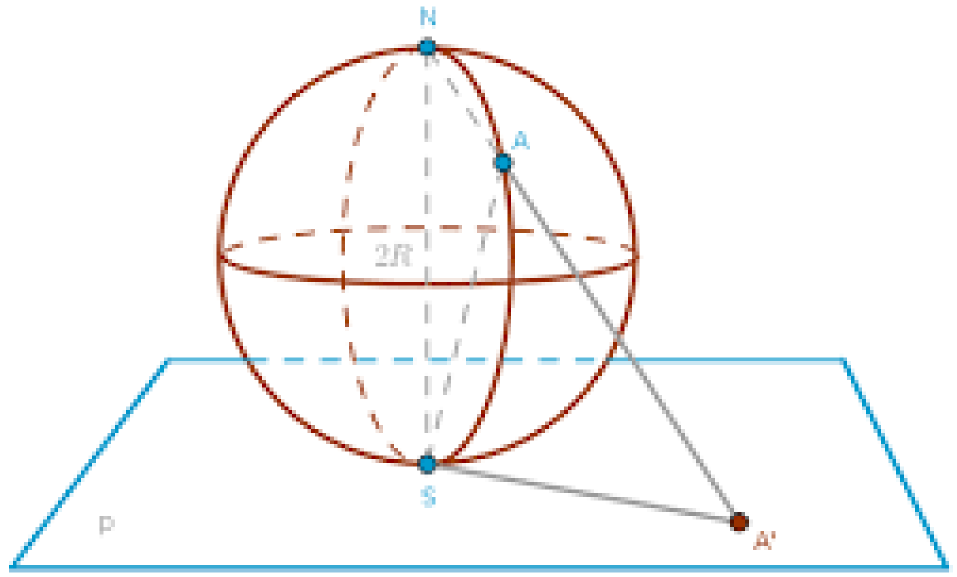
Orthographic Projection

CS 418: Interactive Computer Graphics

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Eric Shaffer

Projections



In computer graphics,
eventually we need to move from 3D space to 2D space

More accurately:
from 4D homogeneous coordinates to 3D homogeneous coordinates

A ***projection***
is a transformation that maps from a high-dimensional space into a lower-dimensional space.

We will look at some common projections…and then we will discuss projection within WebGL

