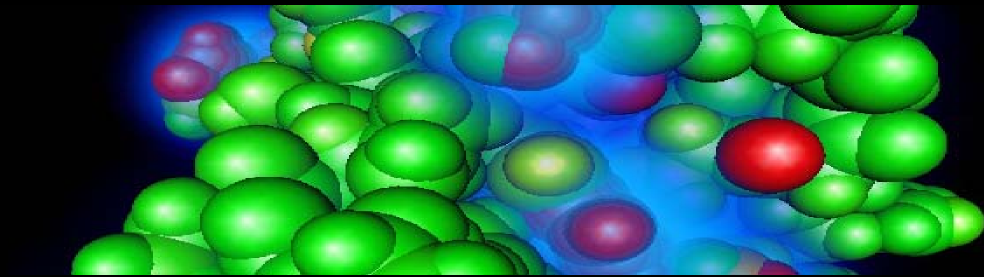


Parallelism in Graphics and Visualization

Wes Bethel

Lawrence Berkeley National Laboratory

May 9, 2005

Outline

- Why computer graphics and visualization?
- Ray Tracing and the Shading Equation.
- Graphics APIs and the Graphics Pipeline.
- Parallelism in Graphics.
- The Visualization Pipeline, and parallelization.

Why Computer Graphics and Visualization?

Office of Science
U.S. DEPARTMENT OF ENERGY

BERKELEY LAB

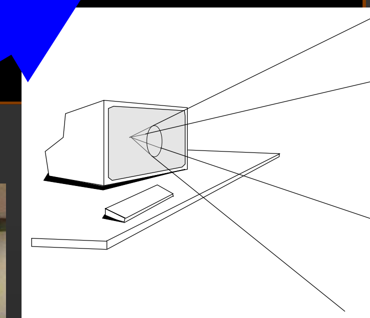
Why Computer Graphics and Visualization?

- The visual cortex and associated machinery occupy more than half our brains – we are innately visual creatures.
- Vision is our principal means for understanding and interacting with the world.
- The best connection between humans and computers is through the high-bandwidth connection of our highly evolved visual system.

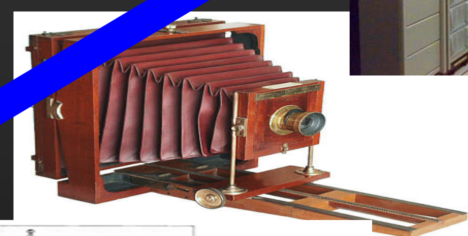
Office of Science
U.S. DEPARTMENT OF ENERGY

BERKELEY LAB

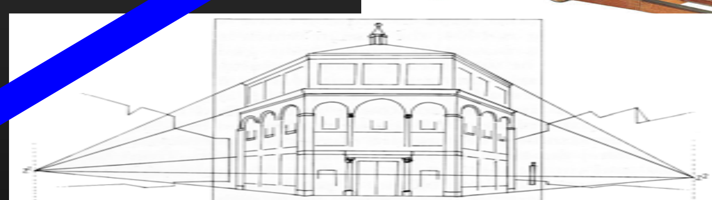
The Path Towards Graphics and Visualization



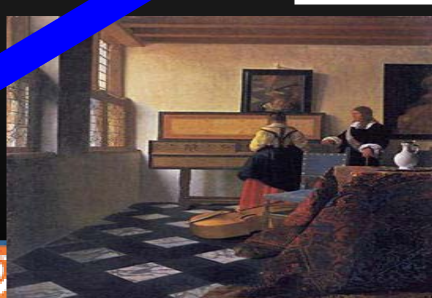
Computers



Photography



Geometry and perspecitive



Art and painting

The Invention of Drawing

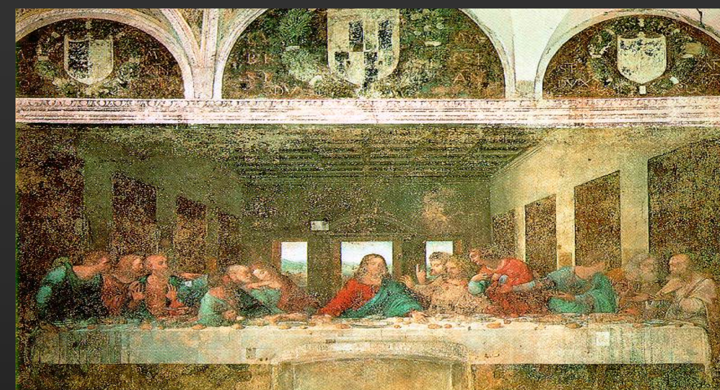
- **Painting based on mythical tale as told by Pliny the Elder: Corinthian man traces shadow of departing lover. Detail from** The Invention of Drawing**, 1830: Karl Friedrich Schinkle (Mitchell p.1)**

