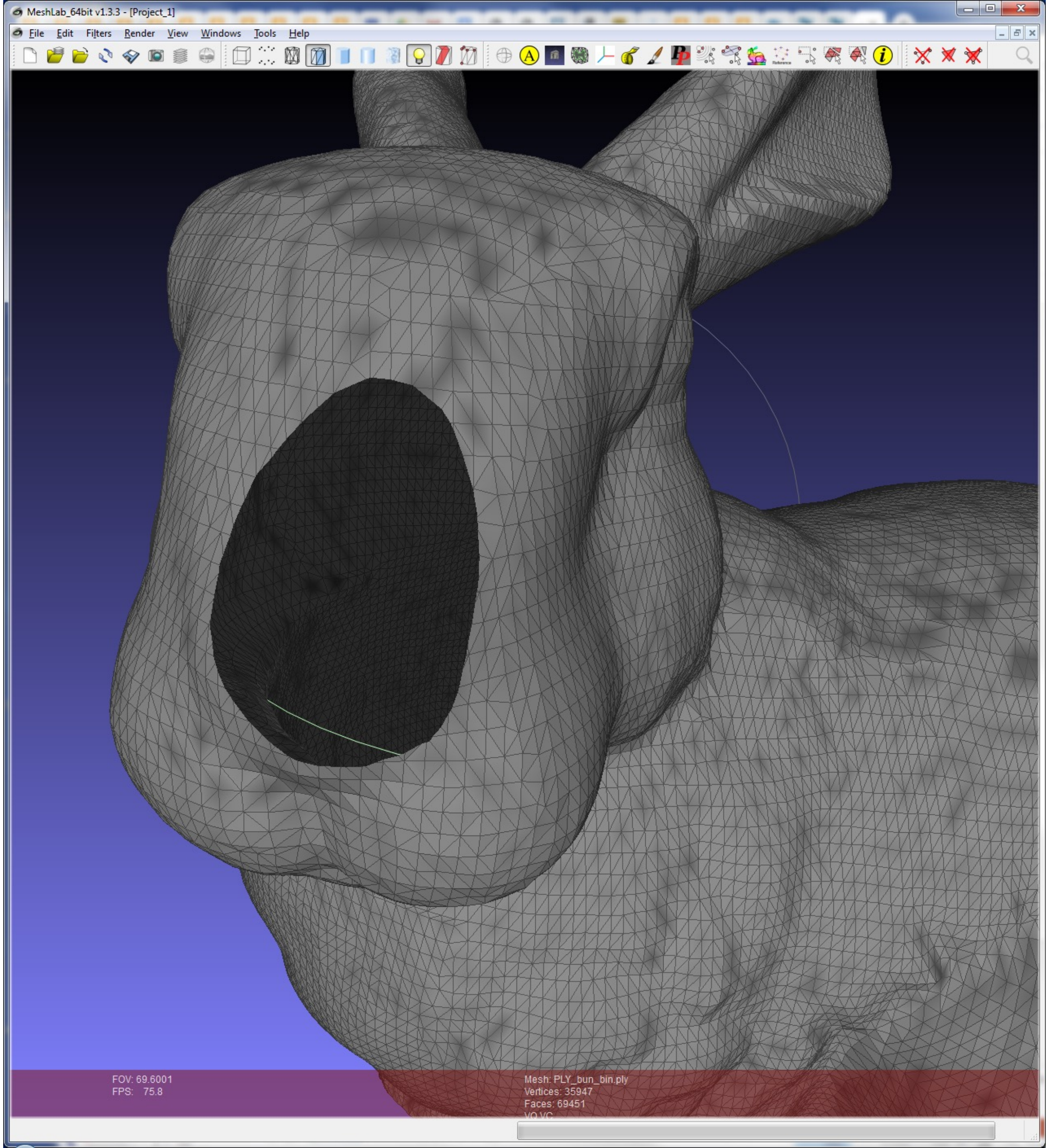
Near Plane Clipping Example



Near Plane Clipping Example



Near Plane Clipping Example




Objective: Talk to the Villagers (3/12)
Near plane clipping of villagers head
Z-fighting
Purse: 30
Objective
Talk to the Villagers
(3/12)
www.hypixel.net
100/100
100/100 Mana
[VIP] Witless: enchanted cobblestone (x64) on my ah 2 minutes left, cheap
[MVP+] ItMistas: Selling Stack OF PURPLE CANDY ON my Ah 500 COins 15 MIN left
ThePh03nix: cheap lapis blocks in my ah ending soon


Objective: Talk to the Villagers (3/12)
SKYBLOCK
02/09/20 mega1C
Spring 5th
2:50am
Village
Purse: 30
Objective
Talk to the Villagers
(3/12)
www.hypixel.net
100/100❤
100/100✎ Mana
Saved screenshot as 2020-02-09_17.37.24.png
lxzq was killed by Sven Packmaster.
11
3
21
3
2


Objective: Talk to the Villagers (3/12)
SKYBLOCK
02/09/20 mega1C
Spring 5th
3:00am
Village
Purse: 30
Objective
Talk to the Villagers
(3/12)
www.hypixel.net
100/100❤
100/100✎ Mana
lxzq was killed by Sven Packmaster.
Saved screenshot as 2020-02-09_17.37.30.png
[VIP] Yoonsoocraft: selling maxed out aotd with crit 6 party me if interested


Objective: Talk to the Villagers (3/12)
SKYBLOCK
02/09/20 mega1C
Spring 5th
3:10am
Village
Purse: 30
Objective
Talk to the Villagers
(3/12)
www.hypixel.net
13
12
11
10
9
8
7
6
5
4
3
2
1
100/100❤ 100/100✎ Mana
lxzq was killed by Sven Packmaster.
Saved screenshot as 2020-02-09_17.37.30.png
[VIP] Yoonsoocraft: selling maxed out aotd with crit 6 party me if interested
Saved screenshot as 2020-02-09_17.37.38.png
[MVP] azuru_el_mejor: selling demonic sword on my ah!!!!
11
3
21
3
2