Planar Geometric Projections

Standard projections project onto a plane

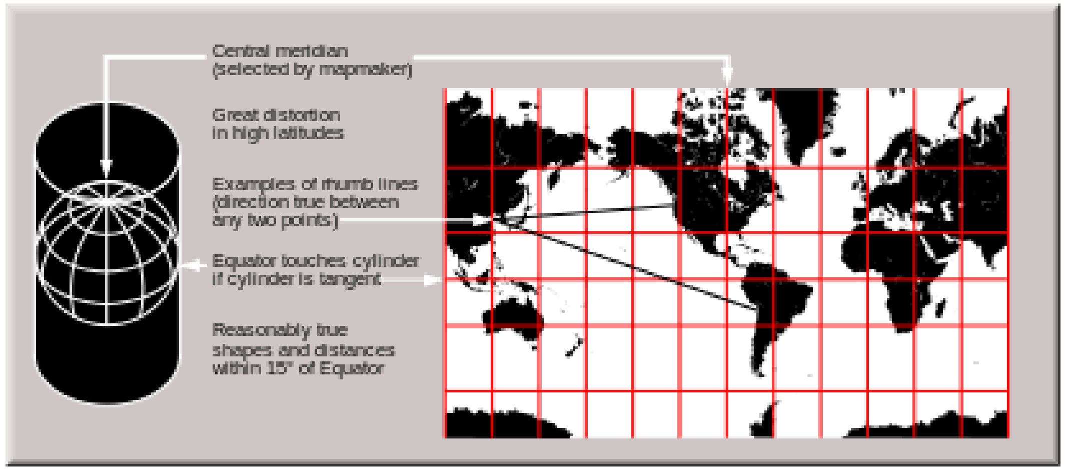
Projectors are lines that either

- converge at a center of projection
- are parallel

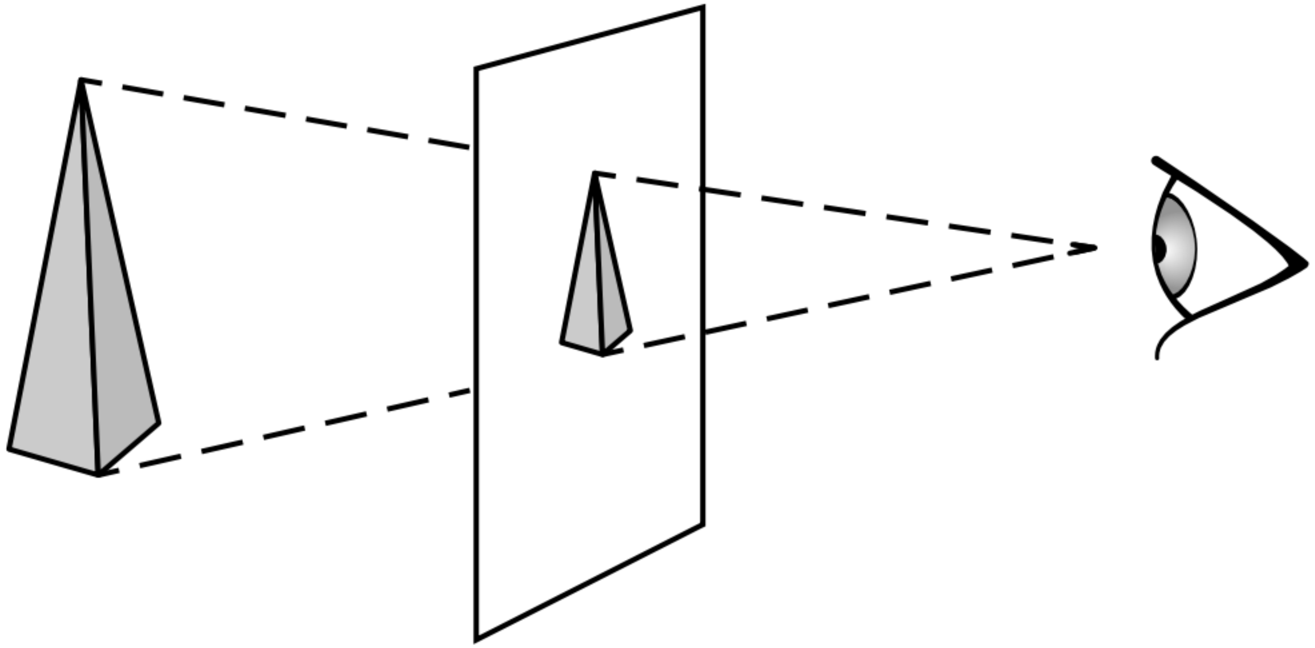
Such projections preserve lines

- but not necessarily angles

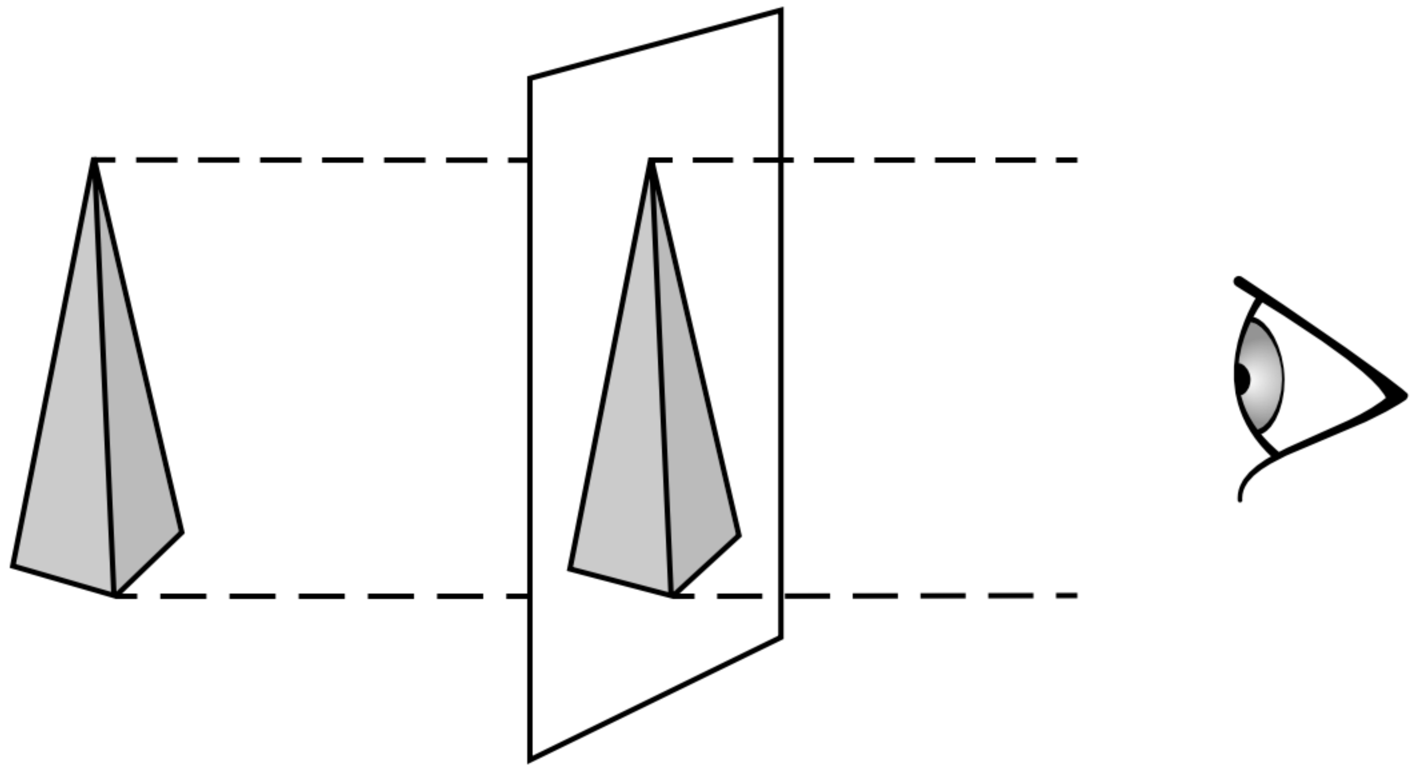
Non-planar projections are often used in map construction


Central meridian (selected by mapmaker)
Great distortion in high latitudes
Examples of rhumb lines (direction true between any two points)
Equator touches cylinder if cylinder is tangent
Reasonably true shapes and distances within 15° of Equator

