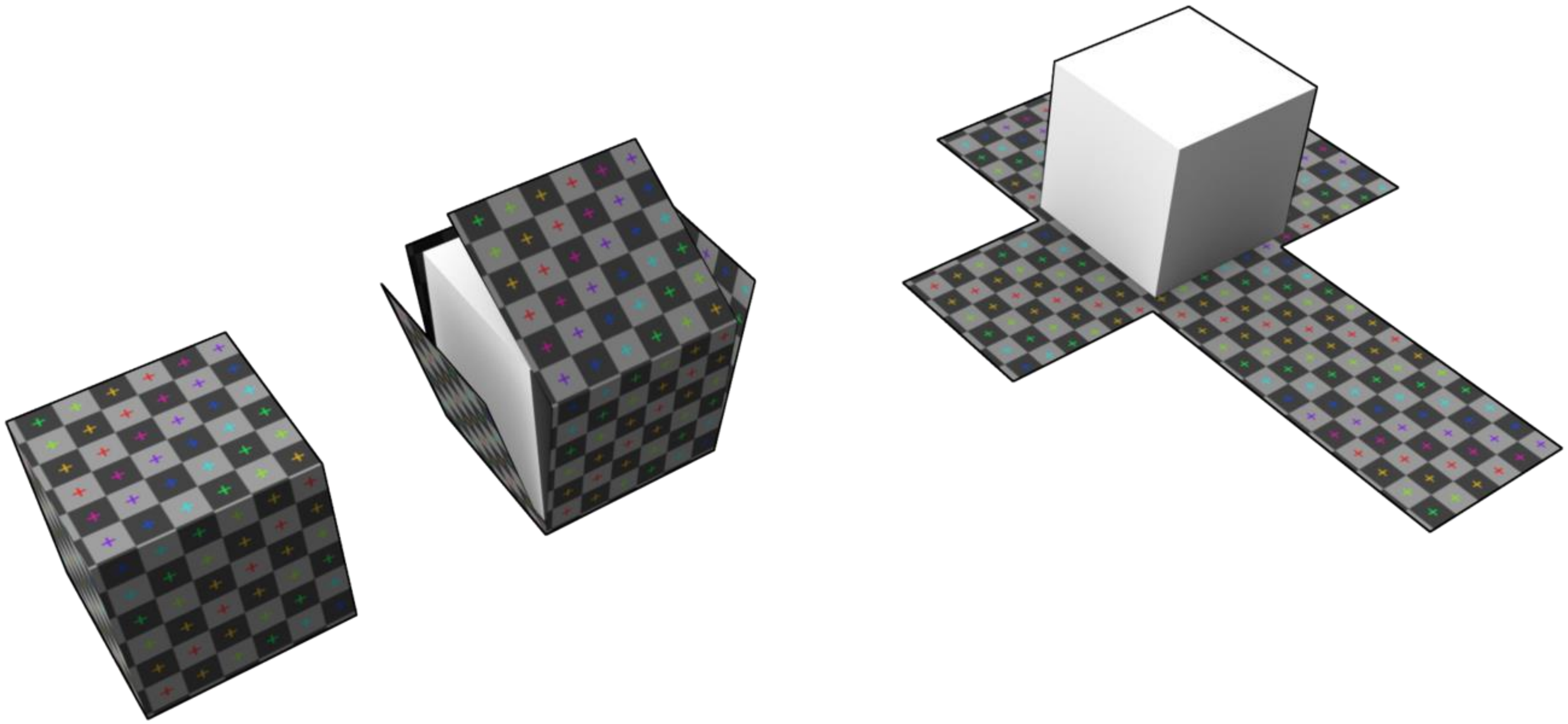
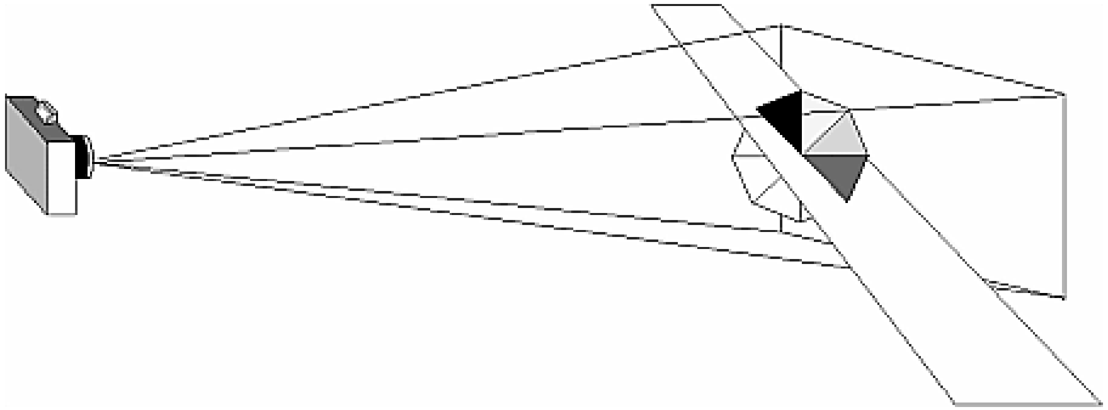
Viewing and Texture Mapping

In OPENGL

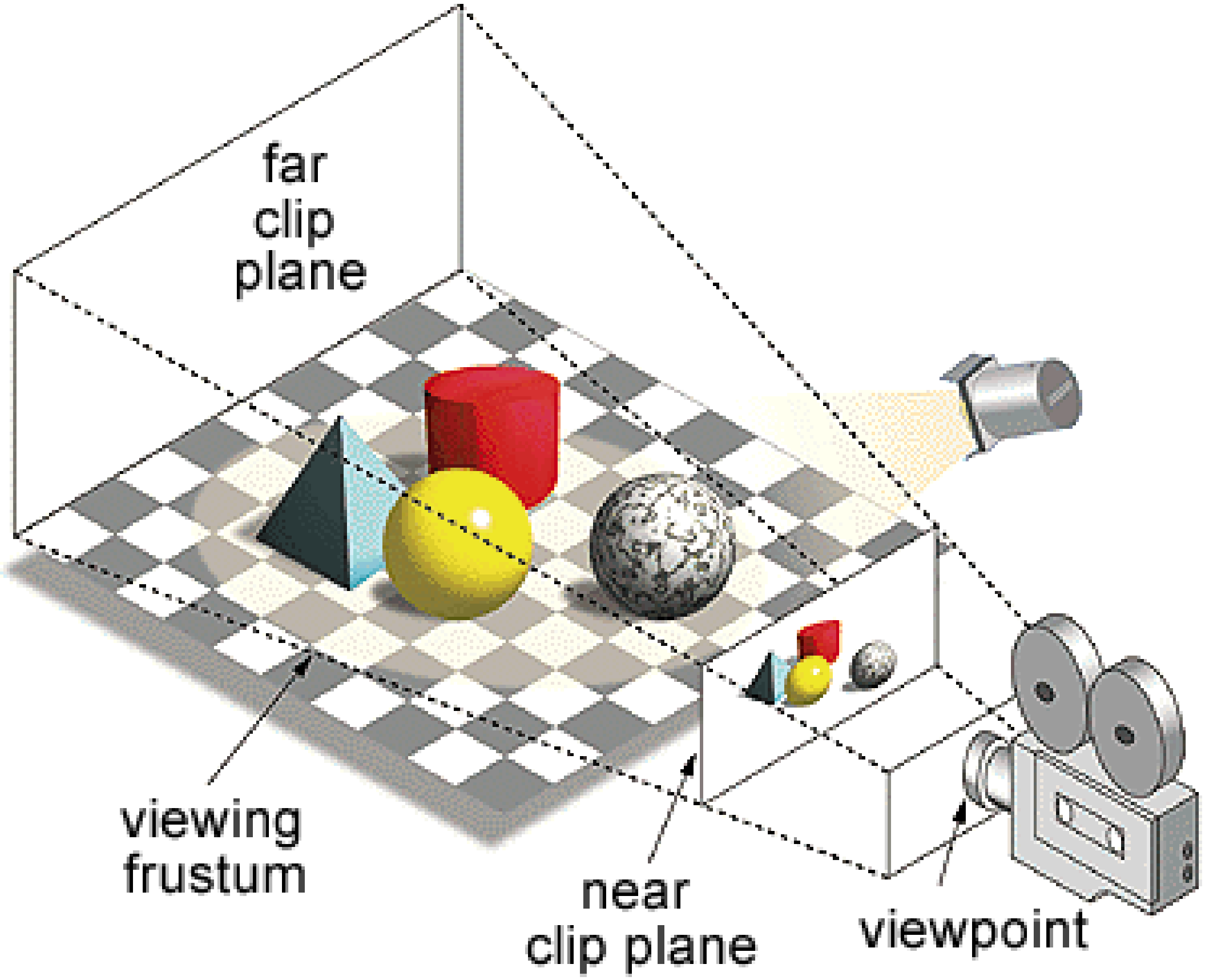


VIEWING

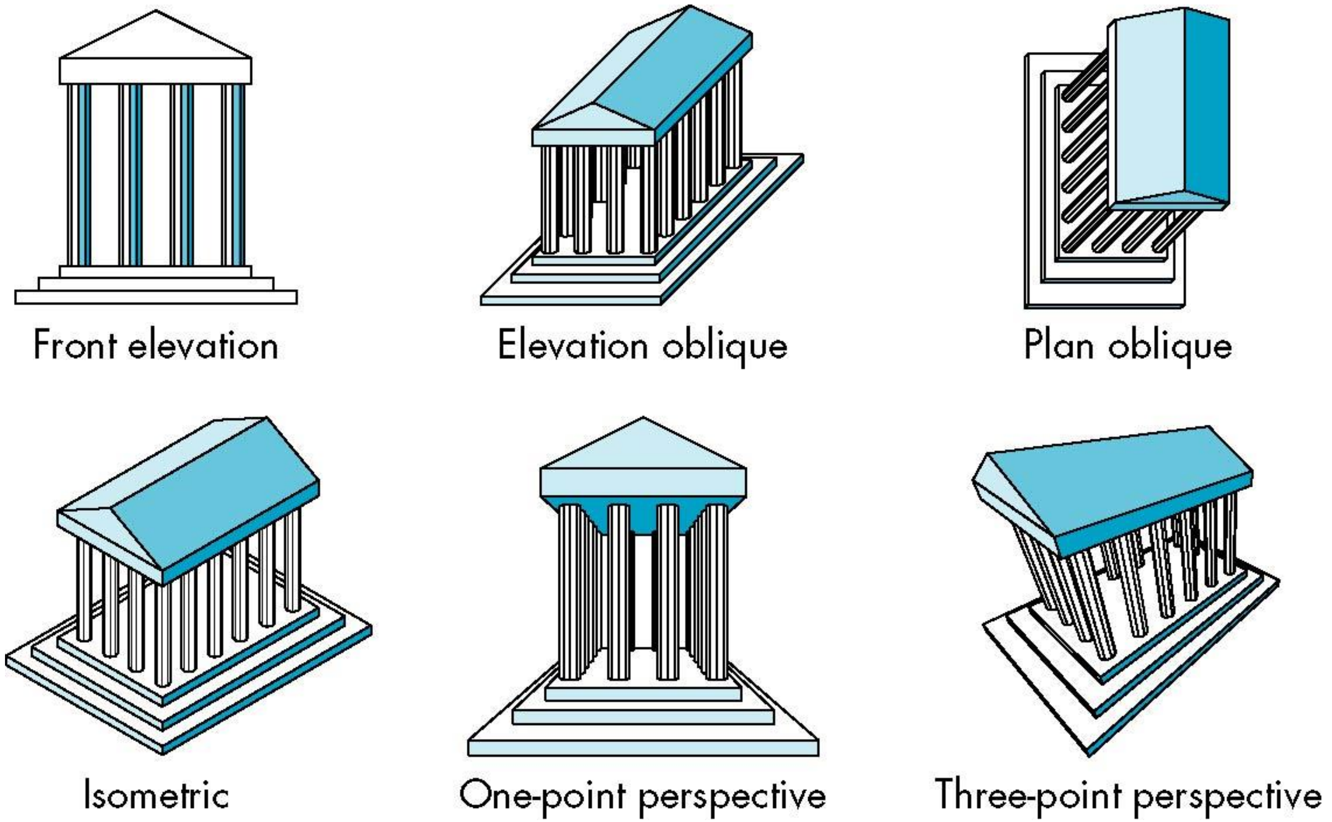
1. One or more objects
2. A viewer with a projection surface
3. Projectors that go from the object(s) to the projection surface



VIEWING

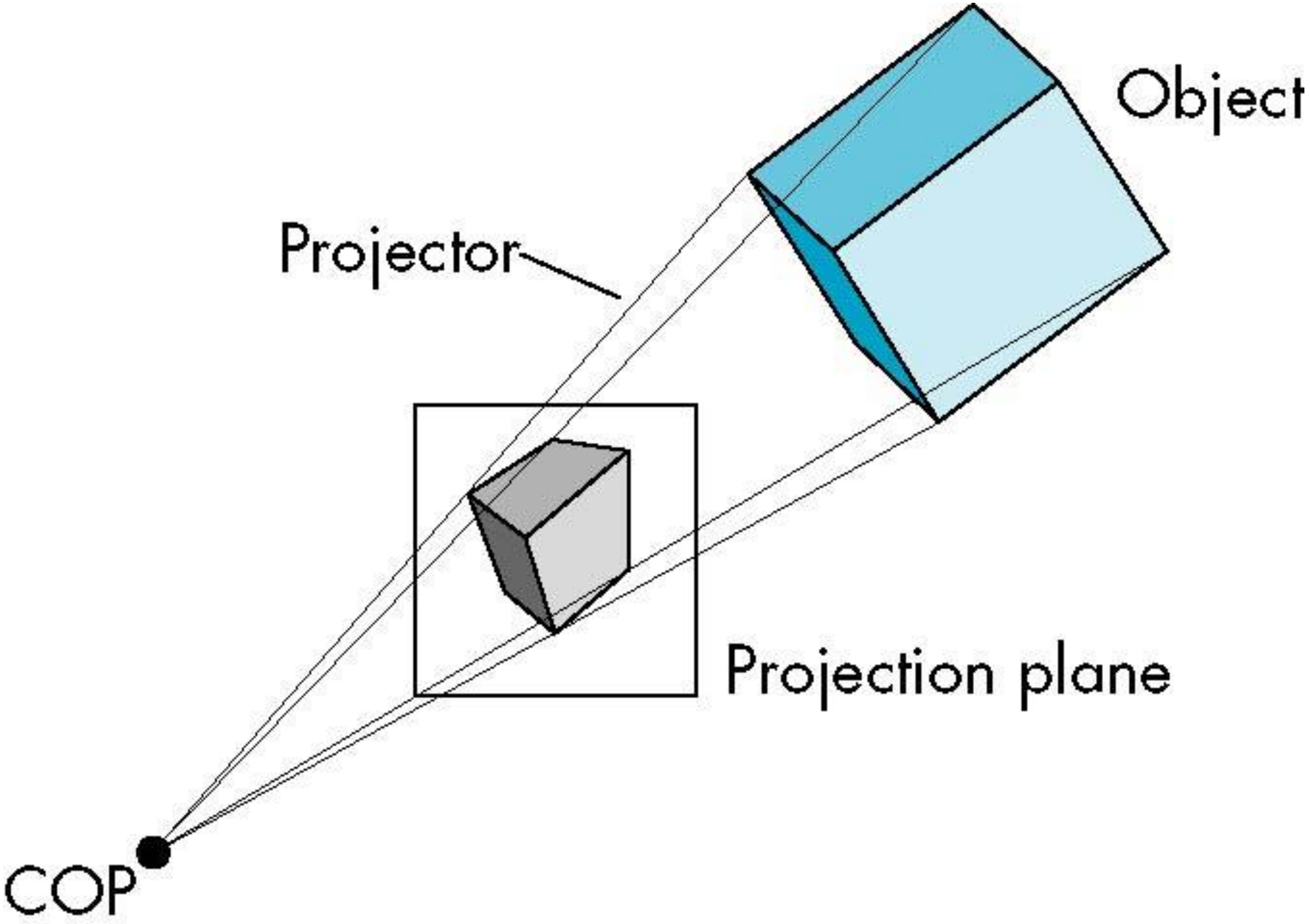


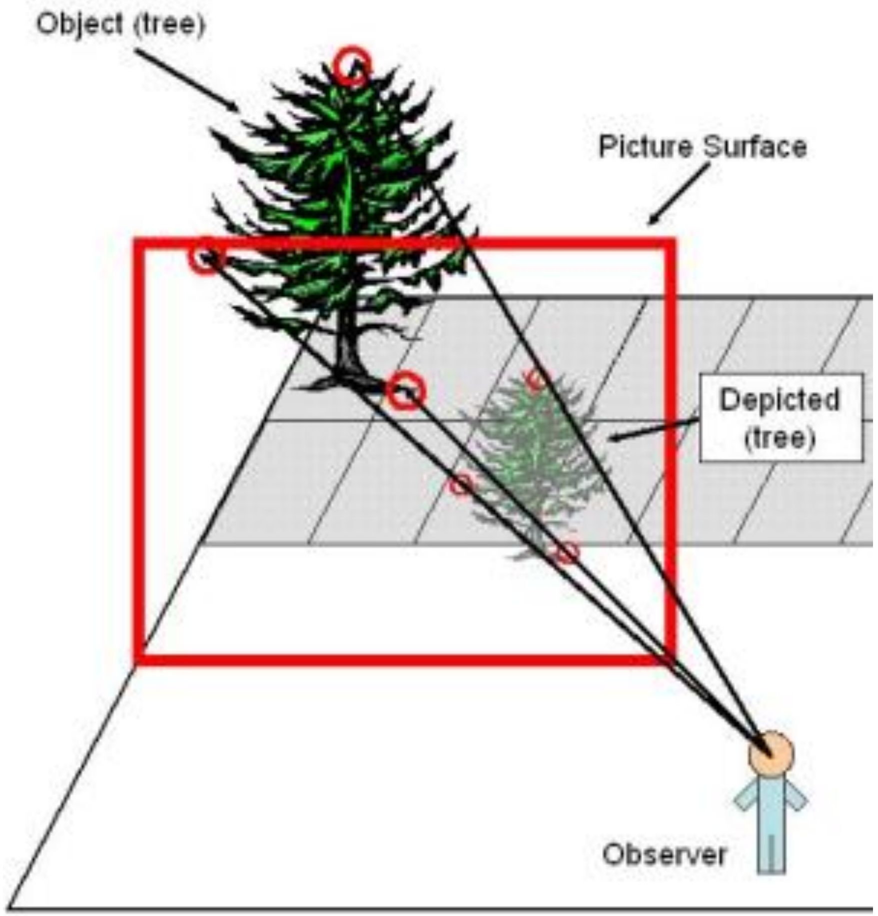
VIEWING

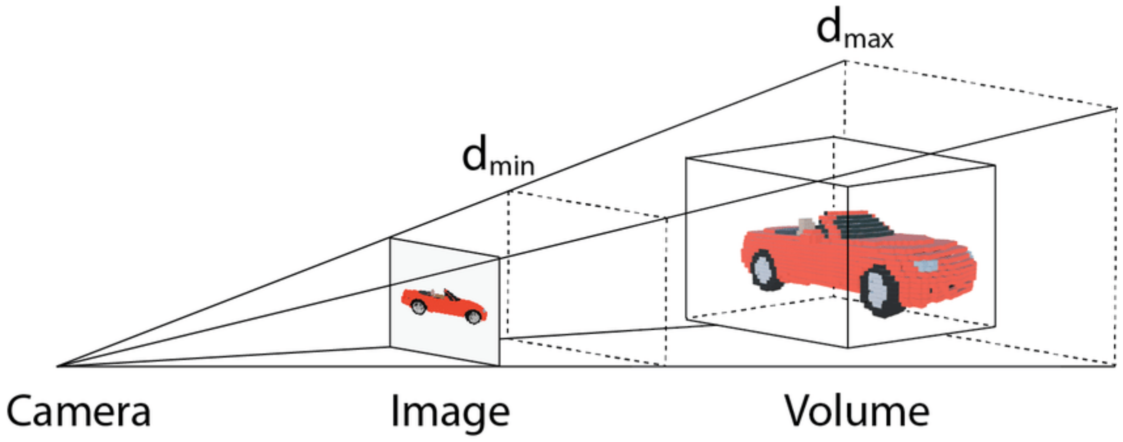

Front elevation
Elevation oblique
Plan oblique
Isometric
One-point perspective
Three-point perspective

