A New Algorithm for Interactive Graphics on Multicomputers

David A. Ellsworth

- Multicomputer: non-shared memory multi-cpu system
  - Then: Touchstone, iWarp, Paragon
  - Now: commodity clusters

**Sort-first**

- Primitives initially assigned arbitrarily
- *Pre-transformation* is done to determine which screen regions are covered
- Primitives are then redistributed over the network to the correct renderer
- Renderer performs the work of the entire pipeline for that primitive from that point on

**Granularity Radio**

*Number of screen regions per processor*

Higher granularity ratio =
  + Higher probability of being able to assign regions equitably
  + Increase in required communication and per-primitive rasterization overhead

**Load Balancing**

- Between Regions
  - Processor asks for regions on the fly
  - Each assignment individually broadcasted
- Between Stages
  - Wait for transformation completion, then assign regions all at once (single broadcast)
- Between Frames
  - Processors immediately begin rendering based on last frame’s primitive counts
  - Frame-to-frame coherence is key

- Mueller uses first person singular
- Hughes Hoppe doesn’t
“A New Algorithm”

• Transformation and Rasterization not overlapped (it was too efficient and caused the computer to overheat)
  – But previous frame’s primitive counts used
    • Single processor computes and then broadcasts region assignments for next frame
    • Fixed granularity ratio: 8 regions/processor

“A New Algorithm”

“All-to-all communication does not scale very well”

Future Work

• Try statically assigning regions to processors instead
  – (Would it have been that hard to test this when the dynamic assignment tests were performed?)
• High hopes for big iron multicomputers

“A New Algorithm”

• Two-tier’d communication system
  – Optimal when x routers for x² total processors
• Implemented on 512 node Touchstone Delta
  – Only 17MB/s because it’s 1994
  – Primary bottleneck: collection of region primitive counts limits framerate
    • (Why not re-use last completed region distribution if new assignment not ready yet? Anything is better than a suffering framerate…)

The Sort-First Rendering Architecture for High-Performance Graphics
Carl Mueller

“millions of polygons for zillions of pixels”
WANTED: Interactivity
(Low Latency and 30fps)

• Sort-last: too much bandwidth required
• Sort-middle: many-to-many communication -> limited scalability
• Sort-first: load balancing hard

Coherence

• Htha wast Yugli jix mallie nop sequin
• Sudden view changes also make for bad coherence
• The faster your framerate, the better your frame-to-frame coherence

Offscreen Primitives

• Keep on the processor where they were on-screen
  – Can lead to overload
• Send to neighboring processors
  – Still leads to overload or redundant communication
• Send to underloaded processor
  – Requires broadcast of load information
• Send to a random processor
  – Randomness is cool
• When to get rid of them?
  – (popping in and out of view)

Load Balancing

• Static assignment
• Adaptive methods
  – Roble’s Method
    • split high-primitive regions, join low-primitive ones
  – Whelan’s Method
    • split according to primitive centroid distribution
  – Whitman’s Method
    • use uniform grid to tally primitive distribution
  – MAHD
    • use uniform grid, weight primitive tallies by inverse of size

Results

• Using previous frame’s region assignment has little detrimental impact
• Static method requires 9-25 regions/proc and 3-5 times communication bandwidth to get same load-balance as adaptive 1 region/proc

Results

• Static method
  – requires no per-primitive overhead
  – fixed-size regions
  – fixed processor-to-framebuffer mapping
• Static method suitable for low-end system (few processors = little overhead)
• To get around screen subdivision scalability limitations, use sort-last compositing on top of sort-first rendering
  – Good enough of an idea to score Mueller a citation in the wiregl paper
Hierarchical Graphics Databases in Sort-First
Carl Mueller (again)

- HGD (scenegraph)
  - Sort-middle / Sort-last
    - Divide primitives equally among processors
  - Sort-first
    - Can divide structures among processors
    - But requires state replication or resolution

Database Representation

- Minimal view
  - “Connectivity” information only, no instances
- Maximal view
  - Explicit instancing

Min-set method

- Processor knows connectivity information, but must check bounding volume of each structure against processor’s region to determine whether to render it or not
- Primitive migration: Push vs. Pull
  - Push wins – less communication, less latency, less computation

Max-set method

- Given min-set representation, processor is assigned a pointer into a structure in min-set for each primitive it needs to render
- Primitive migration
  - Harder - no shared memory
    - Pointers into min-set must use global IDs to be able to find address of structure on new processor

Results

- Max-set slightly better than min-set in terms of transformations
  - Due to bounding-box calculations in min-set method
  - Instancing is a natural application of pointers into the min-set
    - For little or no instancing, min-set pulls ahead

- How many total equations in all three of these papers?
• How many total equations in all three of these papers?
  –1
  • (for optimal number of routing groups in 2-tier communication network in Ellsworth paper)