May 15, 2009

Parallel Rendering

- Fast Graphics

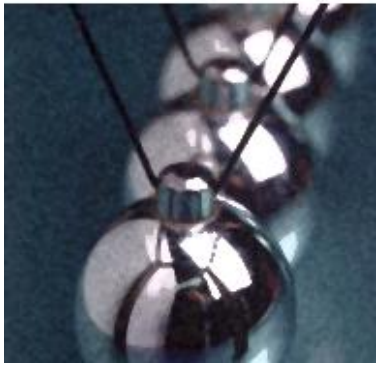
Lei Sun

159.735 Parallel Computing Seminar

Overview

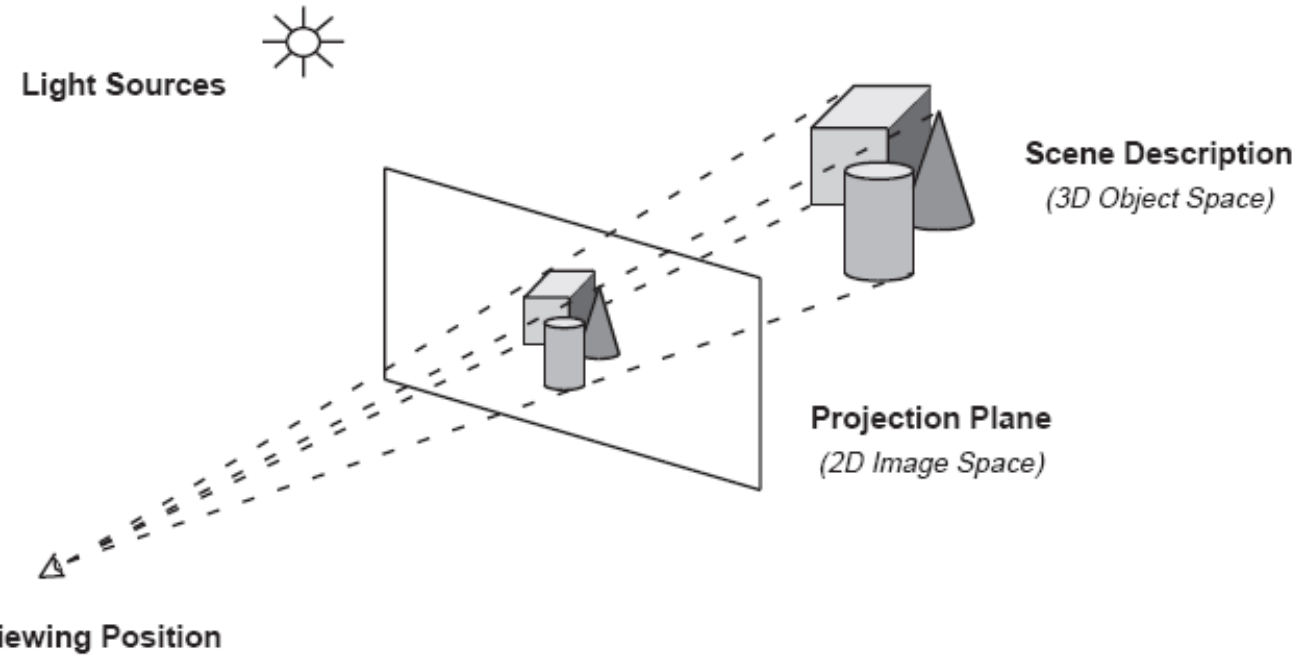
- Introduction
  - Rendering, Problem, Parallel Rendering
- Types of Parallelism in Rendering Process
  - Functional, Data, Temporal, Hybrid
- Algorithms
  - Sort first, sort middle, sort last

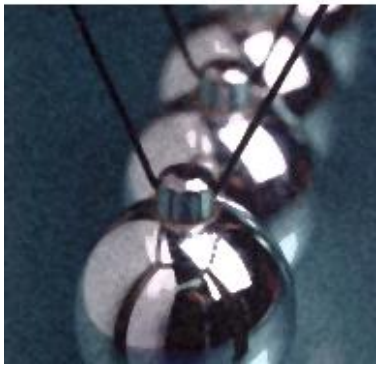
Introduction



Rendering

In computer graphics, *rendering is the process by which an abstract de*scription of a scene is converted to an image.





