Alternative Rasterization

David Sleeper

david.sleeper@gmail.com

November 22nd, 2021

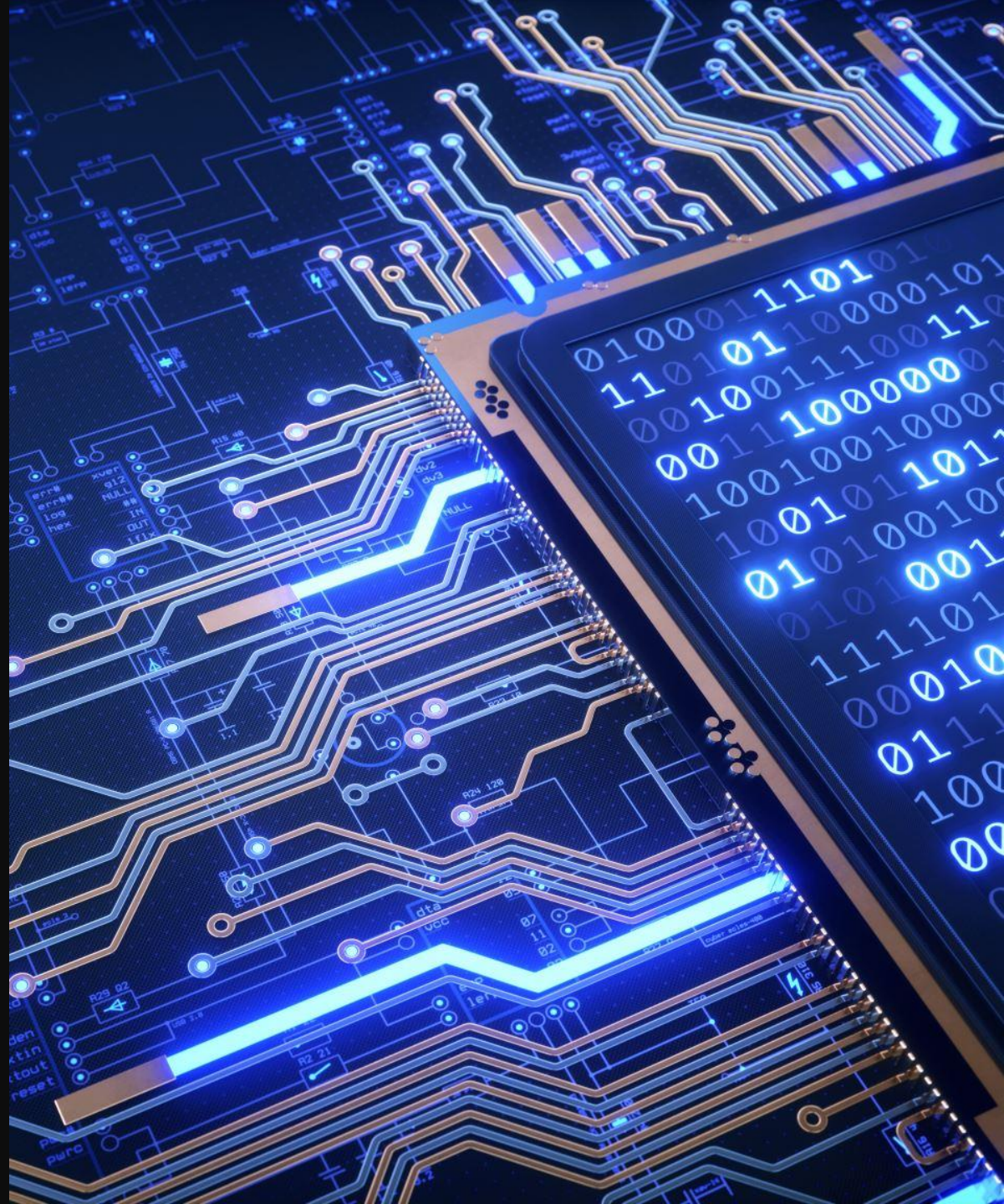
Past

- FFP (Fixed function Pipeline)
  - Based on a few functions able to do basic texturing and rendering of meshes
  - CPU mesh animation and skinning
- Shading (Programmable Pipeline *kinda*)
  - Vertex
    - Still fixed for input assembly
  - Pixel
    - Flexible but early forms had fairly limited resources, both in type and in numbers
- PBR (Physically based rendering)
- Deferred Renderers (variations of this in most modern games)
  - Draw info to offscreen render targets that you will later composite into something useful for final image
  - Common to draw diffuse, normal, specular and other information into these render targets

