CS418
Discussion Section
(III)
Shading & Texture

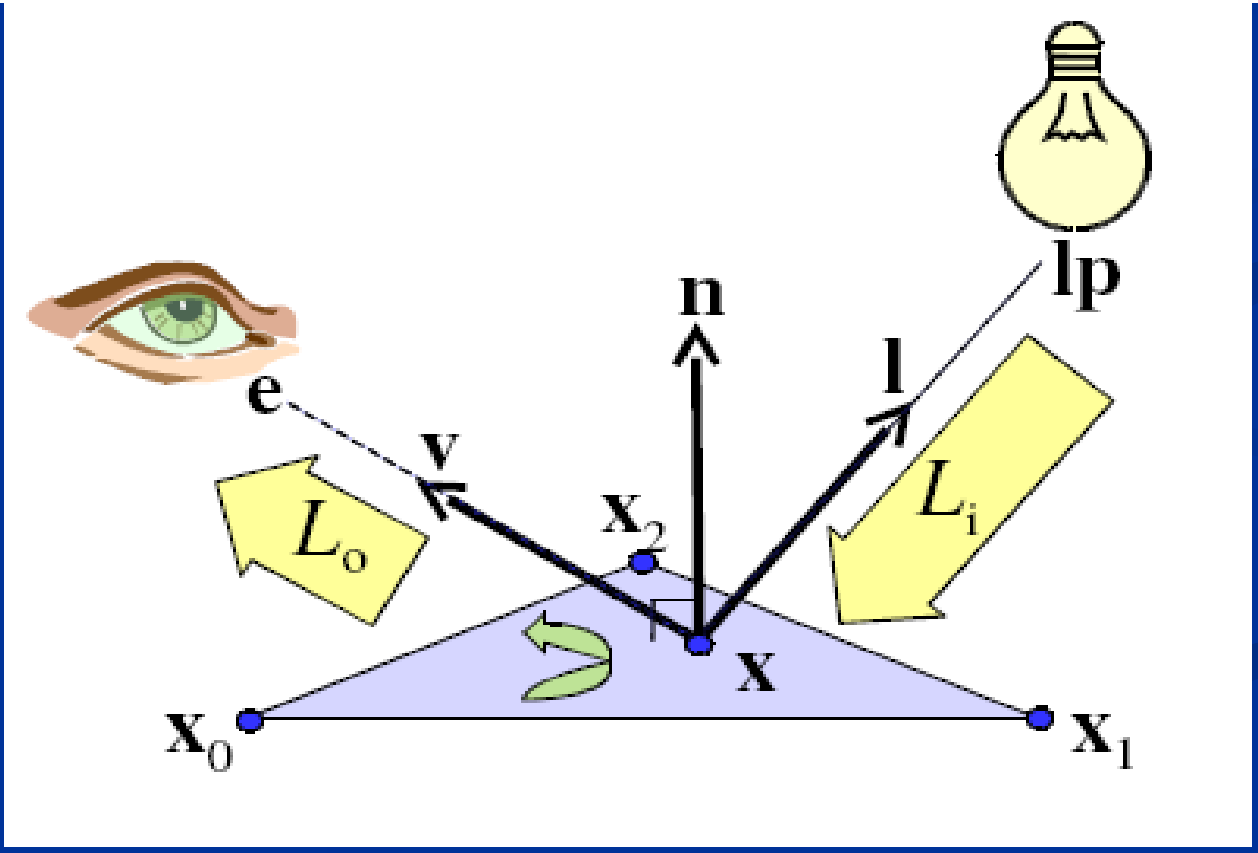
Presented by : Wei-Wen Feng
2/18/2009 (Part of slides from Brett Jones & John Hart)

MP2 : Flight Simulator

- Due on March 2 (Mon)
  - Terrain Texturing / Lighting
  - Camera Control (Flight Simulator)
  - Bonus (20%)
    - Multiple Object rendering (Model Transformation)
    - Object picking/control

Lighting

v	– view vector:	$\mathbf{v} = (\mathbf{e} - \mathbf{x})/\|\mathbf{e} - \mathbf{x}\|$
l	– light vector:	$\mathbf{l} = (\mathbf{lp} - \mathbf{x})/\|\mathbf{lp} - \mathbf{x}\|$
n	– normal vector:	$\mathbf{n} = (\mathbf{x}_1\text{-}\mathbf{x}_0)\times(\mathbf{x}_2\text{-}\mathbf{x}_0)/\|(\mathbf{x}_1\text{-}\mathbf{x}_0)\times(\mathbf{x}_2\text{-}\mathbf{x}_0)\|$

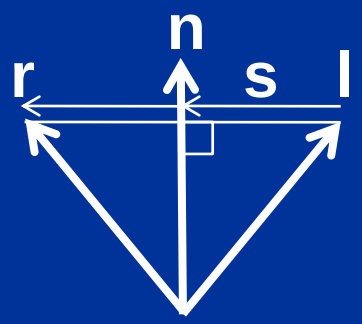


Lambertian (Diffuse)

$L_o = L_i k_d \mathbf{c}_d \cos\theta$
$= L_i k_d \mathbf{c}_d \mathbf{n}\cdot\mathbf{l}$

Specular Reflection

$$L_o = L_i k_s \mathbf{c}_s \cos^n \phi$$
$$= L_i k_s \mathbf{c}_s (\mathbf{v}\cdot\mathbf{r})^n$$



$$\mathbf{s} = (\mathbf{n}\cdot\mathbf{l})\mathbf{n} - \mathbf{l}$$
$$\mathbf{r} = \mathbf{l} + 2\mathbf{s}$$
$$= \mathbf{l} + 2(\mathbf{n}\cdot\mathbf{l})\mathbf{n} - 2\mathbf{l}$$
$$= 2(\mathbf{n}\cdot\mathbf{l})\mathbf{n} - \mathbf{l}$$