- A 3D scene is projected onto an image plane.
- Taking into account the viewing parameters and light sources.
- The rendering operation illuminates the objects and projects them into two-dimensional *image space, where color intensities of* individual pixels are computed to yield a final image.

Problem

- For complex scenes or high-quality images, the rendering process is computationally intensive, requiring millions or billions of floating-point and integer operations for each image. (e.g. medical visualization, iso-surface generation, CAD applications)
- The need for interactive or real-time response in many applications places additional demands on processing power.
- Traditional methods like ray tracing, 3D textures, etc., work extremely slowly in simple machines.
- The only practical way to obtain the needed computational power is to exploit multiple processing units to speed up the rendering task, which is known as

Parallel rendering.

Parallel Rendering

- A method used to improve the performance of computer graphics creation software.
- Divides the work to be done and processes it in parallel.

  For example:

  Non-parallel ray-casting application - send rays one by one to all the pixels in the view frustum.

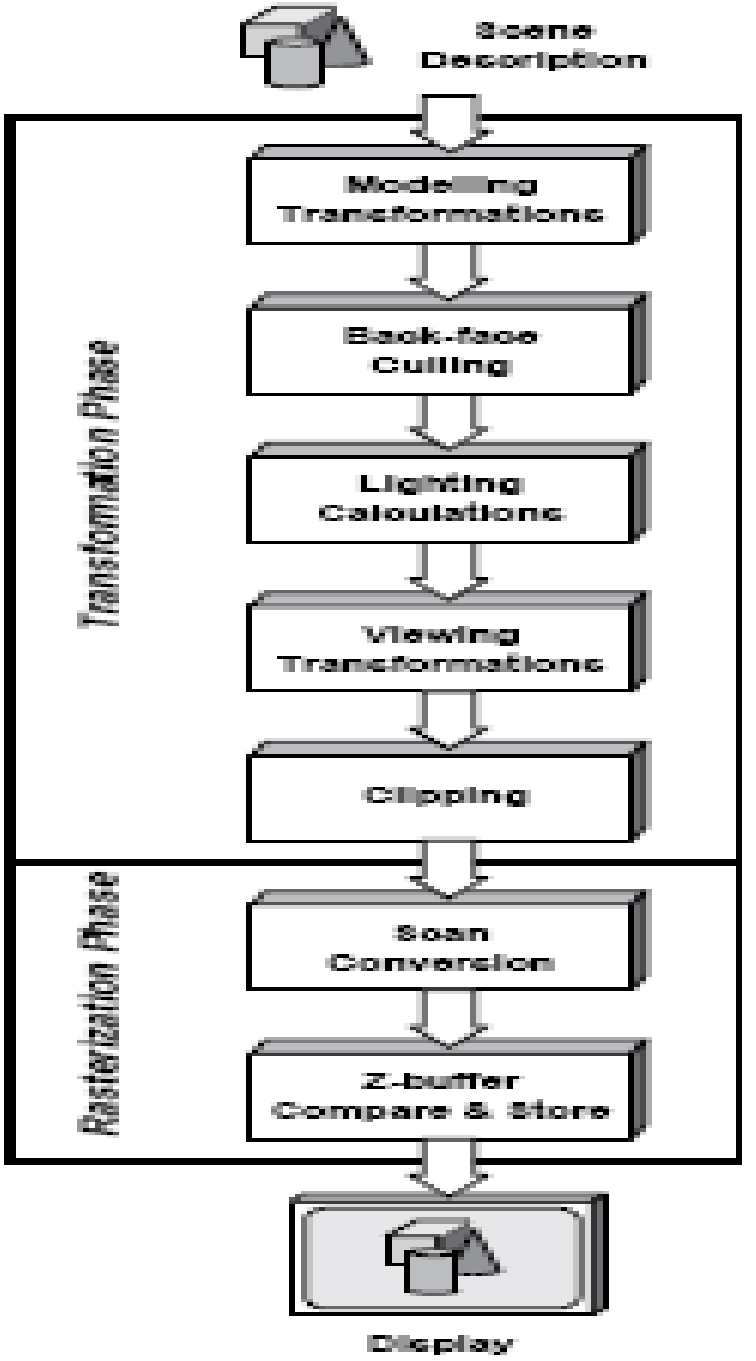
  Parallel - divide the whole frustum into x number of parts, then run that many threads or processes to send rays in parallel to those x tiles.
- Parallel rendering has been applied to virtually every image generation technique used in computer graphics, including surface and polygon rendering, terrain rendering, volume rendering, ray-tracing, and radiosity.