Driven by computer science rather than direct hardware updates

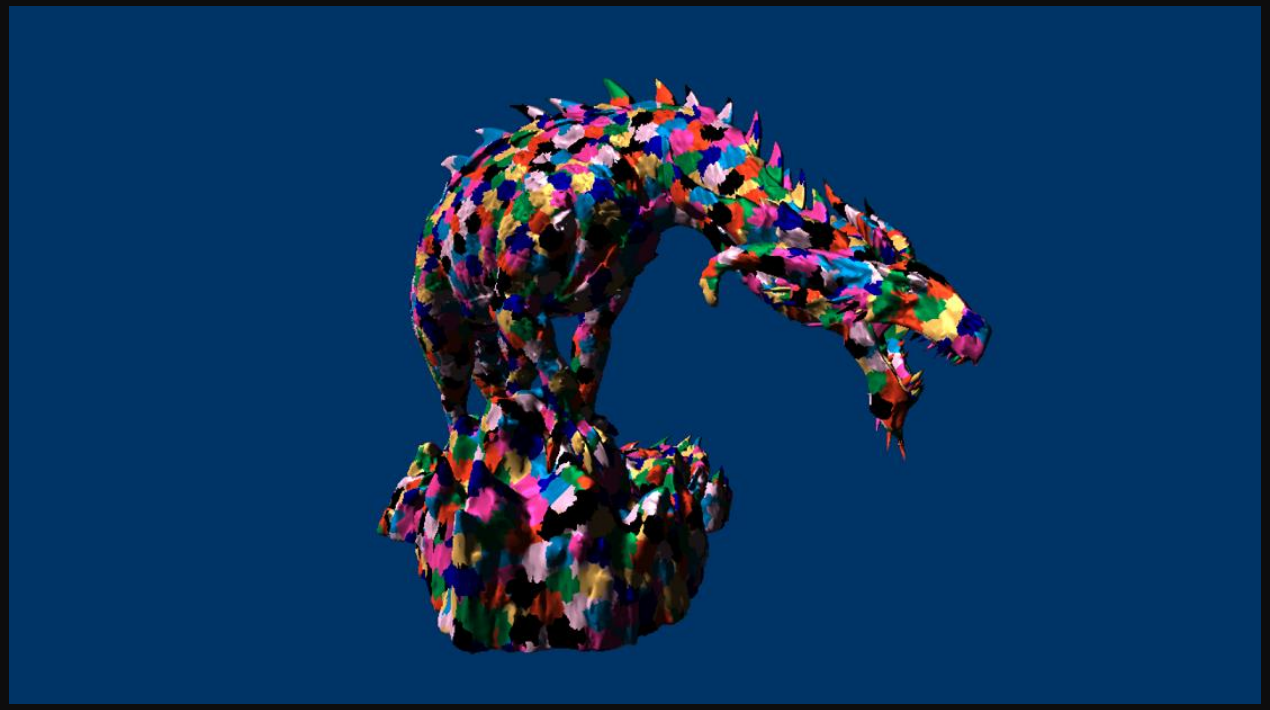
Nanite and Compute Rasterization

- Meshlets/clusters
- DAG (Direct Acyclic Graph)
- Compute/Software rasterizer
- Link to UE5 engineer talk: https://www.youtube.com/watch?v=eviSykqSUUw



