Supplement to Lecture 18

Texturing in OpenGL

CS 354 Computer Graphics
http://www.cs.utexas.edu/~bajaj/
Department of Computer Science

Notes and figures from *Ed Angel: Interactive Computer Graphics, 6th Ed., 2012 © Addison Wesley*

University of Texas at Austin

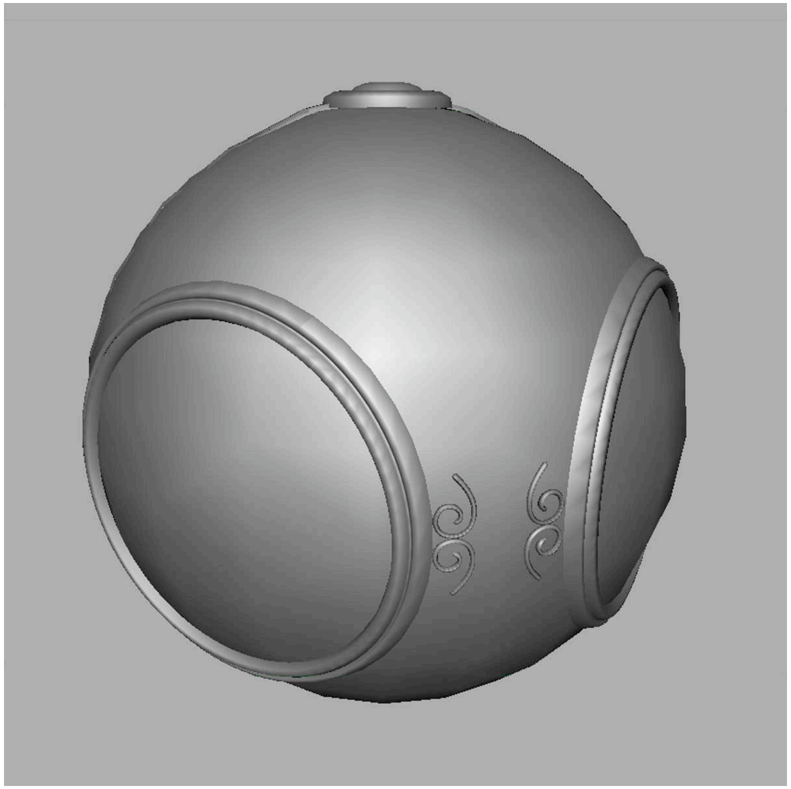
Limits of Geometry

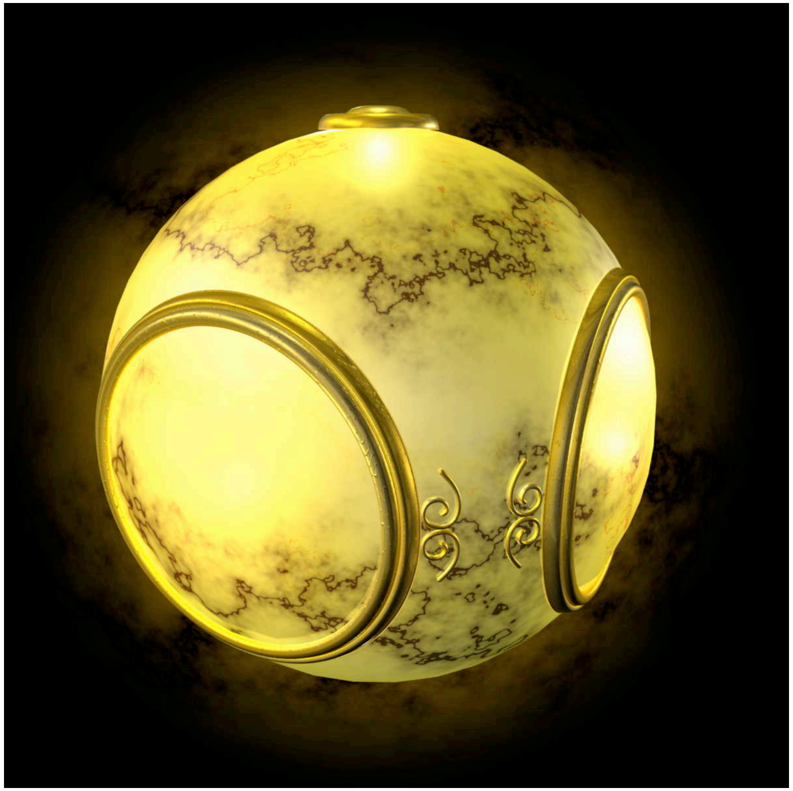
- Although graphics cards can render over 10 million polygons per second, that number is insufficient for many phenomena
  - Clouds
  - Grass
  - Terrain
  - Skin