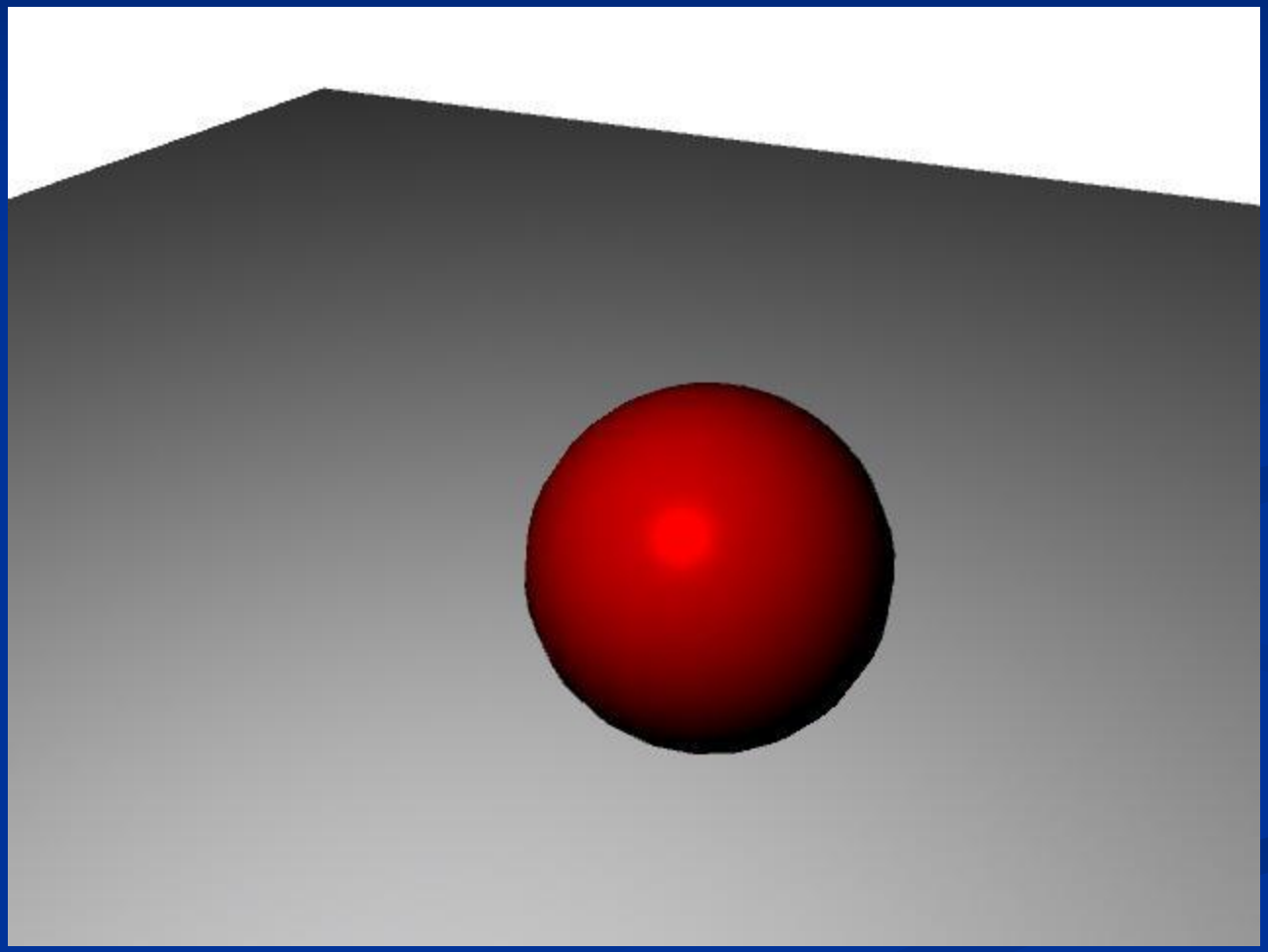
Ambient

OpenGL Lighting

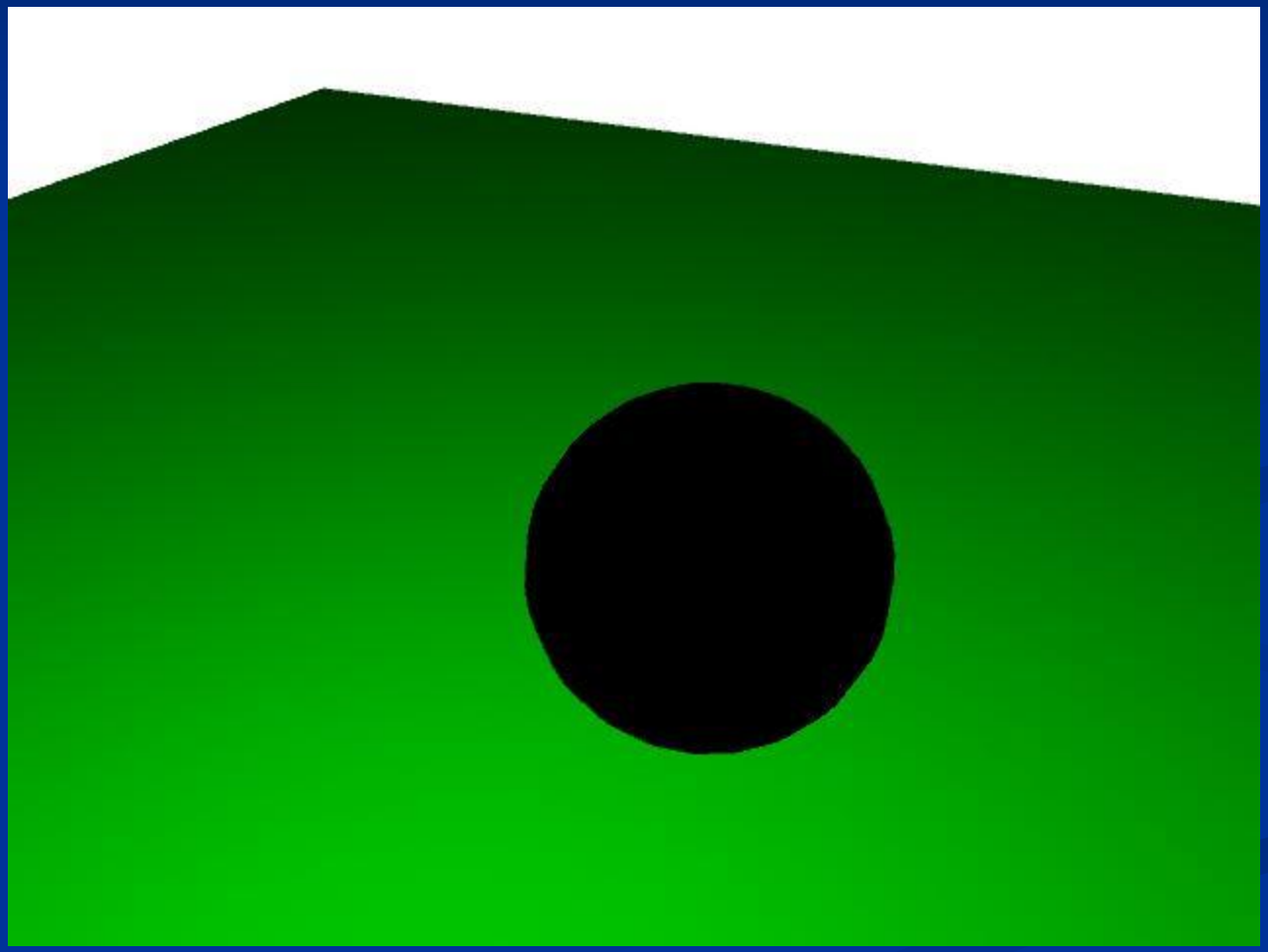
OpenGL: $$L_o = k_a L_{\#a} + L_{\#d} k_d \, \mathbf{n} \cdot \mathbf{l} + L_{\#s} k_s \, (\mathbf{v} \cdot \mathbf{r})^n$$

- Materials
  - Define the surface properties of an object (k_a, k_d, k_s)
- Lights
  - Define the properties of the emitted light (L#a, L#d, L#s)

