Sample-based Rendering for Extremely Large Meshes and Complex Iso-surfaces

Matthäus G. Chajdas
Why voxels?

- In 2D: Instead of extremely dense vector data, we use Bitmaps (uniform resolution)
Voxels, indeed
Overview

• Interval Buffer Raycasting
  – Iso-surfaces
  – Meshes

• Voxel rasterization

• Outlook: Beyond octrees
Overview

• Interval Buffer Raycasting
  – Iso-surfaces
  – Meshes

• Voxel rasterization

• Outlook: Beyond octrees
Interval Buffer Raycasting

• Goal: Render very large & complex iso-surfaces

• This work is published:
  – *Hybrid Sample-based Surface Rendering*, F. Reichl, M. G. Chajdas, K. Bürger, R. Westermann, VMV 2012
  – See the paper for more details
Sample-based Rendering for Extremely Large Meshes and Complex Isosurfaces

Matthäus G. Chajdas
computer graphics & visualization
Interval Buffer Raycasting

• Goal: Render very large & complex iso-surfaces
  – ... on commodity hardware
  – ... on large viewports (1080p and higher)
  – ... with low memory usage
  – ... and with good performance
  – ... and with the ability to change the iso-value on the fly
Interval Buffer Raycasting

• What is the problem, actually?
  – Volumes are large!
  – Working set barely fits into RAM, if at all
  – Raycasting needs additional acceleration structure

Assumption: Optimize for memory, and everything else will follow!
Interval Buffer Raycasting

- We want to render a surface!
- What is the bare minimum we can store for a surface?
  - Position
- Not enough for good quality
  - Normals
Sample-based Rendering for Extremely Large Meshes and Complex Iso-surfaces
Matthäus G. Chajdas

Interval Buffer Raycasting
Interval Buffer Raycasting

- Accelerate using an uniform grid
Interval Buffer Raycasting

• Store differential positions as linked lists
Sample-based Rendering for Extremely Large Meshes and Complex Iso-surfaces
Matthäus G. Chajdas
Sample-based Rendering for Extremely Large Meshes and Complex Iso-surfaces
Matthäus G. Chajdas
Interval Buffer

• Each cell in the grid contains a list of intervals
• Ray-casting is done in 2D and must traverse the interval stack for each cell
• Very efficient unless the lists become very long
  – This is not an iso-surface any more, but an “iso-sponge”
  – Still, comparable with direct volume raycasting
Interval Buffer

• 2D grid is not sparse
  – Cannot use a single grid for large scenes
  – Grid is for a single level-of-detail only

• Use an octree on top!
  – Estimate the level-of-detail per octree leaf
  – Leaf nodes contain interval buffers
Interval Buffer

• Are we done?
  – Nearly!
  – Quality up-close not good enough
  – Individual voxels can be seen when getting too close
Hybrid interval buffer

• Add a new level-of-detail: The volume itself
  – Only pay for full volume raycasting close-up
  – Guarantee pixel-error to ensure that there are no visible transitions
Interval Buffer

- Data structure is simple enough to create it *on-the-fly*
- Allows us to change the iso-value interactively
Overview

• Interval Buffer Raycasting
  – Iso-surfaces
    – Meshes
• Voxel rasterization
• Outlook: Beyond octrees
Hybrid sample-based

• Goal: Rendering extremely large meshes
• Lots of challenges!
  – Level-of-detail
  – Streaming
Hybrid sample-based

• Can we apply the previous approach?
• Yes we can!
  – Mesh surface, Iso-surface, doesn’t matter
  – Resample the mesh such that voxel size ≈ triangle size
  – Resample into different levels of detail
  – Decide at render time whether to render voxels or triangles
Hybrid sample-based

- Example: David statue
- 950 million triangles (11 GiB indices, 5.5 GiB vertices)
- Voxel resolution: 32768 x 8192 x 8192
  - 2199023255552 voxels ($\approx 2^{41}$)
  - Very sparse!
Hybrid sample-based

• For magnification, go back to the triangle mesh itself
  – Similar technique as used for switching between interval buffer/volume
  – More complex heuristic (more on that later)
Hybrid sample-based

• Decide when to use the rasterizer and when to use voxels
  – Magnification: Use rasterizer
  – Minification: Heuristic
    • If very small or level-of-detail required: Voxels
    • If both are viable, check average depth complexity
Overview

• Interval Buffer Raycasting
  – Iso-surfaces
  – Meshes

• **Voxel rasterization**

• Outlook: Beyond octrees
Voxel rasterization

• Going simpler: No acceleration structure

• Why?
  – Super-fast conversion from volume to voxels
  – Direct level-of-detail on the voxels
Rendering

- Surface voxel rendering
  - Extract surface voxels and render as small cubes
  - Hybrid between points and marching cubes
Level-of-detail

• Directly on the voxels
• *Dramatically* faster than resampling of the volume
  – Several orders of magnitude less data
  – Very simple, parallel reduction
  – Works efficiently on CPU & GPU
Level-of-Detail
• Level-of-detail by voxel fusion
Level-of-detail

• Parallel
  – Sort buffer
  – Count voxels
  – Merge

• Fast, even on CPU
  – ~ 5 Minutes to rasterize the David in $8192^3$ resolution
  – ~ 3 Minutes for the level-of-detail (single threaded, I/O bound)
Rendering

- Rasterizer can use hardware MSAA
- Very useful for the highly regular cube meshes
- Negligible cost on modern GPUs
Sample-based Rendering for Extremely Large Meshes and Complex Iso-surfaces

Matthäus G. Chajdas

No Anti-Aliasing
Sample-based Rendering for Extremely Large Meshes and Complex Iso-surfaces

Matthäus G. Chajdas

4x MSAA
Overview

• Interval Buffer Raycasting
  – Iso-surfaces
  – Meshes

• Voxel rasterization

• **Outlook: Beyond octrees**
Adaptive trees

- Octree is very popular for voxels
  - Easy level-of-detail
  - Simple traversal
  - Fast construction
- In „normal“ ray-tracing, BVHs and k-D-Trees are the norm
Adaptive trees

• k-D-Trees and BVHs *adapt* better to the data than octrees

• Still need a good heuristic to determine splits
  – SAH?
  – Median?
  – Middle?
Implicit k-D-Tree
(Middle split)
Sparse voxel octree
(Middle split)
Hybrid
(SAH-style split)
Adaptive trees

- Apples to apples comparison
  - Same octree traversal
  - Same k-D-Tree traversal
  - Hybrid combines both into one huge kernel
    - Not optimal, but what can we do?
    - Register & local memory pressure as we need to keep two traversal stacks
    - Still, faster
Sample-based Rendering for Extremely Large Meshes and Complex Iso-surfaces
Matthäus G. Chajdas

Relative rendering time

- Hybrid tree
- Sparse octree (full volume)
- Sparse octree (bricks)
- Implicit kD-tree

Minimum
Average
Maximum
Adaptive Trees

• Very careful tradeoffs between tree-depth, coherence, and quality are necessary
  – Much deeper trees can be faster (30 vs 22 levels)
  – More coherent tree can be faster, even if metrics are worse (hit-rate, average stack depth, etc.)
  – Highest quality does not necessarily mean highest performance
Median split
Middle split
SAH split
Optimized split
Acknowledgements

- Matthias Reitinger
- Jan Sommer
- Andreas Ostermaier
- An Lu
Thank you!
Questions?

Twitter: @NIV_Anteru 🌐
E-Mail: chajdas@tum.de