Office of Science
U.S. DEPARTMENT OF ENERGY

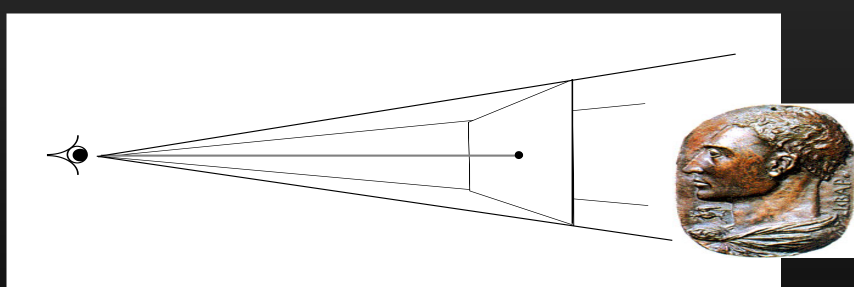
BERKELEY LAB

Understanding Perspective

Incorrect perspective, Giotto, c.1295-1300



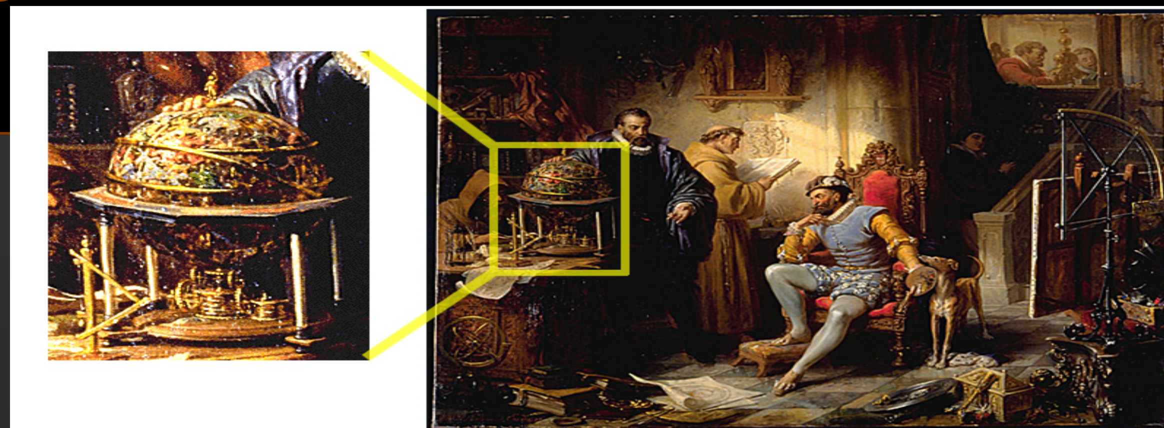
Correct perspective,
Last Supper, Da Vinci, 1495

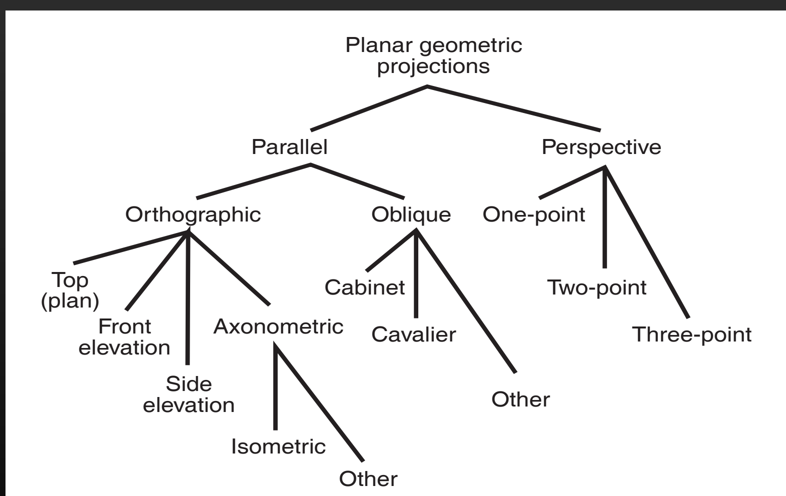


Alberti’s 1435 treatise, *Della Pittura*, explained perspective for the first time