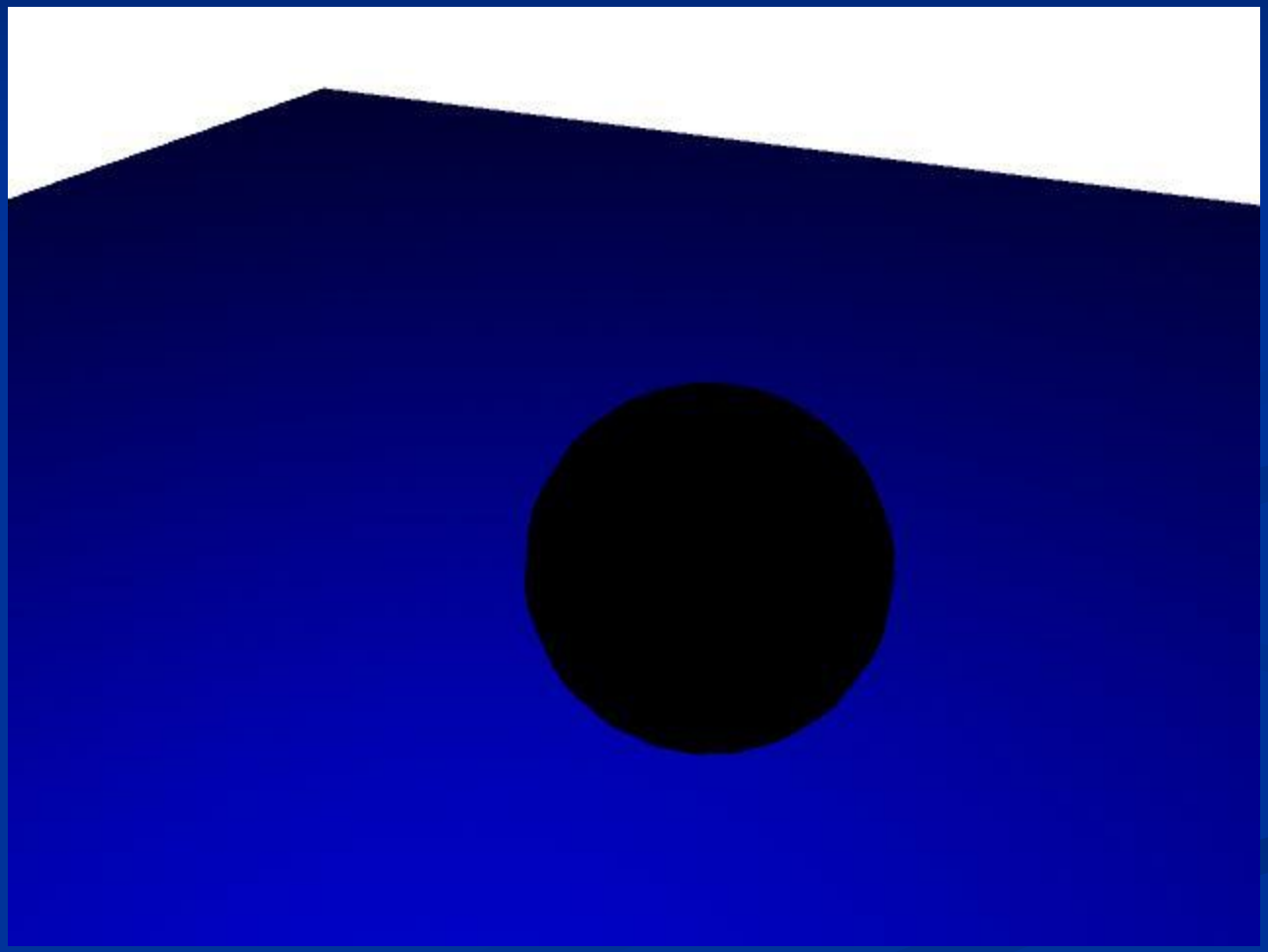
Red Ball + White Light



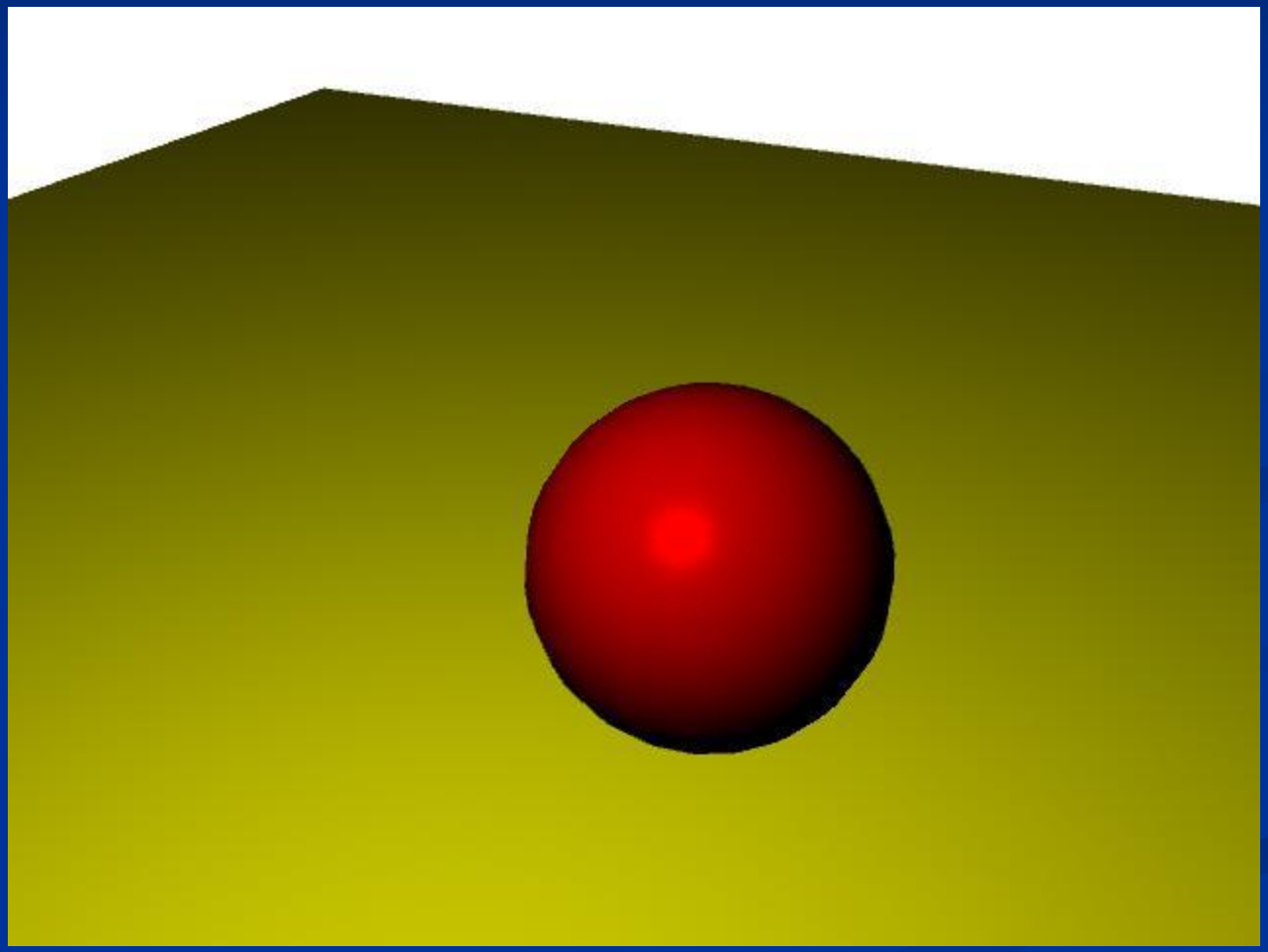
Red Ball + Green Light



Red Ball + Blue Light



Red Ball + Yellow Light



Lighting

- Define the surface properties of a primitive
  - `glMaterialfv( face, property, value );`
  - Follow Phong lighting model for parameters
  - You can define different material for front/back faces.