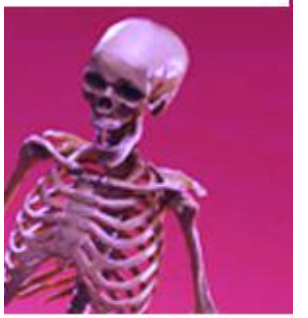
Types of **Parallelism in the Rendering Process**

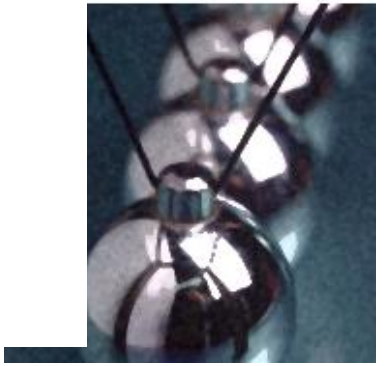
Several different types of parallelism can be applied in the rendering process. These include *functional parallelism, data parallelism, and temporal parallelism*. *These basic types can also be combined into hybrid systems which* exploit multiple forms of parallelism.

Functional parallelism

Split the rendering process into several distinct functions which can be applied in series to individual data items. Rendering *pipeline is formed by assigning* a processing unit to each function (or group of functions) As a processing unit completes work on one data item, it forwards. it to the next unit, and receives a new item from its upstream neighbor.

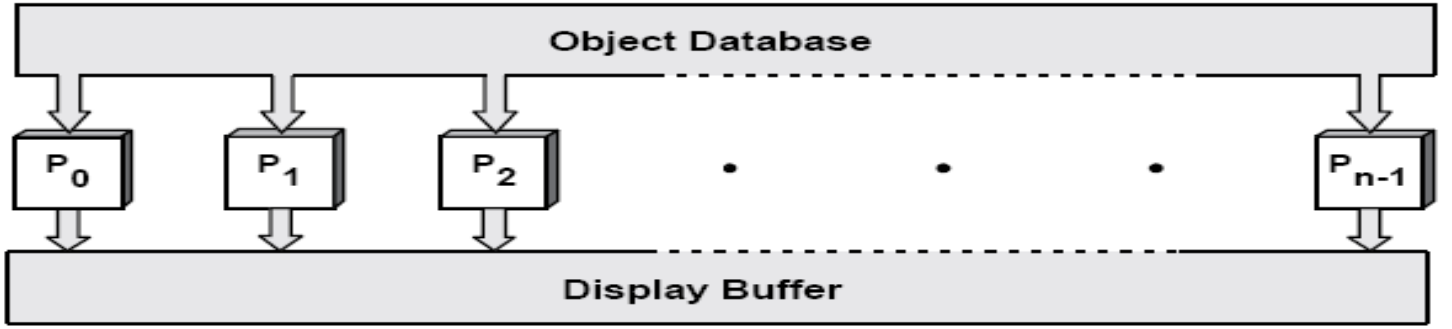

Scene Description
Modeling Transformations
Back-face Culling
Lighting Calculations
Viewing Transformations
Clipping
Transformation Phase
Scan Conversion
Z-buffer Compare & Store
Rasterization Phase
Display





Data parallelism

Split the data into multiple streams and operate on several items simultaneously by replicating a number of identical rendering units.



Multiple data items are processed simultaneously and the results are merged to create the final image

Object parallelism*:* operations performed independently on the geometric primitives which comprise objects in a scene. (*modeling, viewing transformations, lighting computations, clipping*)
- transformation phase

Image parallelism: operations used to compute individual pixel values. (illumination, interpolation, composition, visibility determination.)
- rasterization phase.

