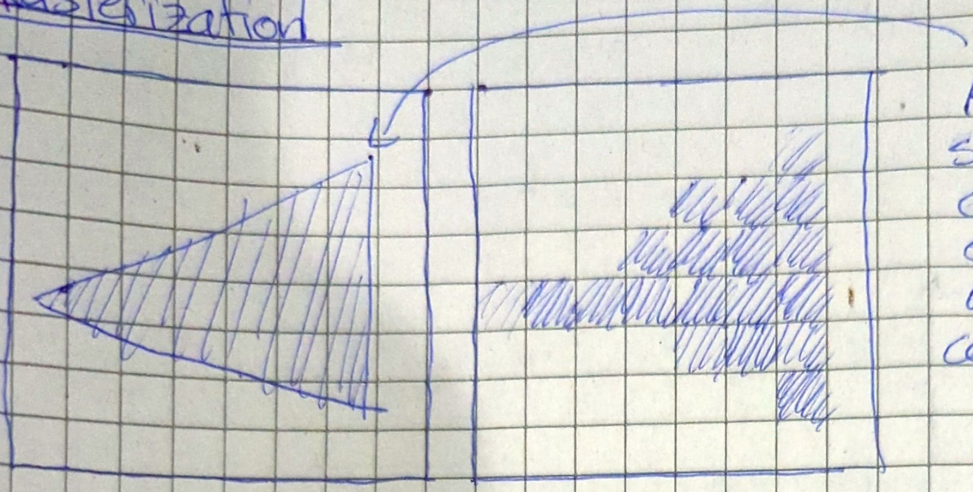
18 - Rendering algorithms

Rasterization



Anti-aliasing selects pixel color based on how much of the triangle cover the pixel

Canonical view volume → Raster image

Painter's algorithm

Starts by sorting the triangles, then draws them starting with the one furthest back.
Cannot handle intersecting geometry, which is a big drawback.

Z-buffer rasterization

doing with RGBA

Most popular, used by GPUs. Saves z-value for each center of pixel. Can handle intersecting geometry, but needs sorting for transparency

Can also lose anti-aliasing.

Super sampling anti-aliasing (SSAA) uses several samples for example ~~4x~~, for each pixel to compute anti-aliasing. 4x SSAA ~~takes 4 times longer to render~~. requires 4 times more operations on each pixel. Renders higher resolution image, then downsizes.

Multisampling anti-aliasing (MSAA) (for RGBA + z) is a mix of not doing AA, and SSAA, in that it takes 4 samples for z but only 1 sample for RGBA, for each pixel.

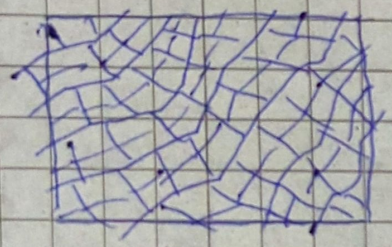
A-buffer rasterization

✓ can handle intersecting geometry
✓ supports order-independent transparency
× requires more (dynamic) memory

Works by having a linked list for each pixel, with each entry containing depth, RGBA and coverage. A new triangles pixel value can then be added to the list based on depth, and calculated a new color value with the other entries. Does not need SAA or MSAA. Mostly for offline rendering as it can be very expensive.

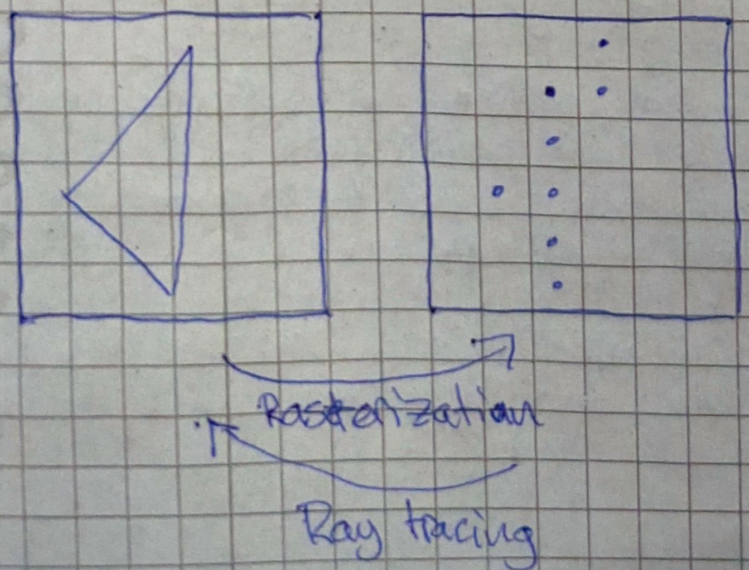
REYES (Renders Everything You Ever Saw)

Used by Pixars RenderMan. Dices primitives into micropolygons, smaller than a pixel.

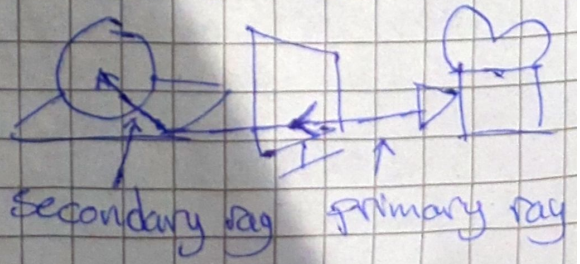


Ray tracing (standard for offline rendering, because higher quality but slower)

Rasterization – for each primitive, find pixel samples
Ray tracing – for each pixel sample, find the closest primitive



secondary rays:
reflections, refractions, shadows
realistic illumination...



Rasterization vs. ray tracing

Rasterization
for each primitive ← Good: linear memory access
find pixel samples ← fast
Bad: linear complexity

Ray tracing
for each pixel sample
find the closest primitive ← Good: logarithmic complexity
Bad: random memory access, slow

Linear complexity means it gets slower the more primitives we have.
Logarithmic complexity means the lookup of primitives can be faster the more primitives we have based on which data structure we are using. I.e. using a tree we can use lookup algorithms based on that.

Rasterization + ray tracing

Rasterization – for primary visibility

Ray tracing – for secondary effects: - reflections/refractions
- shadows
- realistic illuminations
...

Both can be used, but for offline rendering mostly ray tracing is used for both as rasterization can be expensive because of linear complexity.

Raytracing can be implemented in software, either on CPU or GPU, or on the hardware.
Hardware ray tracing is mostly used for secondary effects, because it's still very expensive.

Utah is working on a pure hardware ray tracing GPU, and the ray tracing on GPUs we have today is spun out from Utah.