`GL_DIFFUSE`	Base color
`GL_SPECULAR`	Highlight Color
`GL_AMBIENT`	Low-light Color
`GL_EMISSION`	Glow Color
`GL_SHININESS`	Surface Smoothness

OpenGL Materials

```
GLfloat mat_ambient[] = { 0.1, 0.1, 0.1, 1.0 };
GLfloat mat_diffuse[] = { 0.8, 0.8, 0.8, 1.0 };
GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat mat_shininess[] = { 50.0 };

glMaterialfv(GL_FRONT,GL_AMBIENT, mat_ambient)
glMaterialfv(GL_FRONT,GL_DIFFUSE, mat_diffuse)
glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular)
glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
```

Lighting

- Define light source property
  - OpenGL support at least 8 lights. (GL_LIGHT0 ~ GL_LIGHTn)
  - `glLightfv( light, property, value );`
  - ***light*** specifies which light
  - OpenGL light can emit different colors for each material property
  - Lighting type
    - Point light
    - Directional light
    - Spot light

GL_DIFFUSE	Base color
GL_SPECULAR	Highlight Color
GL_AMBIENT	Low-light Color

Lighting

- Change Light Type : Set related properties in
  `glLightfv( light, property, value );`
- Point Light
  - Set position to (x,y,z,1.0)
- Directional Light
  - Set direction to (x,y,z,0)
- Spot Light
  - Set spot cut-off for point light

- Be careful when setting light positions
  - Light also get transformed by ModelView matrix.

OpenGL Lights

```
GLfloat light_ambient[] = { 0.0, 0.0, 0.0, 1.0 };
GLfloat light_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat light_specular[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };

glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
glLightfv(GL_LIGHT0, GL_POSITION, light_position);}
```

Lighting

- Turn on the Light in OpenGL after setting up each light source.
- Flip each light's switch

```
glEnable( GL_LIGHTn );
```

- Turn on the power

```
glEnable( GL_LIGHTING );
```

OpenGL Lighting

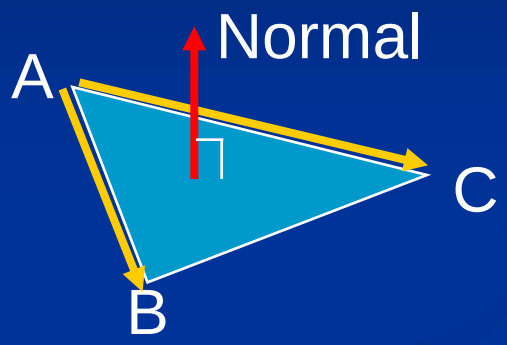
```
glShadeModel (GL_SMOOTH);
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glBegin(GL_POLYGON);
  glNormal3f(nx0,ny0,nz0)
  glVertex3f(x0,y0,z0);

  glNormal3f(nx1,ny1,nz1)
  glVertex3f(x1,y1,z1);

  glNormal3f(nx2,ny2,nz2)
  glVertex3f(x2,y2,z2);
glEnd();
```

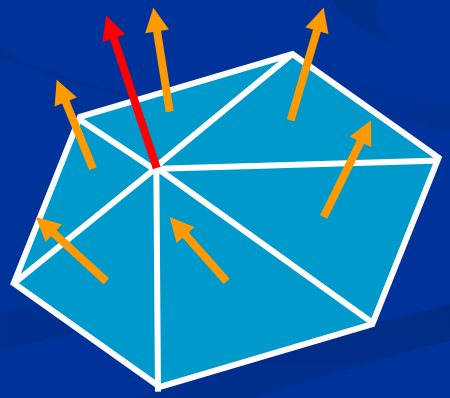
Normal Computation

- Normals define how a surface reflects light

  `glNormal3f( x, y, z )`

- Face Normal :
  - Cross-product of edge vectors
- Vertex Normal :
  - Average of adjacent face normals