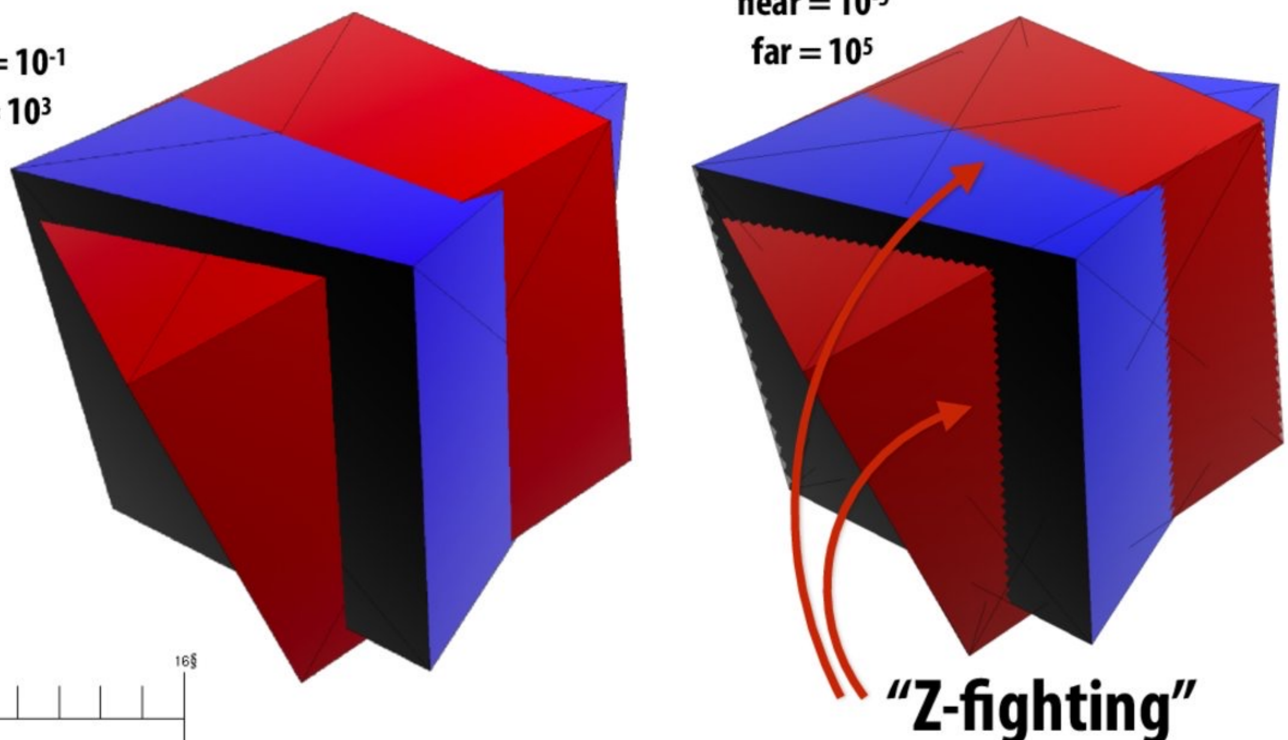

Objective: Talk to the Villagers (3/12)
SKYBLOCK
02/09/20 mega1C
Spring 5th
3:10am
Village
Purse: 30
Objective
Talk to the Villagers
(3/12)
www.hypixel.net
13
12
11
10
9
8
7
6
5
4
3
2
1
100/100❤
100/100✎ Mana
[VIP] Yoonsoocraft: selling maxed out aotd with crit 6 party me if interested
Saved screenshot as 2020-02-09_17.37.38.png
[MVP] azuru_el_mejor: selling demonic sword on my ah!!!
Saved screenshot as 2020-02-09_17.37.40.png
11
3
21
3
2

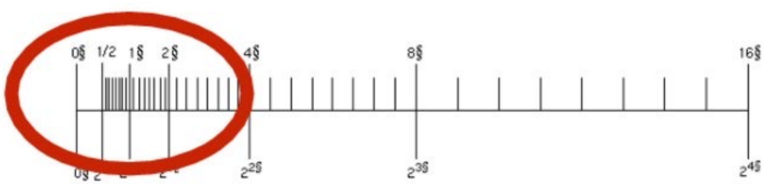
More detailed aside: why near/far plane clipping?

- Primitives (e.g., triangles) may have vertices both in front and behind camera! (Causes headaches for rasterization, e.g., checking if fragments are behind camera
- Avoid divide by zero in perspective divide (near plane clipping)
- Also important for dealing with finite precision of depth buffer

near = 10^{-1}
far = 10^{3}

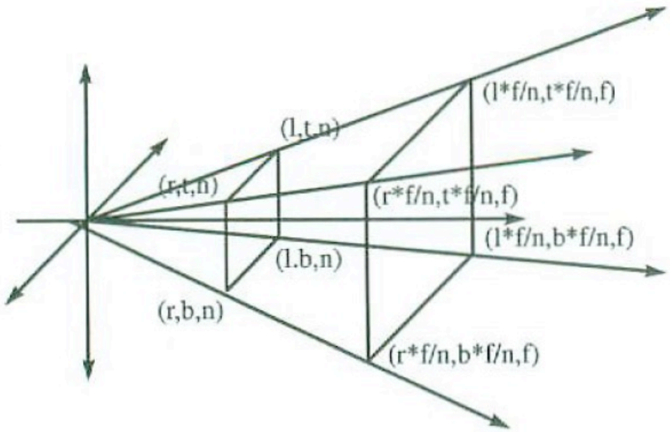
near = 10^{-5}
far = 10^{5}





floating point has more "resolution" near zero—hence more precise resolution of primitive-primitive intersection

Perspective Frustum



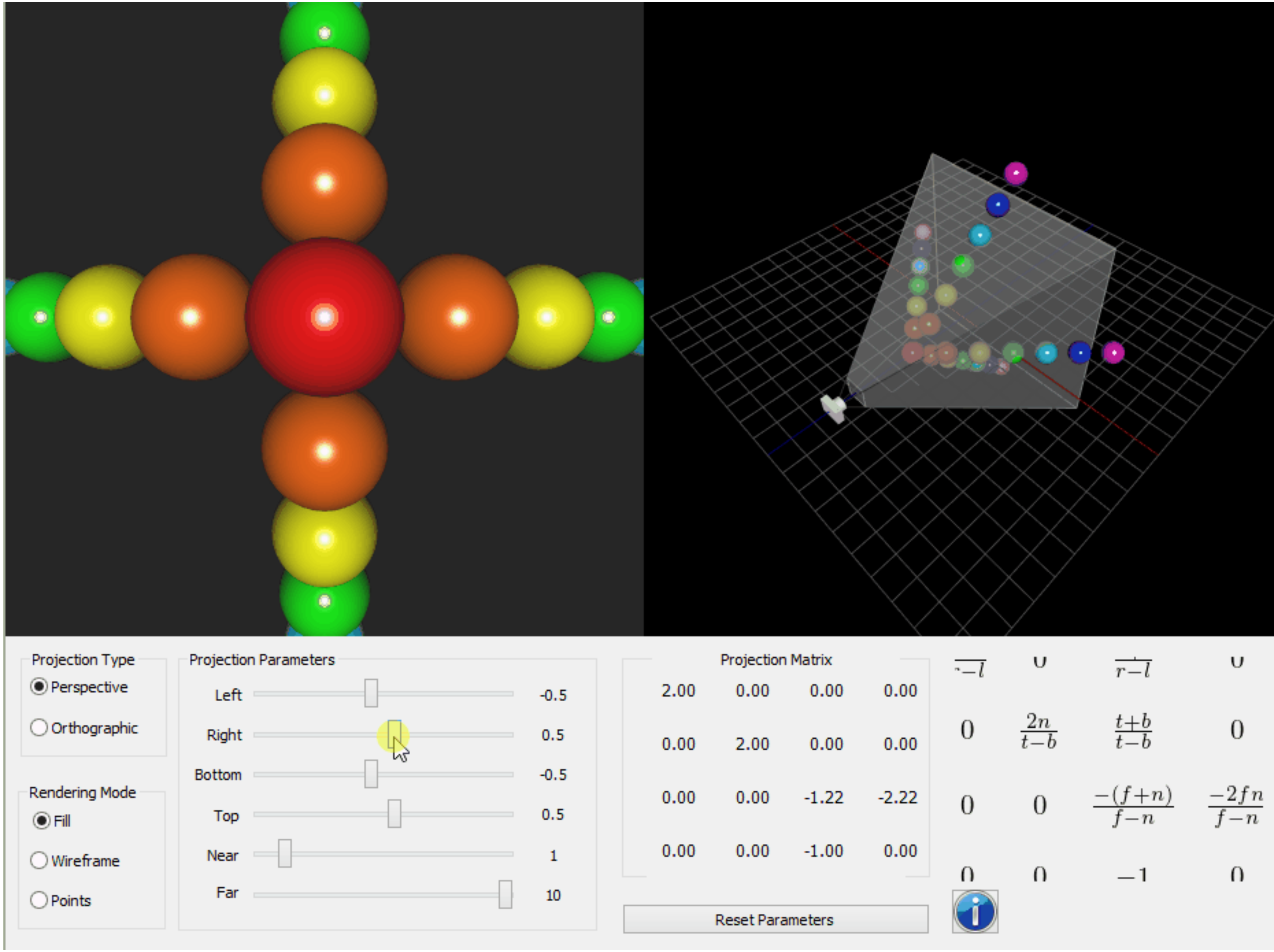
$$\begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

```
glFrustum(float l, r, b, t, n, f);
```

