Parallel Rendering

Ed Angel
Professor of Computer Science, Electrical and Computer Engineering, and Media Arts
Director, Arts Technology Center
University of New Mexico
Introduction

• In many situations, a standard rendering pipeline might not be sufficient
  - Need higher resolution display
  - More primitives than one pipeline can handle

• Want to use commodity components to build a system that can render in parallel

• Use standard network to connect
Power Walls

• Where do we display large data sets?
  - CRTs have low resolution (1 Mpixel)
  - LCD panels improving but still expensive
  - Need resolution comparable to data set to see detail
    • CT/MRI/MEG
    • Ocean data

• Solution?
  - Multiple projectors
    • Commodity
    • High-end
    • See IEECE CG & A (July)
Tiled Display

[Image of a tiled display with multiple panels showing a underwater scene with sharks]
CS Power Wall

- 6 dual processor Intellestations
- G Force 3 Graphics cards
- 6 commodity projectors (1024 x 768)
- Gigabit ethernet
- Back projected screen
- Shared facility with scalable system group
  - Investigate OS and network issues
CS Power Wall
CS Power Wall
Power Wall

- Inexpensive solution but there are some problems
  - Color matching
  - Vignetting
  - Alignment
    - Overlap areas
  - Synching
  - Dark field
Graphics Architectures

• Pipeline Architecture
  - SGI Geometry Engine
  - Geometry passes through pipeline
  - Hardware for
    • clipping
    • transformations
    • texture mapping

Transform → Clip → Project/Sort → Rasterize → Screen
Building Blocks

- Graphics processors consist of geometric blocks and rasterizers
- Geometric units: transformations, clipping, lighting
- Rasterization: scan conversion, shading
- Parallelize by using multiple blocks
- Where to do depth check?
Sorting Paradigm

• We can categorize different ways of interconnecting blocks using a sorting paradigm: each projector is responsible for one area of the screen. Hence, we must sort the primitives and assign them to the proper projector.

• Algorithms can be categorized by where this sorting occurs.
Three Rendering Methods

Sort-First Rendering

Sort-Middle Rendering

Sort-Last Rendering

Composite
Sort First

- Each rasterization unit assigned to an area of the screen
- Each geometric unit coupled to its own rasterizer
- Must sort primitives first
- Can use commodity cards
Sort-First Rendering for a Random Triangles Application
Sort Middle

- Geometric units and rasterization units decoupled
- Each geometric unit can be assigned any group of objects
- Each rasterizer is assigned to an area of the screen
- Must sort between stages
Sort Last

- Couple rasterizers and geometric units
- Assign objects to geometric units to load balance or via application
- Composite results at end
Tree Compositing

- Composite in pairs
- Send color and depth buffers
- Each time half processors become idle

Diagram:

```
<table>
<thead>
<tr>
<th>application</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
</tr>
<tr>
<td>↓</td>
</tr>
<tr>
<td>P1</td>
</tr>
<tr>
<td>↓</td>
</tr>
<tr>
<td>P2</td>
</tr>
<tr>
<td>↓</td>
</tr>
<tr>
<td>P3</td>
</tr>
<tr>
<td>↓</td>
</tr>
<tr>
<td>P3</td>
</tr>
<tr>
<td>↓</td>
</tr>
<tr>
<td>display</td>
</tr>
</tbody>
</table>
```
Binary Swap Compositing

- Each processor responsible for one part of display
- Pass data to right n times

![Diagram of binary swap compositing]

Angel: Interactive Computer Graphics 3E © Addison-Wesley 2002
Sort-Last Rendering for a Random Triangles Application
Comparison

- **Sort first**
  - Appealing but hard to implement

- **Sort middle**
  - Used in hardware pipelines
  - More difficult to implement with add-on commodity cards

- **Sort last**
  - Easy to implement with a compositing stage
  - High network traffic
Mapping to Clusters

- Different architectures
  - Shared vs distributed memory
  - Communication overhead
  - Parallel vs distributed algorithms
- Easy to do sort last
- Must evaluate communication cost
- Standard visualization strategies are incorrect if transparency used
Vista Azul

• Experimental architecture from IBM donated to AHPCC
• Half Intel nodes, half AIX nodes
• Only one (PCI) graphics card per four processors
• Contained a Scalable Graphics Engine (SGE): high speed-high resolution color buffer that is accessible by all processors
Vista Azul
Comparison Between Sort-First and Sort-Last

Sort Last Rendering vs. Sort First Rendering

CPU Time (seconds)

Number of Processors

Sort Last Rendering vs. Sort First Rendering

- Sort Last Rendering
- Sort First Rendering
Performance on a PC Cluster

• Following experiments were done by Ye Cong on the CS cluster
  - 6 Intellestations
  - Gigabit Ethernet
  - GForce 3 graphics

• Show the effect of network
Sort-First vs Sort Last
Random Triangles

Sort-First vs Sort-Last (Intellistations)

Number of Triangles per second

Sort last
Sort first

Number of Processors

Angel: Interactive Computer Graphics 3E © Addison-Wesley 2002
Sort First vs Sort Last

Teapot

![Graph showing the comparison of Sort First vs Sort Last with a teapot as the dataset, indicating the time taken in seconds for different numbers of processors.](image-url)
Azul vs Intellistations

Sort-First (azul vs intellistations)

Number of Triangle per second

Number of Processors

Azul
Intellistations
Software for Parallel Rendering

• Write your own sort-first sort-last
• WireGL/Chromium (Stanford)
• Embed inside package (VTK)
WireGL: A Distributed Graphics System

- A software-based parallel rendering system that unifies the rendering power of a collection of cluster nodes
- Scalability is achieved by integrating parallel applications into its sort-first parallel renders
- Each node in the cluster can be either a rendering client or a rendering server
- Clients submit OpenGL commands concurrently to servers, which render the final physical image
Chromium

• Successor to WireGL
• Allows both sort first and sort last rendering
• Implemented on CS cluster
• Most of gain in performance is because Chromium and WireGL can group state-changing commands separately from rendering commands
Chromium vs Sort First

Sort-First Marching Cube (my-sf vs chromium, Tri=2250818)

MRI rotation

Time (Second)

Number of Processors

chromium

my-sf