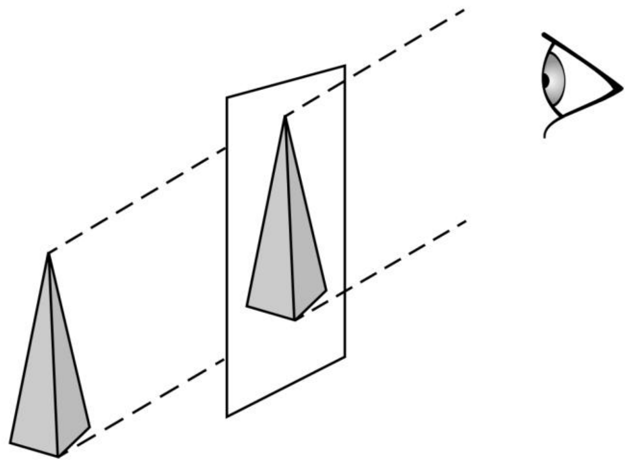
Perspective Projection

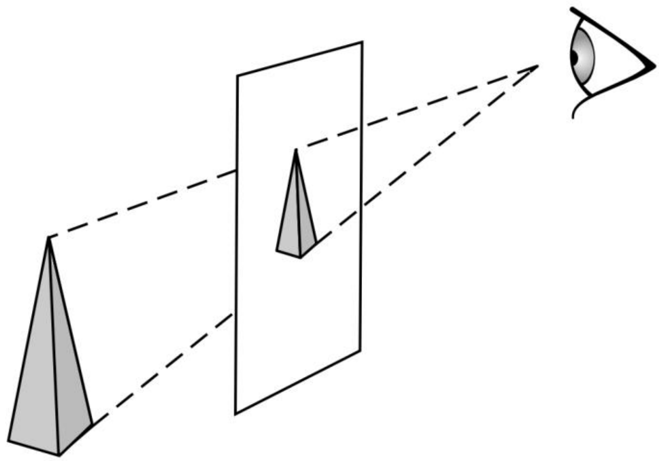


Orthographic Projection



Oblique Projections

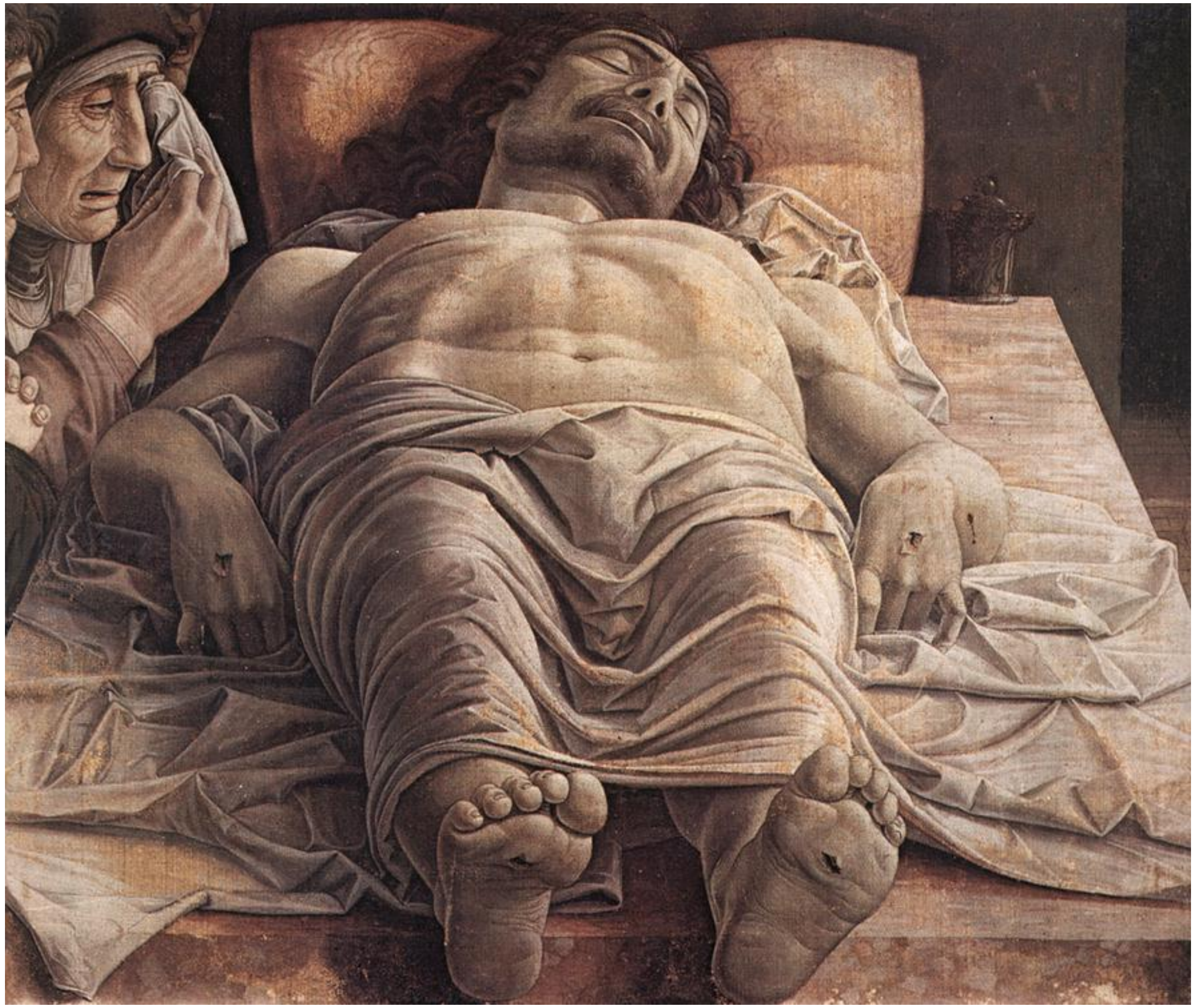

Oblique parallel projection


Oblique perspective projection

Linear projections can be categorized
- By whether the projectors are parallel
- By whether the projectors are orthogonal to the view plane

We will not use these…..

Definition to Know: Foreshortening



Foreshortening is the visual effect or optical illusion that causes an object or distance to appear shorter than it actually is because it is angled toward the viewer.

i.e. projections squash receding surfaces

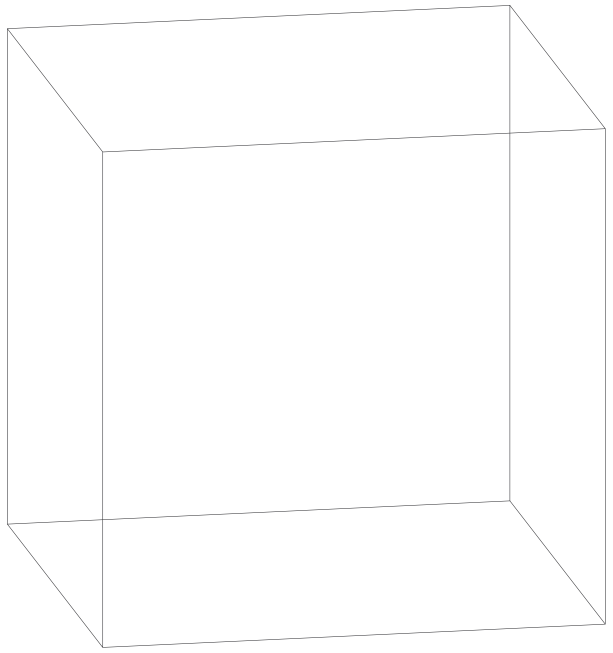
Can foreshortening happen in orthographic projection?