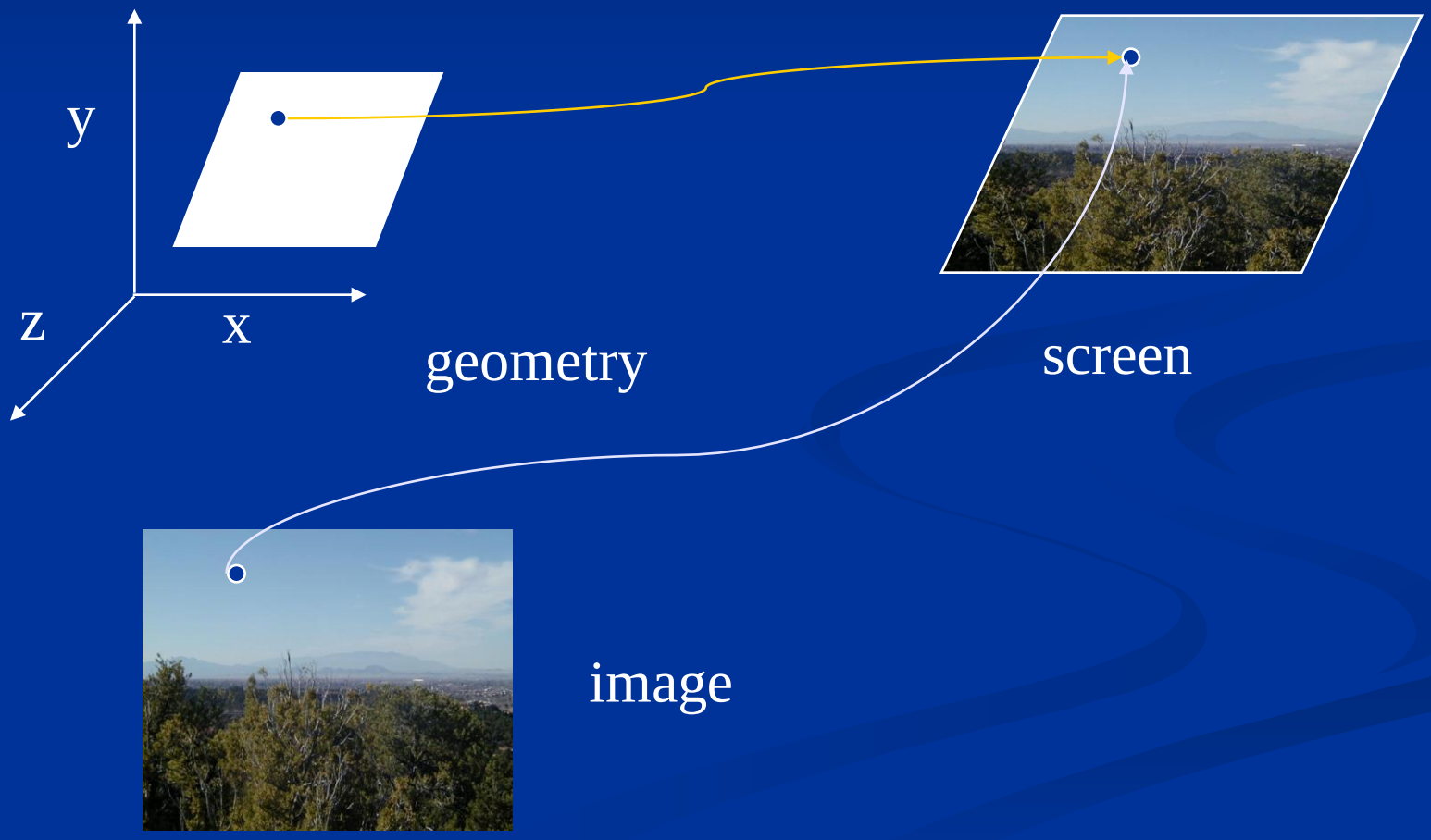




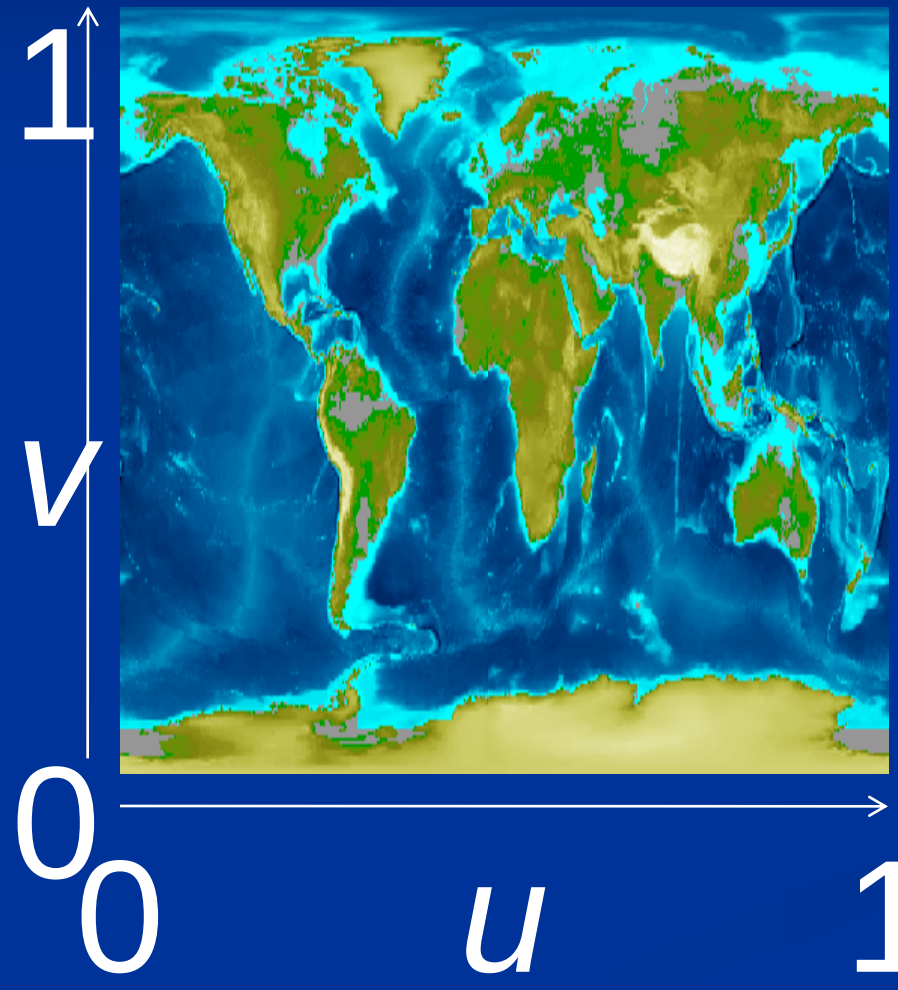
Phong Lighting Model

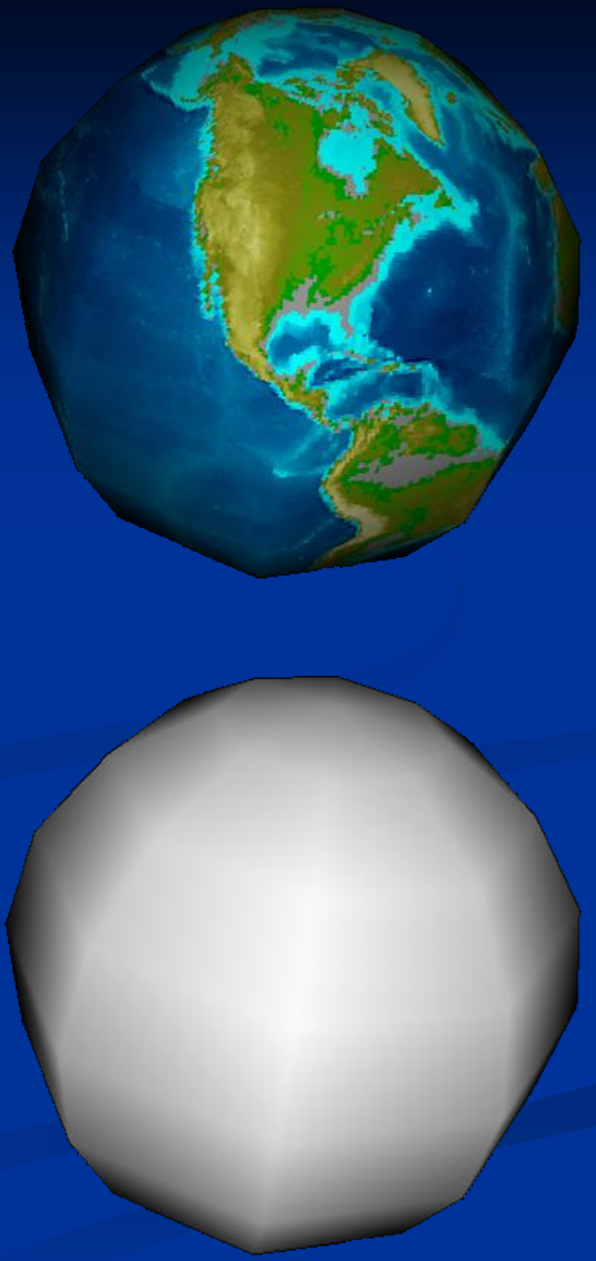
- Default OpenGL lighting.
- Simple model with no physical meaning.
- Usually result in a “plastic” look material.
- Cook-Torrence Demo