Three Mappings

- 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



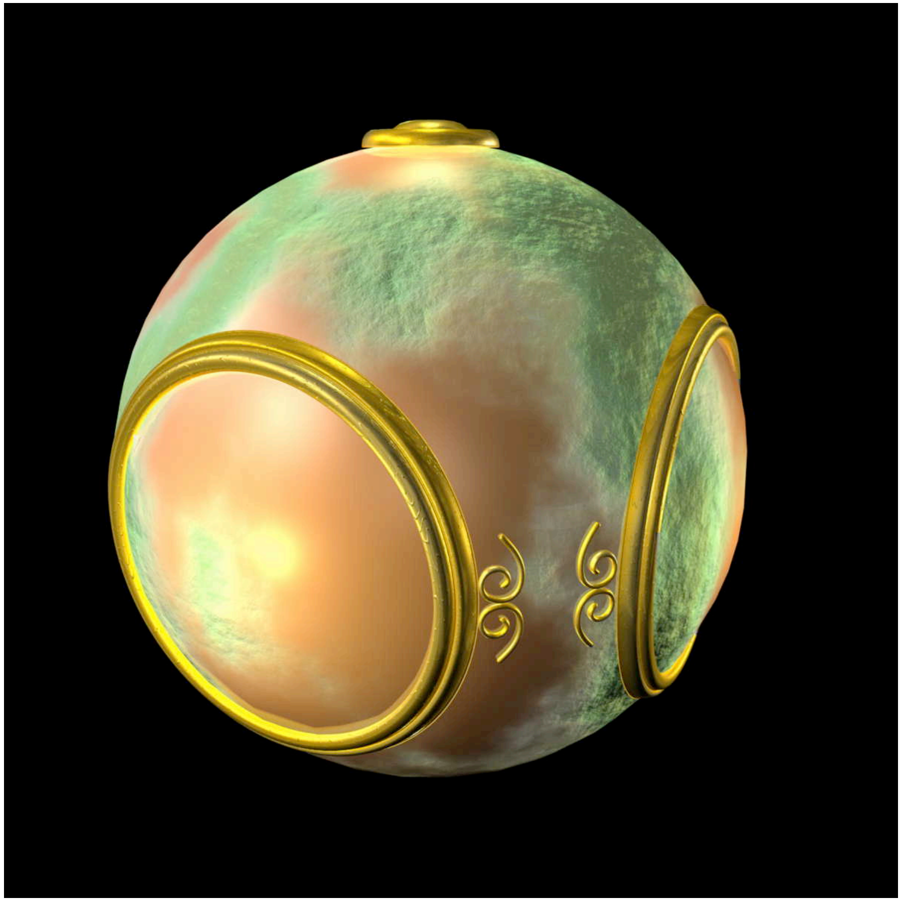


Notes and figures from *Ed Angel: Interactive Computer Graphics, 6th Ed., 2012 © Addison Wesley*

Environment Mapping

Notes and figures from *Ed Angel: Interactive Computer Graphics, 6th Ed., 2012 © Addison Wesley*