Andrea Mantegna
The Lamentation over the Dead Christ

Clip Space View Volume

Before looking at projections in more detail…
…let’s experimentally determine what the default view and view volume are.

So, if we don’t do any transformations…where are we and what direction are we looking?



Take a look at
https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API/WebGL_model_view_projection

http://jsfiddle.net/2x03hdc8/

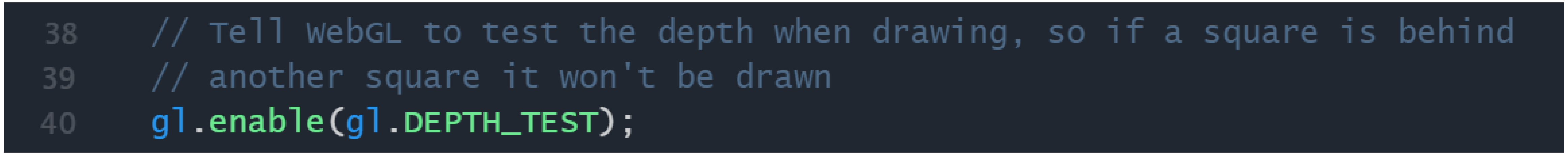
It is a simple WebGL program that lets you draw rectangles

No transformations

By altering the rectangle coordinates, you can figure out the view….

Clip Space View Volume

- Something new – the code enables hidden surface removal



```
// Tell WebGL to test the depth when drawing, so if a square is behind
// another square it won't be drawn
gl.enable(gl.DEPTH_TEST);
```

Side note – the overall code uses different conventions than used in our course
(e.g. no glmatrix library)

So, probably don't base your assignments on this code

Hidden Surface Removal

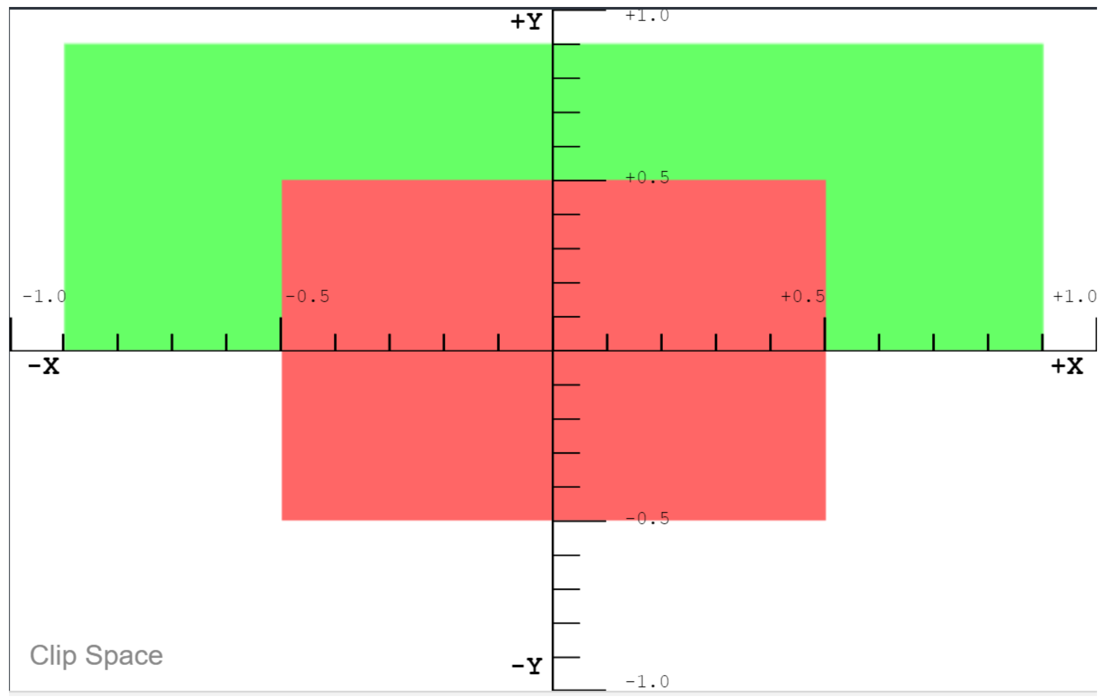
If you are animating, you need to re-initialize the depth buffer each frame

```
/**
 * Draw call that applies matrix transformations to model and draws model in frame
 */
function draw() {
  gl.viewport(0, 0, gl.viewportWidth, gl.viewportHeight);
  gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);

  glMatrix.mat4.identity(mvMatrix);
  glMatrix.mat4.identity(pMatrix);


  gl.bindBuffer(gl.ARRAY_BUFFER, vertexPositionBuffer);
  gl.vertexAttribPointer(shaderProgram.vertexPositionAttribute,
                         vertexPositionBuffer.itemSize, gl.FLOAT, false, 0, 0);
  gl.bindBuffer(gl.ARRAY_BUFFER, vertexColorBuffer);
  gl.vertexAttribPointer(shaderProgram.vertexColorAttribute,
                           vertexColorBuffer.itemSize, gl.FLOAT, false, 0, 0);

  setMatrixUniforms();
  gl.drawArrays(gl.TRIANGLE_FAN, 0, vertexPositionBuffer.numberOfItems);
}
```

Clip Space View

- Red rectangle has z=0
- Green has z=0.5

What direction are we looking?

