Batched Multi Triangulations

Paolo Cignoni
Fabio Ganovelli
Enrico Gobbetti
Fabio Marton
Federico Ponchio
Roberto Scopigno
Accurate adaptive efficient rendering of very large static meshes

➢ Isosurfaces, scanned objects etc
➢ Hundreds of papers.
➢ Recent trends in multires surface representation (some)
  • To harness the power of graphics hw
  • Out of core management
  • Low CPU usage approaches
MT is a well known framework to describe a multiresolution models

Consider a sequence of local modifications over a given description $D$

- Each modification replaces a portion of the domain with a different conforming portion (simplified)
  - $f_i$ floor
  - $g_i$ the new fragment

$D' = D \setminus f \cup g$

$D_{i+1} = D_i \oplus g_{i+1}$
Multi Triangulations

Dependence between modifications can be arranged in a DAG.
Multi Triangulations

- Adding a sink to the DAG we can associate each fragment to an arc.
MT Cuts

A cut on the DAG defines a new representation

\[ D^* = D_0 \oplus g_1 \oplus g_2 \oplus g_4 \]
A cut on the DAG defines a new representation

Collect all the fragment floors of cut arcs and you get a new conforming mesh
  • just load and render
  • low cpu workload
  • fit very well in extended memory hierarchies

\[ D^* = D_0 \oplus g_1 \oplus g_2 \oplus g_4 = f_{0\infty} \cup f_{02} \cup f_{03} \cup f_{13} \cup f_{1\infty} \cup f_{4\infty} \]
Rendering a MT

- Fragment are patches of triangles
  - Optimized (tristripped) and stored compressed on disk
  - Obtained with a high quality simplification algo.
  - DAG << original mesh (patches composed by 8k tri)

- At run time two threads: Render and PatchServer
  - **Render**
    - Update the cut (moving in and out MT nodes)
    - Choose patches to be prefetched
    - Render the selected fragments
  - **PatchServer**
    - Load and uncompressed requested patches from disk into memory

- Culling
  - Standard frustum and hw based occlusion techniques defined for generic hierarchies can be easily adapted to DAGs
To update the cut we use two priority queues R & C for refinement and coarsening of MT nodes

- R top is \textit{max} screen space error
- C top is \textit{min} screen space error

Cost function for each node
- According to its size and if it is loaded

Refine until budget is full
Coarsen otherwise
Take care of node feasibility
Possible DAG problems:

- The topology of dependencies lowers the adaptivity of the multiresolution structure
- Cascading dependencies are BAD
Goal: Building a nice sequence of fragments

- Consider a partition $\Psi$ of the space into disjoint regions $\{\psi_j\}$
  - Partitions can be applied to triangulations
- Given two partitions $\Psi, \Phi$ we define the crossing of the partitions as

$$
\Psi \otimes \Phi = \bigcup_{i=0 \ldots |\Psi|, j=0 \ldots |\Phi|} \{\psi_i \cap \phi_j\}
$$
Crossing Partitions

$V_0 \otimes V_1 \quad V_1 \otimes V_2$
At step i we substitute independently all the regions of a partition $V_i$ with a simpler representation.

Iterating this process generates a set of fragments belonging to the various crossing of the partitions $V_{i-1} \otimes V_i$ and $V_i \otimes V_{i+1}$.

These pieces (the portions of floor fragments) are the building blocks that we will use to build new conforming triangulations.
Partitions in practice: Simplification

- Load the patches of $V_{i-1} \otimes V_i$ that compose a region of $V_i$
- Join and simplify preserving just the blue border
  - Simplification of patches is independent (out of core and parallel)
- Save this region partitioned according to $V_i \otimes V_{i+1}$
MT in action

\[ V_0 \]

\[ V_0 \otimes V_1 \]

\[ V_1 \otimes V_2 \]

\[ V_2 \]

high resolution  low resolution
Partition in practice: building

- We need a sequence of nice partitions
- Voronoi partitioning of space
  - Nice shape of the regions
  - Easy to attribute triangles to regions
  - generating seeds
- Regularly sampled seeds
  - Finer and finer grids
- Adaptive distribution of seed
  - Choose a patch radius $r$
  - Stream the triangles adding a seed every time we encounter a triangle farther than $r$ from all previous seeds
  - Apply Lloyd relaxation
General Framework

- Encompasses existing partitioning schemas like right angle hierarchies and Tetrapuzzles/SlowGrowingSubdivisions
Yet another multiresolution algorithm for rendering large static meshes.

- General framework
- Easy to implement
- State of the art performance
  - >4Mtri/frame at >30 fps