Texture Mapping

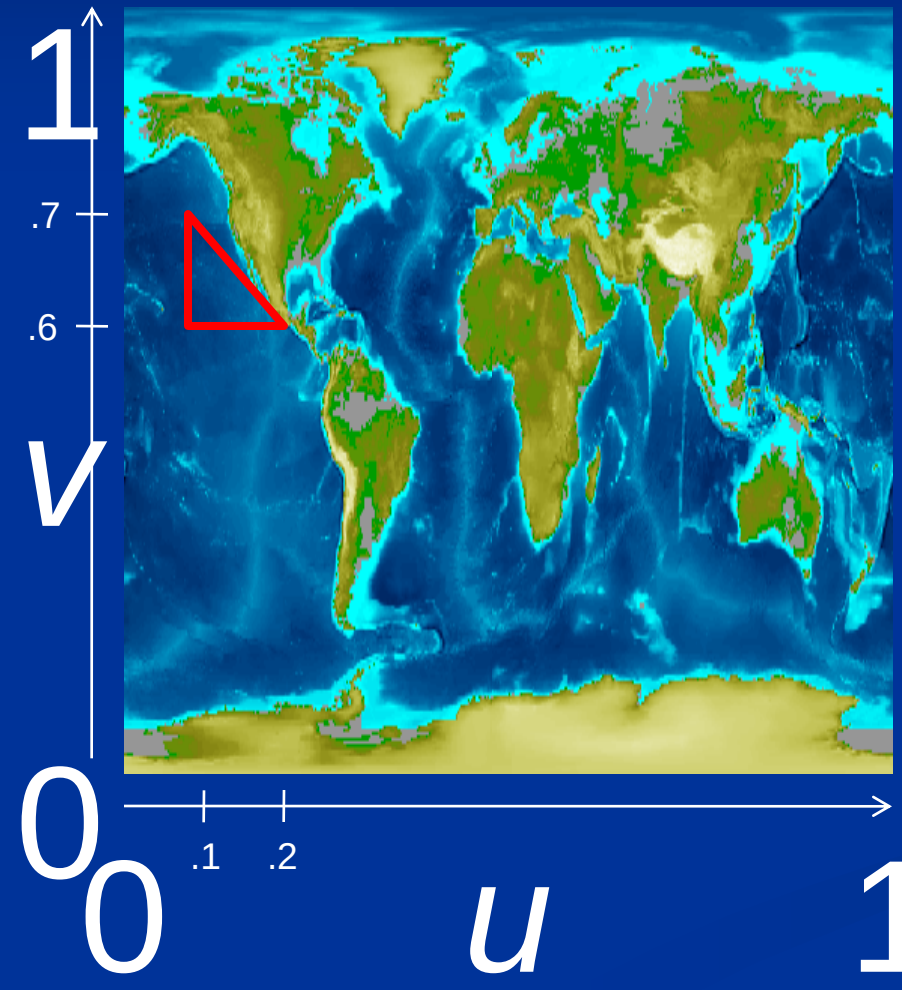


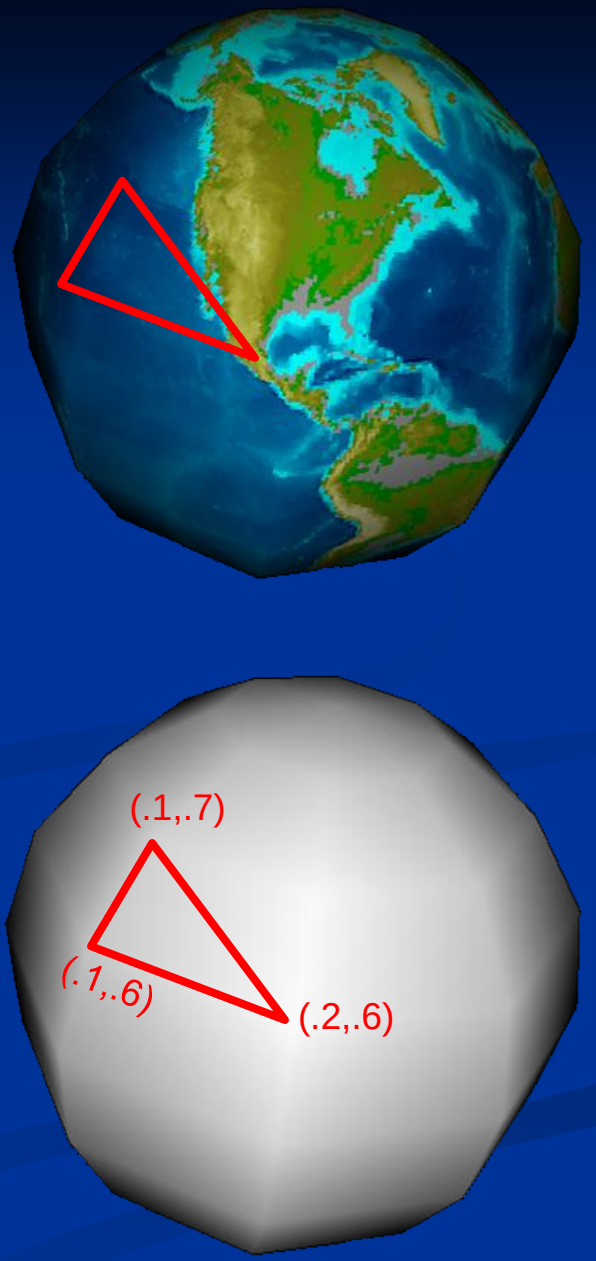
Texture Mapping





Texture Mapping





Texture Mapping

- Steps
  - Specify texture
    - Read or generate image
    - Link image to texture
  - Set texturing parameters.
  - Assign texture coordinates to vertices.
  - Draw the scene with texture mapping.

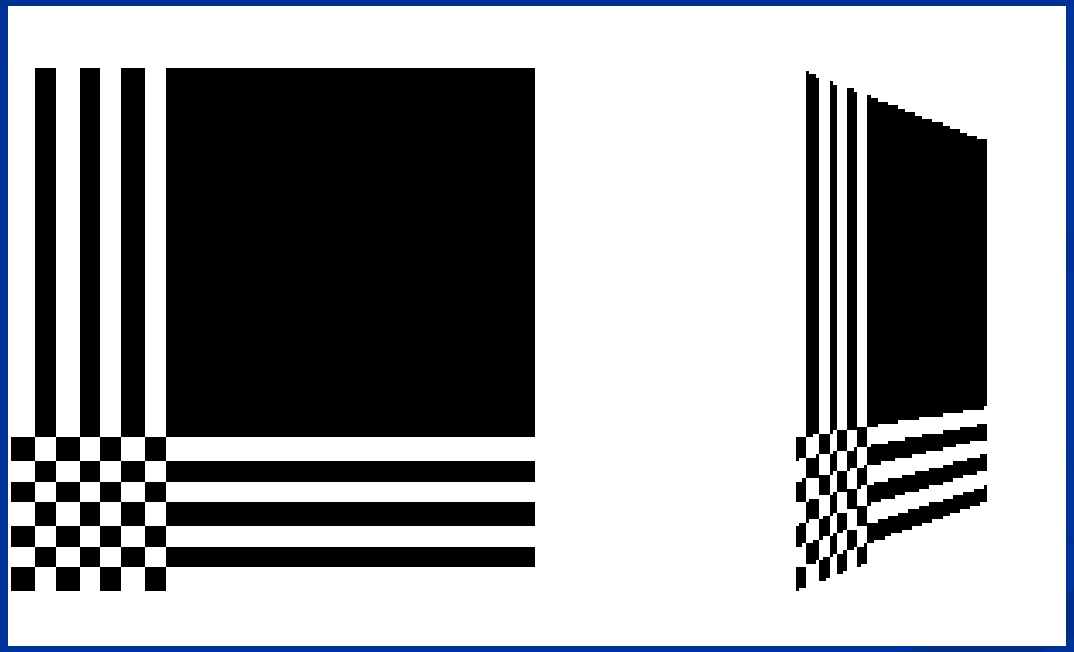
Texture Mapping

- Texture Objects
  - one image per texture object
  - Faster to use an existing object than reload a texture image
- Generate texture ***names***
  - **`glGenTextures( n, *texIds );`**
  - Only the name(s) is created. texId = zero is reserved.
  - No information about its data/dimension is created.
- Create/Bind textures with the ***name*** before using
  - **`glBindTexture( target, id );`**
  - Create a new texture object if called for first time
  - Activate the texture object with the same ID already exist.
  - If (id==0), unbind the current texture object