Temporal parallelism

Decomposing the problem in the time domain. The fundamental unit of work is a complete image, and each processor is assigned a number of frames to render, along with the data needed to produce those frames.

Hybrid approaches

Incorporate multiple forms of parallelism in a single system. E.g. functional + data parallel.

Algorithms

Embarrassingly parallel algorithms

Non-interactive parallel rendering

Problems can be parallelized trivially, requiring little or no inter-processor communication, and with no significant computational overheads.

Rendering algorithms which exploit temporal parallelism typically fall into this category.

Rendering methods based on ray-casting (such as ray-tracing and direct volume rendering) also have embarrassingly parallel implementations

Interactive parallel rendering

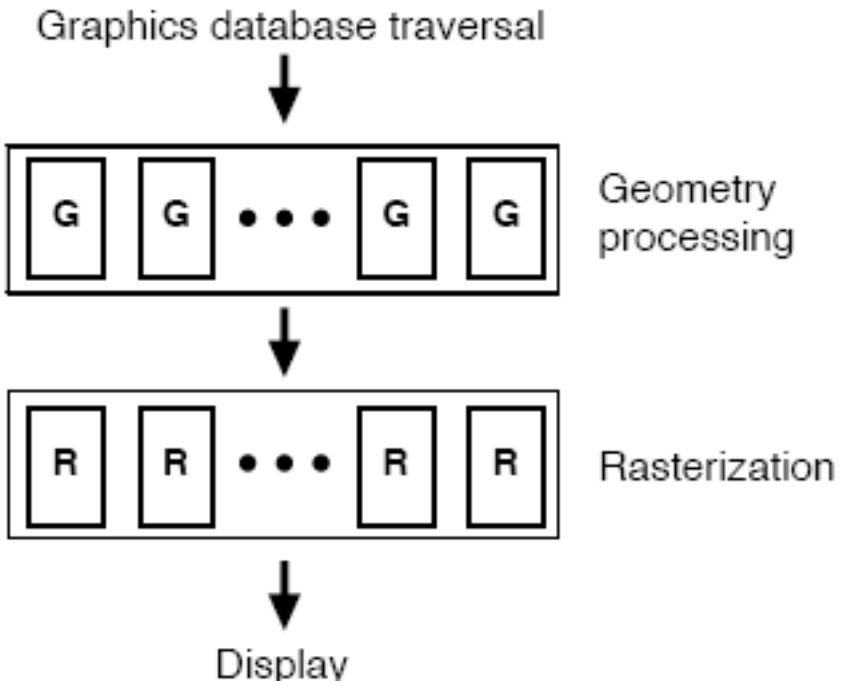
Sort-first, Sort middle, Sort last.

Pixel Decompositions: Divide the pixels of the final view evenly, either by dividing full pixels or sub-pixels, No sorting of rendered primitives takes place.

DPlex rendering: Distributes full, alternating frames to the individual rendering nodes. It scales very well, but increases the latency between user input and final display, which is often irritating for the user.

Stereo decomposition: Used for immersive applications, where the individual eye passes are rendered by different rendering units. Passive stereo systems are a typical example for this mode.