$$\lim_{f\to\infty} -\frac{f+n}{f-n} = -1$$

$$\lim_{f\to\infty} -\frac{2fn}{f-n} = -2n$$

Do we ever want the frustum to be non symmetric for left/right?


Projection Type
Perspective
Orthographic
Rendering Mode
Fill
Wireframe
Points
Projection Parameters
Left -0.5
Right 0.5
Bottom -0.5
Top 0.5
Near 1
Far 10
Projection Matrix
2.00 0.00 0.00 0.00
0.00 2.00 0.00 0.00
0.00 0.00 -1.22 -2.22
0.00 0.00 -1.00 0.00
Reset Parameters
0 $\frac{2n}{t-b}$ $\frac{t+b}{t-b}$ 0
0 0 $\frac{-(f+n)}{f-n}$ $\frac{-2fn}{f-n}$

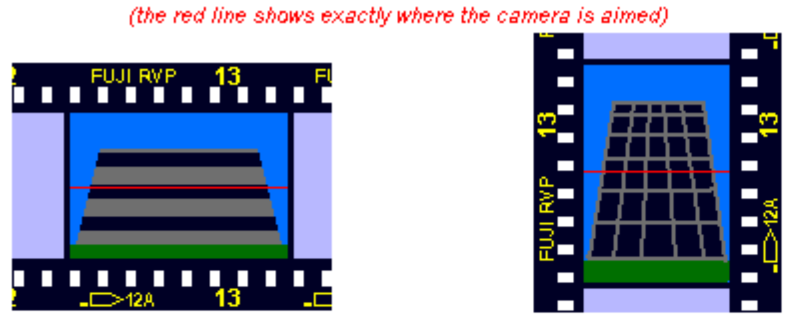


Image 1: Bad Perspective with Normal Lens

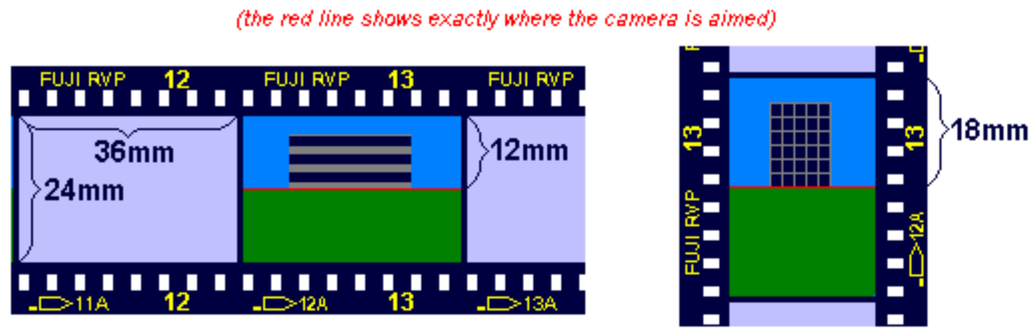


Image 2: Correct Perspective with Normal Lens

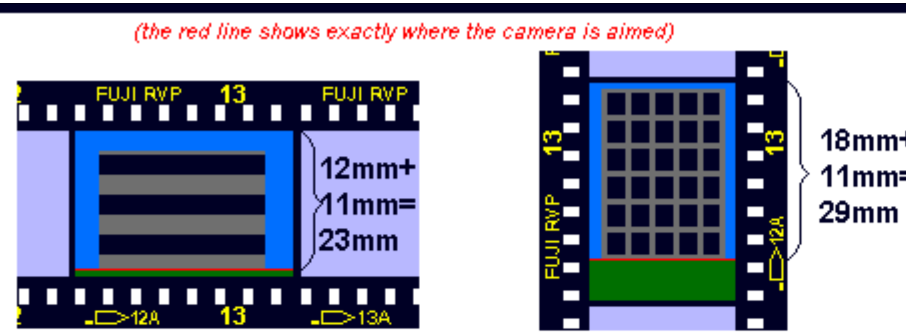
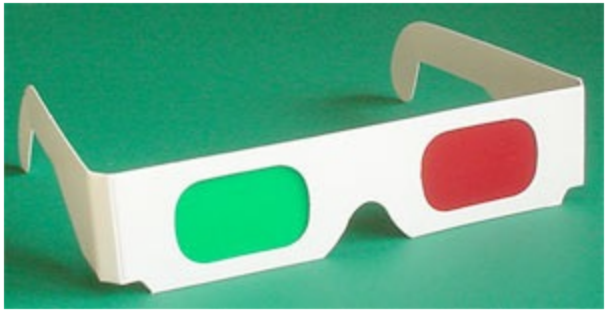
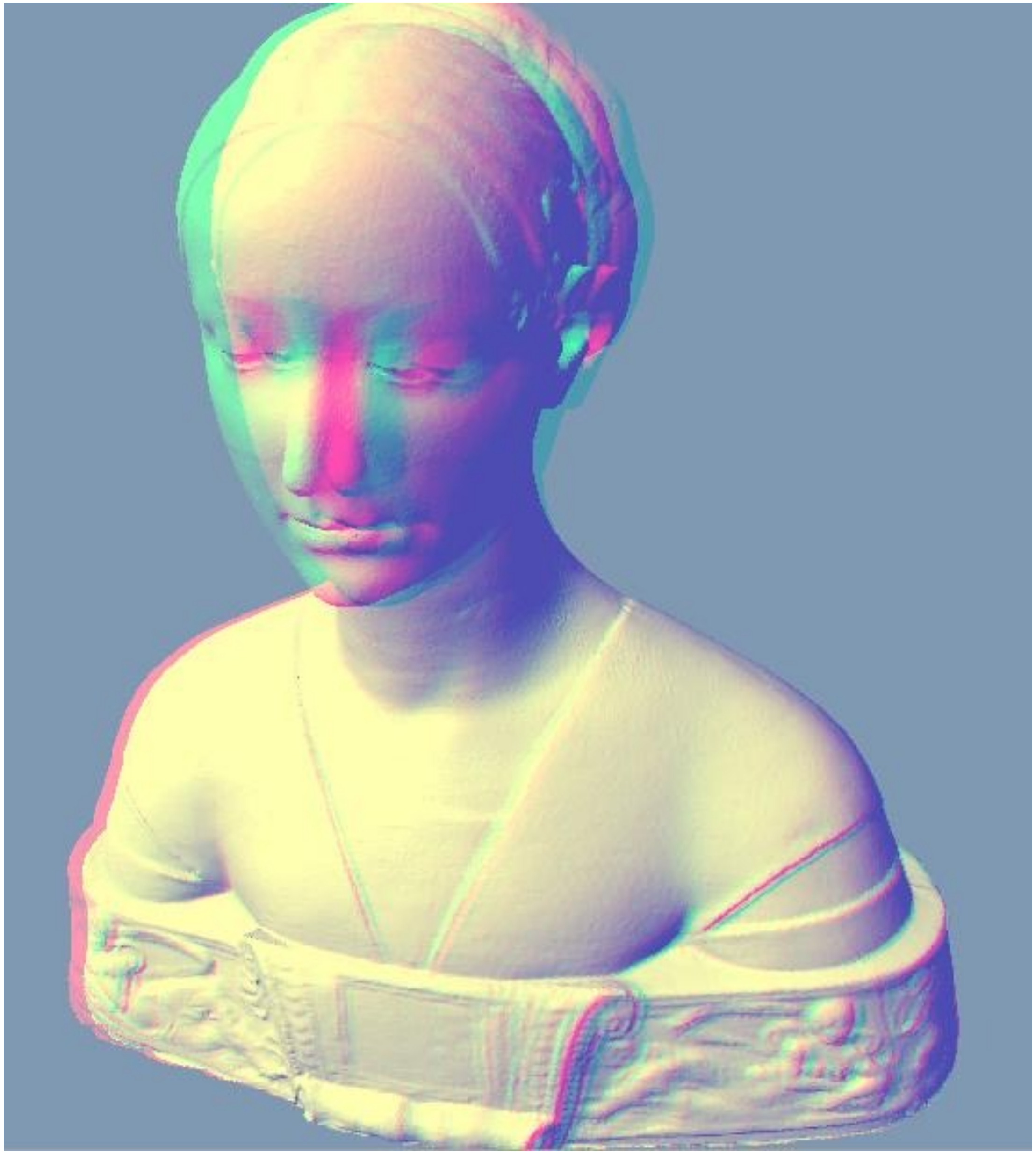
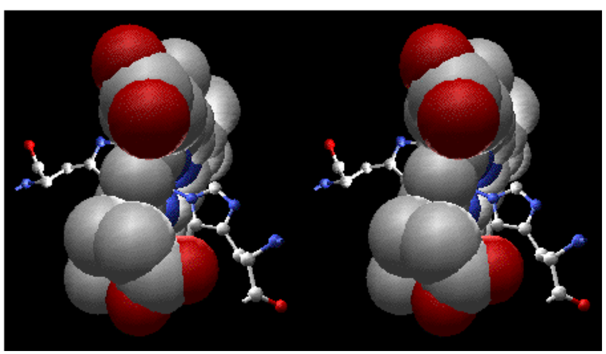
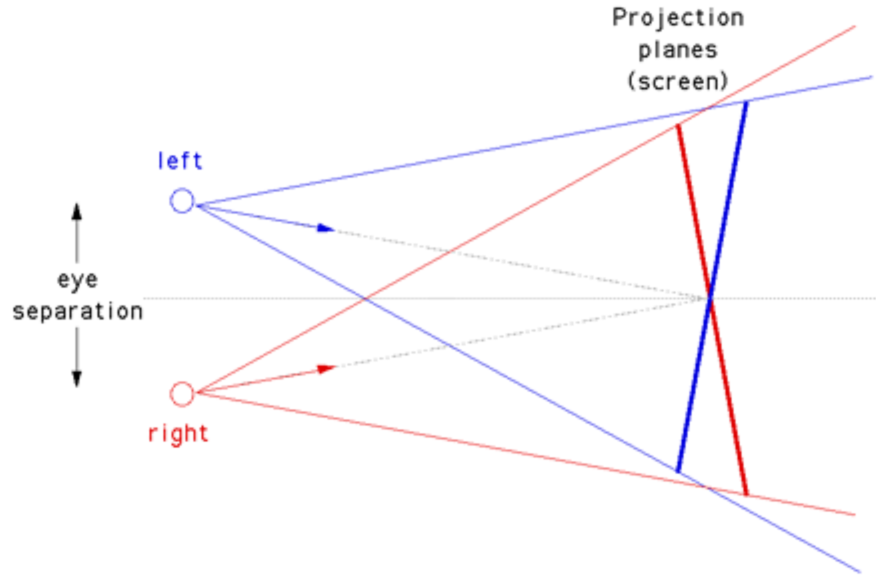


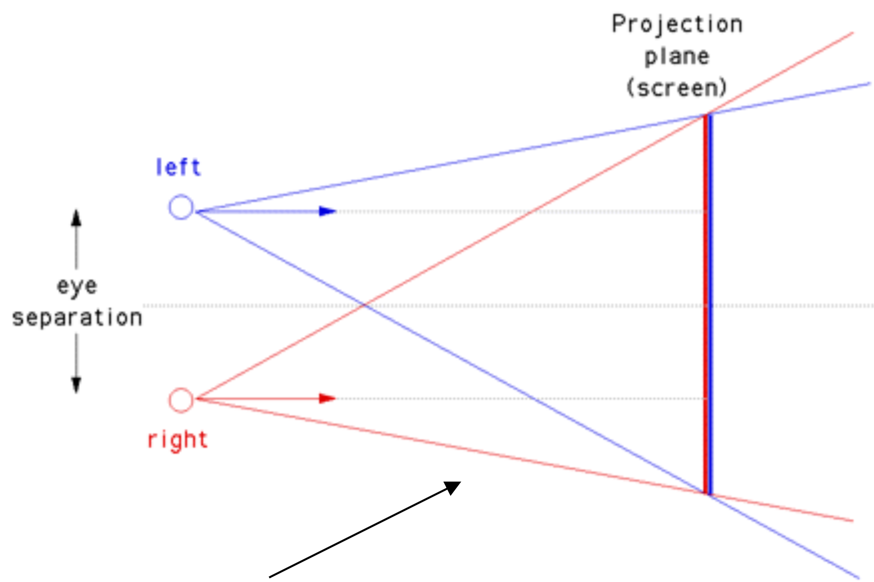
Image 3: Correct Perspective with Shift Lens







