**Outline**

- What is anti-aliasing?
  - Sampling theory
  - Anti-aliasing for object boundaries
    - Accumulator buffer algorithm
    - Super-sampling & post filtering
  - Anti-aliasing for textures
    - MIP mapping
    - Summed-area

---

**What is Anti-aliasing**

- **Aliasing:**
  - Jagged edges and random noises appeared in the computer synthesized images
- **Anti-aliasing:**
  - The technique of minimizing aliasing effects

---

**Sampling Theory**

- In order to reconstruct a signal from a set of samples, the sampling frequency must be at least twice of the maximum signal frequency
  - \( f_s > 2f_{\text{max}} \)
- Aliasing is caused by lack of samples

---

**Aliasing Caused by Object Boundaries**

- Pixels along the object boundaries are not fully covered by one object
  - Using one object to calculate pixel color causes jagged edges
- For anti-aliasing:
  - Model each pixel as a square window
  - Calculate pixel color based on the percentage covered by different objects

---

**A-Buffer Algorithm**

- A straightforward anti-aliasing approach is proposed by Catmull in 1978
  - Based on scan-line approach
  - Involves clipping polygons against the square window & compute the area of overlapping region
- Accumulator buffer (A-buffer) algorithm is proposed by Carpenter in 1984
  - Based on the Z-buffer algorithm
  - Use bitwise logical operators between the masks that representing polygon fragments
  - Floating point calculations are avoided
- More efficient than Catmull's approach
A-Buffer Algorithm (Cont’d)

- The area covered by a given polygon is represented using a 32-bit integer
  - Each bit indication whether the corresponding position is covered
  - Clipping one polygon against another becomes a simple bitwise Boolean operation

```
11111000111111101111100000000000
00000010000001110001111100111111
```

Super-Sampling & Post Filtering

- Basic idea:
  - Generate a set of samples at higher frequency than the image resolution required
  - The color of a given pixel is calculated using the average of samples within the window

Uniform Sampling

- Increases the sampling density uniformly across the image
- For an 1024x768 image:
  - First render at 4096x3072 resolution
  - Then calculate the average of 4x4 samples

Adaptive Sampling

- Uniform sampling introduces high computational costs
- For pixels that are covered by a single polygon, high sampling rate is unnecessary
- Adaptive sampling only increases sampling rate in locations that is needed

Stochastic Sampling

- Uniform & adaptive sampling samples at predefined locations
  - Aliasing may still appear for objects with patterns
  - Stochastic sampling samples a pixel at random locations
  - For each pixel 10 locations randomly determined and are used for sampling

Aliasing Caused by Textures

- Textures provide details (high frequency signal) and are prone to aliasing effects
  - The surface covered by a single pixel may map to a large area in texture space
  - Sample at one location causes aliasing
  - For anti-aliasing, the average color within an area should be used
Averaging in Texture Space

- The area covered by a square pixel in texture space can be an arbitrary shape.
- The size and shape of the area depends on surface orientation and distance to the viewpoint.
- The further away the object, the larger the area is.

MIP Mapping

- Works by creating lower resolution, pre-filtered versions of the original texture.
- MIP (multum in parvo): much in a small space.
- When rendering, the appropriate resolution MIP map is chosen.
- A single pixel in lower resolution keeps the average of a square area.

MIP Map

Summed-Area Approach

- A pixel in a MIP map covers a square region:
  - The averaging is always isotropic.
  - Cannot support anisotropic filtering.
- Summed-area approach uses a rectangle of arbitrary aspect ratio.
- The average within the rectangle region can be calculated using summed-area table.

Summed-Area Table

- A summed-area table pre-calculates the sum of different rectangles starting from the top-left corner:
  \[ S_{\text{area}} = \sum_{i,j} t_{i,j} \]
- The average within an arbitrary rectangle can be calculated using:
  \[ \frac{S_{ij} - S_{il} - S_{jl} + S_{ilj}}{(r-i)(b-j)} \]