VIEWING

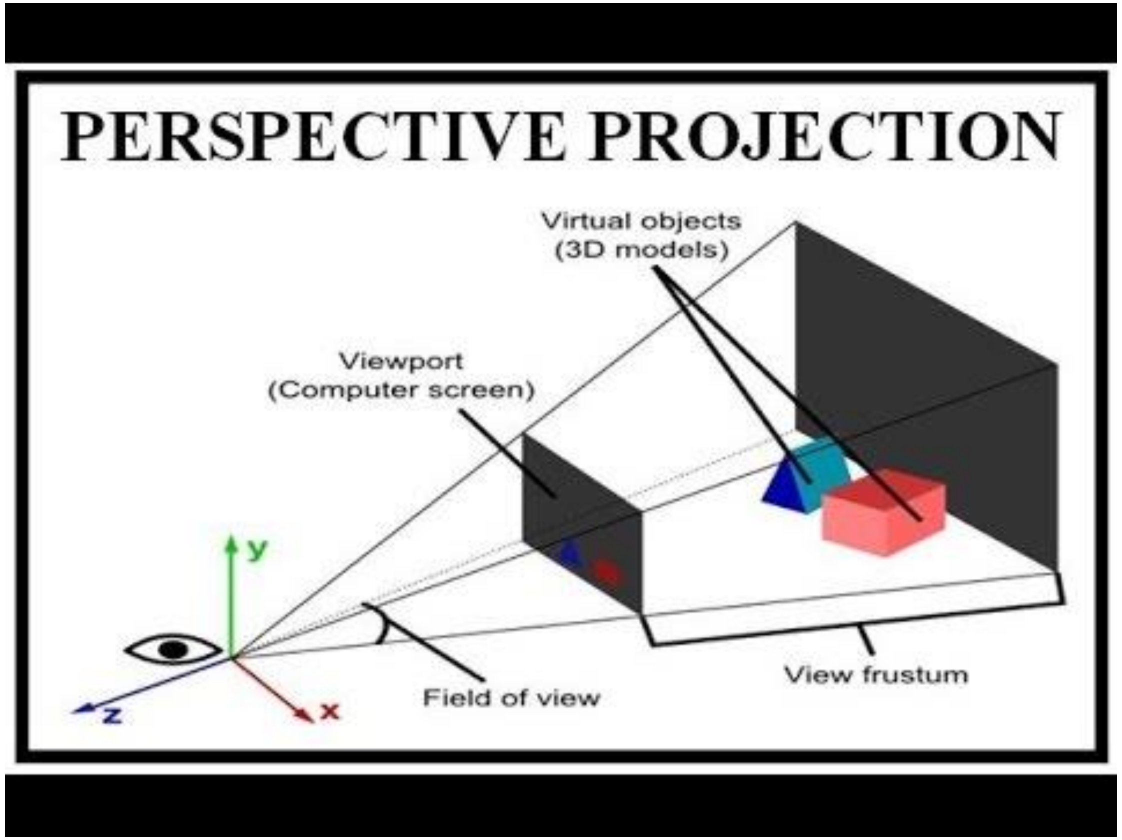
Perspective Projection







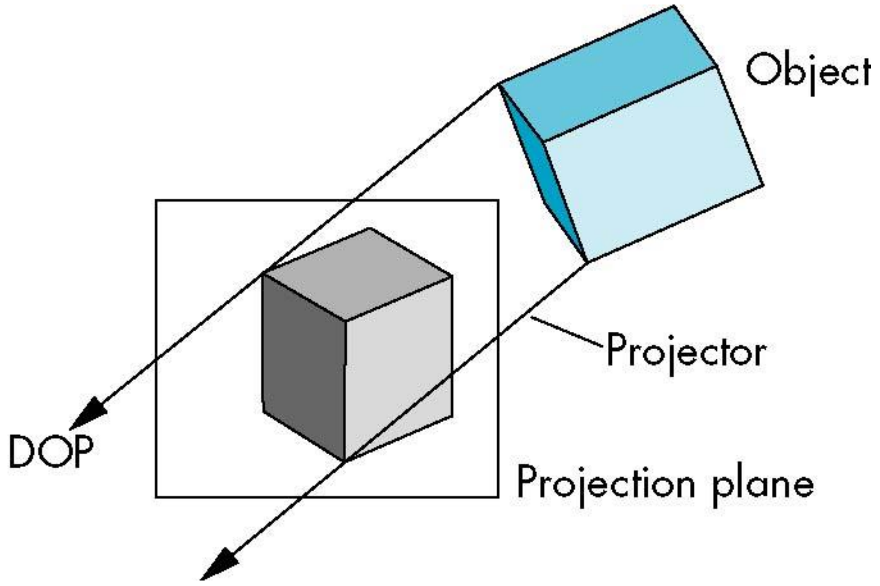
d_{max}
d_{min}
Camera
Image
Volume

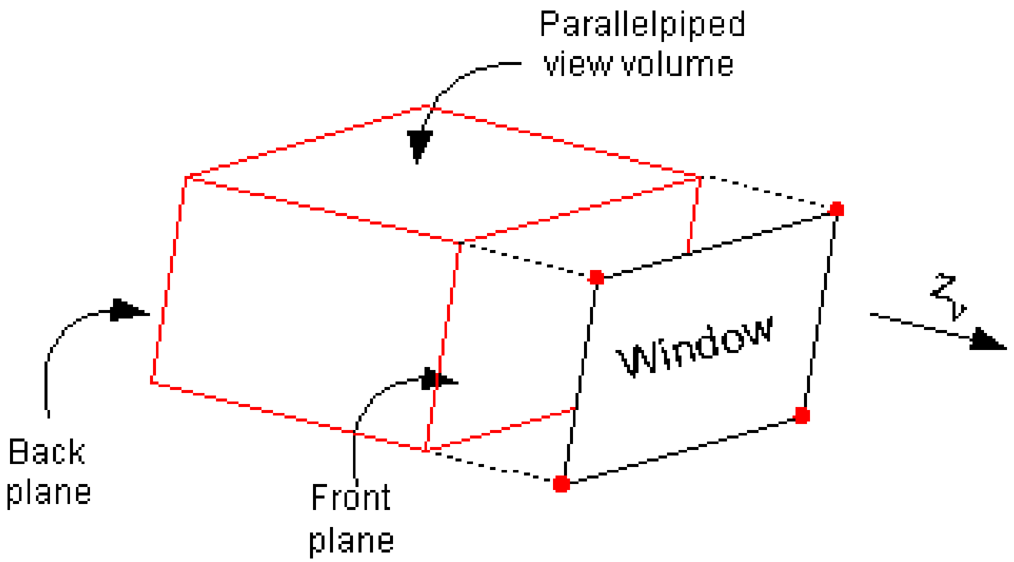


PERSPECTIVE PROJECTION
Virtual objects
(3D models)
Viewport
(Computer screen)
y
z
x
Field of view
View frustum