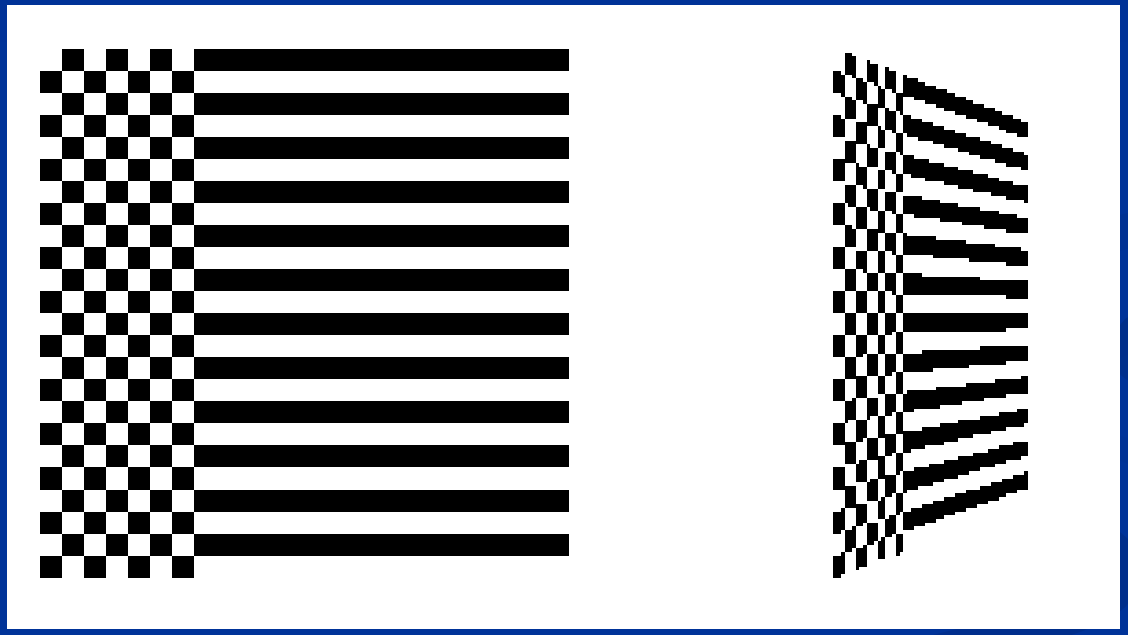
Set texturing parameters

```
// Set texturing parameters
glBindTexture(GL_TEXTURE_2D, texName);
glTexParameteri(GL_TEXTURE_2D,
   GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D,
   GL_TEXTURE_WRAP_T, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D,
   GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D,
   GL_TEXTURE_MIN_FILTER, GL_LINEAR);
```

Texturing Clamping/Repeating



Texturing Clamping/Repeating

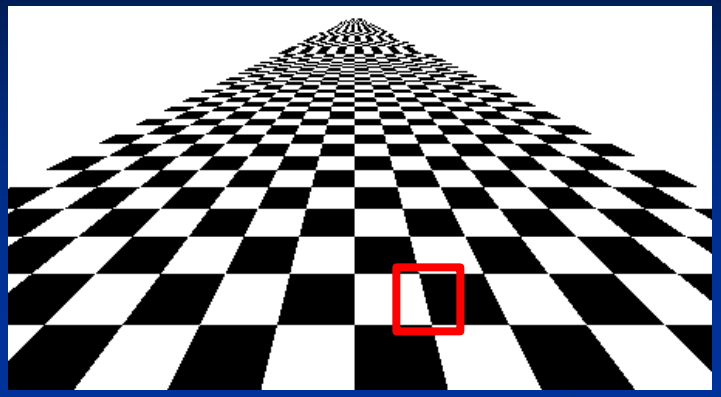


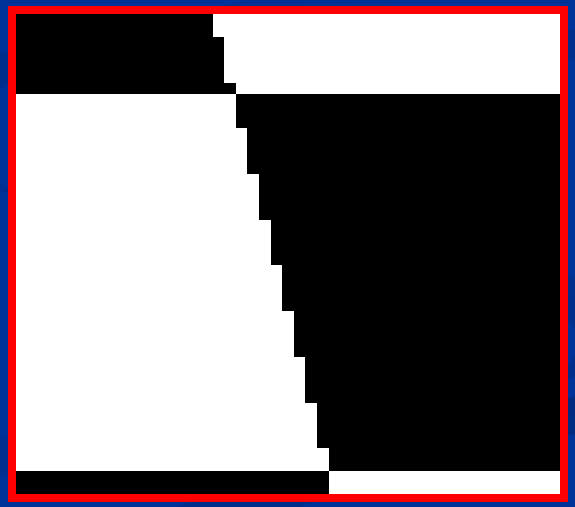
Texturing Clamping/Repeating

Parameter	Values
GL_TEXTURE_WRAP_S	GL_CLAMP, GL_REPEAT
GL_TEXTURE_WRAP_T	GL_CLAMP, GL_REPEAT

Magnification Aliasing

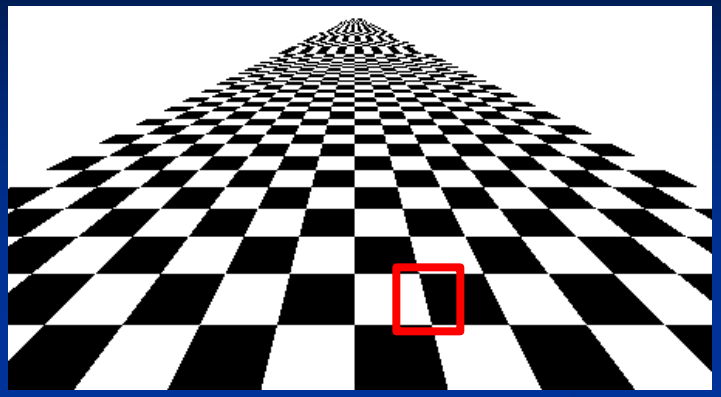
- "Jaggies" – lines have a staircased edge appearance
- Occur when a single texture sample (texels) projects to multiple screen pixels
- (Also occurs when rasterizing lines or polygon edges)

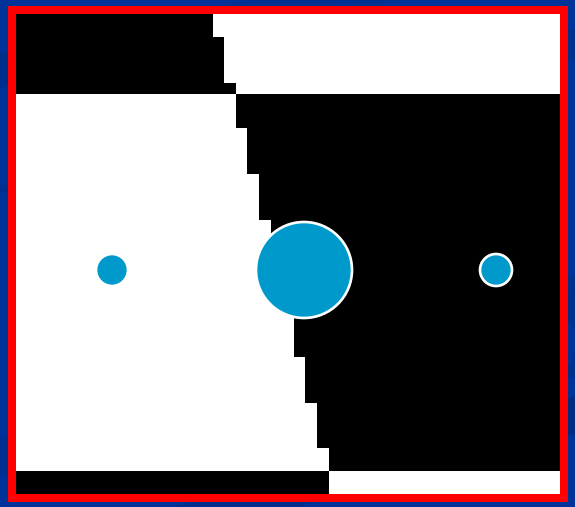




Bilinear Filtering

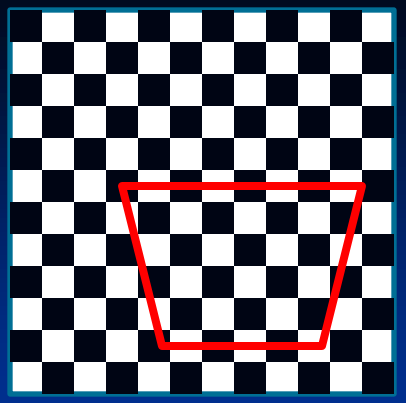
- "Jaggies" – lines have a staircased edge appearance
- Occur when a single texture sample (texels) projects to multiple screen pixels
- (Also occurs when rasterizing lines or polygon edges)
- Fixed by averaging neighboring samples to find the value between samples

