Texture Filtering

MipMaps

“Optimal” case
Minification

Magnification
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.

Pixel Footprint
Pyramid Textures (Mipmapping)
Linear vs. Nearest

Trilinear
**Mipmapped Textures**

- *Mipmapping* allows for prefiltered texture maps of decreasing resolutions
- Lessens interpolation errors for smaller textured objects
- Declare mipmap level during texture definition
  ```c
  glTexImage2D( GL_TEXTURE_*D, level, …
    )
  ```
- GLU mipmap builder routines will build all the textures from a given image
  ```c
  gluBuild*DMipmaps( … )
  ```

**Example**

- Point sampling
- Linear filtering
- Mipmapped point sampling
- Mipmapped linear filtering
Anisotropic Filtering

Isotropic Filter

Anisotropic Filter

bilinear

trilinear
Lightmaps

• Creating local contribution

Unlit Scene  Lightmap Intensity  Local Light Contribution

Lightmaps

• Adding local light to scene

OpenGL Lighting  Combined Image
Lightmaps

- **Cached Lighting Results**
  - Reuse lighting calculations
    - Multiple local lights (same type)
    - Static portion of scene’s light field
    - Sample region with texture instead of tessellating
  - Low resolution sampling
    - Local lighting; rapid change over small area
    - Global lighting; slow change over large area

Lightmaps

- **Segmenting Scene Lighting**
  - Static vs. dynamic light fields
  - Global vs. local lighting
  - Similar light shape
Lightmaps

• Segmenting the lighting

- Dominant Lighting
- Local lighting

Lightmaps

• Moving Local Lights
  – Recreate the texture; simple but slow
  – Manipulate the lightmap
    • Translate to move relative to the surface
    • Scale to change spot size
    • Change base polygon color to adjust intensity
  – Projective textures ideal for spotlights
  – 3D textures easy to use (if available)
Spotlights as Lightmap
Special Case

• Mapping Single Spotlight Texture Pattern

Use texture matrix to perform spotlight texture coordinates transformations.

Lightmaps

• Creating a lightmap
  – Light white, tesselated surface with local light
  – Render, capture image as texture
  – Texture contains ambient and diffuse lighting
  – glLight() parameters should match light
  – Texture can also be computed analytically
Lightmaps

- Creating a lightmap

  Render surface lit by local light → Create a Texture Map from Image

Lightmaps

- Lightmap building tips

  Boundary should have constant value
  Intensity changes from light should be minimal near edge of lightmap
Lightmaps

• Lighting with a Lightmap
  – Local light is affected by surface color and texture
  – Two step process adds local light contribution:
    • Modulate textured, unlit surfaces with lightmap
    • Add locally lit image to scene
  – Can mix OpenGL, lightmap lighting in same scene

Lightmaps

• Creating local contribution

Unlit Scene | Lightmap Intensity | Local Light Contribution
Lightmaps

• Adding local light to scene

OpenGL Lighting

Combined Image

Lightmaps in Quake2

lightmaps only

×

(modulate)

decal only

combined scene
Packing Many Lightmaps into a Single Texture

- **Quake 2 light map texture image example**
  - Lightmaps typically heavily magnified.
  - Permits multiple lightmaps packed into a single texture.
  - Quake 2 computes lightmaps via off-line radiosity solver.

Lightmaps

- **Lightmap considerations**
  - Lightmaps are good:
    - Under-tessellated surfaces
    - Custom lighting
    - Multiple identical lights
    - Static scene lighting
Lightmaps

• **Lightmap considerations**
  – Lightmaps less helpful:
    • Highly tessellated surfaces
    • Directional lights
    • Combine with other surface effects (e.g. bump-mapping)
      – *eats a texture unit/access in fragment programs*
      – *may need to go to multi-pass rendering (fill-bound app)*

Multitexturing

• **Multitexturing** allows the use of multiple textures at one time.

• It is a standard feature of OpenGL 1.3 and later.

• An ordinary texture combines the base color of a polygon with color from the texture image. In multitexturing, this result of the first texturing can be combined with color from another texture.

• Each texture can be applied with different texture coordinates.
Texture Units

- Multitexturing uses multiple texture units.
- A texture unit is a part of the rendering pipeline that applies one texture to whatever is being drawn.
- Each unit has a texture, a texture environment, and optional texgen mode. That is, its own complete and independent OpenGL texture state.
- Most current hardware has from 2 to 16 texture units.
- To get the number of units available: `glGetIntegerv(GL_MAX_TEXTURE_UNITS)`
Texture Units

- Texture units are named GL_TEXTURE0, GL_TEXTURE1, etc.

- The unit names are used with two new functions.

  - `glActiveTexture(texture_unit)`
    - selects the current unit to be affected by texture calls (such as `glBindTexture`, `glTexEnv`, `glTexGen`).

  - `glMultiTexCoord2f(texture_unit, s, t)`
    - Sets texture coordinates for one unit

OpenGL Multitexture Quick Tutorial

- Configuring up a given texture unit:
  ```
  glActiveTextureARB(GL_TEXTURE1);  // Sets active texture unit
  glBindTexture(GL_TEXTURE_2D, texObject);
  glTexImage2D(GL_TEXTURE_2D, ...,);
  glTexParameterfv(GL_TEXTURE_2D, ...);
  glTexEnvronf(GL_TEXTURE_ENV, ...);
  glTexGenf(GL_S, ...);
  glMatrixMode(GL_TEXTURE);
  glLoadIdentity();
  ```

- Setting texture coordinates for a vertex:
  ```
  glMultiTexCoord4f(GL_TEXTURE0, s0, t0, r0, q0);
  glMultiTexCoord2f(GL_TEXTURE1, s1, t1);
  glMultiTexCoord3f(GL_TEXTURE2, s2, t2, r2);
  glVertex3f(x, y, z);
  ```
OpenGL Multitexture Texture Environments

• Chain of Texture Environment Stages

Pre-texturing color
\texttt{glColor3f(r,g,b)}

\texttt{glMultiTexCoord2f(GL\_TEXTURE0\_ARB, …)} → \texttt{GL\_MODULATE} → \texttt{#0}

\texttt{glMultiTexCoord2f(GL\_TEXTURE1\_ARB, …)} → \texttt{GL\_DECAL} → \texttt{#1}

\texttt{glMultiTexCoord2f(GL\_TEXTURE2\_ARB, …)} → \texttt{GL\_BLEND} → \texttt{Post-texturing color} → \texttt{#2}

Texture Combiners

• Multitexture is just the start
  – Standard OpenGL texturing has just \texttt{GL\_REPLACE}, \texttt{GL\_MODULATE}, \texttt{GL\_DECAL}, and \texttt{GL\_BLEND}
  – Texturing combining: \texttt{GL\_ADD}, \texttt{GL\_COMBINE}
  – E.g.
    • Powerful \((A \times B + C) \times \text{scale} + \text{bias}\)
    • \(A, B, C\) variables use various color inputs
  – Table 9-8, 9-9 in the Redbook.
Detail Texture

Multitexture Lightmapping