VIEWING

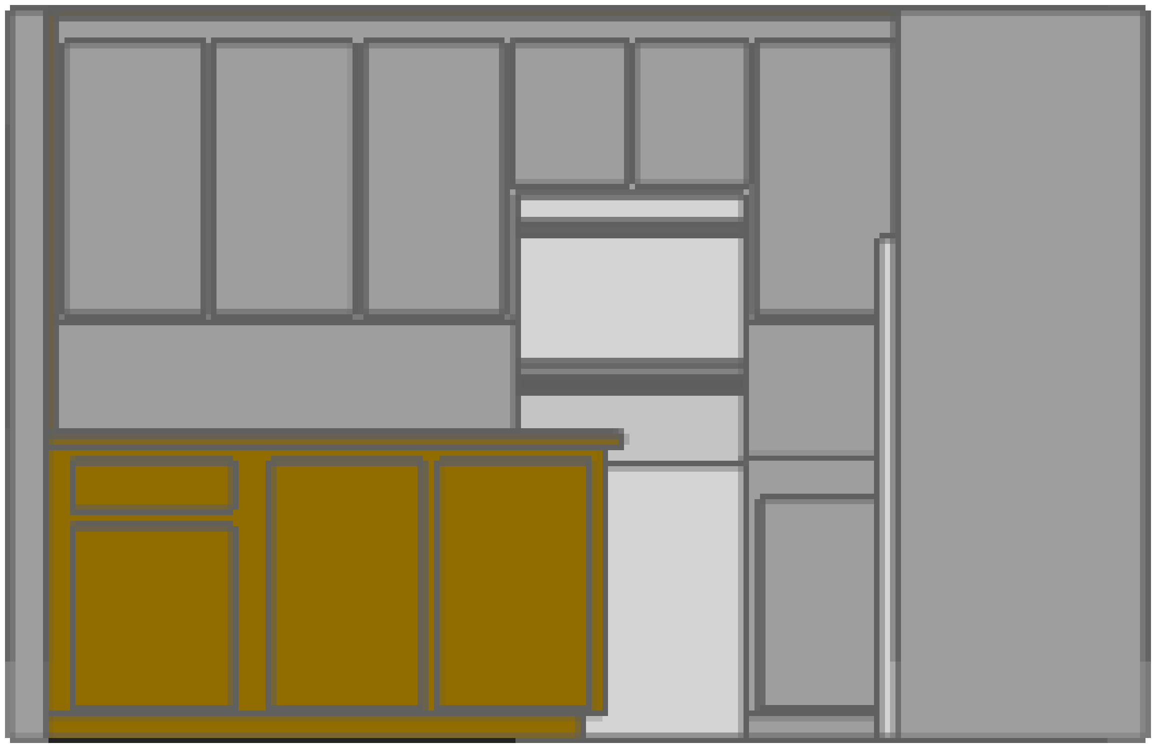
- Parallel Projection



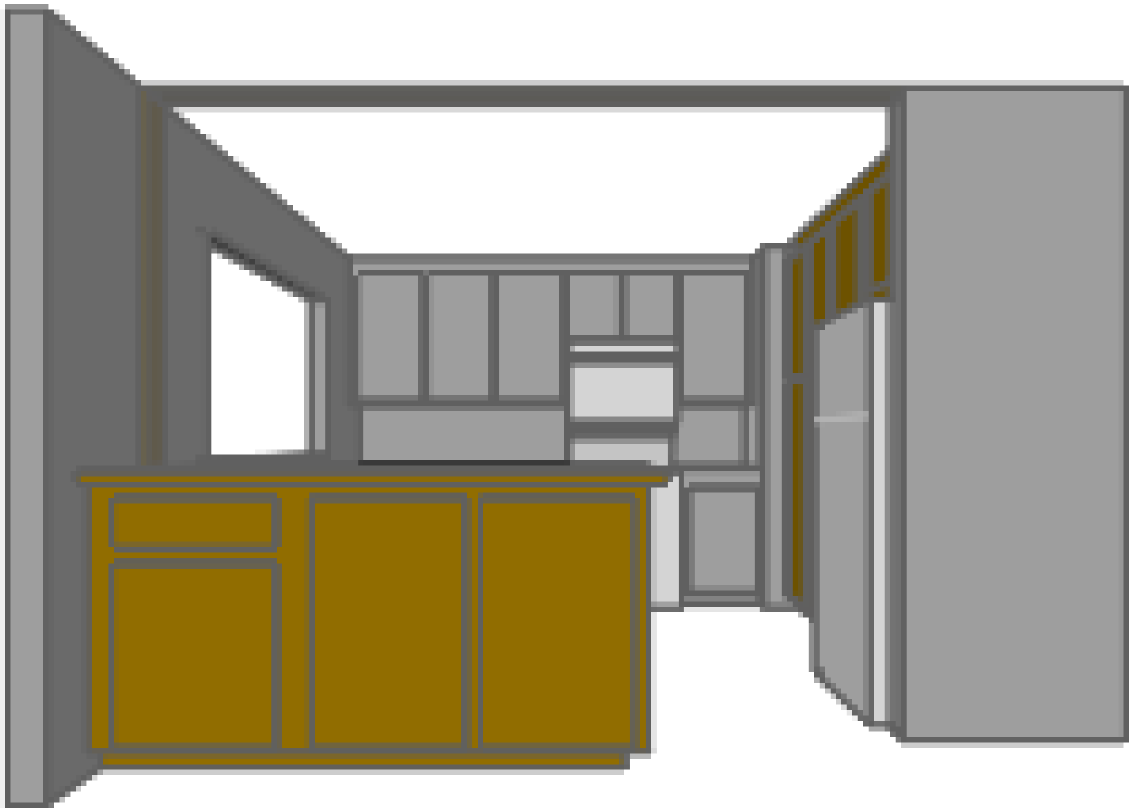


VIEWING

Both



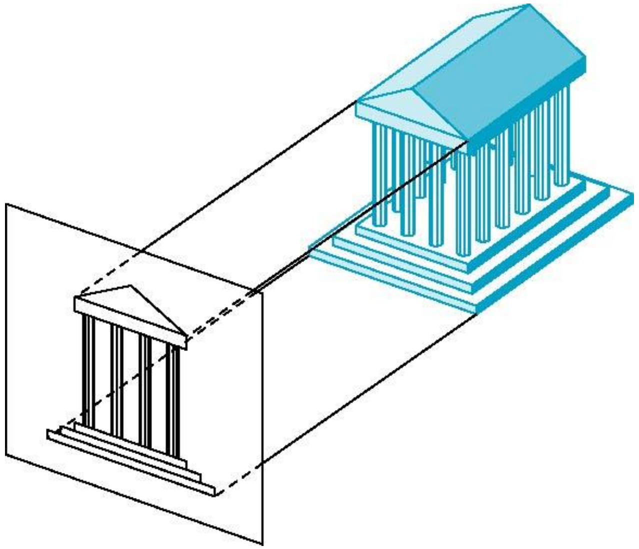
Parallel projection

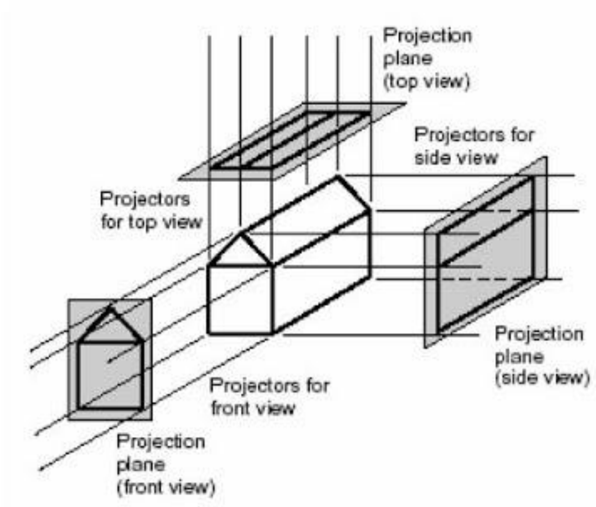


Perspective projection

VIEWING

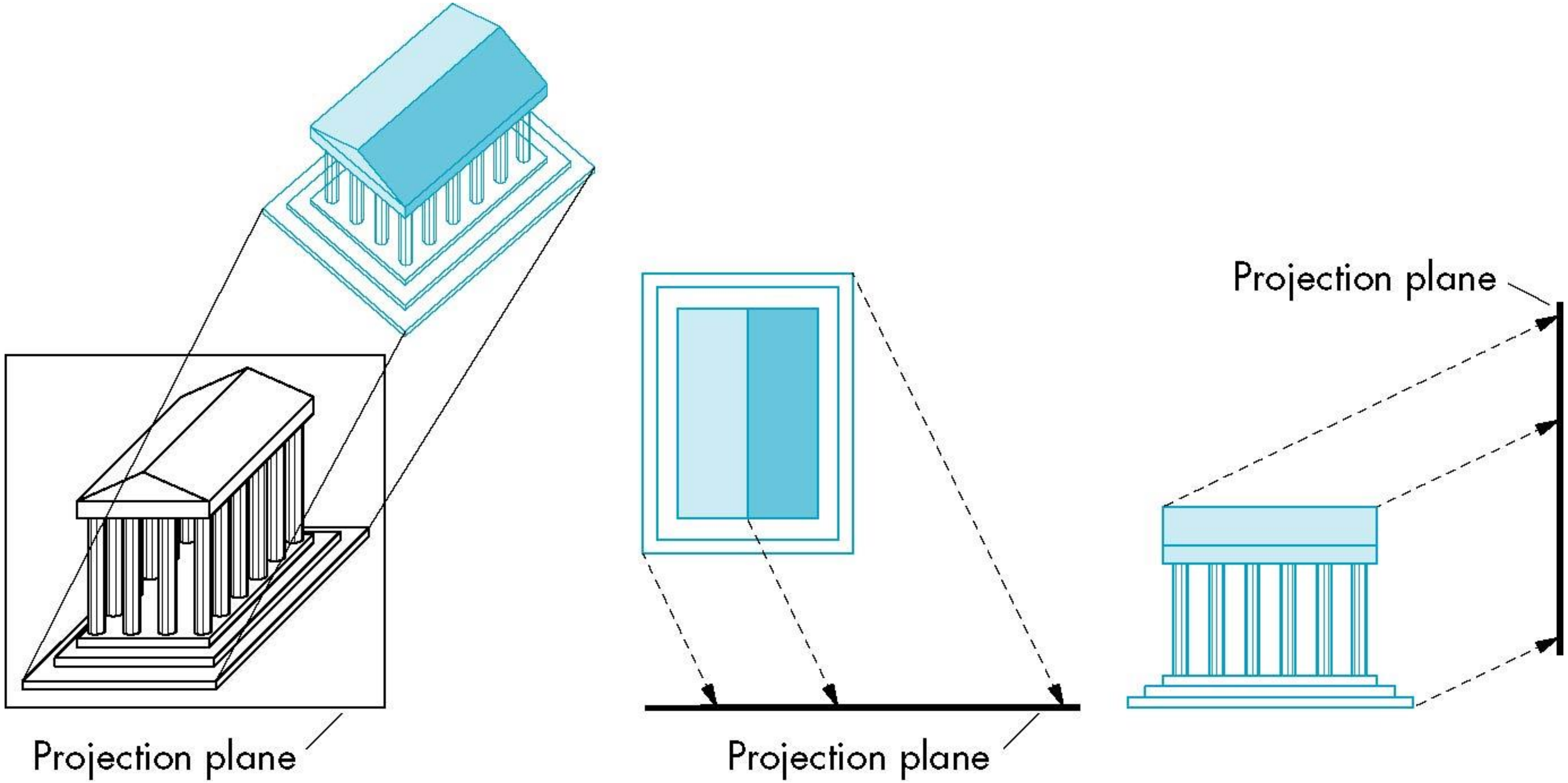
- Orthographic Projection





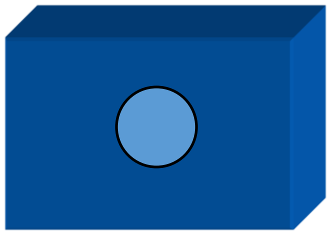
Oblique Projection

Arbitrary relationship between projectors and projection plane

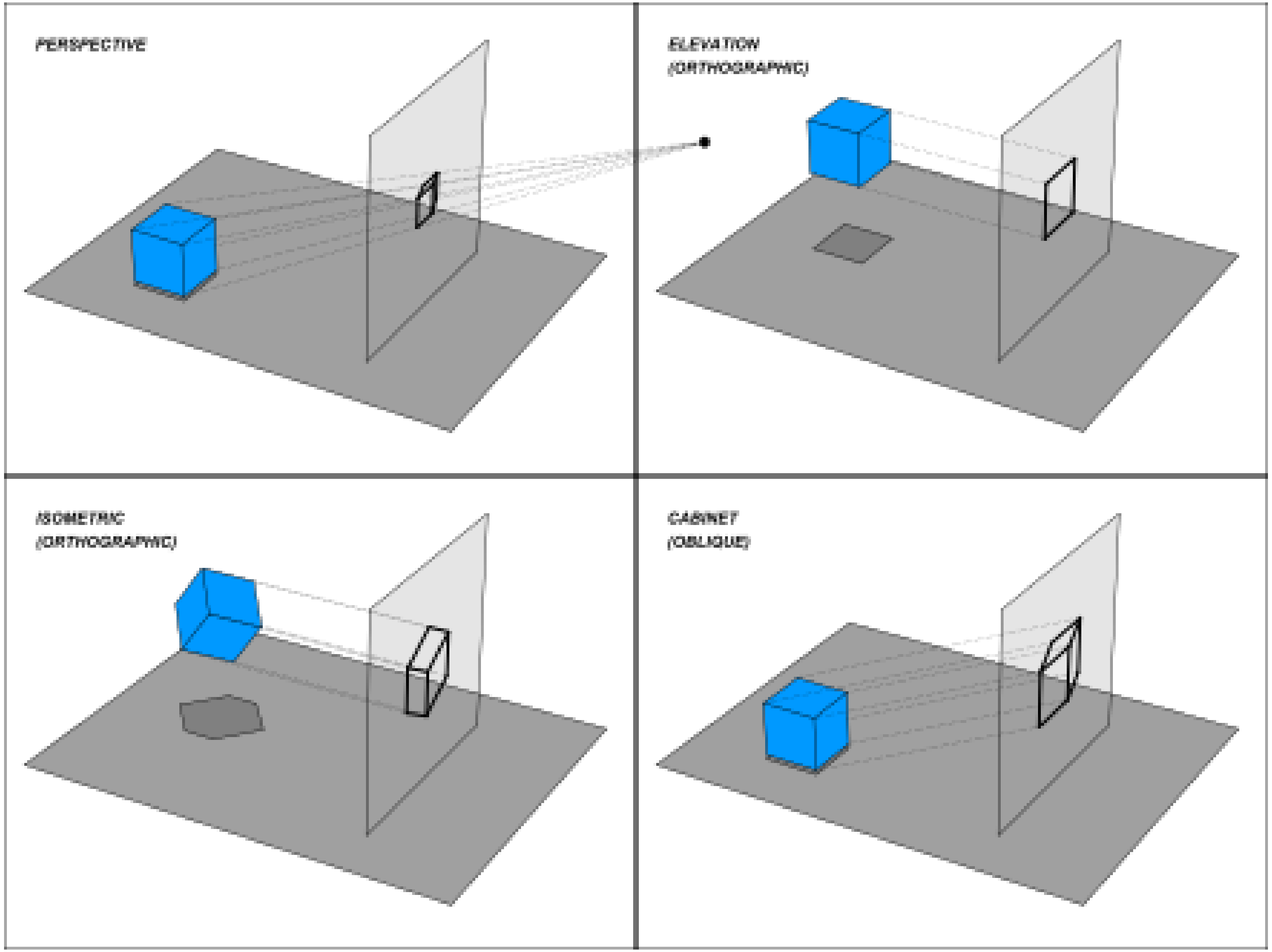


Advantages and Disadvantages

- Can pick the angles to emphasize a particular face
  - Architecture: plan oblique, elevation oblique
- Angles in faces parallel to projection plane are preserved while we can still see “around” side



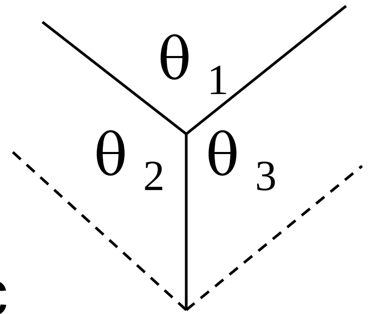
- In physical world, cannot create with simple camera; possible with bellows camera or special lens (architectural)

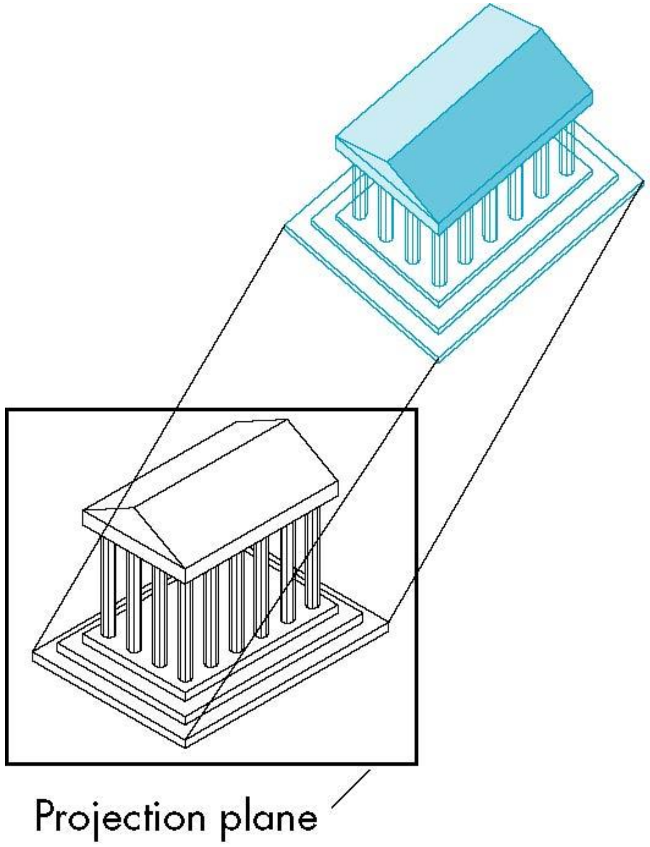

PERSPECTIVE
ELEVATION
(ORTHOGRAPHIC)
ISOMETRIC
(ORTHOGRAPHIC)
CABINET
(OBLIQUE)

Allow projection plane to move relative to object

classify by how many angles of a corner of a projected cube are the same

none: trimetric
two: dimetric
three: isometric

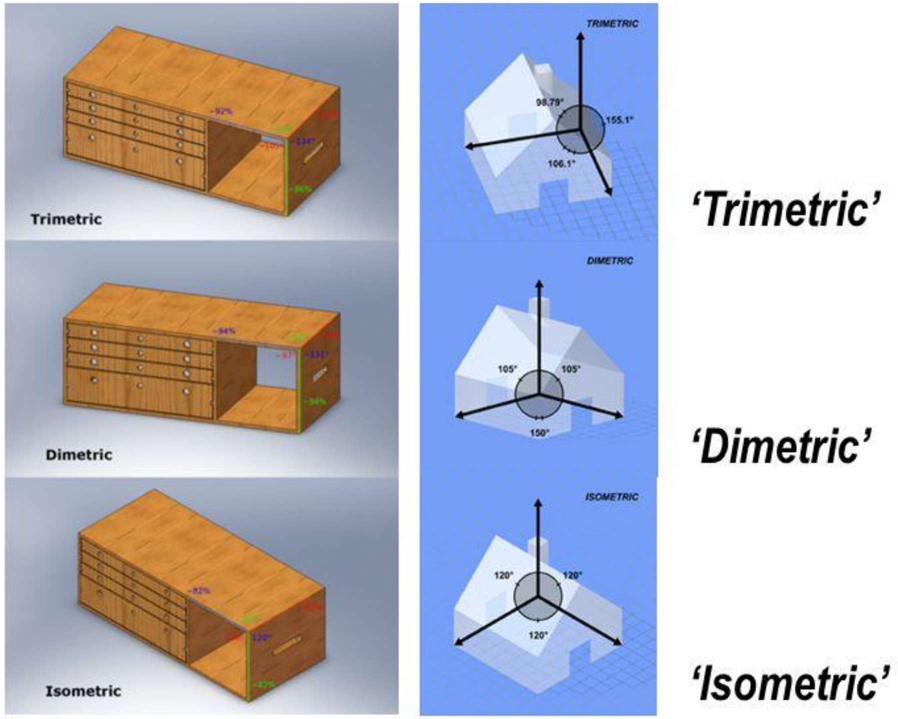




VIEWING

- Axonometric projections

Non-perspective 3D drawings: Axonometric Projections



(Source: www.en.wikipedia.org)

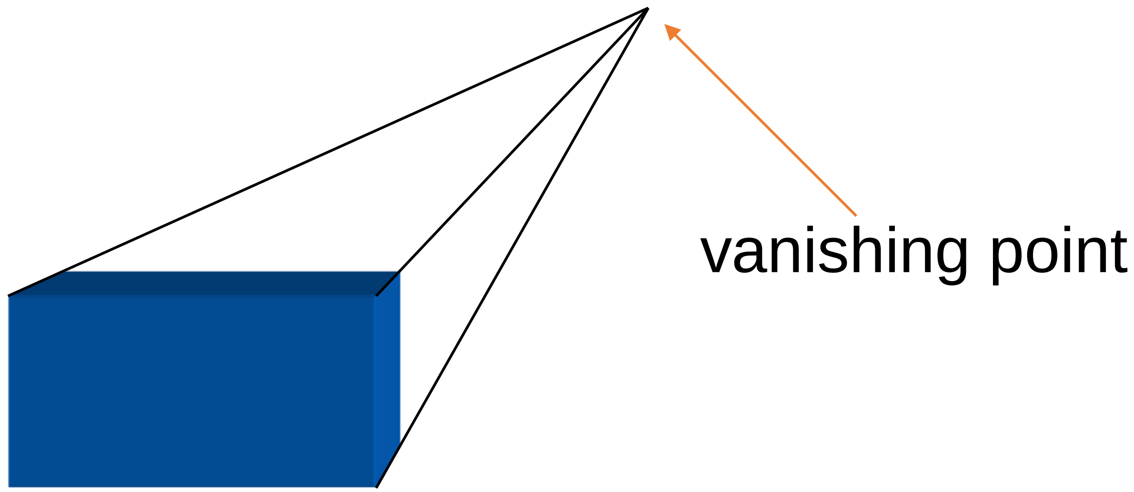
VIEWING

Axonometric projections - Advantages and Disadvantages

- Lines are scaled (*foreshortened*) but can find scaling factors
- Lines preserved but angles are not
  - Projection of a circle in a plane not parallel to the projection plane is an ellipse
- Can see three principal faces of a box-like object
- Some optical illusions possible
  - Parallel lines appear to diverge
- Does not look real because far objects are scaled the same as near objects
- Used in CAD applications

Vanishing Points

- Parallel lines (not parallel to the projection plan) on the object converge at a single point in the projection (the *vanishing point*)
- Drawing simple perspectives by hand uses these vanishing point(s)



Three-Point Perspective

- No principal face parallel to projection plane
- Three vanishing points for cube

Two-Point Perspective

- On principal direction parallel to projection plane
- Two vanishing points for cube

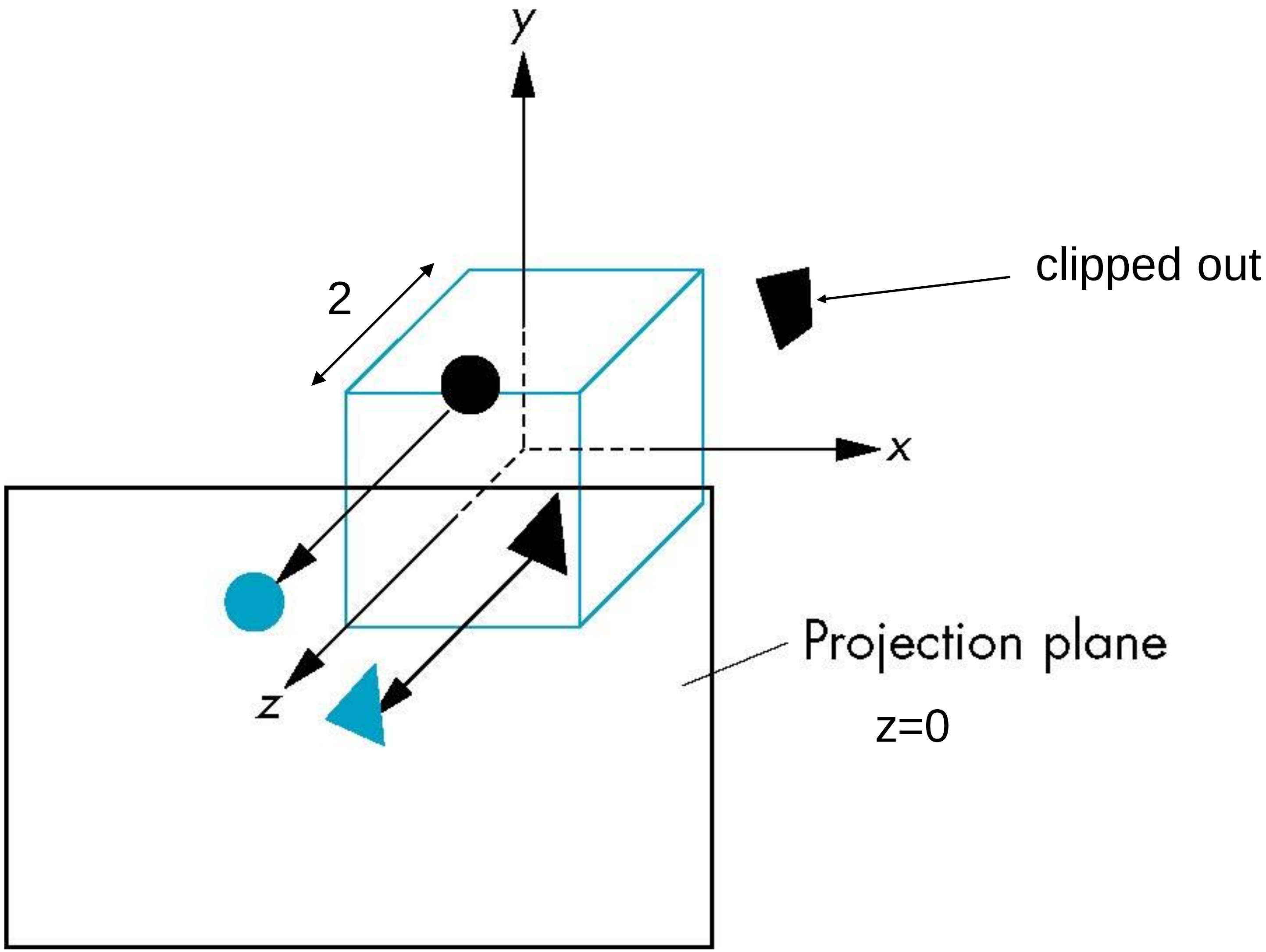
One-Point Perspective

- One principal face parallel to projection plane
- One vanishing point for cube

Advantages and Disadvantages

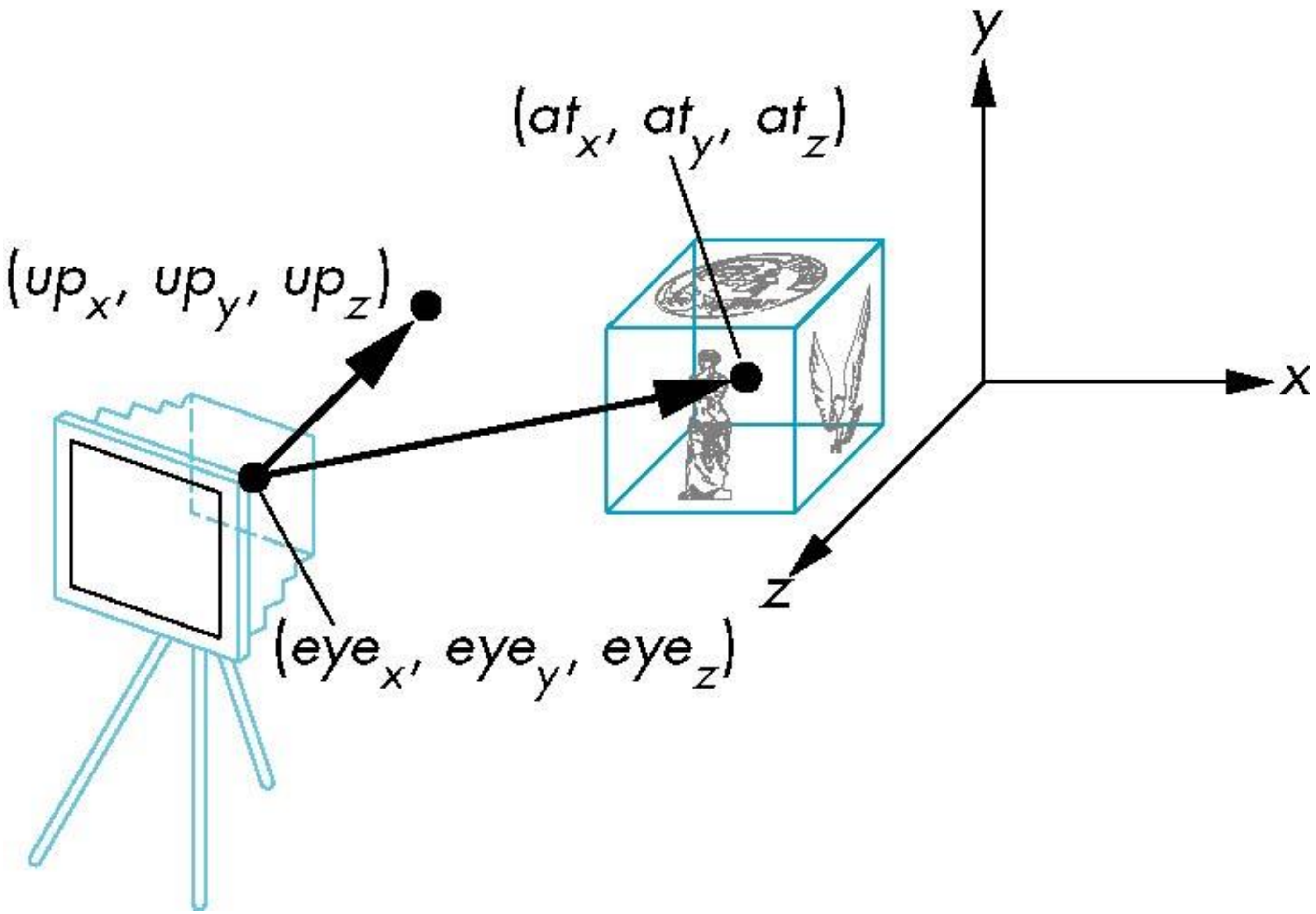
- Objects further from viewer are projected smaller than the same sized objects closer to the viewer (*diminution*)
  - Looks realistic
- Equal distances along a line are not projected into equal distances (*nonuniform foreshortening*)
- Angles preserved only in planes parallel to the projection plane
- More difficult to construct by hand than parallel projections (but not more difficult by computer)