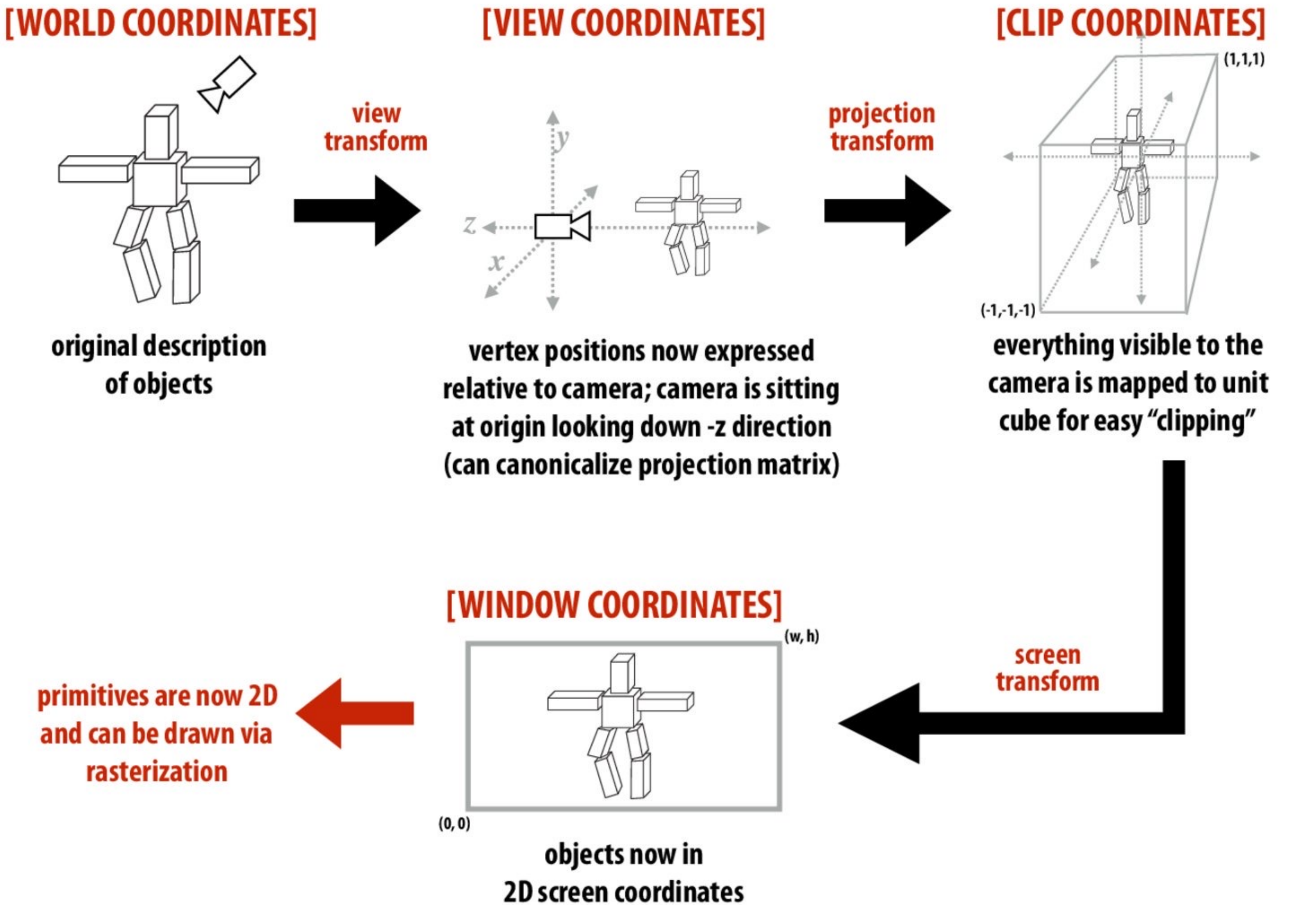




Oblique Perspective Projection

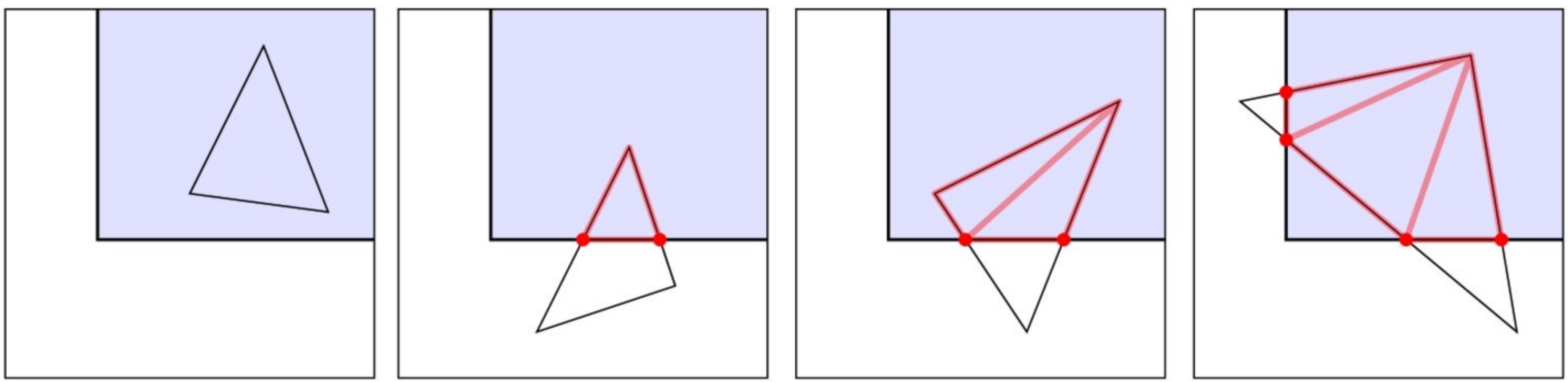
Clipping and Screen Transform

Transformations: from objects to the screen



Clipping

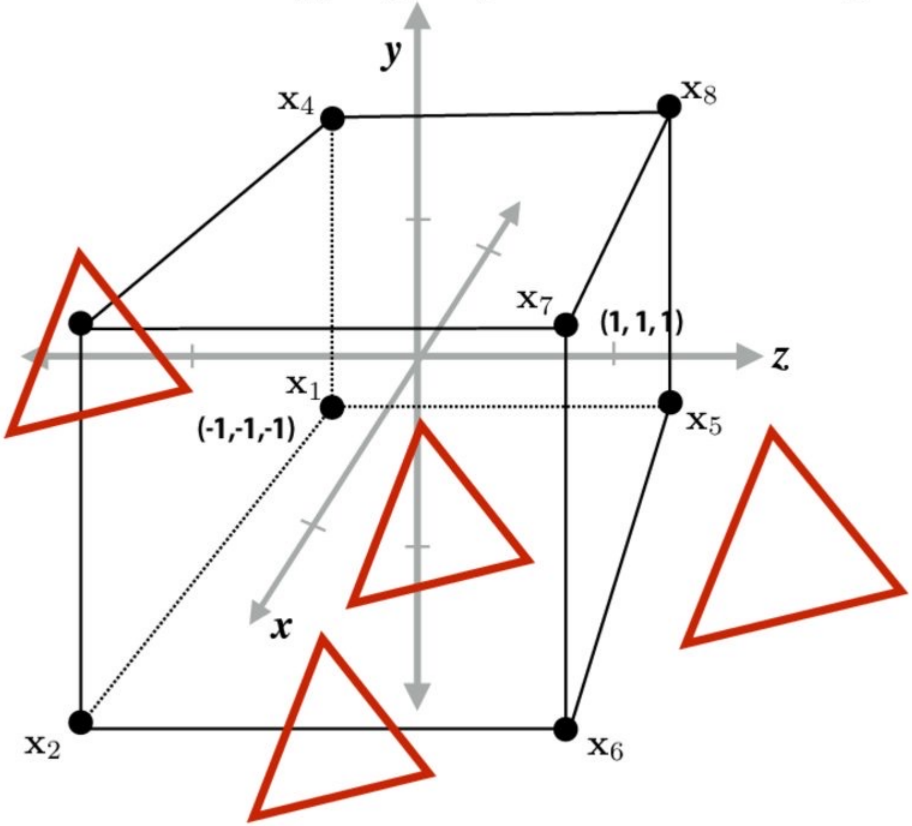
- "Clipping" is the process of eliminating triangles that aren't visible from the camera (because they outside the view frustum)
  - Don't waste time computing appearance of primitives the camera can't see!
  - Sample-in-triangle tests are expensive ("fine granularity" visibility)
  - Makes more sense to toss out entire primitives ("coarse granularity")
  - Must deal with primitives that are partially clipped…



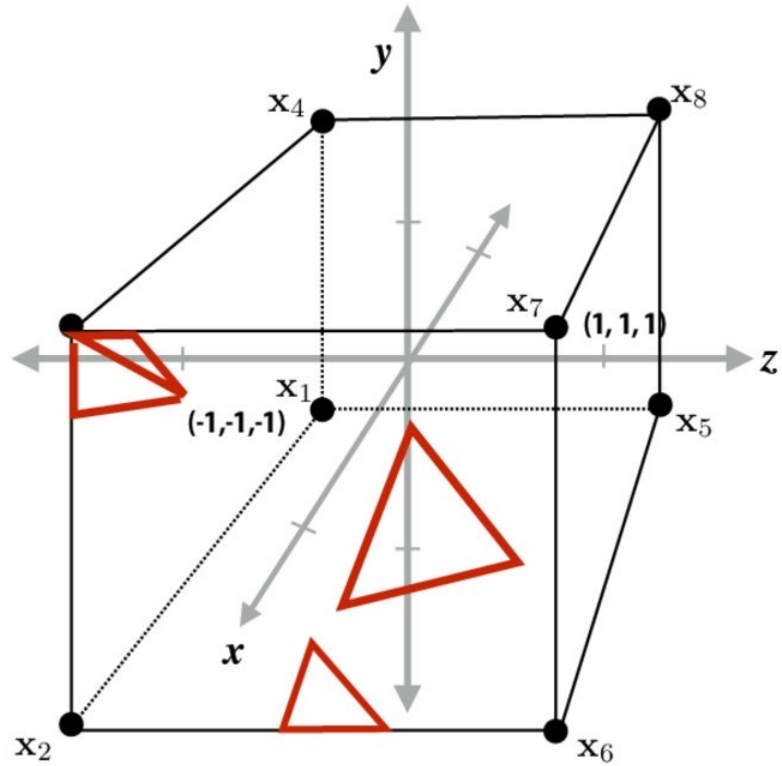
from: https://paroj.github.io/gltut/

Clipping in normalized device coordinates (NDC)