Meshlet/Cluster

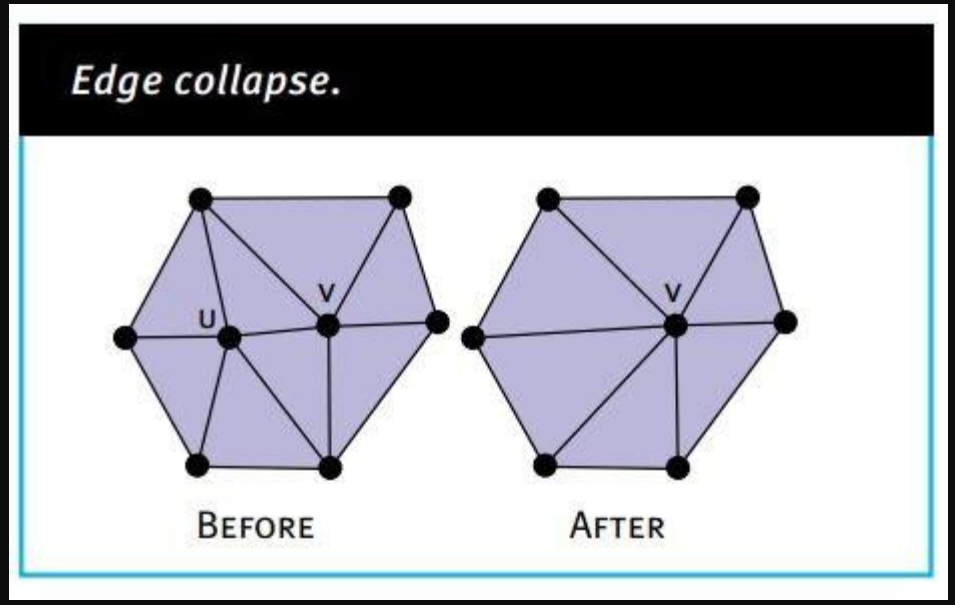
- Smaller subsets of the original mesh of vertices/indices
- Clusters and meshlets for mesh shading very similar as the idea is to replace the input assembler
- Both approaches have a small and finite requirement about how many vertices and primitives you want grouped at time of draw
- More shared edges the better! This naturally reduces vertex count.



DX12 Mesh Shader Example:
https://docs.microsoft.com/en-us/samples/microsoft/directx-graphics-samples/d3d12-mesh-shader-samples-win32/

Mesh Simplification

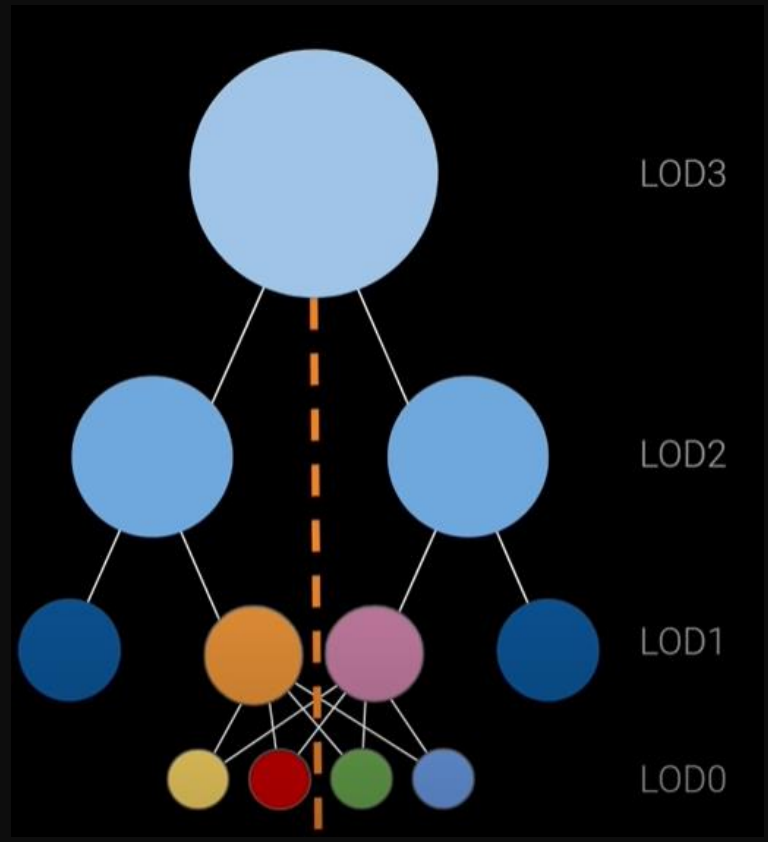
- Been around awhile--idea is to collapse edges based on a desired error algorithm
- Can factor in errors based on normal changes, triangle size, etc.
- Ultimate goal is to be able to specify a new target triangle count
- C++ link to a nice little github example: https://github.com/sp4cerat/Fast-Quadric-Mesh-Simplification