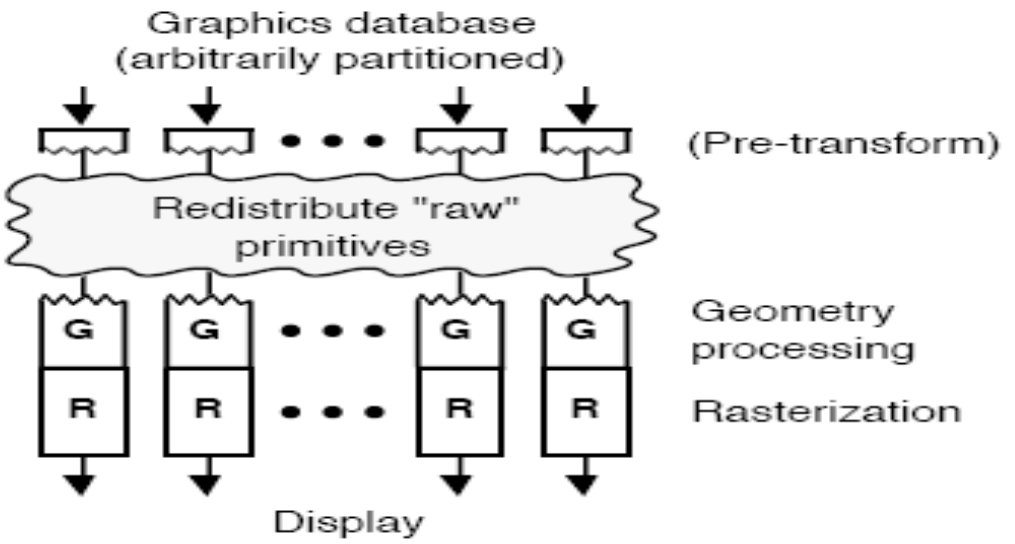
Parallel rendering as a sorting problem



A rendering pipeline consists of two principal parts: **geometry processing** (transformation, clipping, lighting, etc.), and **rasterization** (scan-conversion, shading, and visibility determination). .

- **Geometry processing** usually is parallelized by assigning each processor a subset of the primitives (objects) in the scene.
- **Rasterization** usually is parallelized by assigning each processor a portion of the pixel calculations.
- **Rendering task** is to calculate the effect of each primitive on each pixel. Due to the arbitrary nature of the modeling and viewing transformations, a primitive can fall anywhere on (or off) the screen.
- **Rendering can be viewed as a problem of sorting primitives to the screen**
- The sort can take place anywhere in the rendering pipeline: **during geometry processing (sort first)**, **between geometry processing and rasterization (sort middle), or during rasterization (sort last).**

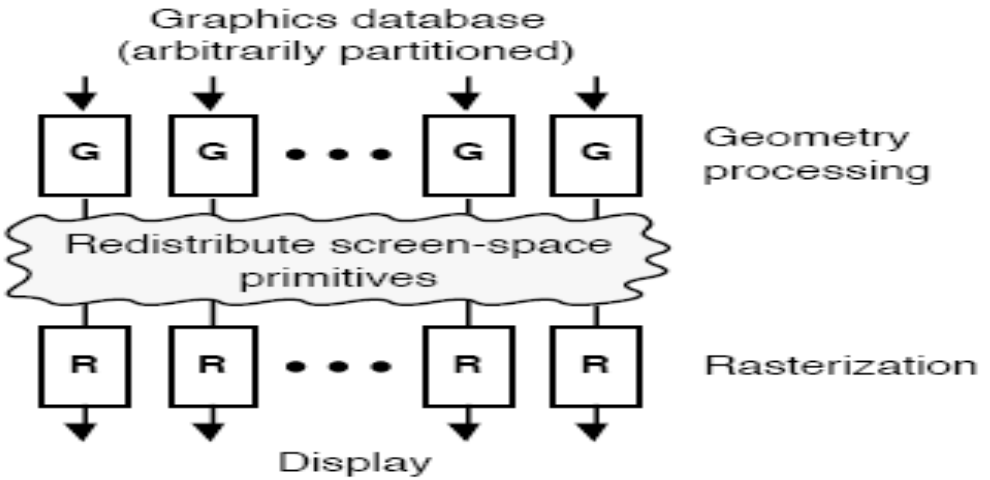
Sort First



Decomposes the final view in screen space, each processor renders a 2D tile of the final view and *responsible for all rendering calculations that* affect their respective screen regions

