MipMap Texturing
Outline

- MipMapping
- Creating MipMaps
- Using MipMaps
- Trilinear MipMapping
- Anisotropic MipMapping
- Exercise Demo
Goals

• You can explain why it is a good idea to use mipmaps
• You know how to generate mipmaps in OpenGL
• You know the different filters for mipmap generation
• You can implement more sophisticated filters by yourself
MipMapping I

**Without mipmapping:**
artifacts/aliasing at details

**Solution:**
filter details before rendering

This happens without mipmapping
MipMapping II

- Textured objects can be viewed at different distances from the viewpoint

**Problem:** Which level of detail (Resolution) should one use for the texture image?

- Too high resolution: Aliasing effects
- Too small resolution: Too few details visible

**Solution:** Use different levels of detail according to the distance between object and viewpoint

→ mipmaps
MipMapping III

• History: 1983 Lance Williams introduced the word “mipmap” in his paper “Pyramidal Parametrics”

• mip = “multum in parvo” (lat.: many things in small place)

• Solves LOD problem by generating a **pyramid of textures**
  – Highest texture resolution at pyramid level 0
  – Halfed Resolution at each subsequent level
MipMapping IV

• MipMap pyramid:
  - needs 1 1/3 times the space

\[
\sum_{i=0}^{\infty} \frac{A}{4^i} = A \cdot \frac{4}{3}
\]

• OpenGL automatically determines the mipmap level to use based on the projected size of the object
Creating MipMaps I

• When creating the mipmap pyramid we have to compute the smaller levels
  - this is done by downsampling the original texture

• Definition:
  \[ c_i(x,y) = \text{color of the texture of level } i \text{ at } (x,y) \]
Creating MipMaps II

1. Nearest Neighbour

\[ c_i(x, y) = c_{i-1}(x \cdot 2, y \cdot 2) \]

sampling from the level below

\[ c_i(x, y) = c_0(x \cdot 2^i, y \cdot 2^i) \]

sampling from the original texture
2. Boxfilter

\[ c_i(x, y) = \frac{1}{4} \left( c_{i-1}(x \cdot 2, y \cdot 2) + c_{i-1}(x \cdot 2 + 1, y \cdot 2) + c_{i-1}(x \cdot 2, y \cdot 2 + 1) + c_{i-1}(x \cdot 2 + 1, y \cdot 2 + 1) \right) \]
3. Gaussian filter

To avoid aliasing effects a low pass filter (like a gaussian or sinc filter) is optimal

Unfortunately this is computational expensive

Therefore we discretize the filter into a matrix and perform a discrete convolution

Filter Matrix: (Gaussian)

\[
\begin{bmatrix}
1 & 4 & 6 & 4 & 1 \\
4 & 16 & 24 & 16 & 4 \\
6 & 24 & 36 & 24 & 6 \\
4 & 16 & 24 & 16 & 4 \\
1 & 4 & 6 & 4 & 1
\end{bmatrix}
\]
Creating MipMaps V

- MipMapping in OpenGL:

```c
void glTexImage2D( GL_TEXTURE_2D, GLint level,
    GLint components, GLsizei width, GLsizei height,
    GLint border, GLenum format, GLenum type, const GLvoid *pixels);
```

→ loads texture for the MipMap level
  (level 0 = original texture)

```c
void gluBuild2DMipMaps();
```

→ calls glTexImage2D(...) for each level
Texture-Lookup 1

• Problems when looking up color in the texture

Minification:
- Pixels map to less than one texel

Magnification:
- Pixels map to more than one texel

Filtering:
Nearest: centre of texel on texture determines color
Bilinear: weighted average of overlapping pixel
• Problem with bilinear:
  - it is visible where the mipmap level changes
Trilinear Filtering I

- linear filtering between two mipmap levels

In this example, the color of the pixel would be:
0.3 * (color of level i) + 0.7 * (color of level i-1)
Trilinear Filtering II

- Colored mipmaps:
  - with bilinear the change of levels is acute
  - with trilinear the levels fade in smoothly
Trilinear Filtering III

- Mipmap filtering in OpenGL:

```c
void glTexParameteri( GL_TEXTURE_2D,
                      GL_TEXTURE_MIN_FILTER,
                      GLenum filter );
```

→ filter:

- GL_NEAREST
- GL_LINEAR
- GL_NEAREST_MIPMAP_NEAREST
- GL_NEAREST_MIPMAP_LINEAR
- GL_LINEAR_MIPMAP_NEAREST
- GL_LINEAR_MIPMAP_LINEAR (trilinear)

↑ filter used to sample texture
↑ filter used when combining mipmap levels
Anisotropic Filtering I

- Trilinear mipmapping blurs for acute angles

trilinear (also bilinear) filtering does not take the perspective into account
Anisotropic Filtering II

- Anisotropic filtering looks at the projection of the pixel onto the texture

\[ k \text{ anisotropic} \text{ means that } k \text{ samples of the texture are used to approximate the projection of the pixel (here } k=8) \]
Anisotropic Filtering III