Bump Mapping

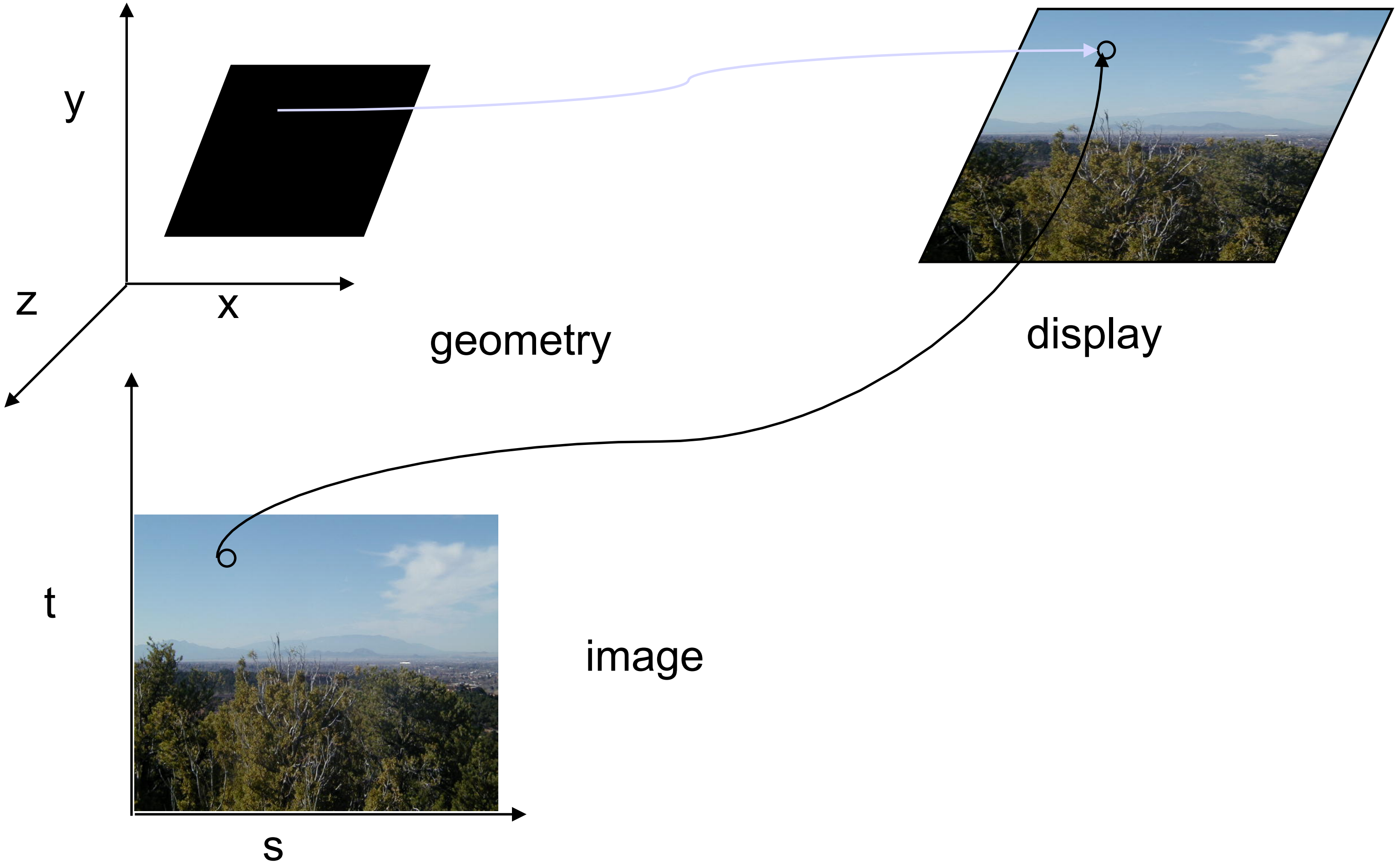


Implementation Strategy

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

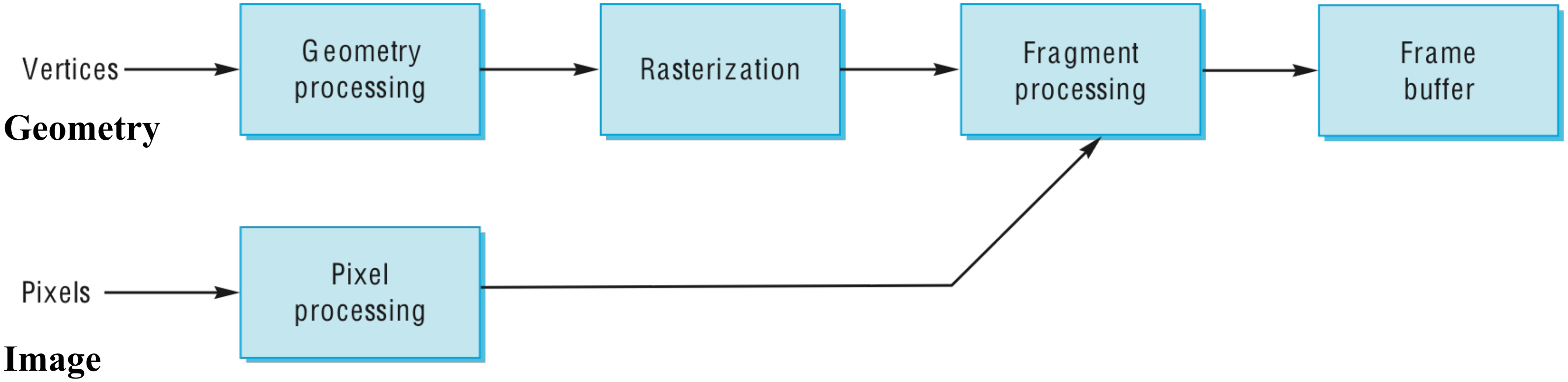
Texture Mapping



Notes and figures from *Ed Angel: Interactive Computer Graphics, 6th Ed., 2012 © Addison Wesley*

Where does mapping occur

- Mapping techniques are implemented at the end of the rendering pipeline
  - Very efficient because few polygons make it past the clipper



Notes and figures from *Ed Angel: Interactive Computer Graphics, 6th Ed., 2012 © Addison Wesley*

Define Image as 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);
```

Specifying a Texture Image

- Define a texture image from an array of *texels* (texture elements) in CPU memory

```
Glubyte my_texels[512][512];
