Antialiasing
Example: Alien Isolation (2014)

https://youtu.be/Js-Az06kGl8?t=12
What is Aliasing?
What is Aliasing?

- A signal-processing problem!
- Reconstruction from sampling distorted from original signal
- Called “jaggies” in graphics
Sampling Problem

- No correlation of pixel and texel size
- Too many texels per pixel
- Can solve by super-sampling (Nyquist–Shannon sampling theorem)
Antialiasing in a Ray Tracer

• One ray per pixel likely to have artifacts
• Cast multiple rays and average result

What’s the problem with this?
Expensive!

- Many of these super samples are unnecessary
- Only required in areas with rapid change in intensity
Adaptive Sampling

Cast more rays in a particular area

Where should we sample more?
Level of Detail

• Decreases complexity based on distance from the camera
• Often used for geometric complexity
  • But can apply to textures and shaders
• Correlates texel and pixel size thus helping with jaggies
Mipmapping

Main idea: store hierarchy of subsampled textures
Mipmapping

How much memory does this take?
Mipmapping

~50% more memory consumed
Applying Textures

What if \((u,v)\) is out of range for the number of pixels?

- repeat
- mirror
- clamp
- background
Apply Textures

What if \((u,v)\) isn’t an integer?

- snap to nearest texel
- linearly interpolate color
Linear Interpolation

Remember linear interpolation using parameter $t$?

$$p(t) = p0(1-t) + p1(t)$$

Can also calculate point along line using $(x,y)$ ratios
Linear Interpolation

Given known points \((x_0, y_0)\) and \((x_1, y_1)\), we can calculate any \(y'\) at \(x'\):

\[
\frac{y' - y_0}{x' - x_0} = \frac{y_1 - y_0}{x_1 - x_0}
\]

\[
y' = y_0 + (x' - x_0)\frac{y_1 - y_0}{x_1 - x_0}
\]

\[
y' = \frac{y_0(x_1 - x')}{x_1 - x_0} + \frac{y_1(x' - x_0)}{x_1 - x_0}
\]
Bilinear Filtering

Average four nearest texels

Eliminates “blockiness”/pixellation
Bilinear Interpolation

Three linear interpolations to calculate a position on a 2D grid

Provides weighted average between all four points
How to Perform Bilinear Interpolation?

R1 = linear interpolation between Q11 and Q21
R2 = linear interpolation between Q12 and Q22
P = linear interpolation between R1 and R2
Bilinear Interpolation

\[ R1 = Q11 \left( \frac{x_2 - x}{x_2 - x_1} \right) + Q21 \left( \frac{x - x_1}{x_2 - x_1} \right) \]

\[ R2 = Q12 \left( \frac{x_2 - x}{x_2 - x_1} \right) + Q22 \left( \frac{x - x_1}{x_2 - x_1} \right) \]

\[ P = R1 \left( \frac{y_2 - y}{y_2 - y_1} \right) + R2 \left( \frac{y - y_1}{y_2 - y_1} \right) \]
Trilinear Filtering

Classic problem in games: popping
Trilinear Filtering

Classic problem in games: popping

Can fix by averaging neighboring levels:
Bilinear interpolation on each level then linearly interpolate
Anisotropic Filtering

Use non-square pyramid levels

Compute them on the fly
Anisotropic Mipmaps
Modern Anti-Aliasing

• Aliasing is still an active area of research!
• Many techniques exist to reduce its effects in real-time applications
  • MSAA (multisample anti-aliasing)
  • TXAA (temporal anti-aliasing)
  • DLSS (deep learning super sampling)
• Efficiency of AA techniques relate to screen resolution (the jump from 2k to 4k requires different approaches)
Example: Control (2019)

https://youtu.be/YWIKzRhYZm4?t=43