VIEWING IN OPENGL



VIEWING IN OPENGL

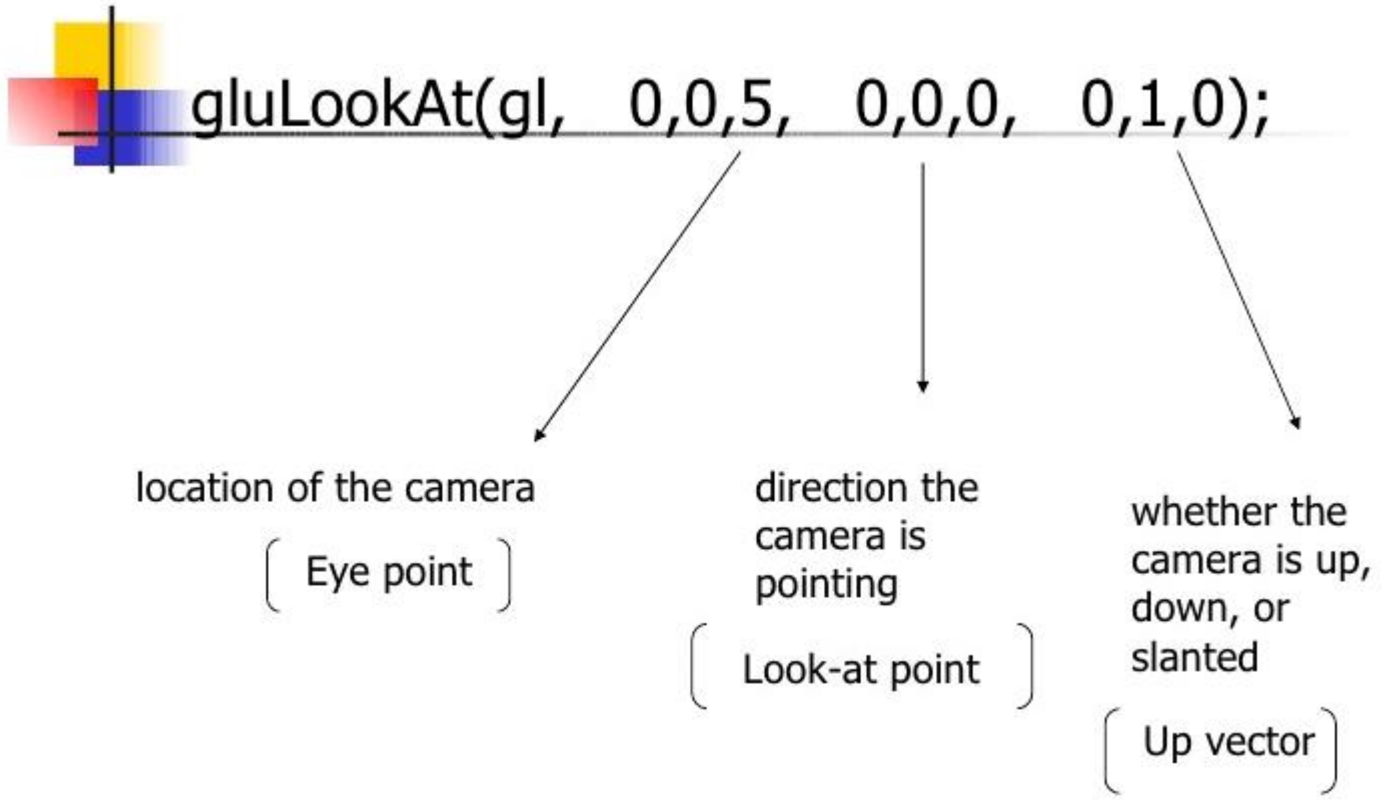
- gluLookAt



VIEWING IN OPENGL

gluLookAt

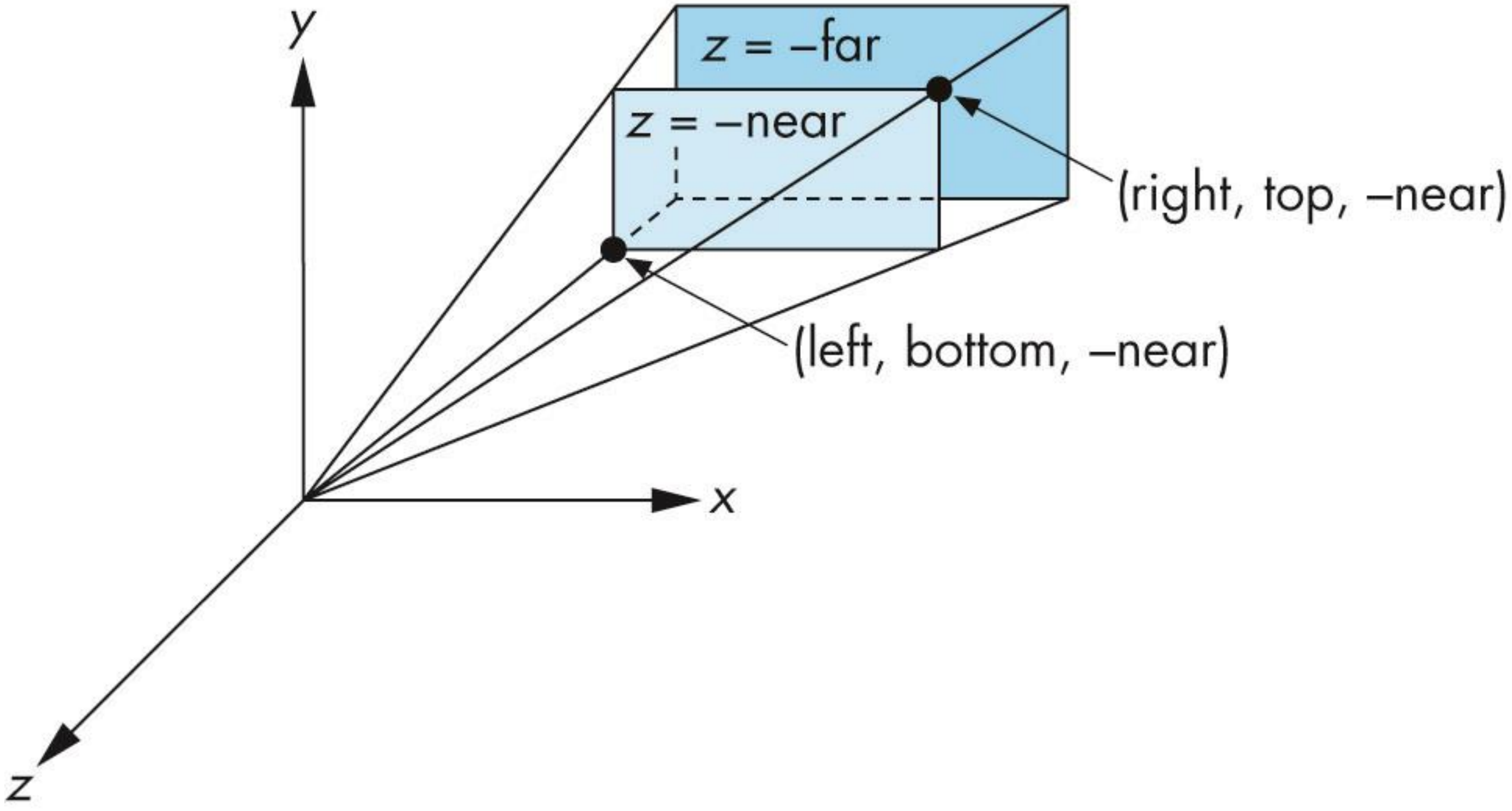
```
LookAt(eye, at, up)
```



VIEWING IN OPENGL

glfrustum

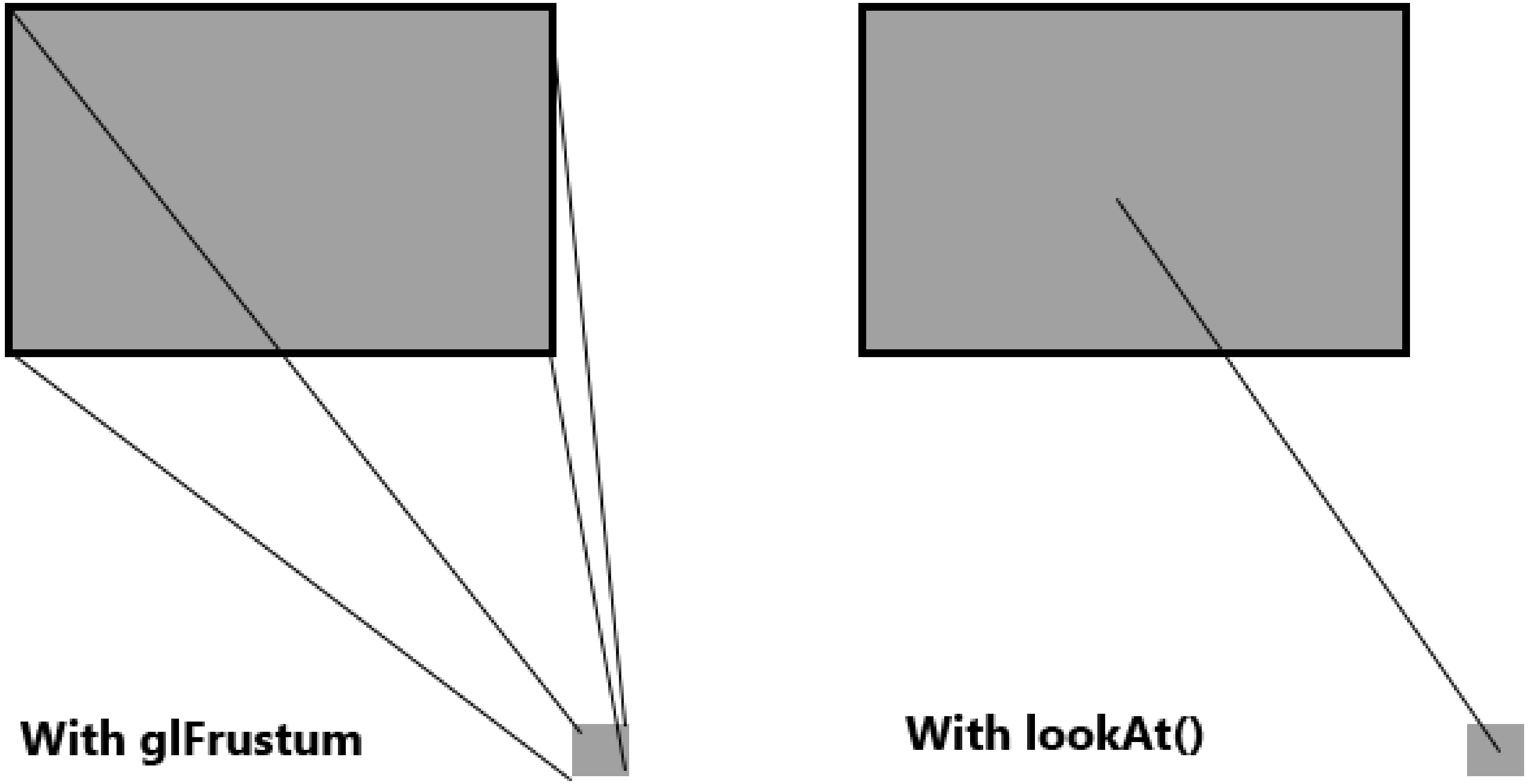
```
Frustum(left,right,bottom,top,near,far)
```



VIEWING IN OPENGL

DIFFERENCES

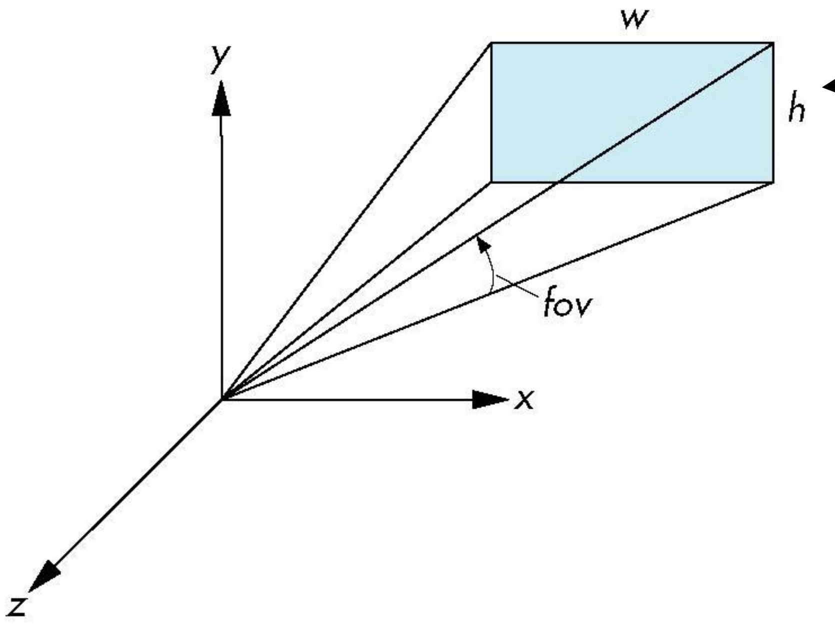
```
Frustum(left,right,bottom,top,near,far)
```



VIEWING IN OPENGL

gluPerspective

```
Perpective(fovy, aspect, near, far)
```



front plane

aspect = w/h

VIEWING IN OPENGL

gluPerspective

```
Perpective(fovy, aspect, near, far)
```

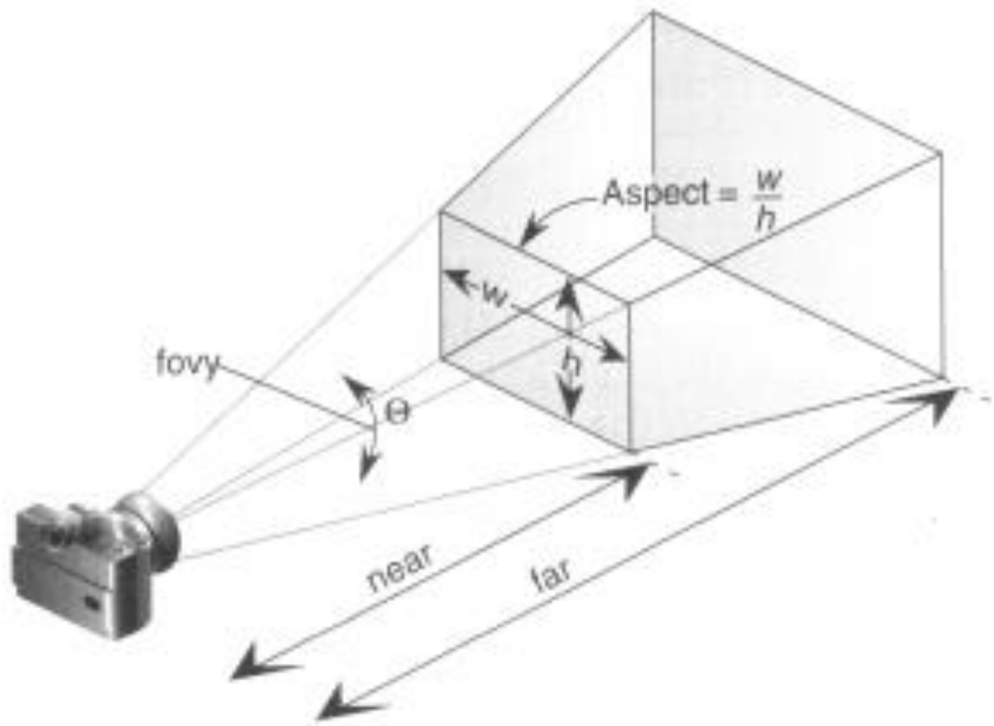
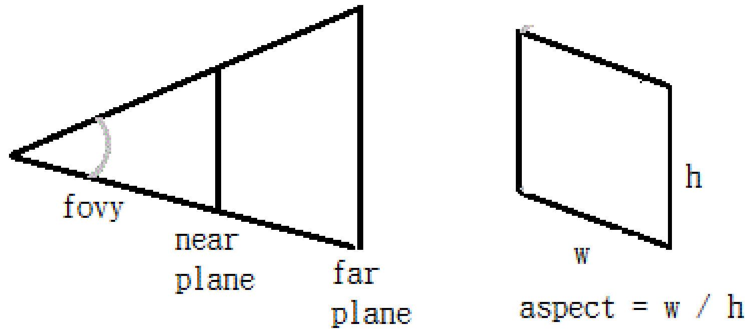
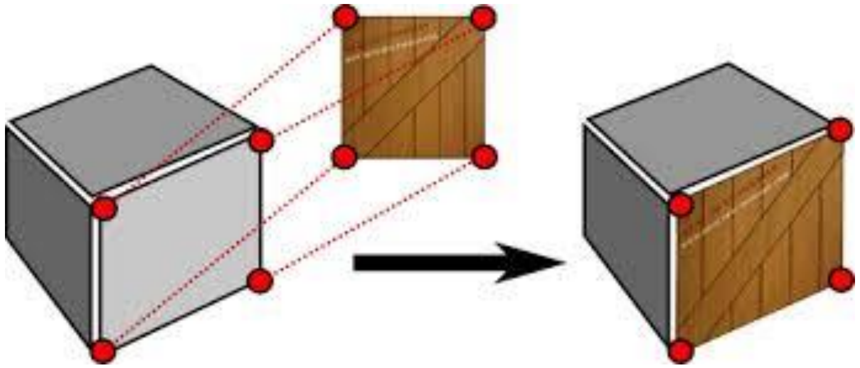