```

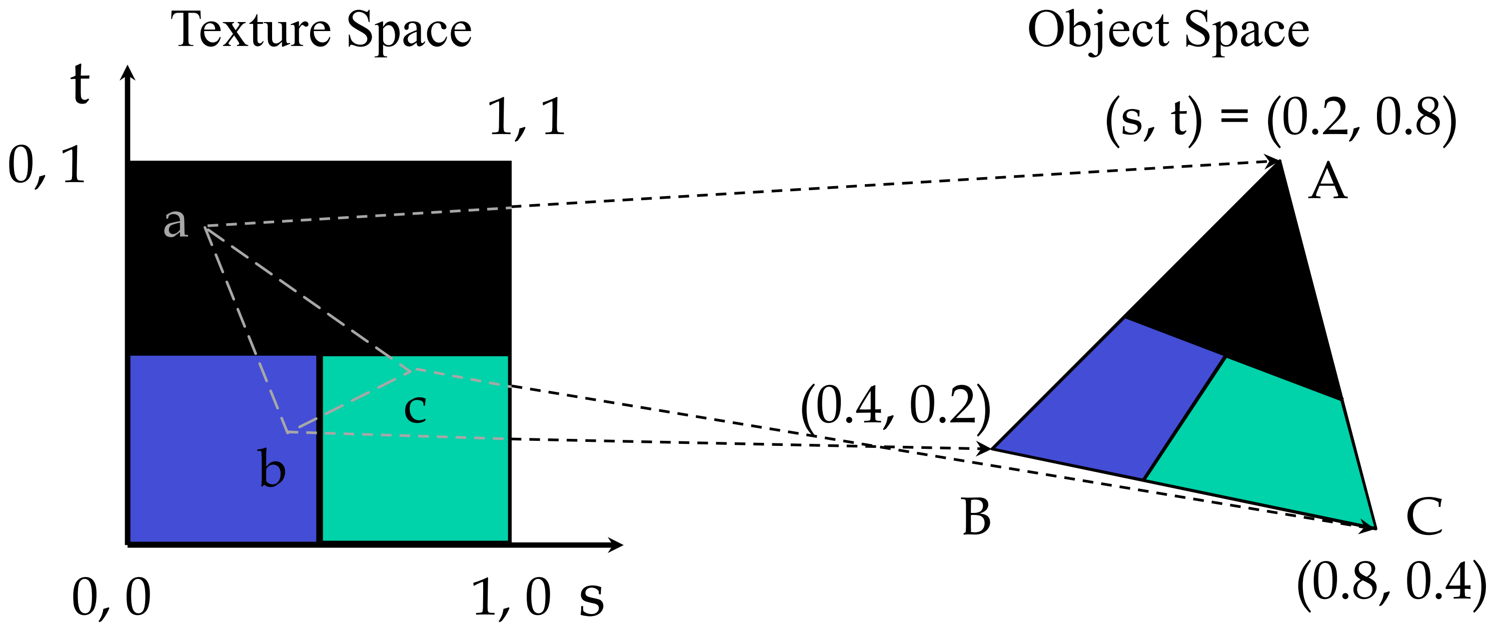
- Define as any other pixel map
  - Scanned image
  - Generate by application code
- Enable texture mapping
  - `glEnable(GL_TEXTURE_2D)`
  - OpenGL supports 1-4 dimensional texture maps

Converting a Texture Image

- OpenGL requires texture dimensions to be powers of 2
- If dimensions of image are not powers of 2
    - `gluScaleImage( format, w_in, h_in, type_in, *data_in, w_out, h_out, type_out, *data_out );`
    - **`data_in`** is source image
    - **`data_out`** is for destination image
- Image interpolated and filtered during scaling

Mapping a Texture

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



Notes and figures from *Ed Angel: Interactive Computer Graphics, 6th Ed., 2012 © Addison Wesley*

Typical Code