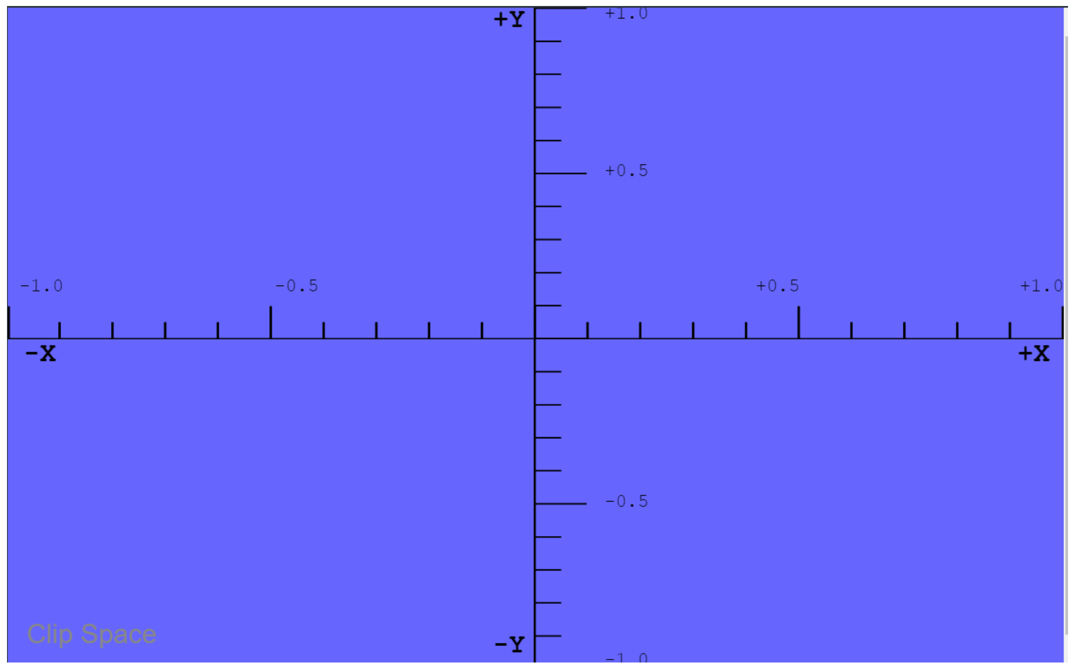


Clip Space View

We add a blue rectangle

It's at z=-1

Where is the near clipping plane?



Clip Space View and View Volume

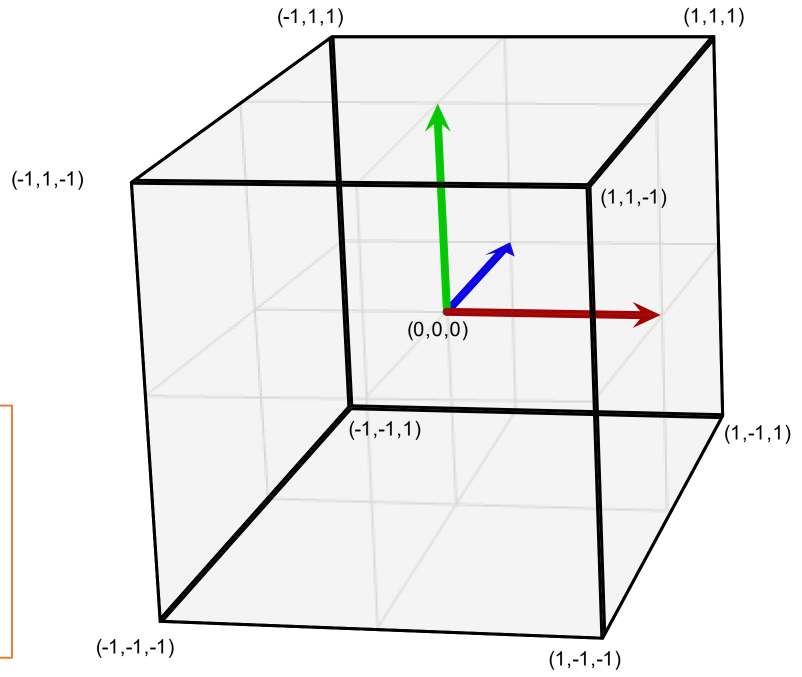
View is looking down Z+

Eyepoint is effectively at (0,0,-1)

This is a left-handed coordinate system

This is a little-known fact….
People often learn to use WebGL without ever learning this.