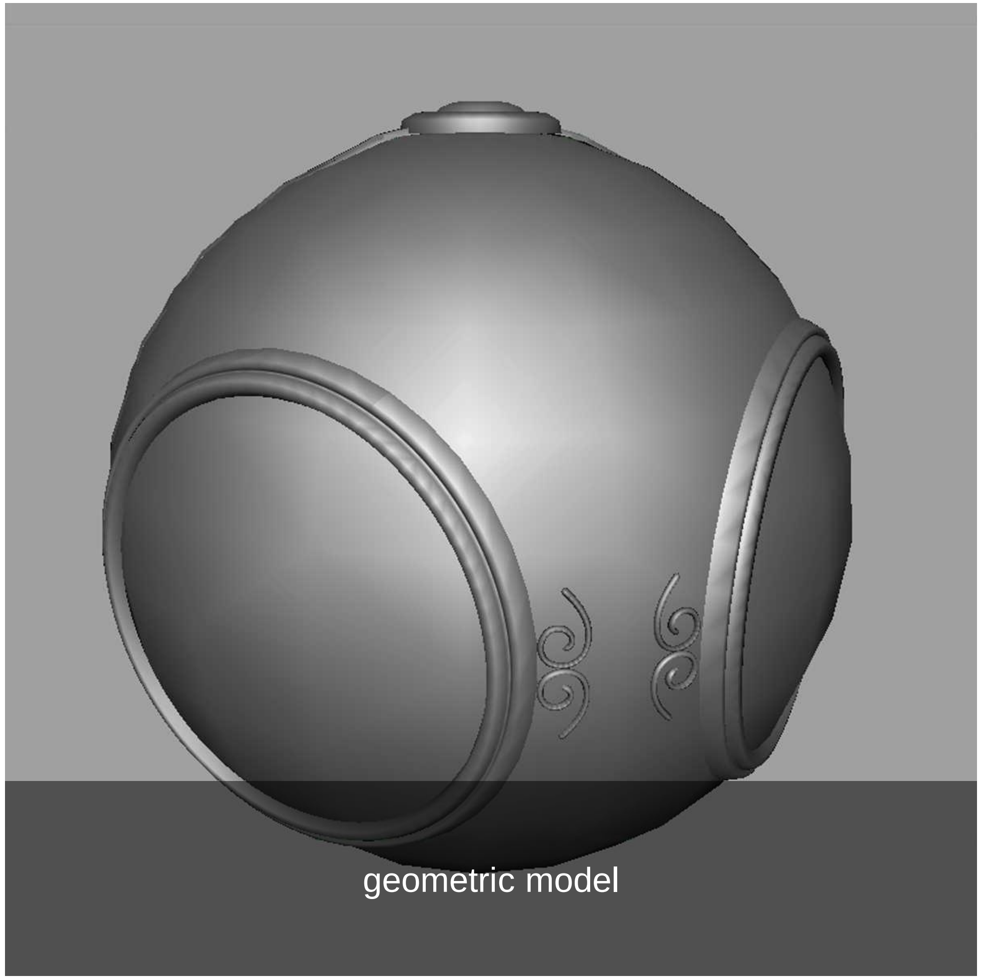
Figure 3-14 Perspective Viewing Volume Specified by gluPerspective()

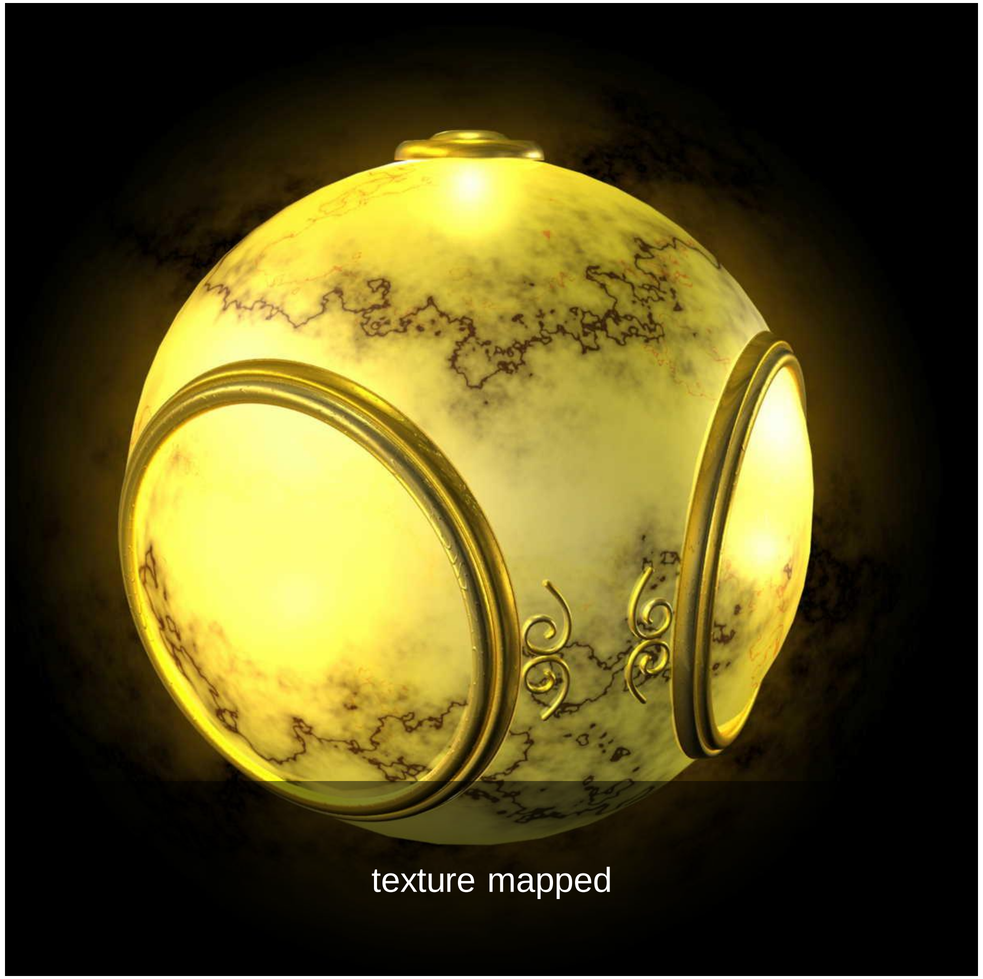
void gluPerspective(GLdouble fovy, GLdouble aspect, GLdouble zNear, GLdouble zFar)





Texture Mapping

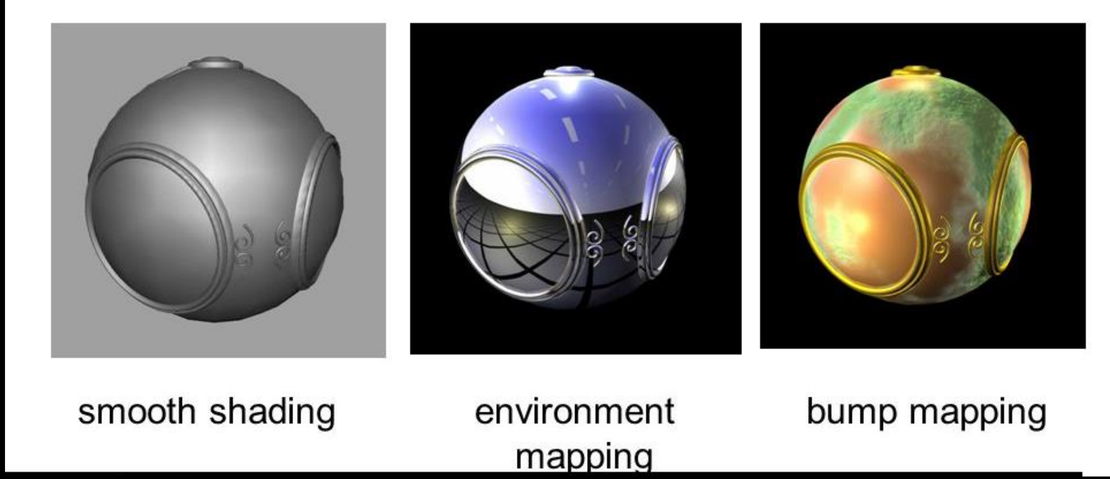


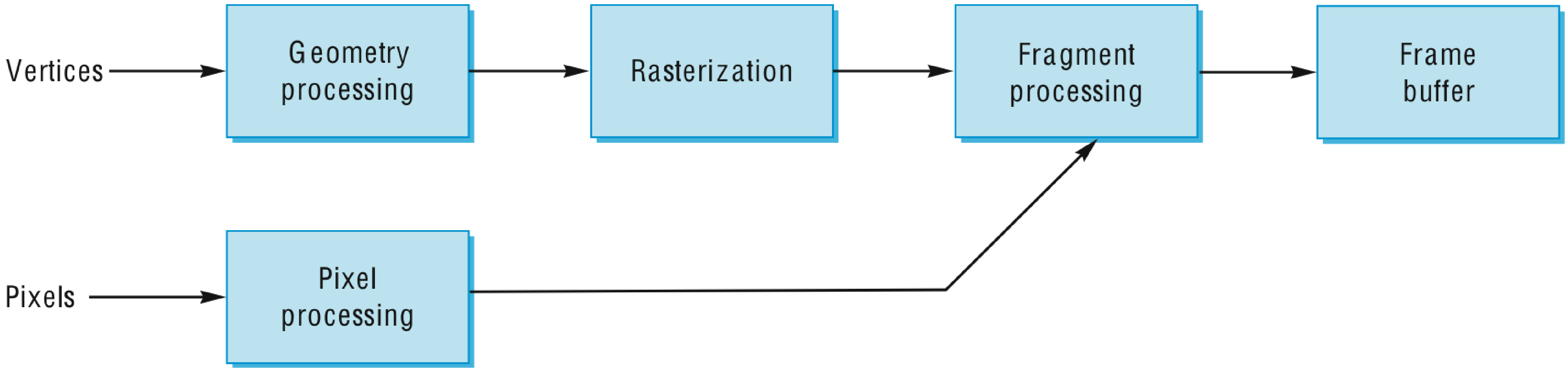


TEXTURE MAPPING

TEXTURE MAPPING

- Texture Mapping
  - Uses images to fill inside of polygons
- Environment (reflection mapping)
  - Uses a picture of the environment for texture maps
  - Allows simulation of highly specular surfaces
- Bump mapping
  - Emulates altering normal vectors during the rendering process

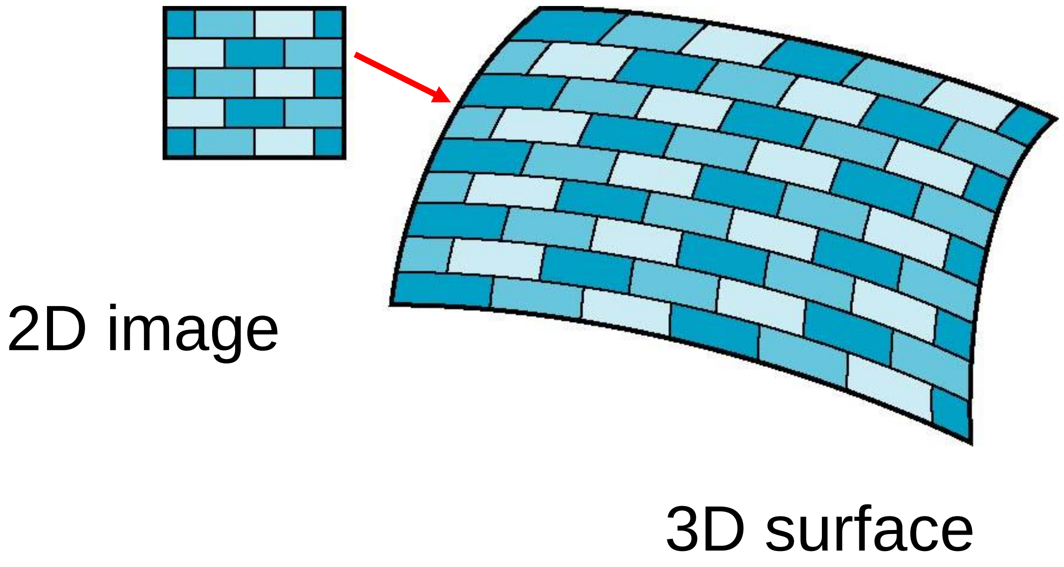




TEXTURE MAPPING

Is it simple?

- Although the idea is simple---map an image to a surface---there are 3 or 4 coordinate systems involved



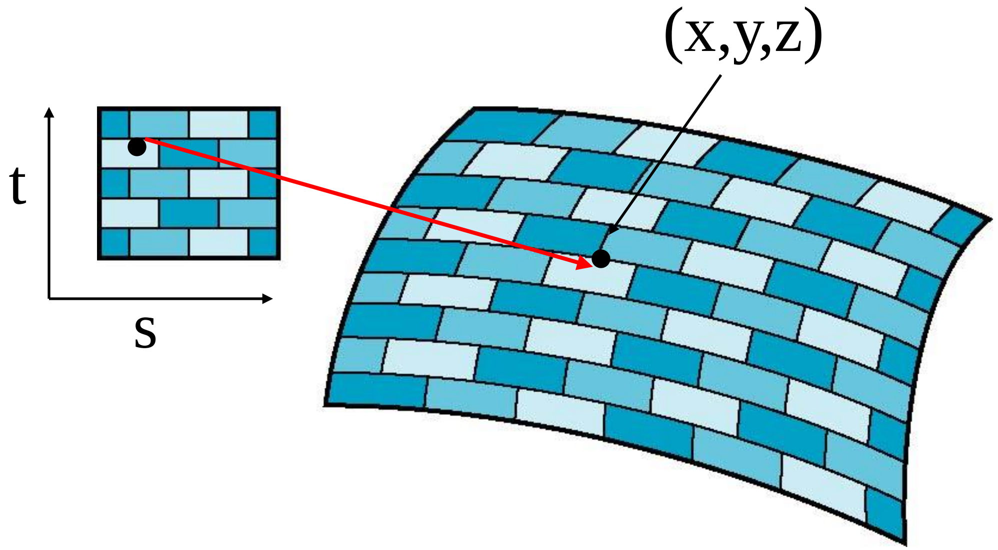
Mapping Functions

- Basic problem is how to find the maps
- Consider mapping from texture coordinates to a point a surface
- Appear to need three functions

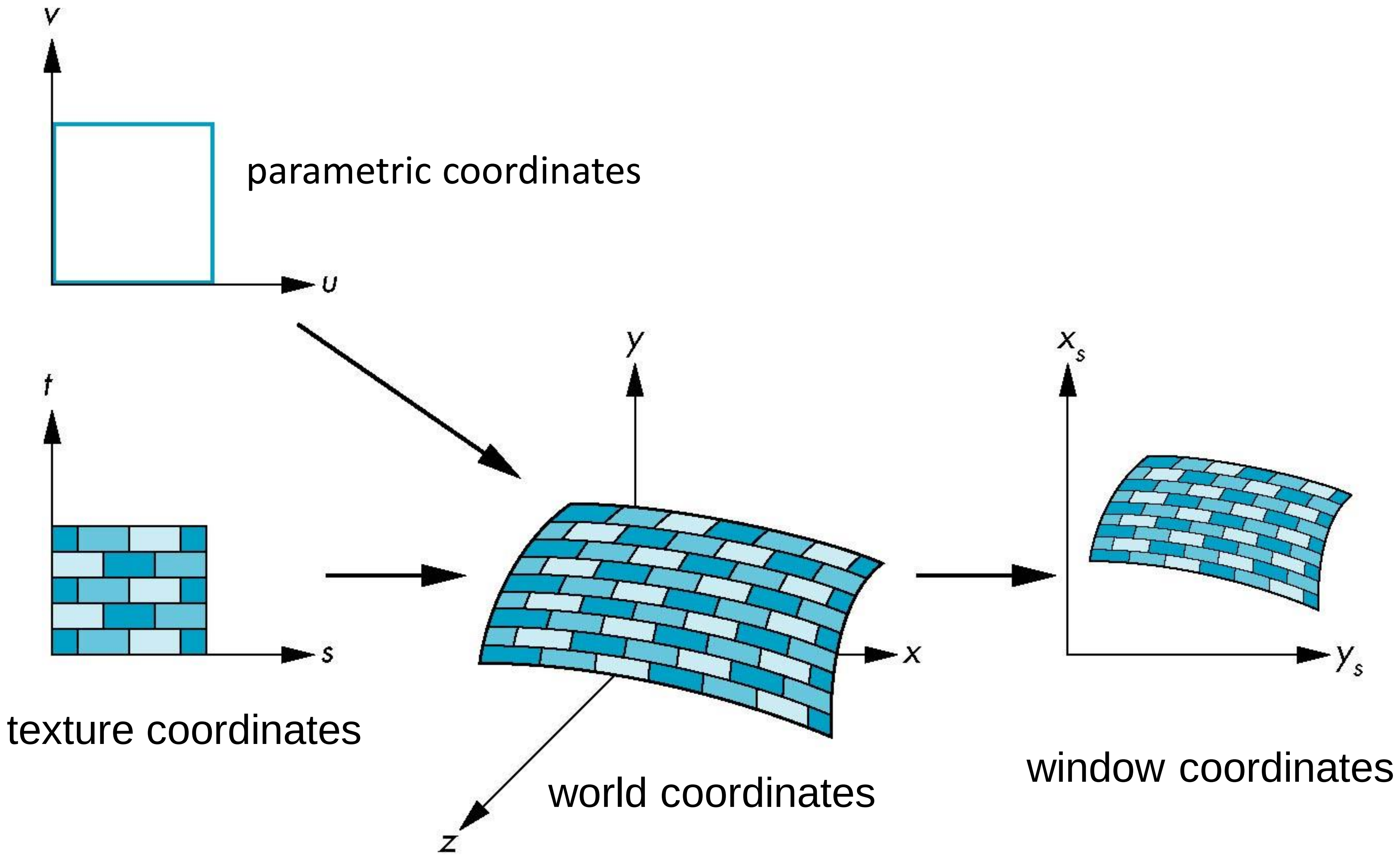
$$x = x(s,t)$$
$$y = y(s,t)$$
$$z = z(s,t)$$

- But we really want

to go the other way

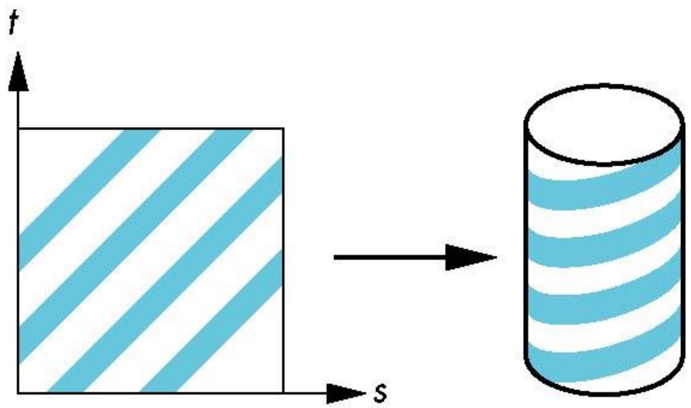


TEXTURE MAPPING



Two-part mapping

- One solution to the mapping problem is to first map the texture to a simple intermediate surface
- Example: map to cylinder



Cylindrical Mapping

parametric cylinder

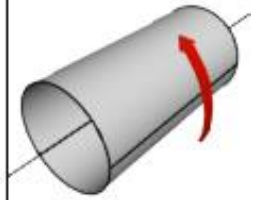
$$x = r \cos 2\pi u$$
$$y = r \sin 2\pi u$$
$$z = v/h$$

maps rectangle in u,v space to cylinder of radius r and height h in world coordinates

$$s = u$$
$$t = v$$

maps from texture space

Cylindrical Image Mapping

Axis / Examples	Diagrams
X Mapping detal to a pipe segment.	
Y Mapping a label onto a soda can or a wine bottle.	
Z Mapping detail onto an engine exhaust.	