```
glBegin(GL_POLYGON);
 glColor3f(r0, g0, b0); //if no shading used
 glNormal3f(u0, v0, w0); // if shading used
 glTexCoord2f(s0, t0);
 glVertex3f(x0, y0, z0);
 glColor3f(r1, g1, b1);
 glNormal3f(u1, v1, w1);
 glTexCoord2f(s1, t1);
 glVertex3f(x1, y1, z1);
       .
       .
glEnd();
```

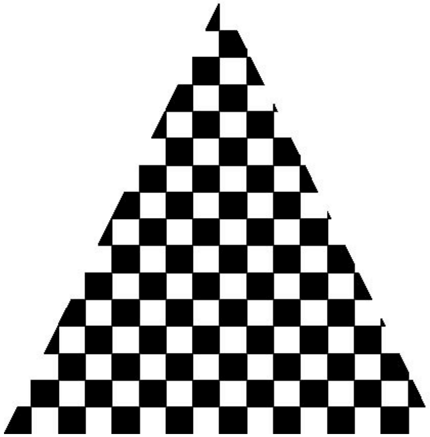
Note that we can use vertex arrays to increase efficiency

Interpolation

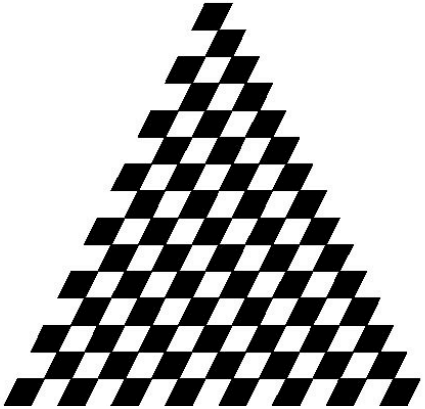
OpenGL uses interpolation to find proper texels from specified texture coordinates

Can be distortions

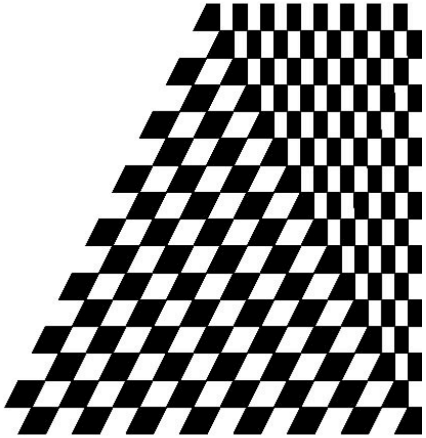
good selection
of tex coordinates



poor selection
of tex coordinates



texture stretched
over trapezoid
showing effects of
bilinear interpolatio



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);
```

Note that linear filtering requires a border of an extra texel for filtering at edges (border = 1)

Mipmapped Textures

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

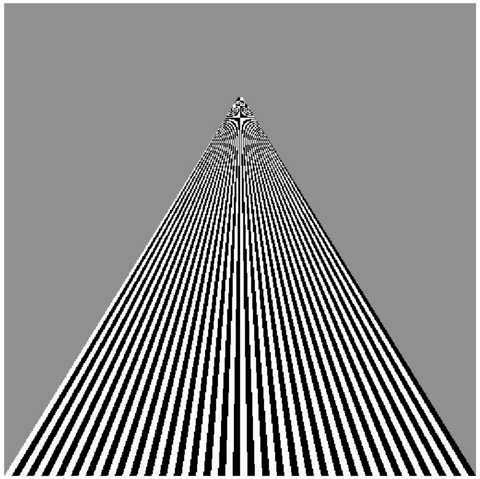
```
glTexImage2D( GL_TEXTURE_*D, level, … )
```