We'll see why in a few slides

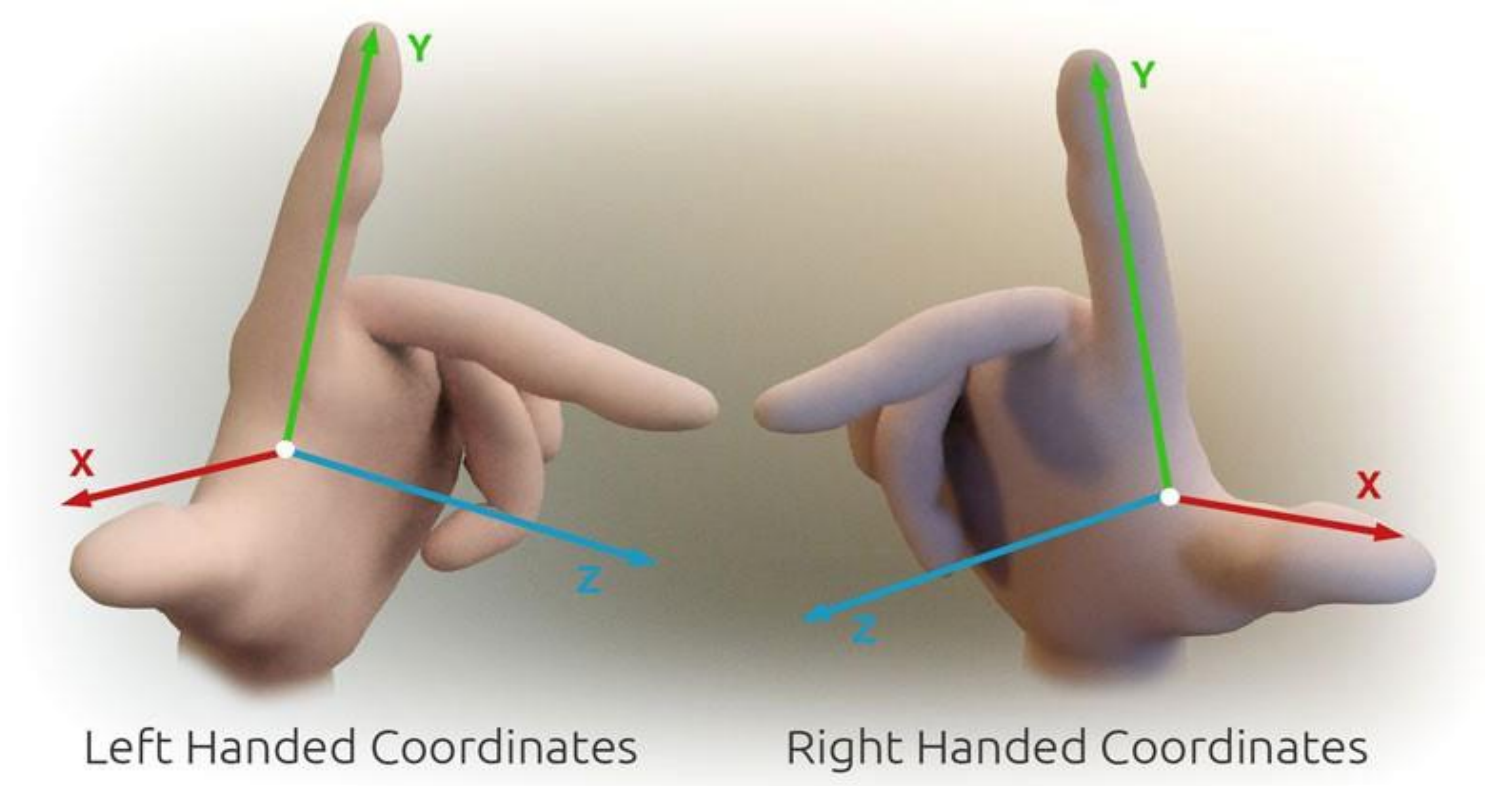


Clipspace

WebGL View Volume

The default view volume is called ***clip space***

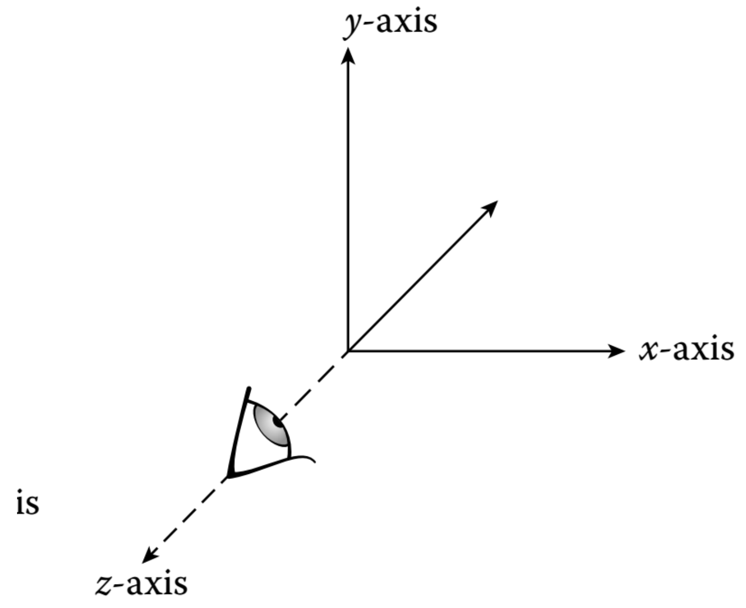
WebGL clip space is a ***left-handed coordinate system***

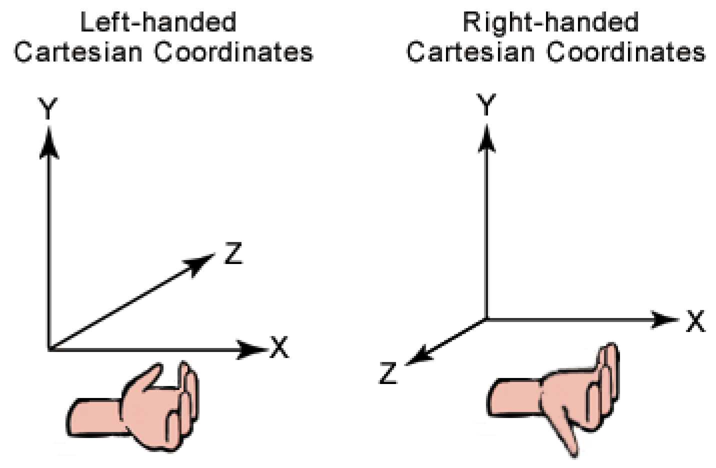


Many (most?) programmers don't realize this…they think WebGL is a right-handed system…this is because the tradition is to model in right-handed system looking down the –Z axis and then use a matrix transformation to invert the Z axis….

Which is +Z in relation to +X and +Y ?