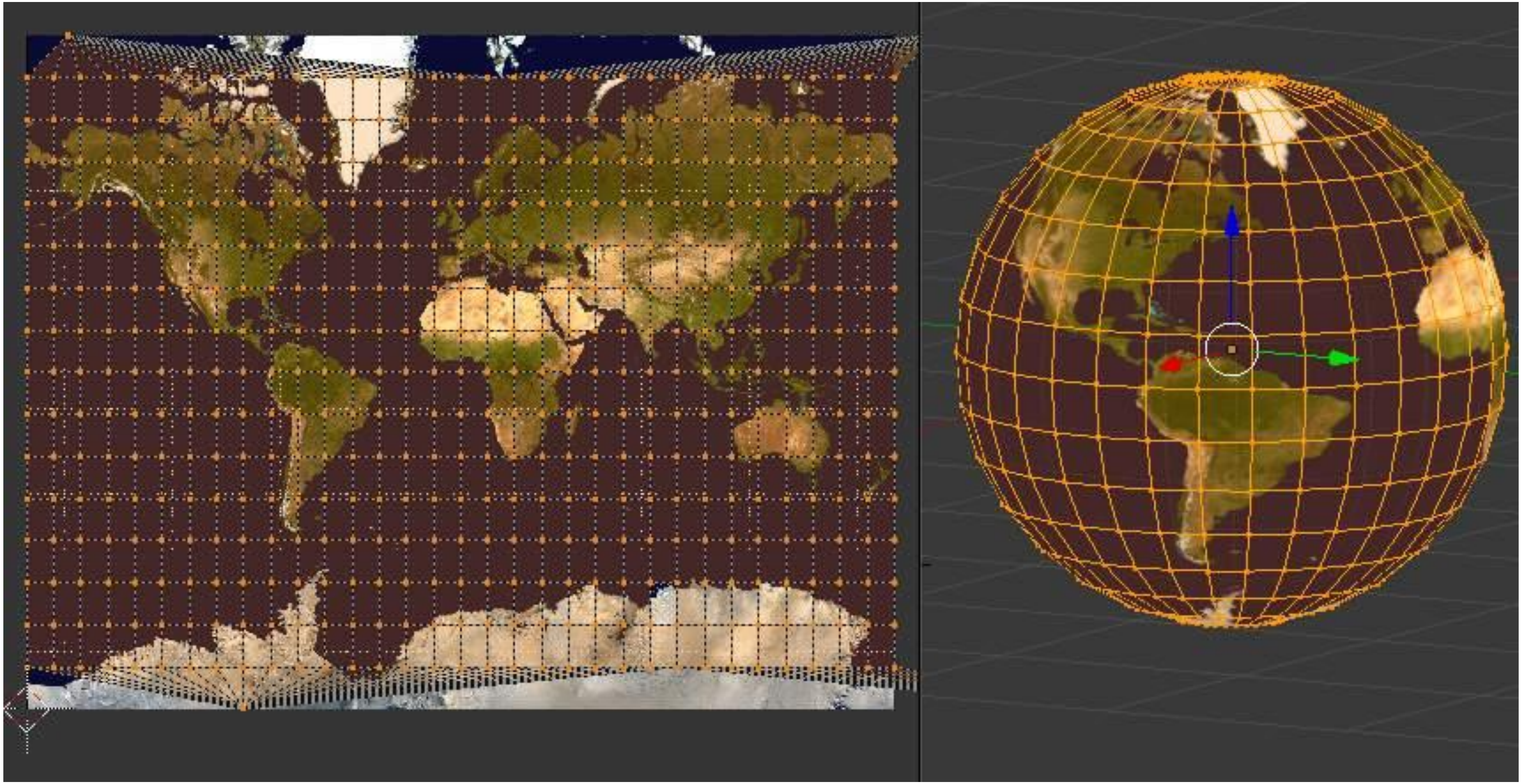
Spherical Map

We can use a parametric sphere

$$x = r \cos 2\pi u$$
$$y = r \sin 2\pi u \cos 2\pi v$$
$$z = r \sin 2\pi u \sin 2\pi v$$

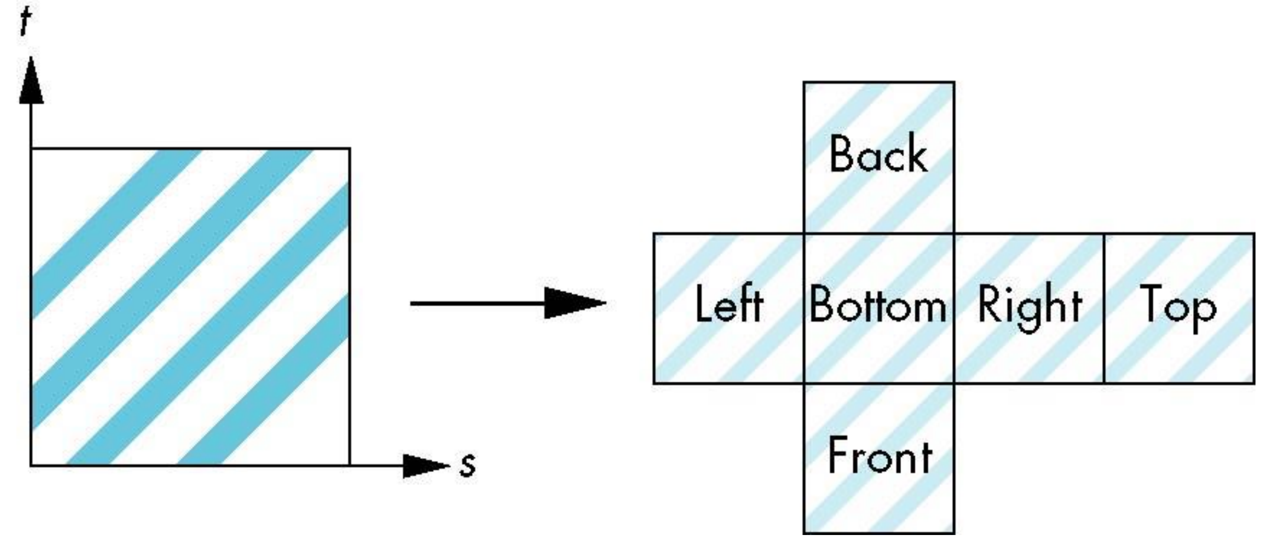
in a similar manner to the cylinder
but have to decide where to put
the distortion

Spheres are used in environmental maps



Box Mapping

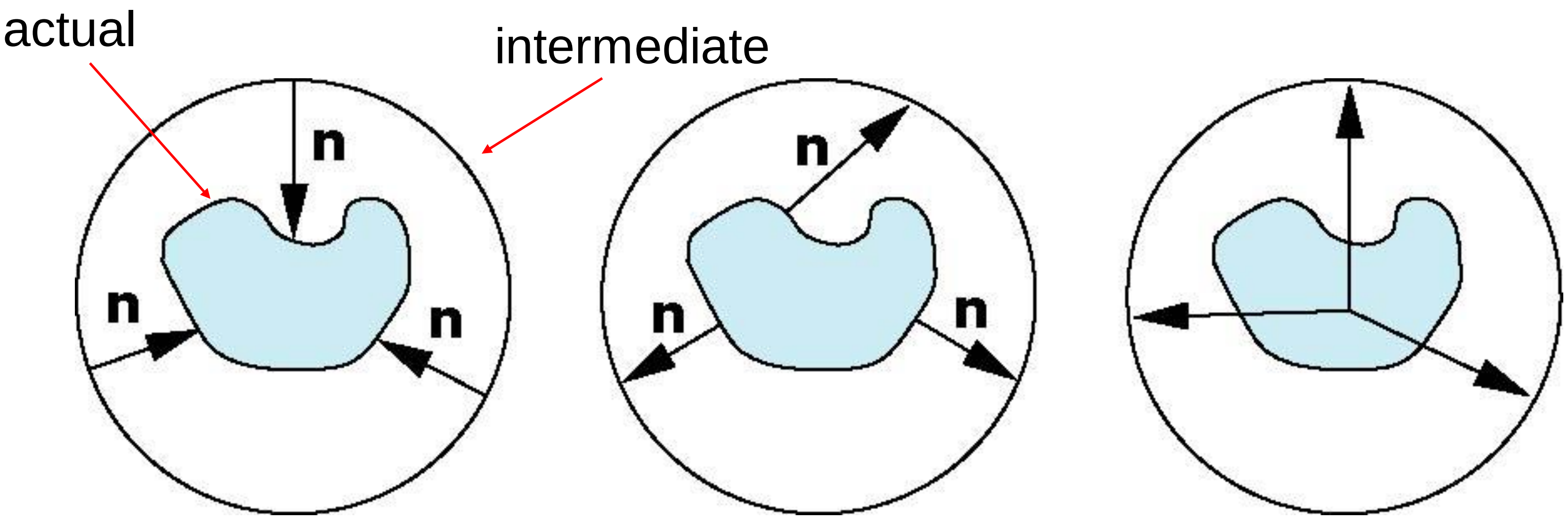
- Easy to use with simple orthographic projection
- Also used in environment maps



Other mapping strategy

Second Mapping

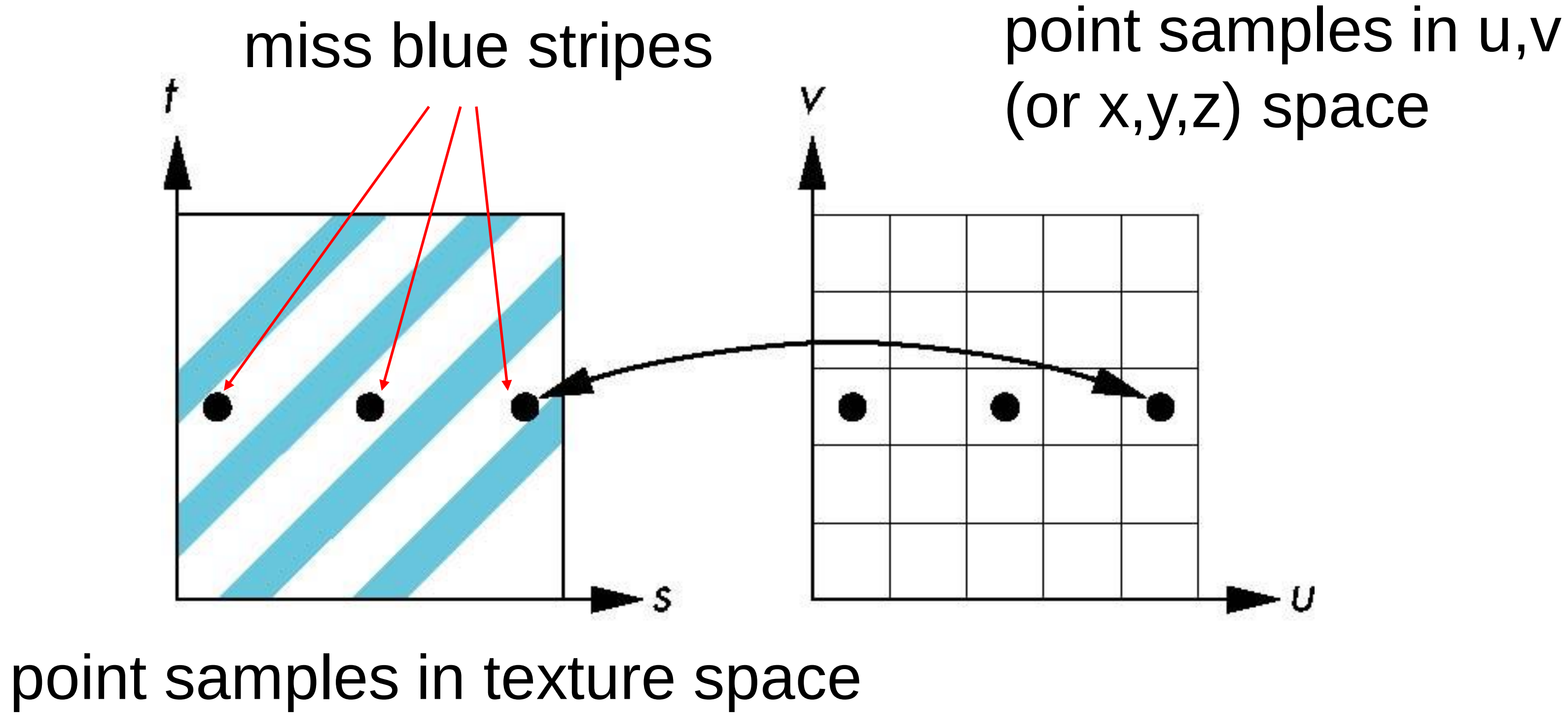
- Map from intermediate object to actual object
  - Normals from intermediate to actual
  - Normals from actual to intermediate
  - Vectors from center of intermediate