- Discard triangles that lie complete outside the normalized cube (culling)
  - They are off screen, don't bother processing them further
- Clip triangles that extend beyond the cube… to the sides of the cube
  - Note: clipping may create more triangles



Triangles before clipping



Triangles after clipping

* These figures are correct: OpenGL normalized device coordinates is left-handed coordinate space

Review: screen transform

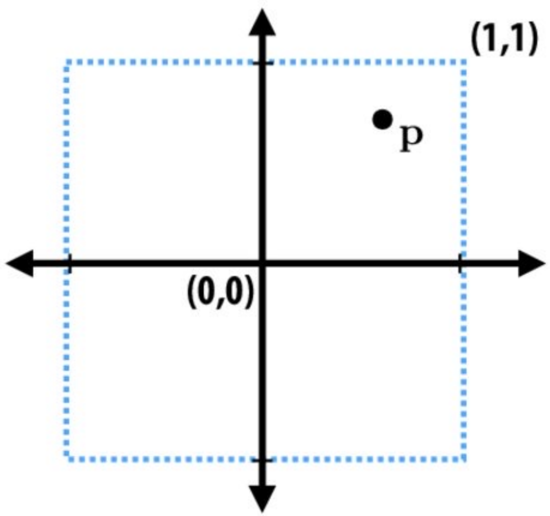
After divide, coordinates in [-1,1] have to be "stretched" to fit the screen

Example:

All points within (-1,1) to (1,1) region are on screen

(1,1) in normalized space maps to (W,0) in screen

Normalized coordinate space:

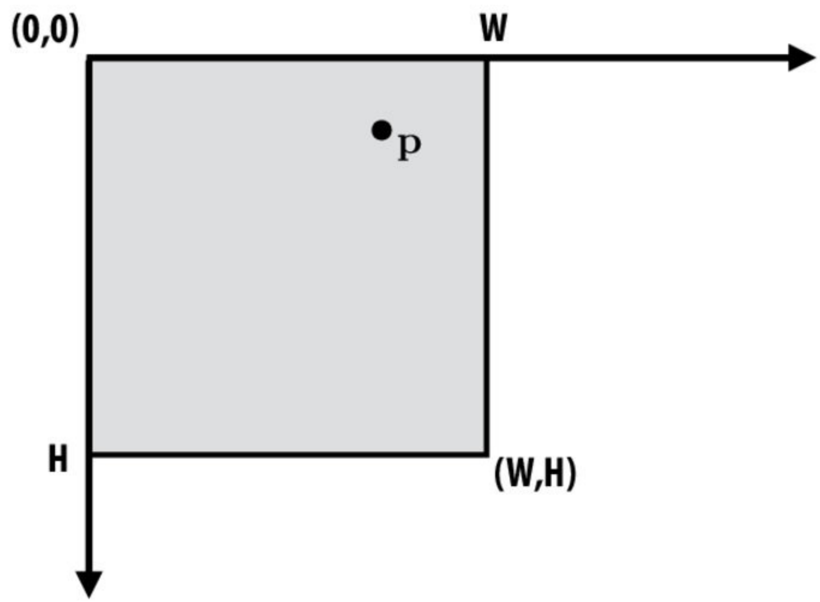


Step 1: reflect about x-axis

Step 2: translate by (1,1)