WebGL Style View

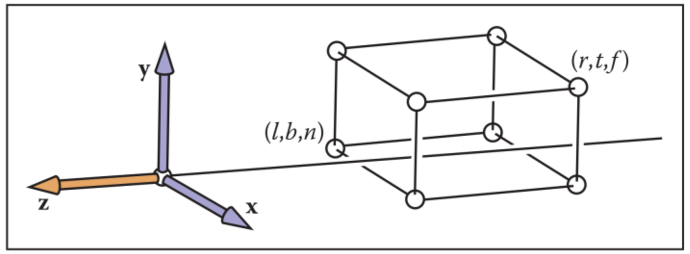


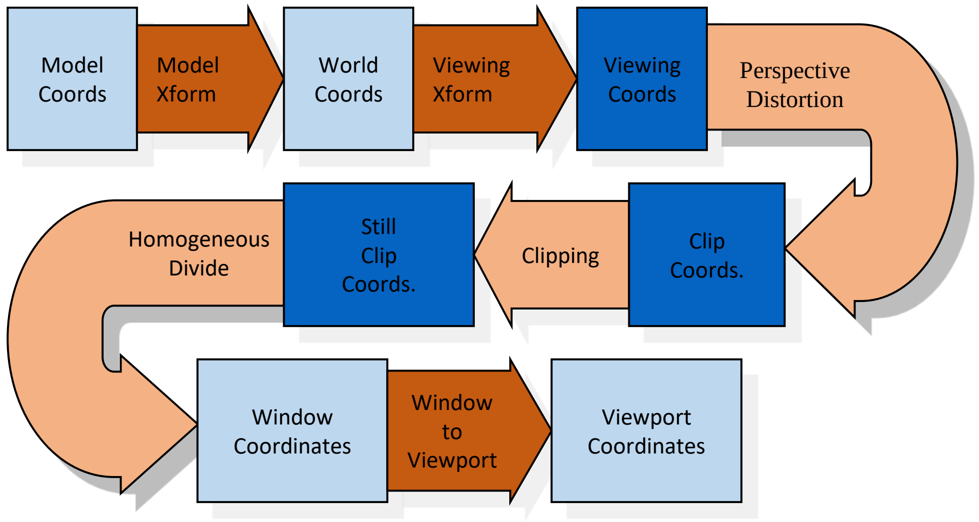

Left-handed Cartesian Coordinates
Right-handed Cartesian Coordinates
Y
Z
X
Y
X
Z

WebGL/OpenGL convention is to assume a right-handed world coordinate system

The ***ortho*** matrix is used to flip this coordinate system by scaling Z by -1
It then matches WebGL clip space

More WebGL Secrets





WebGL only performs an orthogonal projection

- Everything is projected to the z=0 plane in the normalized view volume
- **But you can distort your geometry to achieve a perspective projection**

The projection occurs when the geometry is in clip space (NDC)

- Even then, depth information is kept around to do hidden surface removal
- Depth information means transformed z coordinates

Orthographic Projection

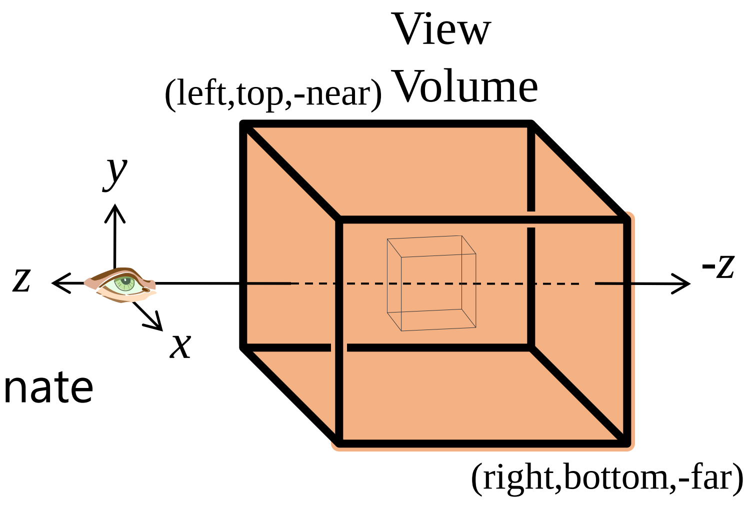
- Foreshortens
- No change in size by depth
- Classic Orthographic Projection matrix simply zeros the z- coordinate

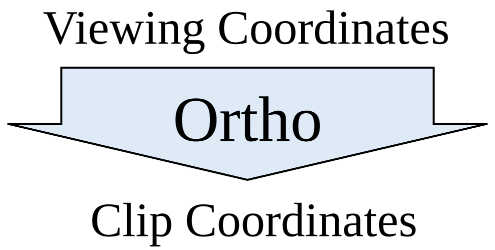
$$\begin{bmatrix} & & & \\ & \text{W2V} & & \\ & & & \end{bmatrix}\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 0 & \\ & & & 1 \end{bmatrix}\begin{bmatrix} & & & \\ & \text{View} & & \\ & & & \end{bmatrix}\begin{bmatrix} & & & \\ & \text{Model} & & \\ & & & \end{bmatrix}$$

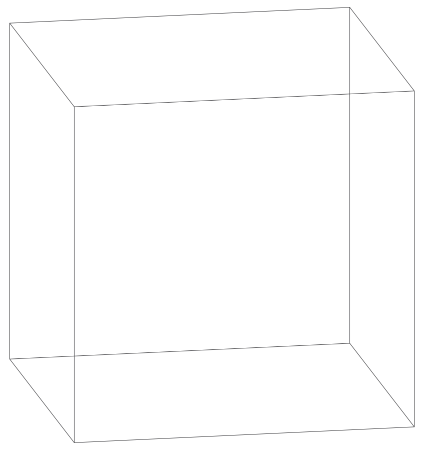
- mat4.ortho(out,left,right,bottom,top,near,far)

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

Does the ortho matrix perform a projection?