E. Angel and D. Shreiner: Interactive Computer Graphics 6E © Addison-Wesley 2012

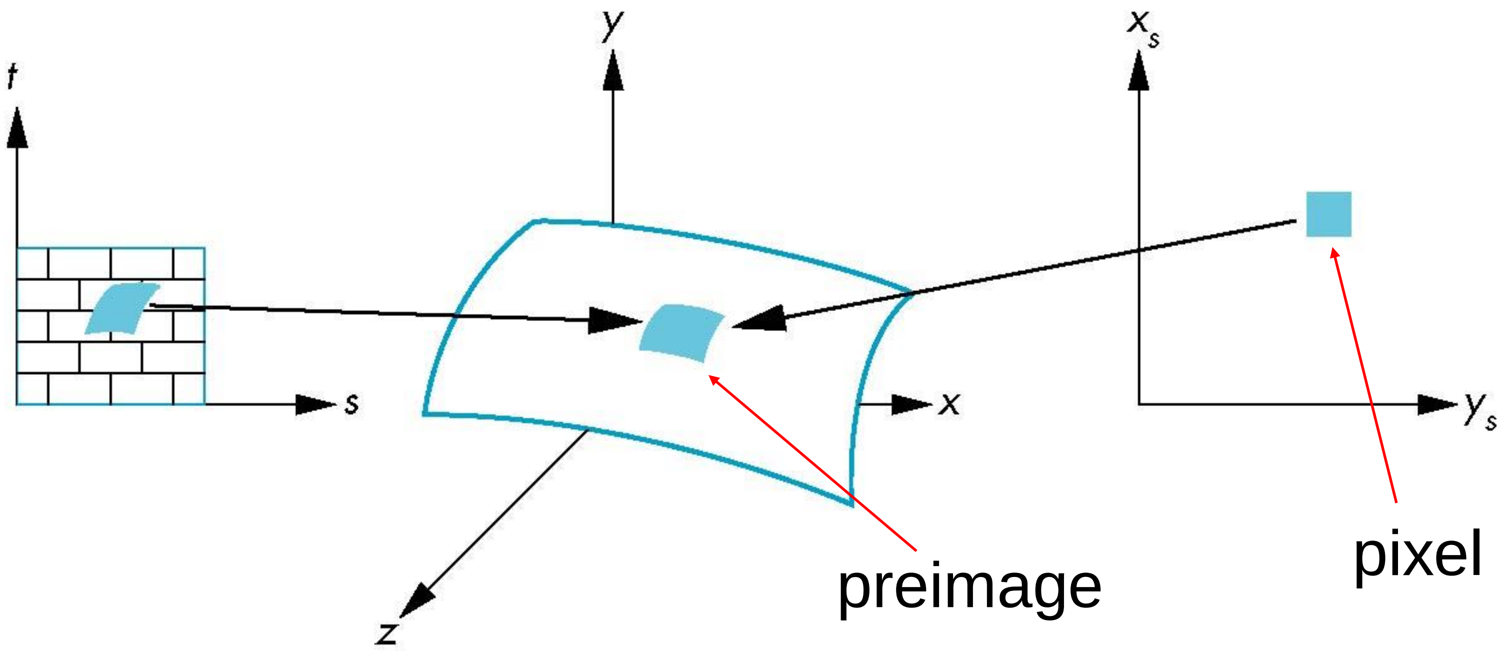
Aliasing

- Point sampling of the texture can lead to aliasing errors



Area Averaging

A better but slower option is to use *area averaging*



Note that *preimage* of pixel is curved

OpenGL Texture Mapping

Ed Angel

Professor Emeritus of Computer Science

University of New Mexico

Objectives

- Introduce the OpenGL texture functions and options

Basic Strategy

Three steps to applying a texture

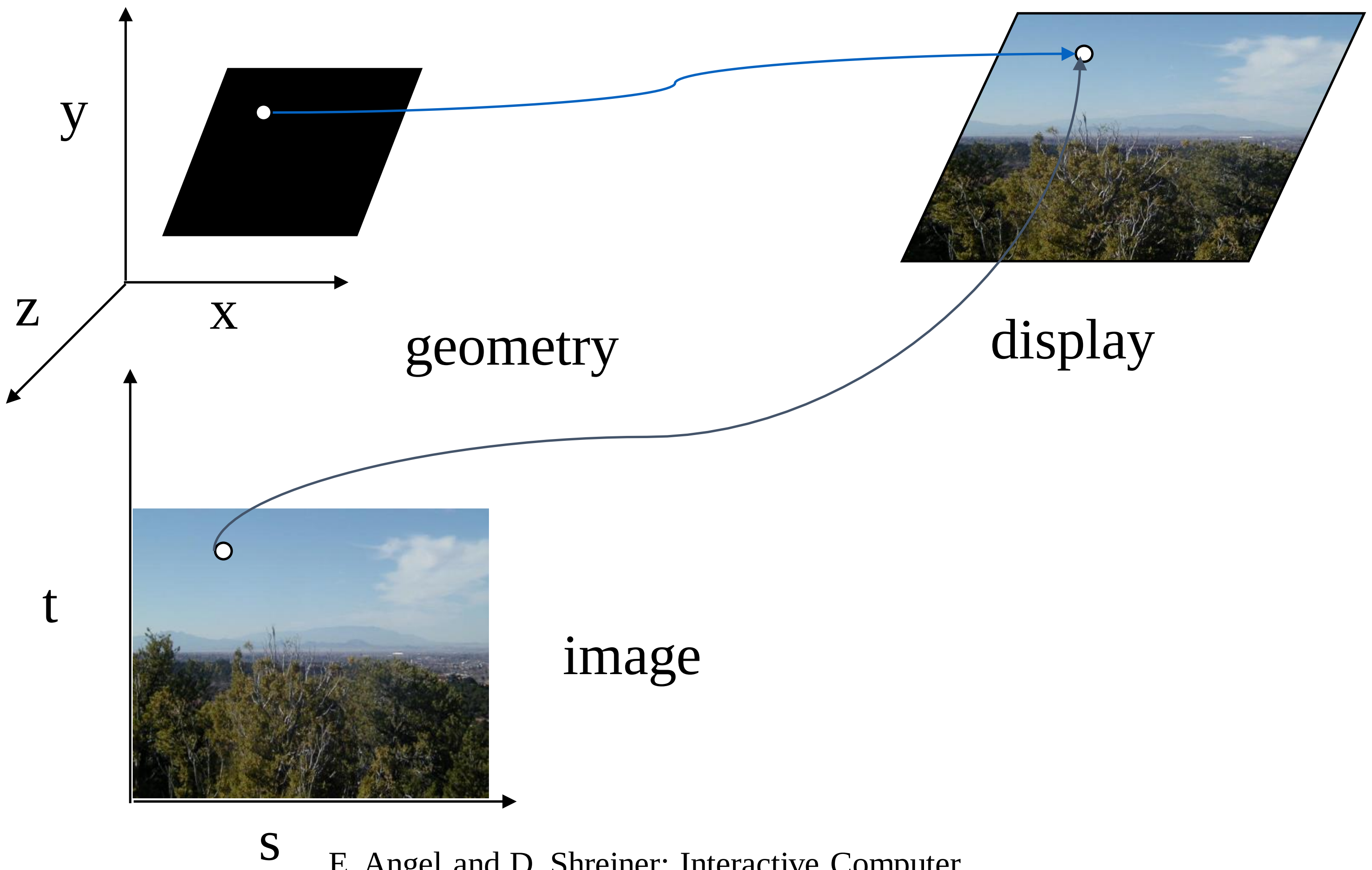
1. specify the texture
   - read or generate image
   - assign to texture
   - enable texturing
2. assign texture coordinates to vertices
   - Proper mapping function is left to application
3. specify texture parameters
   - wrapping, filtering

A Little More Detail

- 1. Load Bitmap
- 2. Generate a Texture Handle
  - `glGenTextures(1, &temptex);`
- 3. Bind and Configure
  - `glBindTexture (GL_TEXTURE_2D, nNewTextureID);`
  - `glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);`
  - `glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);`
  - `glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);`
  - `glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);`
- 4. Build the Texture in OpenGL

```
- gluBuild2DMipmaps (GL_TEXTURE_2D, nBPP, nWidth, nHeight,
-                    (nBPP == 3 ? GL_RGB : GL_RGBA),GL_UNSIGNED_BYTE,
-                    pData);
```

Texture Mapping



E. Angel and D. Shreiner: Interactive Computer Graphics 6E © Addison-Wesley 2012

Texture Example

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

Specifying a 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
  - Scanned image
  - Generate by application code
- Enable texture mapping
  - **glEnable(GL_TEXTURE_2D)**
  - OpenGL supports 1-4 dimensional texture maps

Define Image as a Texture

```
glTexImage2D( target, level, components,
     w, h, border, format, type, texels );
```

`target:` type of texture, e.g. **`GL_TEXTURE_2D`**

`level:` used for mipmapping (discussed later)

`components:` elements per texel

`w, h:` width and height of **`texels`** in pixels

`border:` used for smoothing (discussed later)

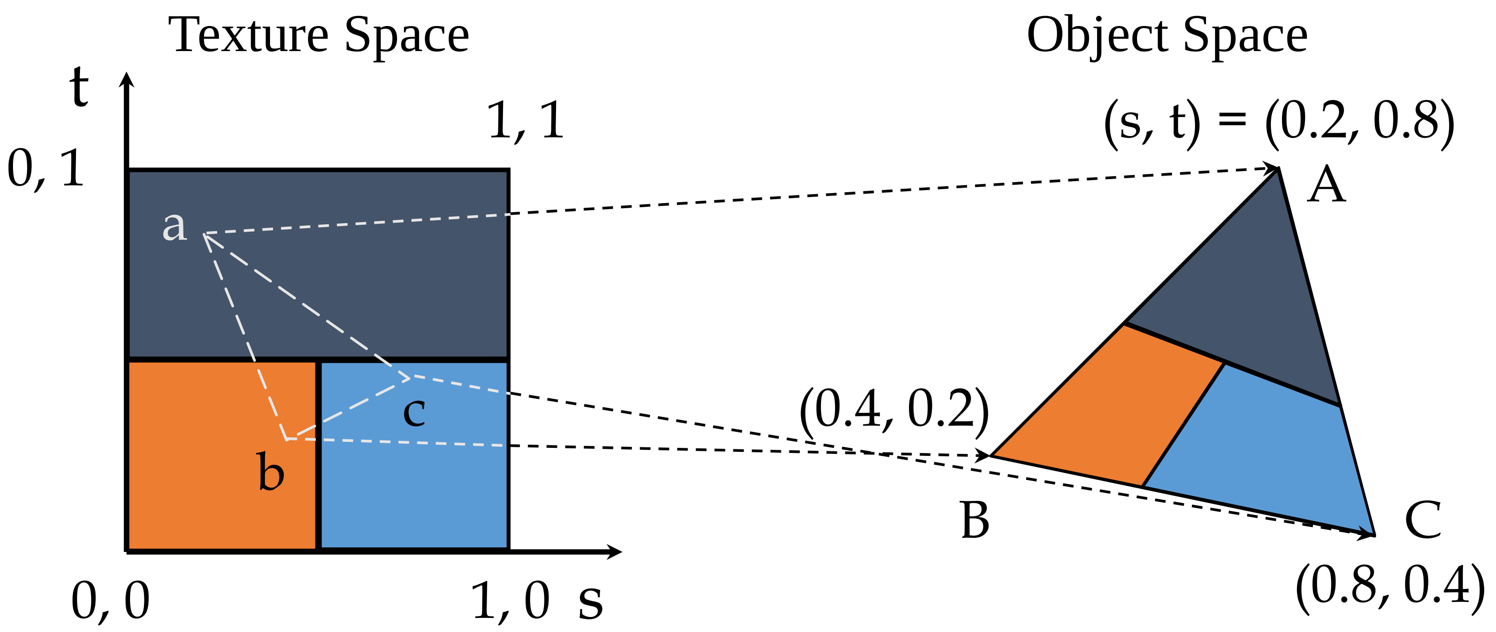
`format and type:` describe texels

`texels:` pointer to texel array

```
glTexImage2D(GL_TEXTURE_2D, 0, 3, 512, 512, 0,
  GL_RGB, GL_UNSIGNED_BYTE, my_texels);
```

Mapping a Texture

- Based on parametric texture coordinates
- **`glTexCoord*()`** specified at each vertex

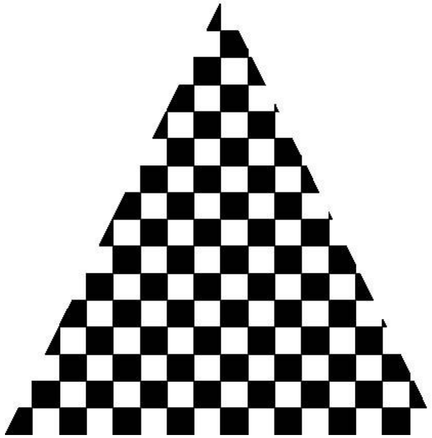


Interpolation

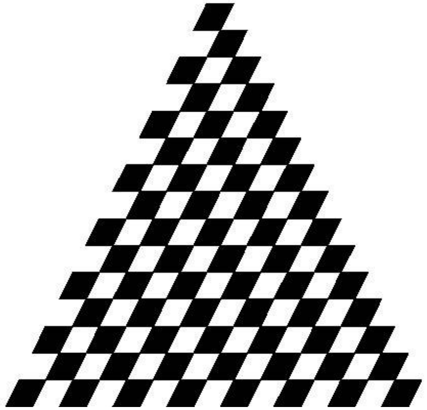
OpenGL uses interpolation to find proper texels from specified texture coordinates

Can be distortions

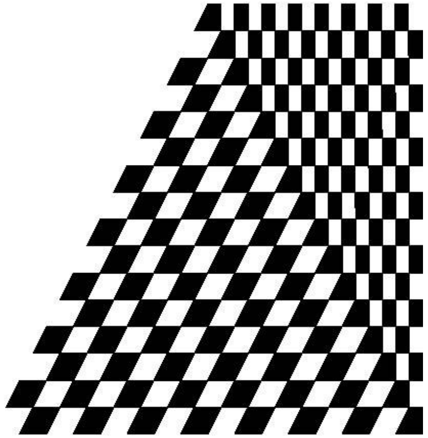
good selection
of tex coordinates



poor selection
of tex coordinates



texture stretched
over trapezoid
showing effects of
bilinear interpolation



Texture Parameters

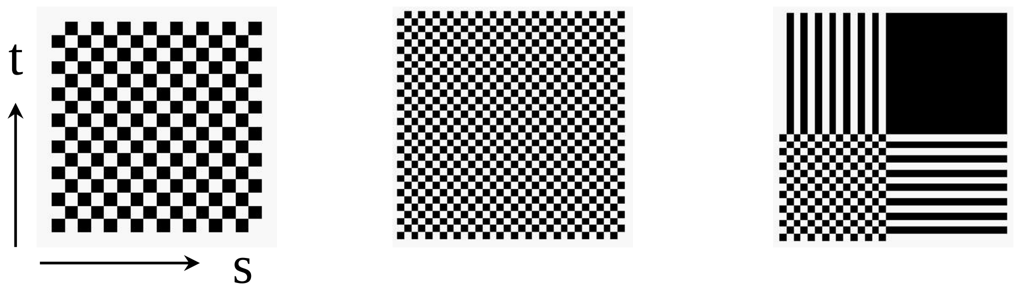
- OpenGL has a variety of parameters that determine how texture is applied
    - Wrapping parameters determine what happens if s and t are outside the (0,1) range
    - Filter modes allow us to use area averaging instead of point samples
    - Mipmapping allows us to use textures at multiple resolutions
    - Environment parameters determine how texture mapping interacts with shading

Wrapping Mode

Clamping: if $s,t > 1$ use 1, if $s,t < 0$ use 0

Wrapping: use s,t modulo 1

```
glTexParameteri( GL_TEXTURE_2D,
     GL_TEXTURE_WRAP_S, GL_CLAMP )
glTexParameteri( GL_TEXTURE_2D,
     GL_TEXTURE_WRAP_T, GL_REPEAT )
```



texture

GL_REPEAT wrapping

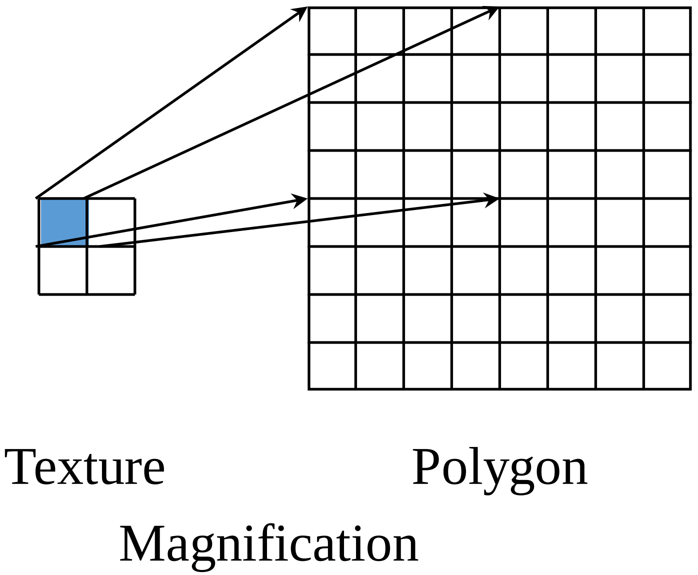
GL_CLAMP wrapping

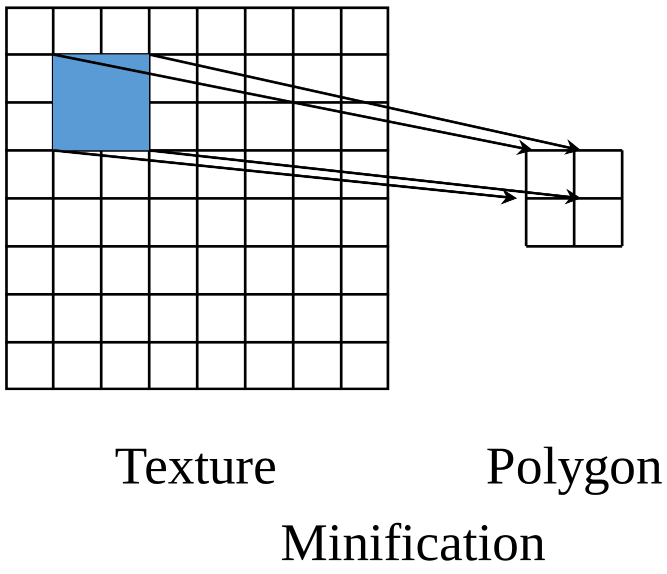
GL_REPEAT GL_MIRRORED_REPEAT GL_CLAMP_TO_EDGE GL_CLAMP_TO_BORDER

Magnification and Minification

More than one texel can cover a pixel (*minification*) or more than one pixel can cover a texel (*magnification*)

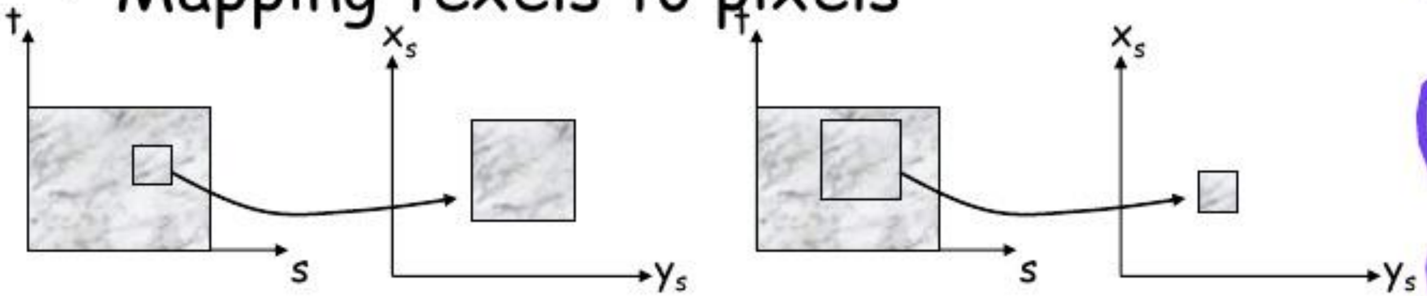
Can use point sampling (nearest texel) or linear filtering (2 x 2 filter) to obtain texture values





2D texture mapping (3)

- Mapping texels to pixels



Magnification: large

Minification: min

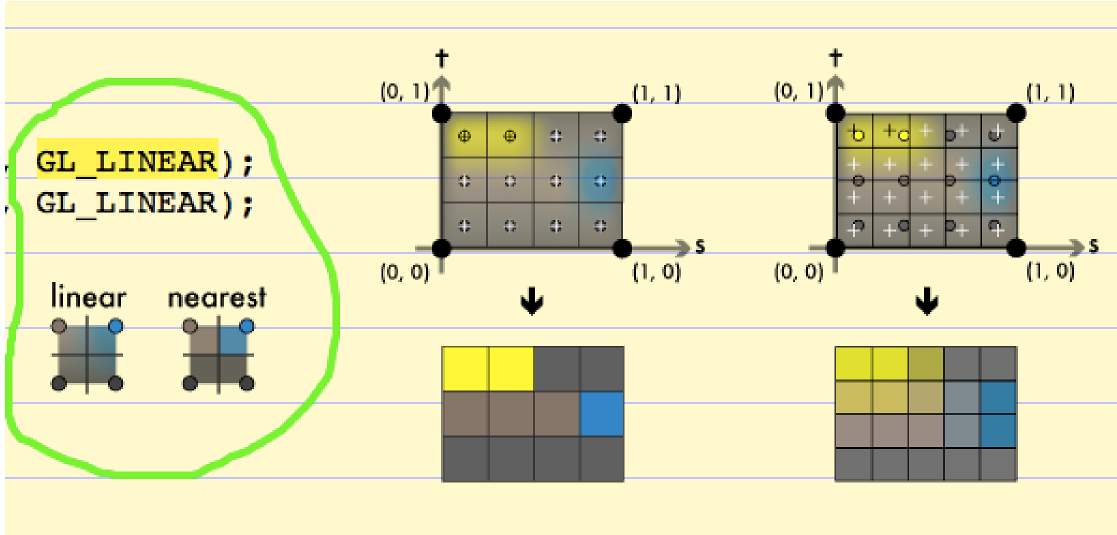
```
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,
GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,
GL_NEAREST);
```