Source of image: https://github.com/andandandand/progressive-mesh-reduction-with-edge-collapse

Directed Acyclic Graph

- Not directly a LOD (level of detail) tree!
- METIS – open-source graph partitioner http://glaros.dtc.umn.edu/gkhome/metis/metis/overview
- Not a tree since child-sharing exists due to nature of locking and unlocking shared edges on mesh simplification
- Ultimate goal is localized LODs within the mesh to minimize micro polys (sub pixel triangles)
- Example: https://youtu.be/q8OuP3SNQxM



Time link from Nanite Deep Dive:
https://youtu.be/eviSykqSUUw?t=1031

Compute Rasterizer

- Taking your clusters and drawing them directly into a render target/texture using a compute shader
- Tend to be called software rasterization not to be confused with CPU rasterization
- EX: https://www.shadertoy.com/view/XdlGzn
- For Nanite they don't always do software rasterization if triangles are larger than a certain threshold (~35 minute mark in deep dive video)

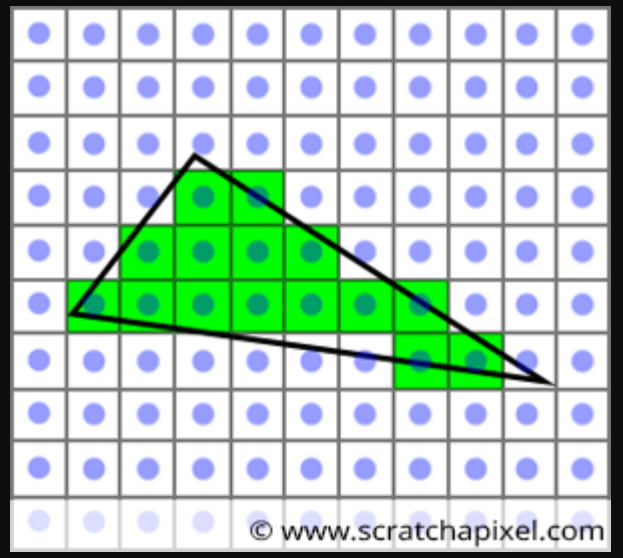


Image source:
https://www.scratchapixel.com/lessons/3d-basic-rendering/rasterization-practical-implementation/rasterization-stage

Nanite Big Old Caveat

- Assuming dense continuous mesh, doesn't really happen unless a photogrammetry mesh or artist planned it out carefully (goal of nanite LOD is to get it down to a single node at the end of simplifying)
- Foliage and many common objects are traditionally NOT rendered or able to be setup as a continuous mesh
- Photogrammetry meshes are just unwieldy in most 3d modelers
- Epic bought Quixel, a photogrammetry company…

Splat, SDF, and Sony Dreams

- Splat
  - Using either point rendering or custom rasterizer to draw little circle patterns or shapes at the point of pixel position of the scene
  - https://github.com/sebastianlipponer/surface_splatting
- SDF (signed distance function)
  - Uses a simple function per shape to return how far your point is from the surface
    - Tremendously simple…
    - Amazing results
    - Crazy slow in the naïve approach
  - Originated in the “demo” scene of graphics development
- Sony Dreams tech
  - Combination of both and some extra in between

Splat, SDF, and Sony Dreams Links

- Shadertoy awesome resource for SDF
  - https://www.shadertoy.com/view/Xds3zN
- Claybook (available on Steam)
  - https://store.steampowered.com/app/661920/Claybook/
  - https://www.youtube.com/watch?v=Xpf7Ua3UqOA
- Sony Dreams
  - https://www.playstation.com/en-us/games/dreams/
  - http://advances.realtimerendering.com/s2015/AlexEvans_SIGGRAPH-2015-sml.pdf

Ray Tracing

- Fairly rigid in design
- Add geometry to the proper 3d structures the driver/card are designed for
- Run groups of dispatched rays with specialized shader to receive those results
- Can be used directly for all rendering or partially for global illumination, ambient occlusion, or various scene effects
- https://www.nvidia.com/en-us/geforce/news/metro-exodus-pc-enhanced-edition-ray-tracing-dlss/