Another name: tilesort

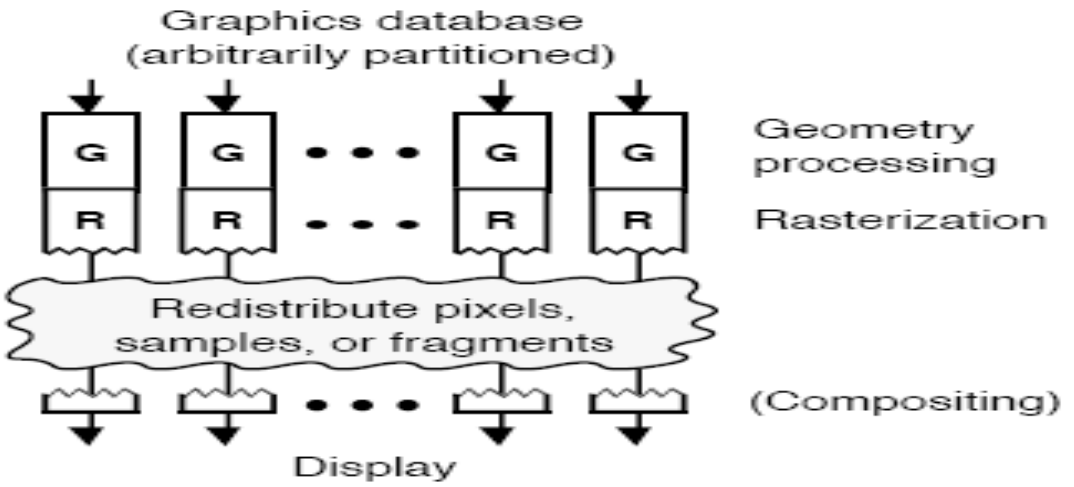
Sort Middle



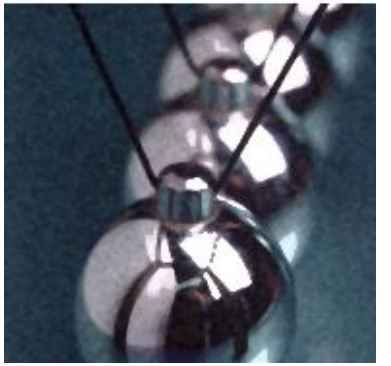
Geometry processors are assigned arbitrary subsets of the primitives, position, Rasterizers are assigned a portion of the display screen. Geometry processors transform, light, etc. their portion of the primitives and classify them with respect to screen region boundaries. They then transmit all of these screen-space primitives to the appropriate rasterizer

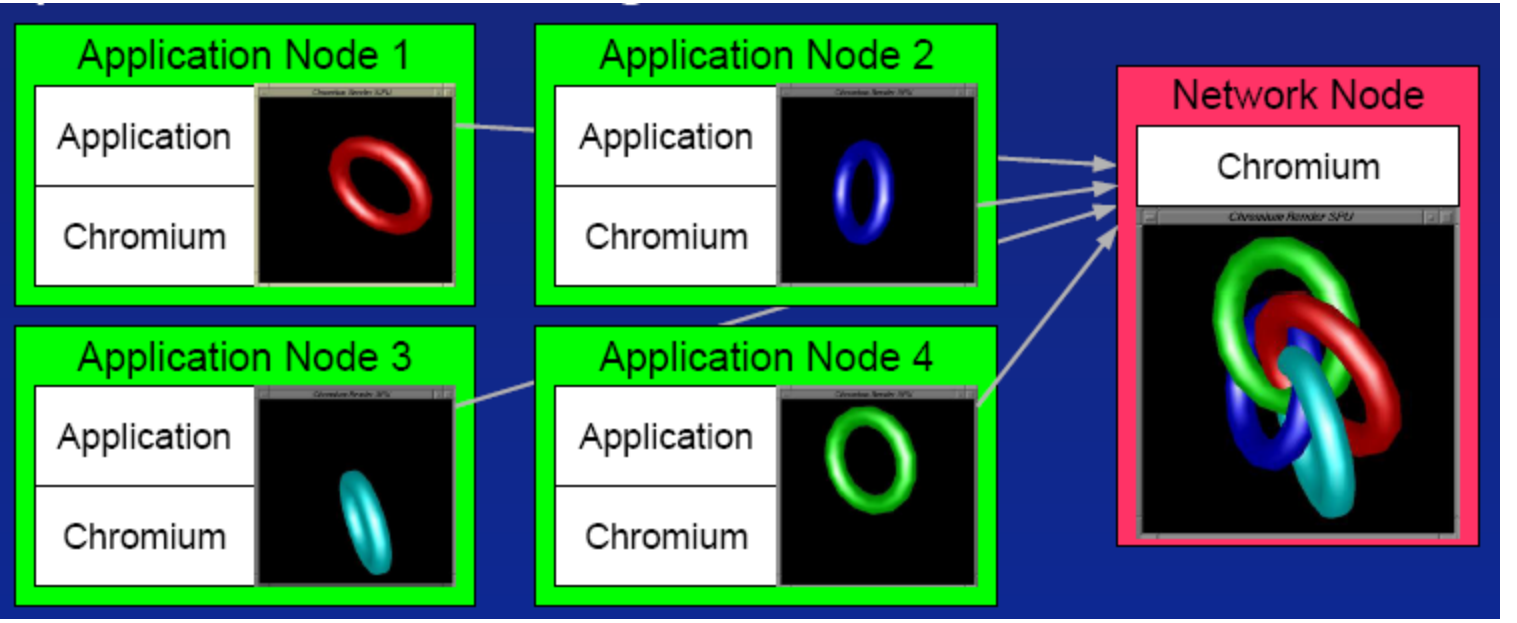
Sort-middle is general and straightforward,
the most common parallel rendering systems.