Step 3: scale by (W/2,H/2)

Screen (W x H output image) coordinate space:



WebGL

Listing 7.8 PerspectiveView.js

```
// PerspectiveView.js
// Vertex shader program
var VSHADER_SOURCE =
  'attribute vec4 a_Position;\n' +
  'attribute vec4 a_Color;\n' +
  'uniform mat4 u_ViewMatrix;\n' +
  'uniform mat4 u_ProjMatrix;\n' +
  'varying vec4 v_Color;\n' +
  'void main() {\n' +
  '  gl_Position = u_ProjMatrix * u_ViewMatrix * a_Position;\n' +
  '  v_Color = a_Color;\n' +
  '}\n';
 ...
function main() {
 ...
  // Set the vertex coordinates and color (blue triangle is in front)
  var n = initVertexBuffers(gl);
 ...
  // Get the storage locations of u_ViewMatrix and u_ProjMatrix
  varu_ViewMatrix = gl.getUniformLocation(gl.program,'u_ViewMatrix');
  var u_ProjMatrix = gl.getUniformLocation(gl.program,'u_ProjMatrix');
 ...
  var viewMatrix = new Matrix4();  // The view matrix
  var projMatrix = new Matrix4();  // The projection matrix

  // Calculate the view and projection matrix
  viewMatrix.setLookAt(0, 0, 5, 0, 0, -100, 0, 1, 0);
  projMatrix.setPerspective(30, canvas.width/canvas.height, 1, 100);
  // Pass The view matrix and projection matrix to u_ViewMatrix and u_ProjMatrix
  gl.uniformMatrix4fv(u_ViewMatrix, false, viewMatrix.elements);
  gl.uniformMatrix4fv(u_ProjMatrix, false, projMatrix.elements);
 ...
  // Draw the rectangles
  gl.drawArrays(gl.TRIANGLES, 0, n);
}
```

u_ModelMatrix

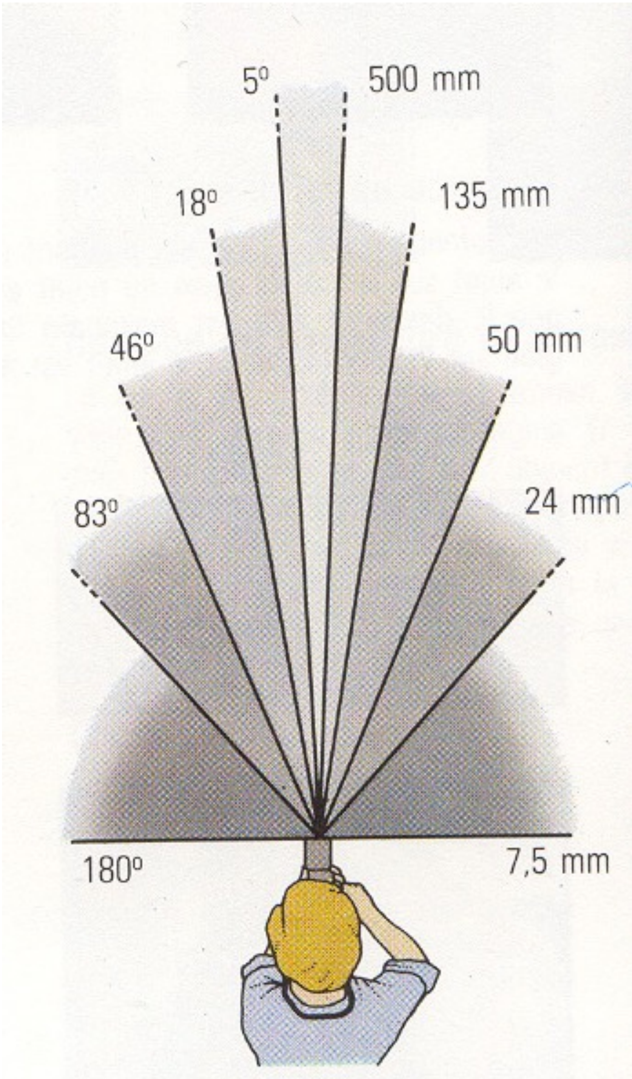
Part of Matrix class

setLookAt(eye, at, up)

setPerspective(fov, aspect, near, far)

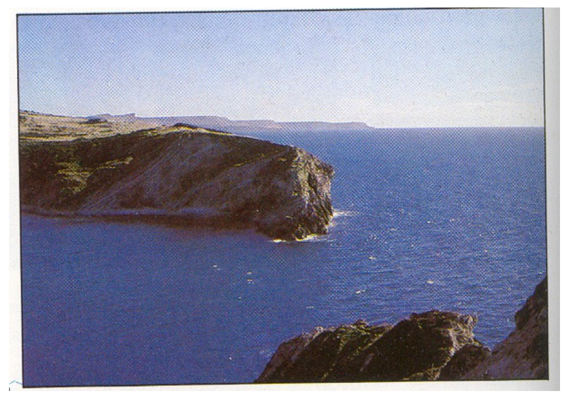
Graphics vs Real Cameras

Lenses



24mm

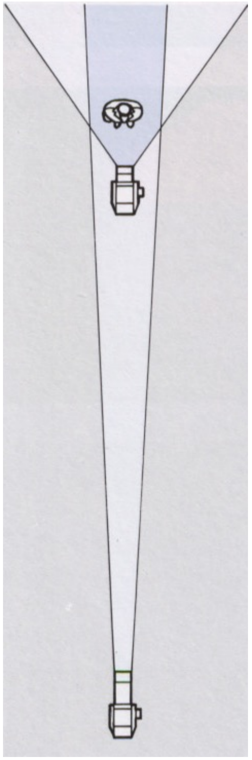
50mm



135mm

Perspective vs. viewpoint

- **Focal lens does NOT ONLY change subject size**
- **Same size by moving the viewpoint**
- **Different perspective (e.g. background)**



Perspective vs. viewpoint

- Portrait: distortion with wide angle
- Why?

