Surface Mapping Two

CS7GV3 – Real-time Rendering
Overview

• Texture Mapping
  • Aliasing
  • Filtering
• Mip-Maps
• Summed Area Table
• Anisotropic Filtering
2D Texturing Issues

• 2D Texture made up of texels mapped on to 3D object
  • Texture object mapped on to 2d screen pixels
  • 2d – 2d warping: Sampling is not always uniform

• Texture **Minification**
  • Many pixels to few texels
• Texture **Magnification**
  • Few texels to many pixels

• Can lead to Aliasing
• Sampling/filtering techniques to account for this
Aliasing

- Leads to:
  - Jaggies
  - Moire-patterns
  - Temporal aliasing

- Nyquist law: maz frequency displayable is half the sampling frequency

Without dot wheel appears to rotate backwards
Texture Minification
Texture Minification
Nyquist Frequency for Textures
Discrete Sampling
Discrete Sampling

• Now suppose that the object moves away from us so that each square occupies 1 pixel.
Discrete Sampling

• Consider one row of pixels and texels
Discrete Sampling

• Using the nearest texel, the pixels will be colored alternately black and white
Discrete Sampling

• Now suppose the surface moves a little further away so that nearly 2 texels cover one pixel.
Discrete Sampling

• Using the nearest texel, there will be long stretches of black and white pixels

![Diagram showing texels and pixels]
Discrete Sampling

• What will happen when the texels are exactly half the width of a pixel?
• What will happen when the texels are exactly one third the width of a pixel?
• Exactly one fourth?
Nearest-Neighbour Filtering

- Use colour of texel closest to the pixel center
- Fast
- Results in a large number of artifacts
  - Texture 'blockiness' during magnification
  - and aliasing and shimmering during minification
Bi-linear filtering

- Get values of four neighbouring texels and linearly interpolate to find a blended value
- removes the blockiness seen during magnification
Texture Magnification

Nearest neighbour  Bi-linear filtering
Texture Magnification

Nearest neighbour

Bi-linear filtering
Minification

• Several texels mapped to single pixels
Texture Minification

Nearest neighbour

Mip-mapping
Mip-Mapping

- mip = multum in parvo = “many things in a small place.”
- Create level of detail simplifications of the entire texture
- Basic technique
  - take 2x2 squares and average
  - Box filter (not great)
- Some are better:
  - gaussian, Lanczos, Kaiser

If the original texture is $64 \times 64$, then we should create copies at the scales of $32 \times 32$, $16 \times 16$, $8 \times 8$, $4 \times 4$, $2 \times 2$, and $1 \times 1$: This is why graphics API’s prefer power of 2 textures
Mip-map storage

10-level Mip Map
Level 0 Mipmap – 64 × 64
Level 1 Mipmap – 32 × 32
Level 0 Mipmap – 64 × 64
Level 1 Mipmap – $32 \times 32$
Level 2 Mipmap – 16 × 16
Level 3 Mipmap – 8 × 8
Level 4 Mipmap – 4 × 4
Level 5 Mipmap – $2 \times 2$
Level 6 Mipmap – $1 \times 1$
Using Mipmaps

• When using mipmaps, we have two separate choices:
  • Whether to use the nearest texel in a mipmap or to interpolate among the 4 nearest texels
  • Whether to use the nearest mipmap or to interpolate between the nearest two mipmaps

• Thus, the choices are:
  • Nearest texel, nearest mipmap
  • Nearest texel, interpolate mipmaps
  • Interpolate texels, nearest mipmap
  • Interpolate texels, interpolate mipmaps
Interpolating Between Mipmaps

• Assume that a single color has been selected from each of the nearest two mipmaps (from either the nearest texel or an average of texels)
• Compute the scale factor r between the level 0 (original) mipmap and the polygon
• Then compute \( \lambda = \log_2 r \)
• \( \lambda = \log(\text{texture}_\text{size}/\text{polygon}_\text{size}) \)
Which Mip Map Level To Use

- Count number of changes in texels along length of pixel cell
Interpolating Between Mipmaps

• The value of $\lambda$ tells us which mipmap to use
  • If $\lambda = 0$, use level 0
  • If $\lambda = 1$, use level 1
  • If $\lambda = 2$, use level 2, etc

• What if $\lambda = 1.5$?
  • Then we interpolate between level 1 and level 2
Example

• Suppose $\lambda = 1.3$ and the level 1 mipmap color is yellow $(1, 1, 0)$ and the level 2 mipmap color is cyan $(0, 1, 1)$

• Then the interpolated color is

$$0.7(1, 1, 0) + 0.3(0, 1, 1) = (0.7, 1.0, 0.3)$$
Trilinear Interpolation

• If we interpolate bilinearly within mipmaps and then interpolate those values between mipmaps, we get *trilinear interpolation*
Trilinear Interpolation

• How many individual interpolations are required to perform trilinear interpolation?
Trilinear Interpolation

• 4 from “left to right” (s direction)
Trilinear Interpolation

• Plus 2 more from “front to back” (t direction)
Trilinear Interpolation

• Plus 1 between levels = 7
Anisotropic Filtering

- Pixel space
- Texture space
- Mipmap samples
- Line of anisotropy
Anisotropic Filtering

Mip Map  Anisotropic