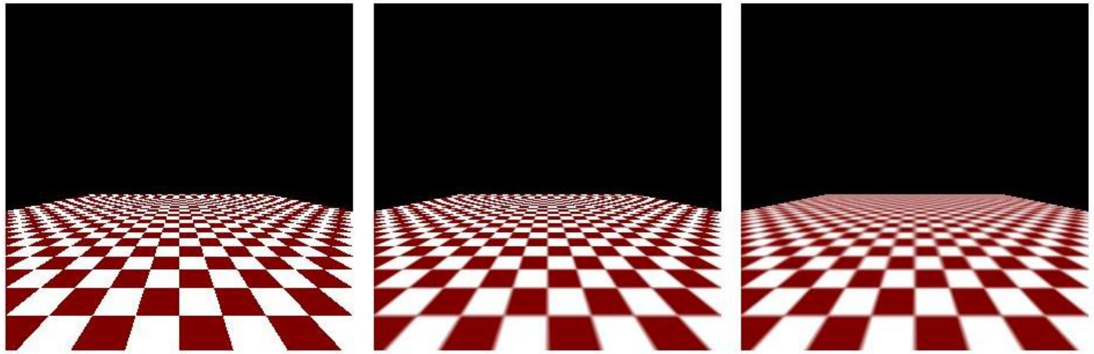
Filter Modes

Modes determined by

- **`glTexParameteri( target, type, mode )`**

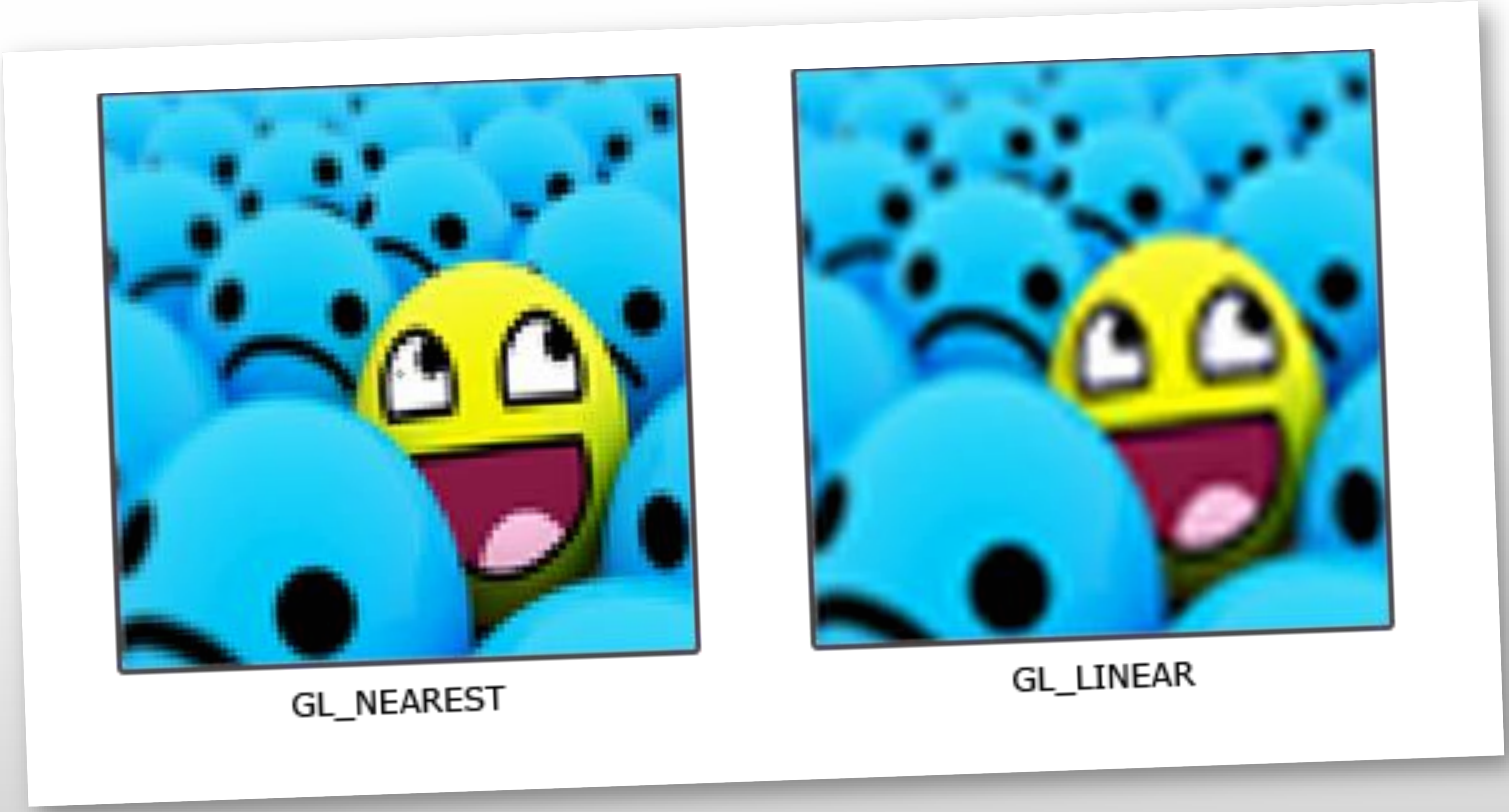
```
glTexParameteri(GL_TEXTURE_2D, GL_TEXURE_MAG_FILTER,
                GL_NEAREST);

glTexParameteri(GL_TEXTURE_2D, GL_TEXURE_MIN_FILTER,
                GL_LINEAR);
```





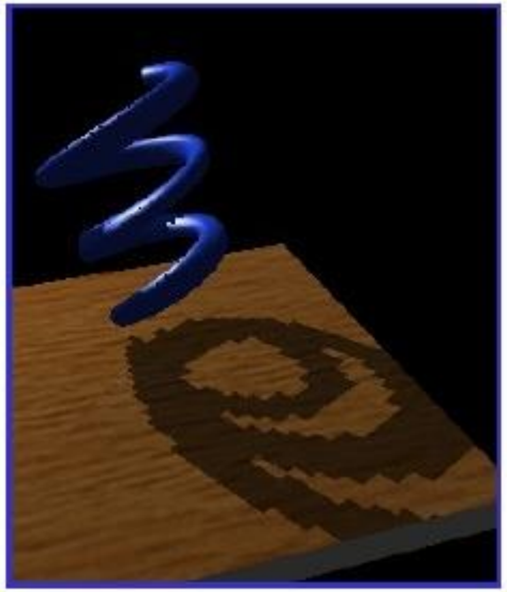
GL_NEAREST GL_LINEAR Mipmappping

https://learnopengl.com/Getting-started/Textures



Hardware Shadow Map Filtering Example

GL_NEAREST: blocky



GL_LINEAR: antialiased edges

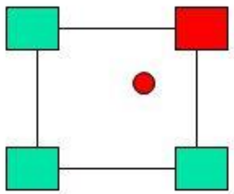
Low shadow map resolution used to heightens filtering artifacts

nVIDIA.

Texture mapping parameters(3)

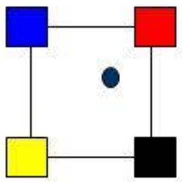
- OpenGL texture filtering:

1) Nearest Neighbor (lower image quality)



glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);

2) Linear interpolate the neighbors (better quality, slower)



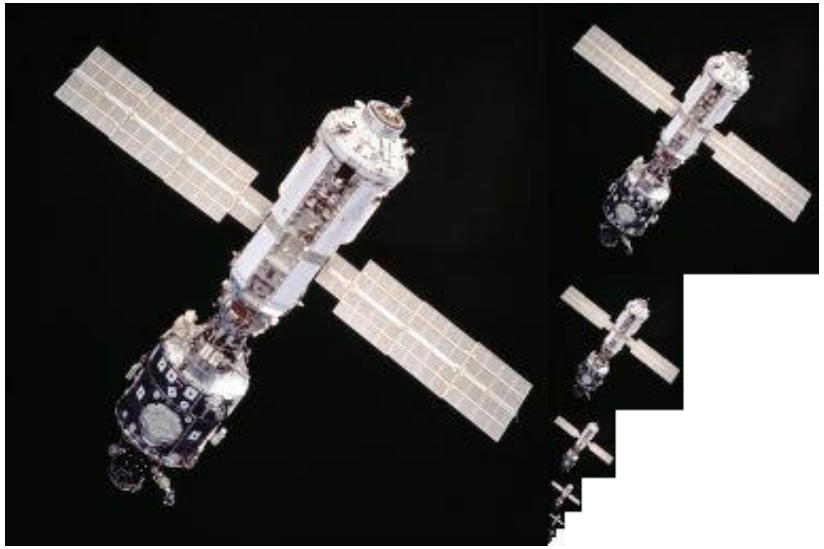
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR)

Or GL_TEXTURE_MAX_FILTER

Mipmapped Textures

- *Mipmapping* allows for prefiltered texture maps of decreasing resolutions
- Lessens interpolation errors for smaller textured objects
- Declare mipmap level during texture definition

```
glTexImage2D(
  GL_TEXTURE_*D, level, … )
```



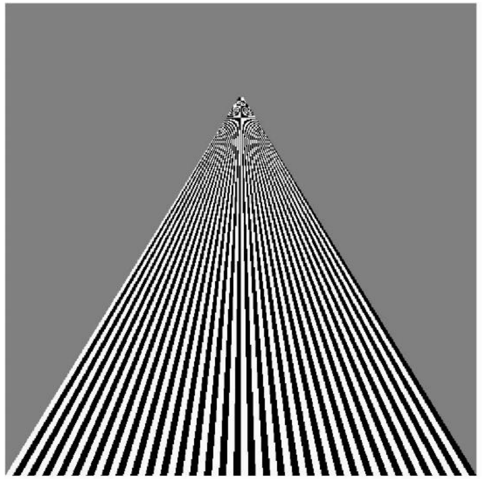
mipmaps

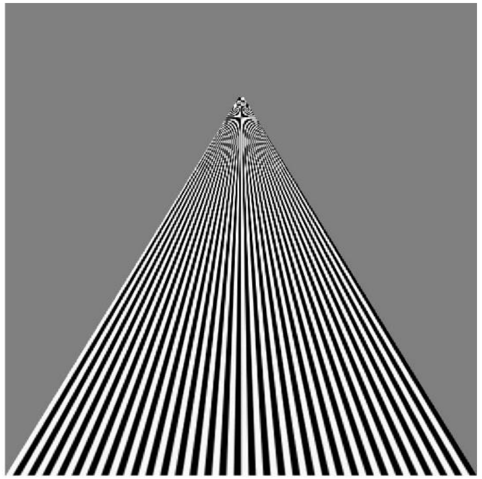
In computer graphics, **mipmaps**

(also **MIP maps**) or **pyramids** [1][2][3] are pre-calculated, optimized sequences of images, each of which is a progressively lower resolution representation of the same image.

Example

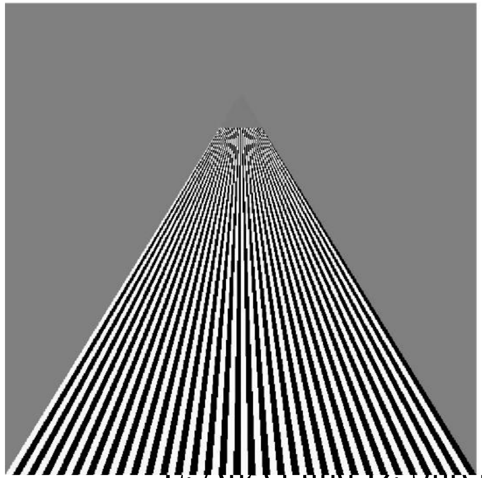
point sampling

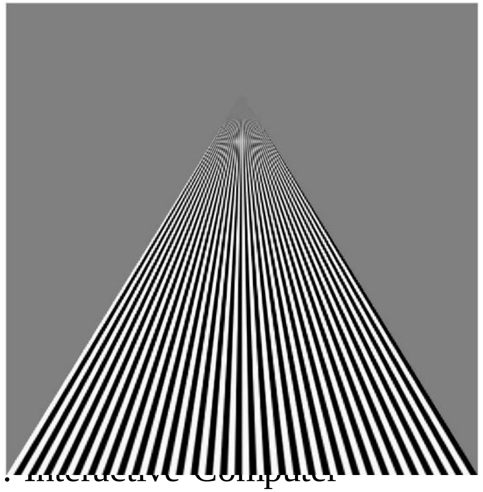




linear filtering

mipmapped point sampling



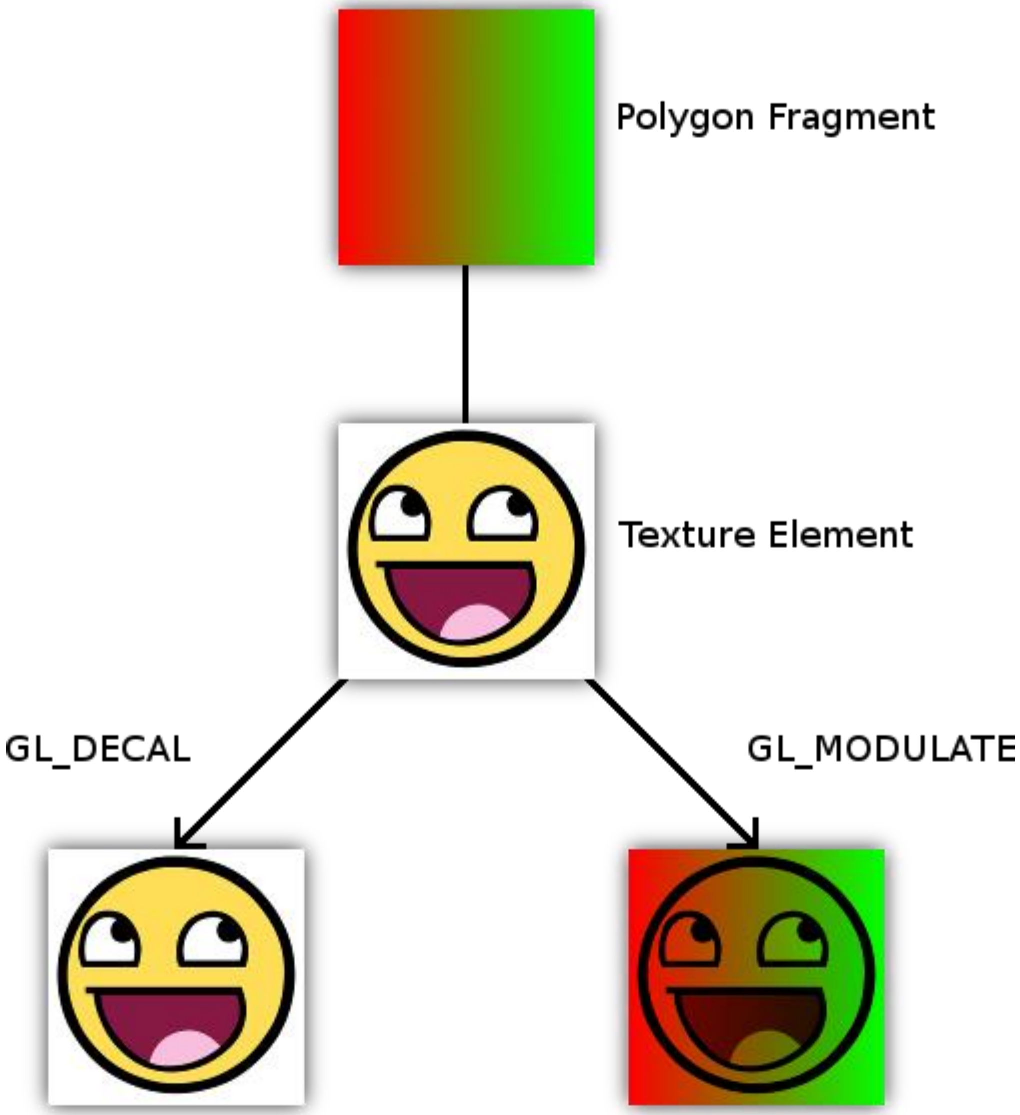


mipmapped linear filtering

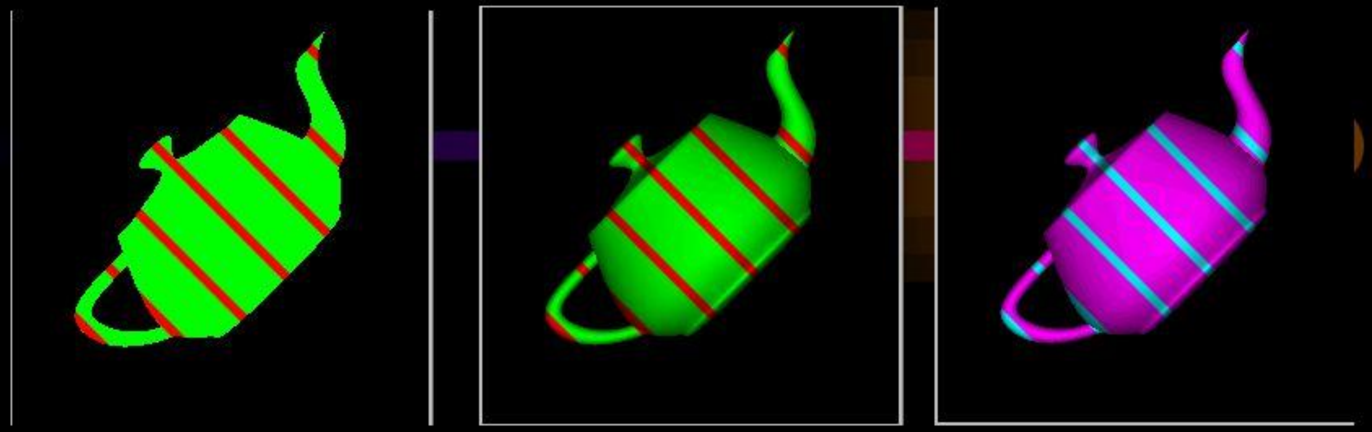
Texture Functions (GL_TEXTURE_ENV_MODE)

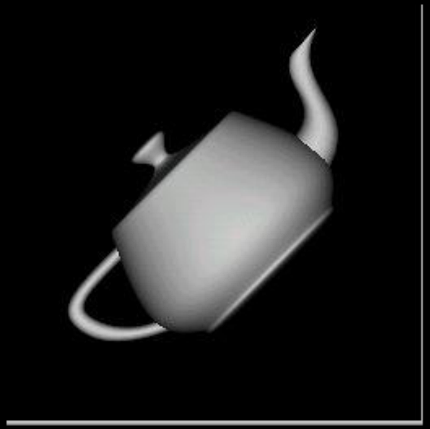
Controls how texture is applied

- `glTexEnv{fi}[v]( GL_TEXTURE_ENV, prop, param )`
- `GL_TEXTURE_ENV_MODE` modes
  - `GL_MODULATE`: modulates with computed shade
  - `GL_BLEND`: blends with an environmental color
  - `GL_REPLACE`: use only texture color
  - `GL(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);`
- Set blend color with `GL_TEXTURE_ENV_COLOR`


Polygon Fragment
Texture Element
GL_DECAL
GL_MODULATE

GL_DECAL **GL_MODULATE** **GL_BLEND**





Non-textured
Shaded Teapot
See texgen.c

GL_RGB	C = Ct, A = Af	C = Cf(1-Ct) + CcCt, A = Af
GL_RGBA	C = Cf(1-At) + CtAt, A = Af	C = Cf(1-Ct) + CcCt, A = AfAt

Using Texture Objects

1. specify textures in texture objects
2. set texture filter
3. set texture function
4. set texture wrap mode
5. set optional perspective correction hint
6. bind texture object
7. enable texturing
8. supply texture coordinates for vertex
   - coordinates can also be generated