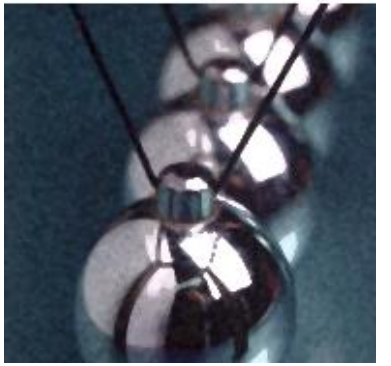
Sort Last



Each processor *assigned arbitrary subsets of the* primitives and computes pixel values for its subset, Renderers then transmits these pixels over an interconnect network to *compositing processors which recompose pixels* from each renderer.




Application Node 1
Application
Chromium
Application Node 2
Application
Chromium
Application Node 3
Application
Chromium
Application Node 4
Application
Chromium
Network Node
Chromium

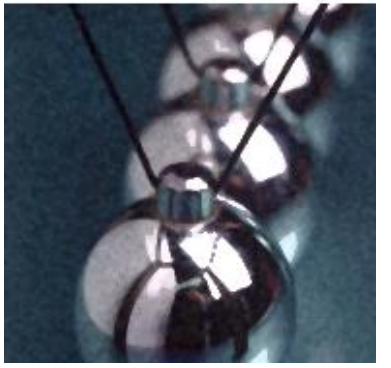


Advantages and Disadvantage

Sort First

- Low communication requirements
- Processors implement entire rendering pipeline for a portion of the screen.

- Primitives may clump into regions, concentrating the work on a few renderers.



Sort Middle

- General and straightforward; redistribution occurs at a natural place in the pipeline.

- High communication costs

Sort Last

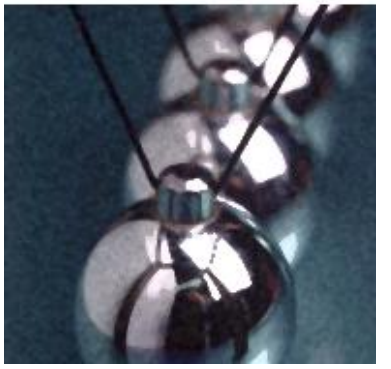
- Scales the rendering very well,

- Recomposition step is expensive due to the amount of pixel data processed during recomposition

Conclusion

Parallel processing techniques have been applied to virtually every computationally-intensive task in computer graphics. Demanding applications such as real-time simulation, animation, virtual reality, photo-realistic imaging, and scientific visualization all benefit from the use of parallelism to increase rendering performance. These applications are also primary motivators in the development of parallel rendering methods.

There are a wide range of implementation strategies in the algorithms. A successful parallel renderer must take into account application requirements, architectural parameters, and algorithmic characteristics.



References:

[1] Steven Molnar - A Sorting Classification of Parallel Rendering
[2] Thomas W. Crockett - Parallel rendering
[3] Rudrajit Samanta - Hybrid Sort-First and Sort-Last Parallel Rendering with a Cluster of PCs.

Any Questions?