- GLU mipmap builder routines will build all the textures from a given image

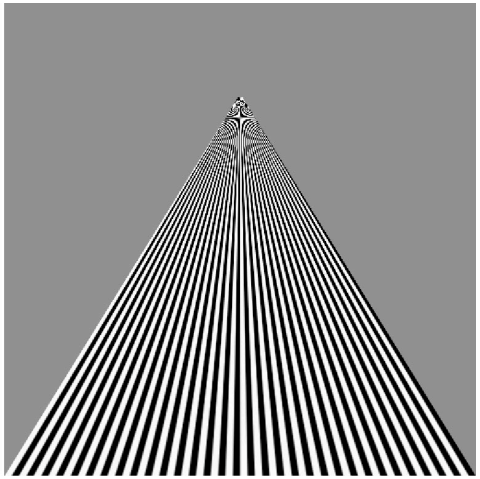
```
gluBuild*DMipmaps( … )
```

Example

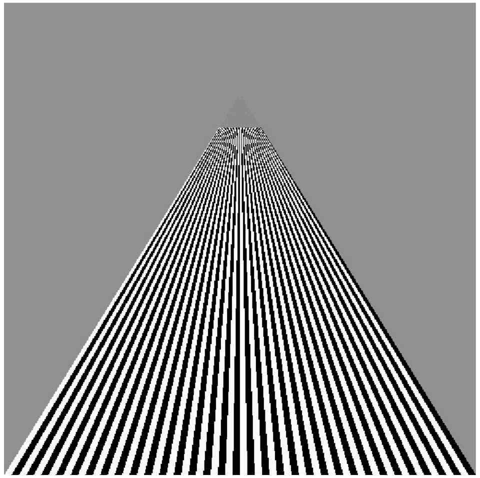
point sampling



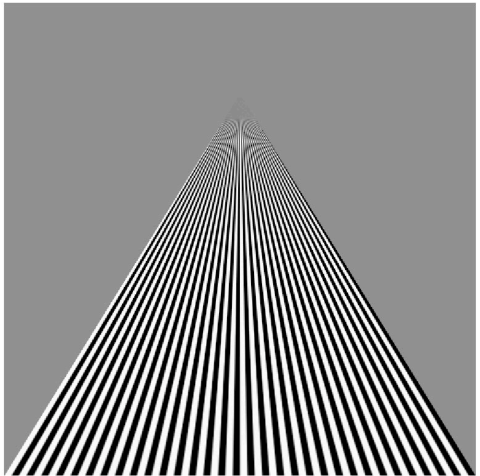
linear filtering



mipmapped point sampling



mipmapped linear filtering



Notes and figures from *Ed Angel: Interactive Computer Graphics, 6th Ed., 2012 © Addison Wesley*

Texture Functions

- 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`**

Perspective Hint Correction

- Texture coordinate and color interpolation
  - either linearly in screen space
  - or using depth/perspective values (slower)
- Noticeable for polygons “on edge”
  - **`glHint( GL_PERSPECTIVE_CORRECTION_HINT, hint )`**

    where **`hint`** is one of
    - **`GL_DONT_CARE`**
    - **`GL_NICEST`**
    - **`GL_FASTEST`**

Generating Texture Coordinates

- OpenGL can generate texture coordinates automatically

  `glTexGen{ifd}[v]()`

- specify a plane
  - generate texture coordinates based upon distance from the plane
- generation modes
  - **`GL_OBJECT_LINEAR`**
  - **`GL_EYE_LINEAR`**
  - **`GL_SPHERE_MAP`** (used for environmental maps)

Notes and figures from *Ed Angel: Interactive Computer Graphics, 6th Ed., 2012 © Addison Wesley*

Other Texture Features

- Environment Maps
  - Start with image of environment through a wide angle lens
    - Can be either a real scanned image or an image created in OpenGL
  - Use this texture to generate a spherical map
  - Use automatic texture coordinate generation
- Multitexturing
  - Apply a sequence of textures through cascaded texture units