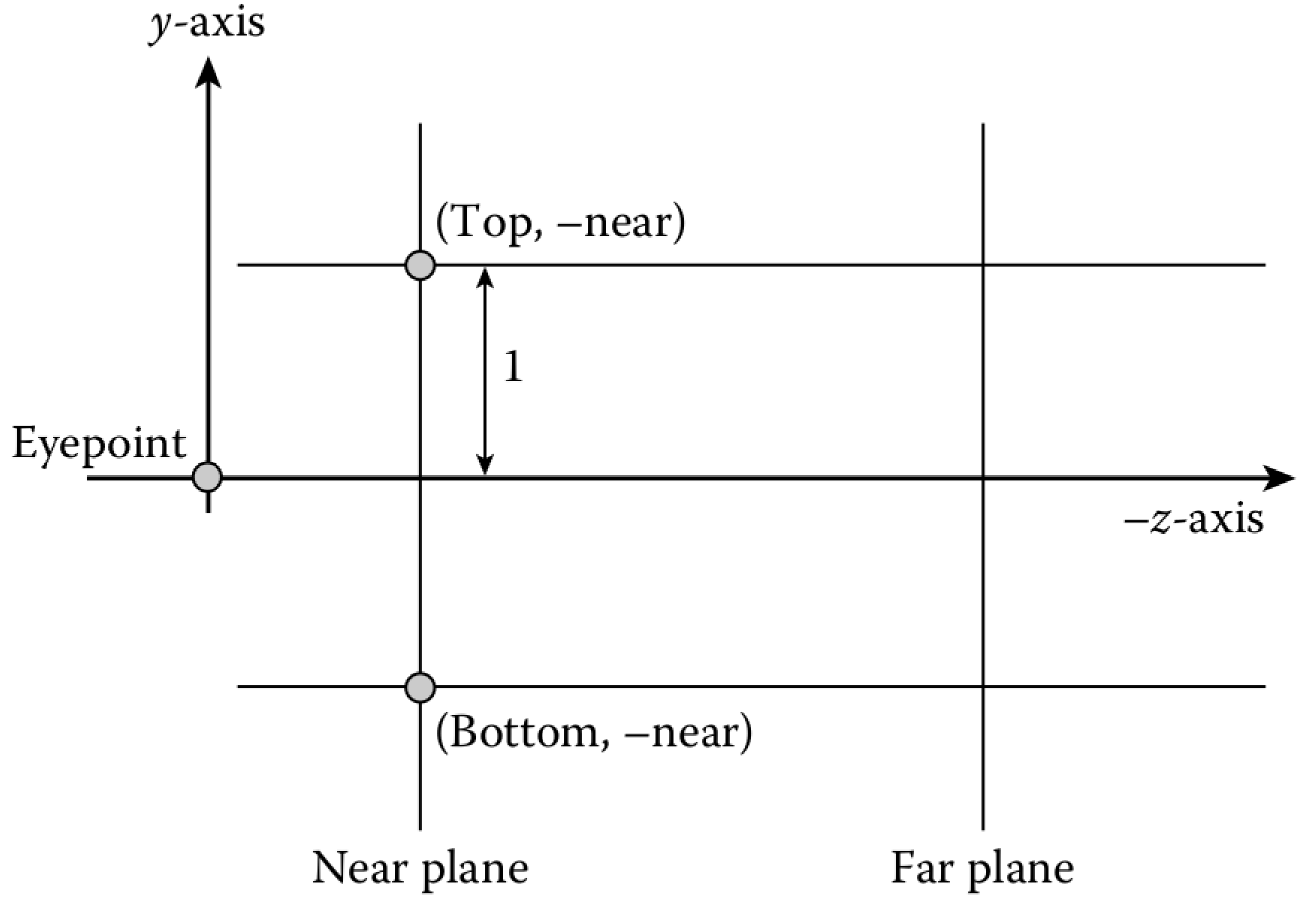




GLMatrix ortho matrix

```
ortho(left,right,bottom,top,near,far)
```

- **near** and **far** **are distances down the –z axis from origin**
- l,r,b,t are coordinates of the bounding planes
- what does the matrix do?



GLMatrix ortho matrix

```
ortho(left,right,bottom,top,near,far)
```

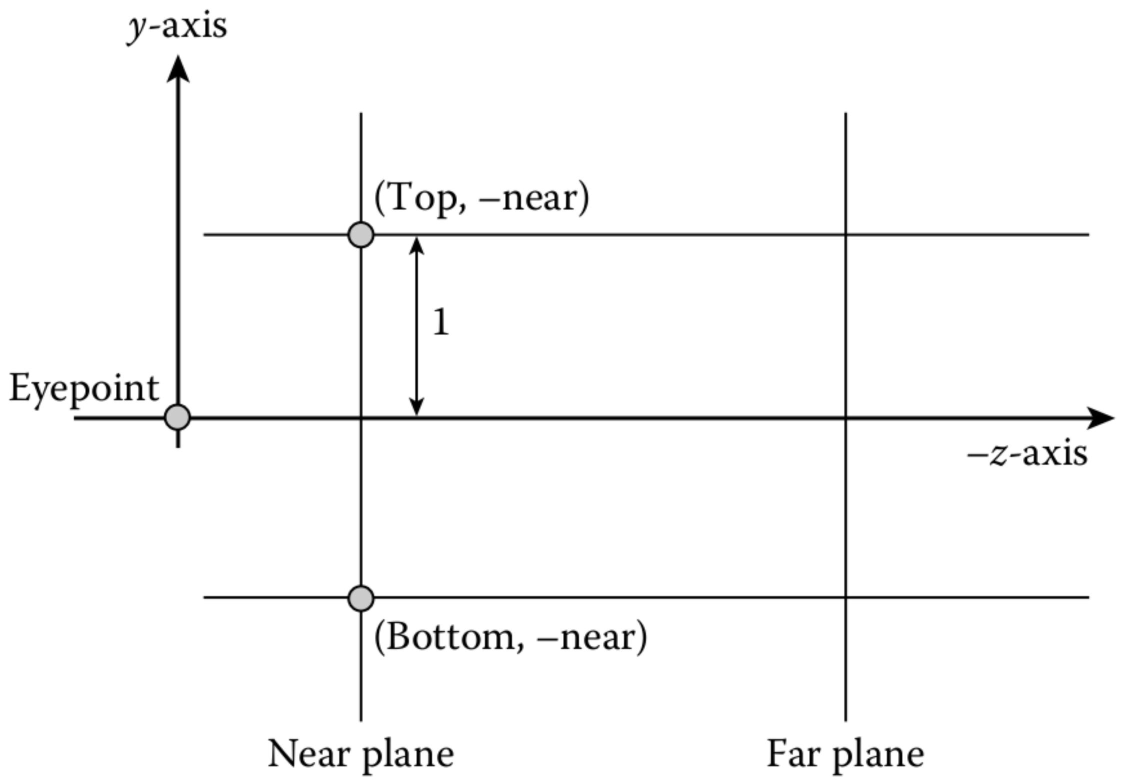
Imagine the eye is at (0,0,0)
We look down the –z axis

The view volume is:
 [left, right]   =[-1, 1]
 [bottom, top]=[-1, 1]

near =0

far   =2

What does the matrix look like?



GLMatrix ortho matrix

`ortho(left,right,bottom,top,near,far)`

Imagine the eye is at (0,0,0)
We look down the –z axis

The view volume is:
[left, right] =[-1, 1]
[bottom, top]=[-1, 1]

near =0

far =2

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where does (0,0,0) get moved to?
What else happens?