## Parallel Rendering

Molnar, Cox, Ellsworth, and Fuchs. "A Sorting Classification of Parallel Rendering." *IEEE Computer Graphics and Applications*. July, 1994.



## Why Parallelism

**Applications need:**

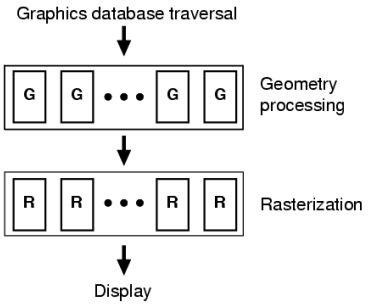
- High frame rates
- High resolution
- Large geometric models
- Stereo
- Antialiasing
- etc.

**Performance implications:**

- Hundreds of MFLOPS compute power
- Gigabytes per second memory bandwidth



## Stages of Parallelism (for object-order rendering)






## Processing Tasks

**Geometry Processors**

- Each processor gets a subset of primitives
- Transformation
- (Lighting)
- Set-up for Rasterization

**Rasterization Processors**

- Each processor gets a subset of pixels
- Visibility computation
- Shading

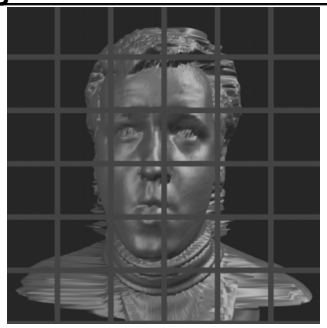


## Rendering as Sorting

- Primitives may lie anywhere on or off screen
- Determine effect of each primitive on each pixel
- Primitives are "sorted" onto screen
- Sorting affects distribution of data on geometry and rasterization processors



## Primitives in Screen-space Regions



## Where to sort

**Sort Middle**

- **Sort between geometry processing and rasterization**

**Sort First**

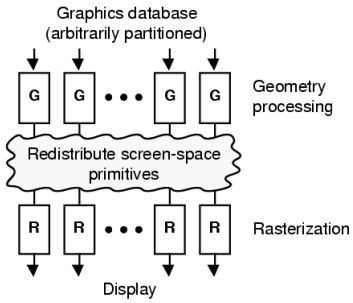
- **Sort during geometry processing**

**Sort Last**

- **Sort during rasterization**



## Sort Middle






## Sort Middle: Data Arrangement

**Geometry processors**

- **Arbitrary (random) distribution of primitives**
- **Good for load balancing**

**Rasterization processors**

- **Screen-space distribution of primitives**
- **Load balancing difficult**



## Sort Middle: Communications

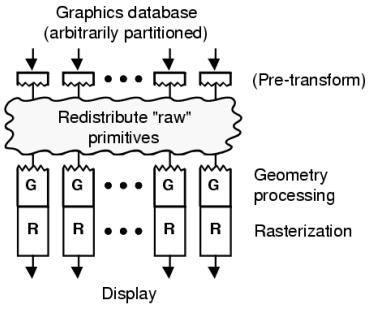
**All geometry transformed primitives must be communicated every frame**

**All geometry processors must communicate with all rasterization processors**

- **$O(n^2)$ communications paths**



## Sort First






## Sort First: Data Arrangement

**Distribute both geometry and rasterization work according to position of primitives on screen**

**Load balancing difficult**

- **Different screen regions of equal sizes may contain different numbers of primitives**
- **May need dynamic region sizes**

## Sort First: Communications

**Must determine primitive screen coverage before full transformation**

**Exploit frame-to-frame coherence**

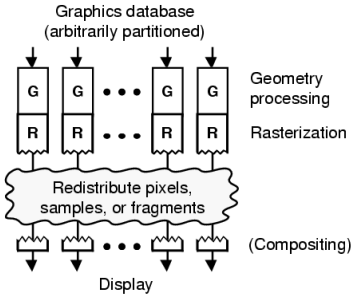
- **Shuffle primitives between geometry processors only if screen coverage changes**

**Possibly employ primitive clustering and bounding volumes**

- **Pre-transform bounding volumes for small groups of primitives**



## Sort Last






## Sort Last: Data Arrangement

**Arbitrary (random) arrangement of data on both geometry and rasterization processors**

**Great for load balancing**

**Each rasterization processor makes image of entire screen, with subset of primitives**



## Sort Last: Communications

**Rasterization processors must communicate final pixel data**

**Composition of pixel data may take place along linear or tree-shaped network**

**Requires high bandwidth, assuming pixel data is much larger than primitive data**



## Advantages and Disadvantages

| | Advantages | Disadvantages |
|---|---|---|
| **SF** | • Low communications when good coherence<br>• Each processor implements entire pipeline | • Susceptible to load imbalance<br>• Retained mode and complex data handling |
| **SM** | • General and straightforward<br>• Natural communications placement | • High communication cost<br>• Rasterizer load imbalance |
| **SL** | • Each processor implements entire pipeline<br>• Easier load balancing<br>• Linear scalability | • Large communication cost, especially for high resolution or multisampling |