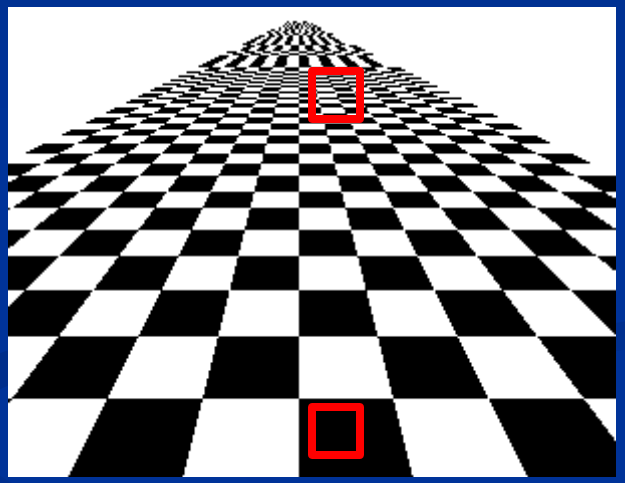


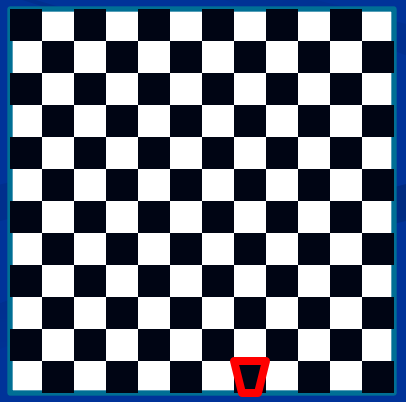


Minification Aliasing

- Many texture pixels (texels) map into a single screen pixel

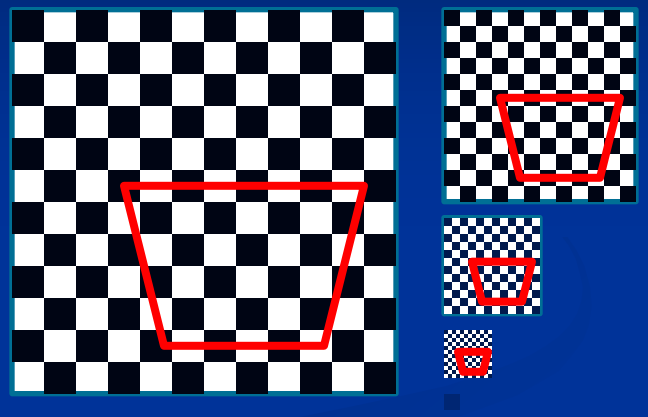


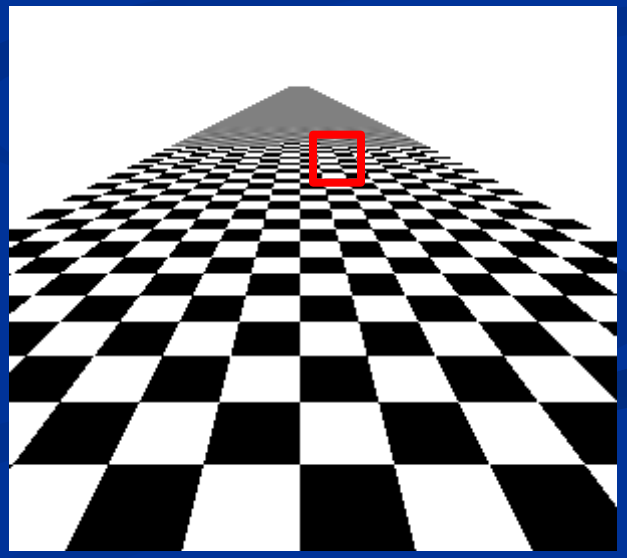




MIP Mapping

- Many texture pixels (texels) map into a single screen pixel
- Cannot simply add them up because some pixels would take longer than others to add
- Create an image pyramid from the initial texture
- Each level of the pyramid half the resolution of the one below it
- Choose the texture resolution whose projected texel size most closely matches pixel size





Texture Filtering

Parameter	Values
GL_TEXTURE_MAG_FILTER	GL_NEAREST or GL_LINEAR
GL_TEXTURE_MIN_FILTER	GL_NEAREST, GL_LINEAR, GL_NEAREST_MIPMAP_NEAREST, GL_NEAREST_MIPMAP_LINEAR, GL_LINEAR_MIPMAP_NEAREST, or GL_LINEAR_MIPMAP_LINEAR

Caveat

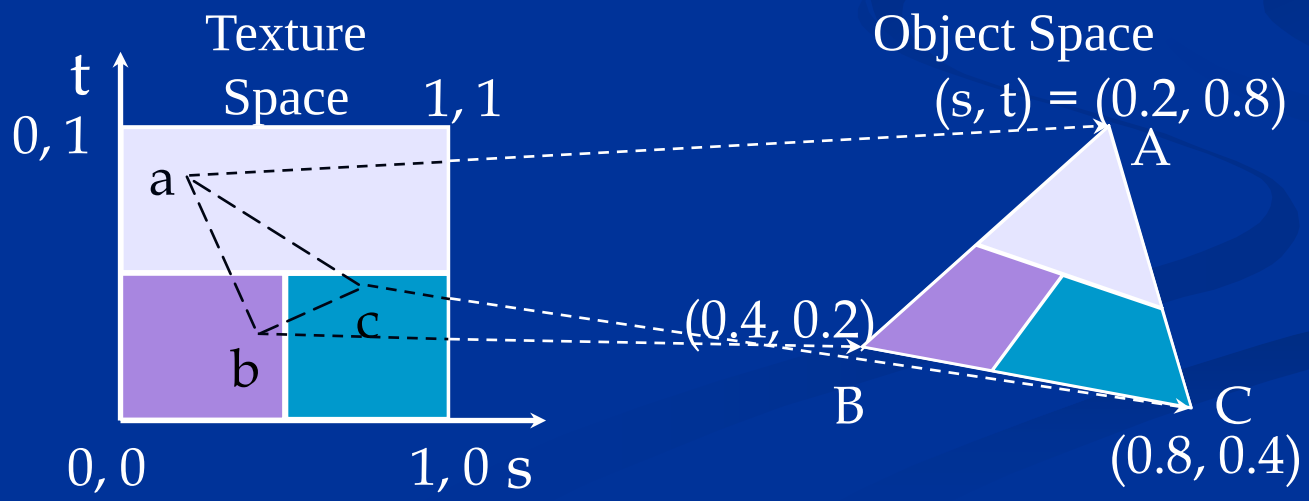
- Default setting for Minification filter is GL_NEAREST_MIPMAP_LINEAR
  - Change it to GL_LINEAR/GL_NEAREST since you don't have mipmap yet.
  - Otherwise your texture binding will fail.

Texture Mapping

- Now we have a texture object, but it contains no image.
- Define a texture image from an array of data in CPU memory
  - `glTexImage2D( target, level, components, w, h, border, format, type, *texels );`
  - dimensions of image are usually powers of 2
  - Target is usually "GL_TEXTURE_2D".
  - Variations : glTexImage1D, glTexImage3D

Texture Mapping

- Applying Texture
  - Remember to call **`glBindTexture & glEnable(GL_TEXTURE2D)`**
  - Based on parametric texture coordinates
  - **`glTexCoord*()`** specified at each vertex



Draw

```
glBindTexture(GL_TEXTURE_2D, texName);
glBegin(--------);
    glTexCoord2f(0.0, 0.0);
    glVertex3f(-2.0, -1.0, 0.0);
    …
glEnd();
```

Applications

- Environment Mapping
- Bump/Normal Mapping

Q&A

- Demo : Geometry Clipmap Rendering