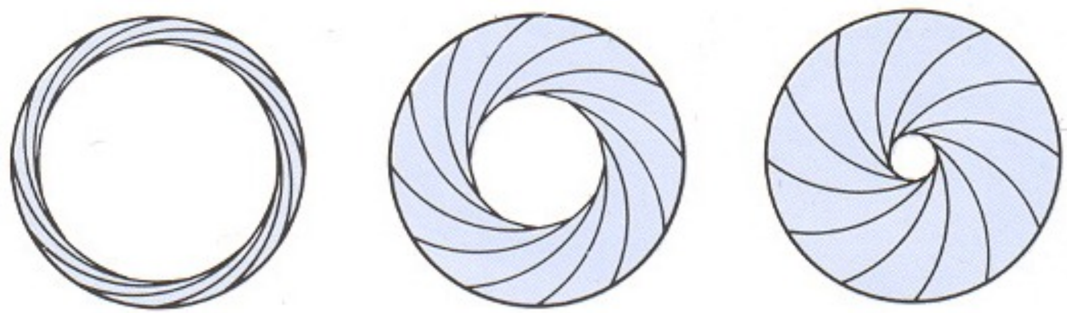
Wide angle

Standard

Telephoto

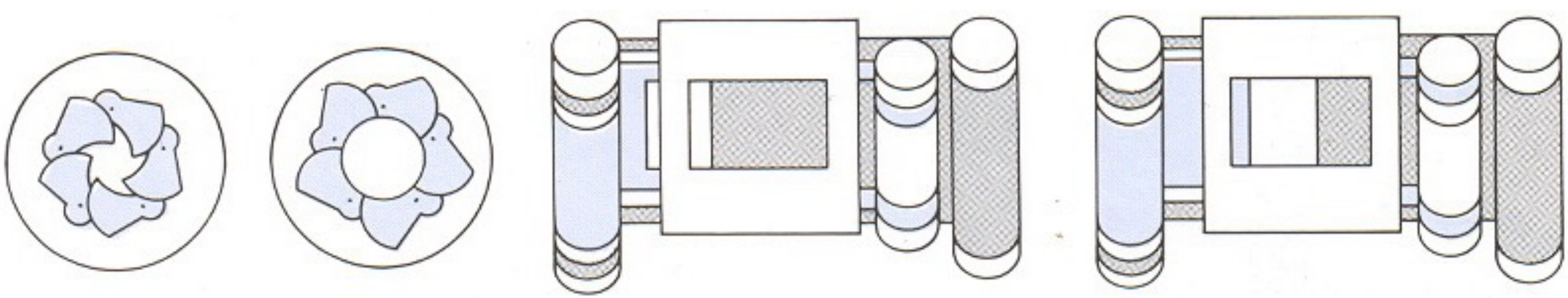
Exposure

- **Two main parameters:**
  - Aperture (in f stop)



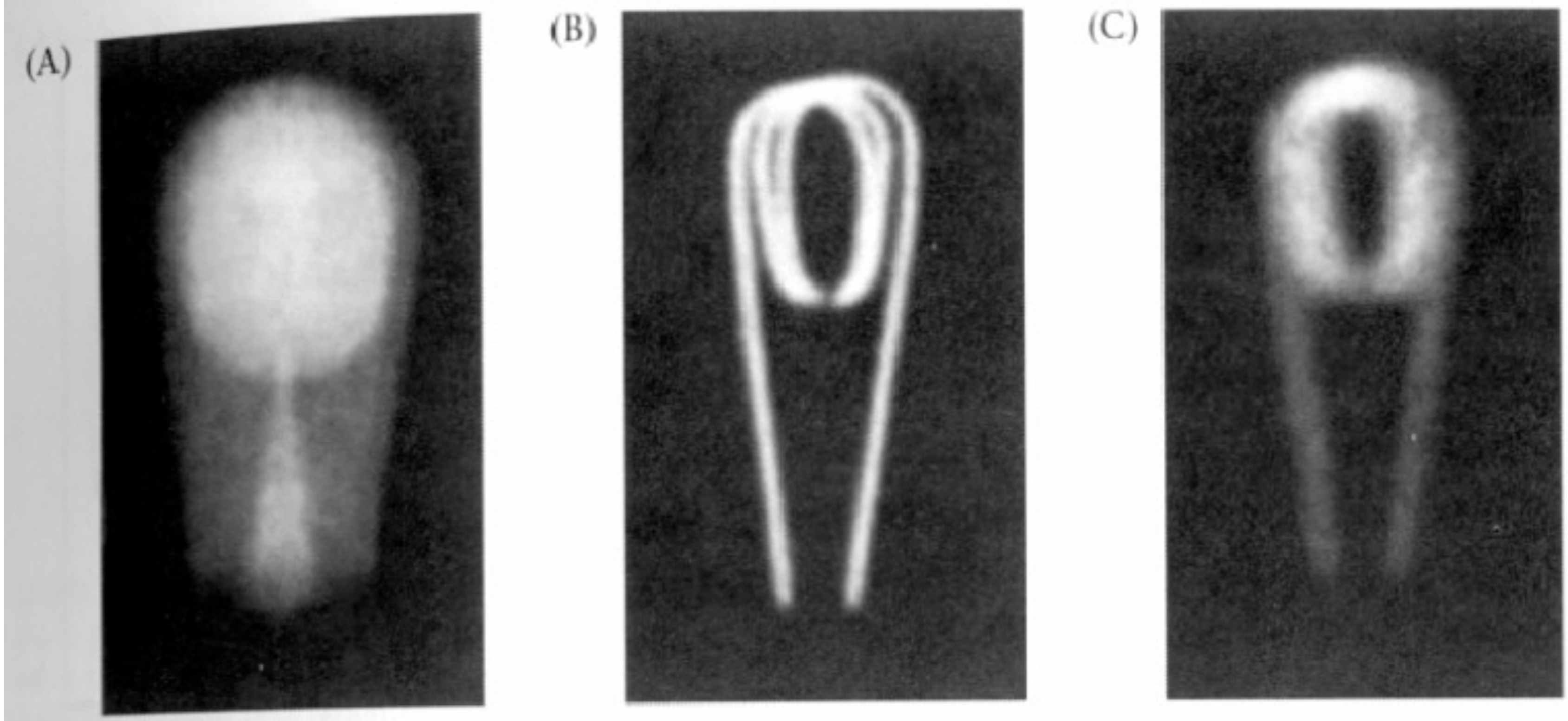
Full aperture Medium aperture Stopped down

  - Shutter speed (in fraction of a second)



Blade (closing) Blade (open) Focal plane (closed) Focal plane (open)

Pinhole limit



2.18 DIFFRACTION LIMITS THE QUALITY OF PINHOLE OPTICS. These three images of a bulb filament were made using pinholes with decreasing size. (A) When the pinhole is relatively large, the image rays are not properly converged, and the image is blurred. (B) Reducing the size of the pinhole improves the focus. (C) Reducing the size of the pinhole further worsens the focus, due to diffraction. From Ruechardt, 1958.

Administrative

Q&A

End