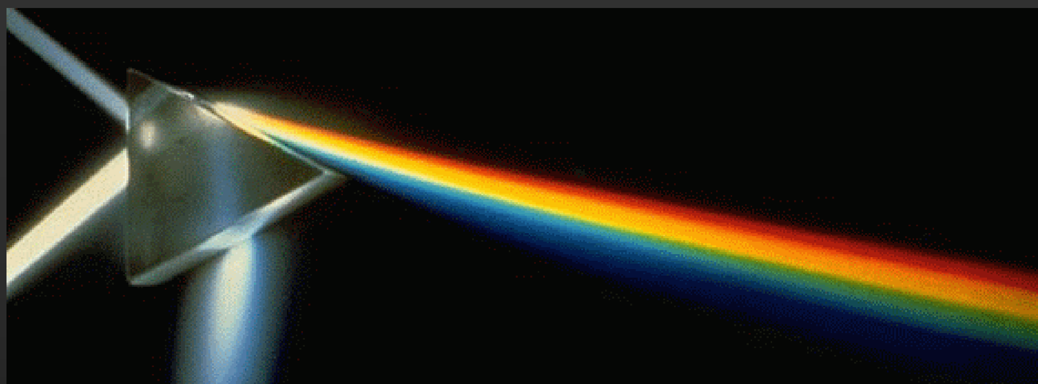
Projections

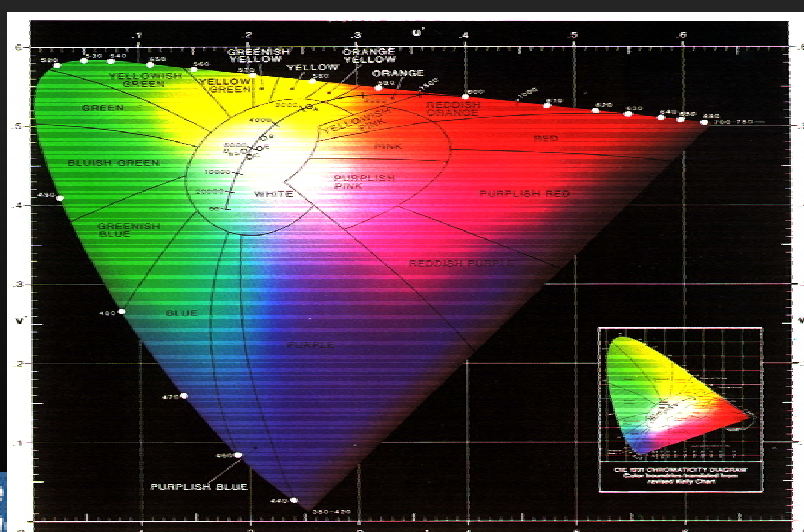
Ender, c. 1855
(detail of clockwork)



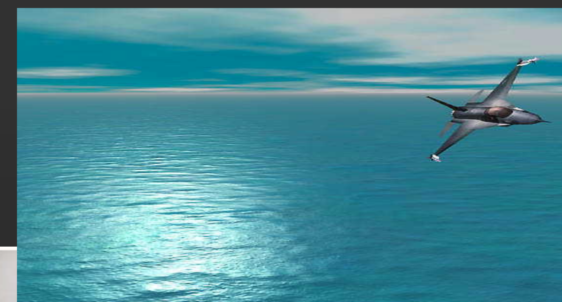


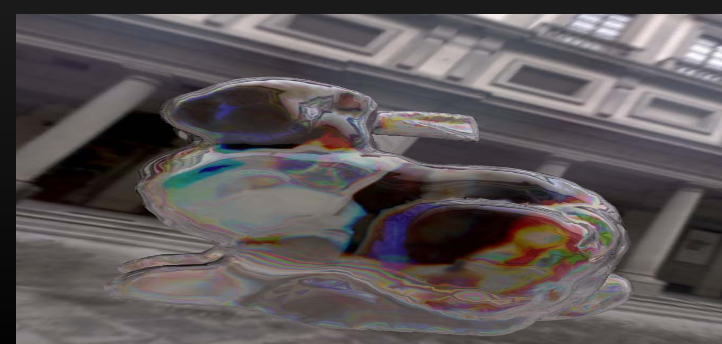
Color and Perception





Decades of Research in Computer Graphics

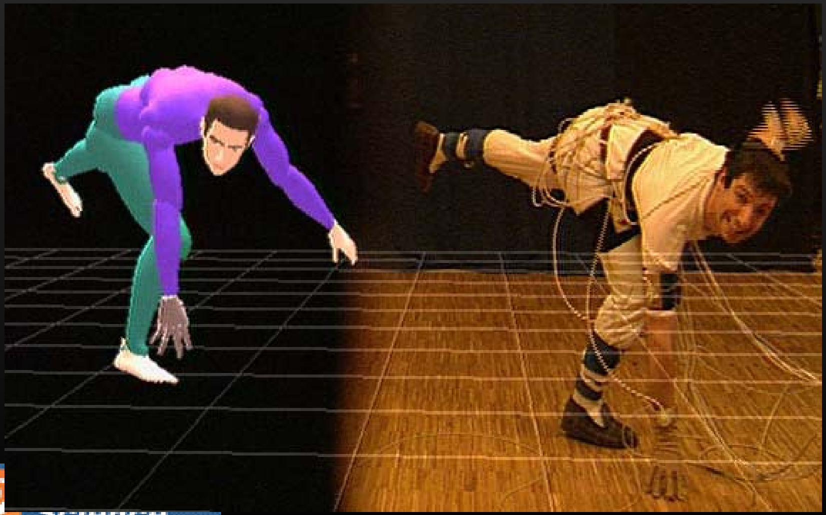




The Quest for Photorealism

Movies and Games

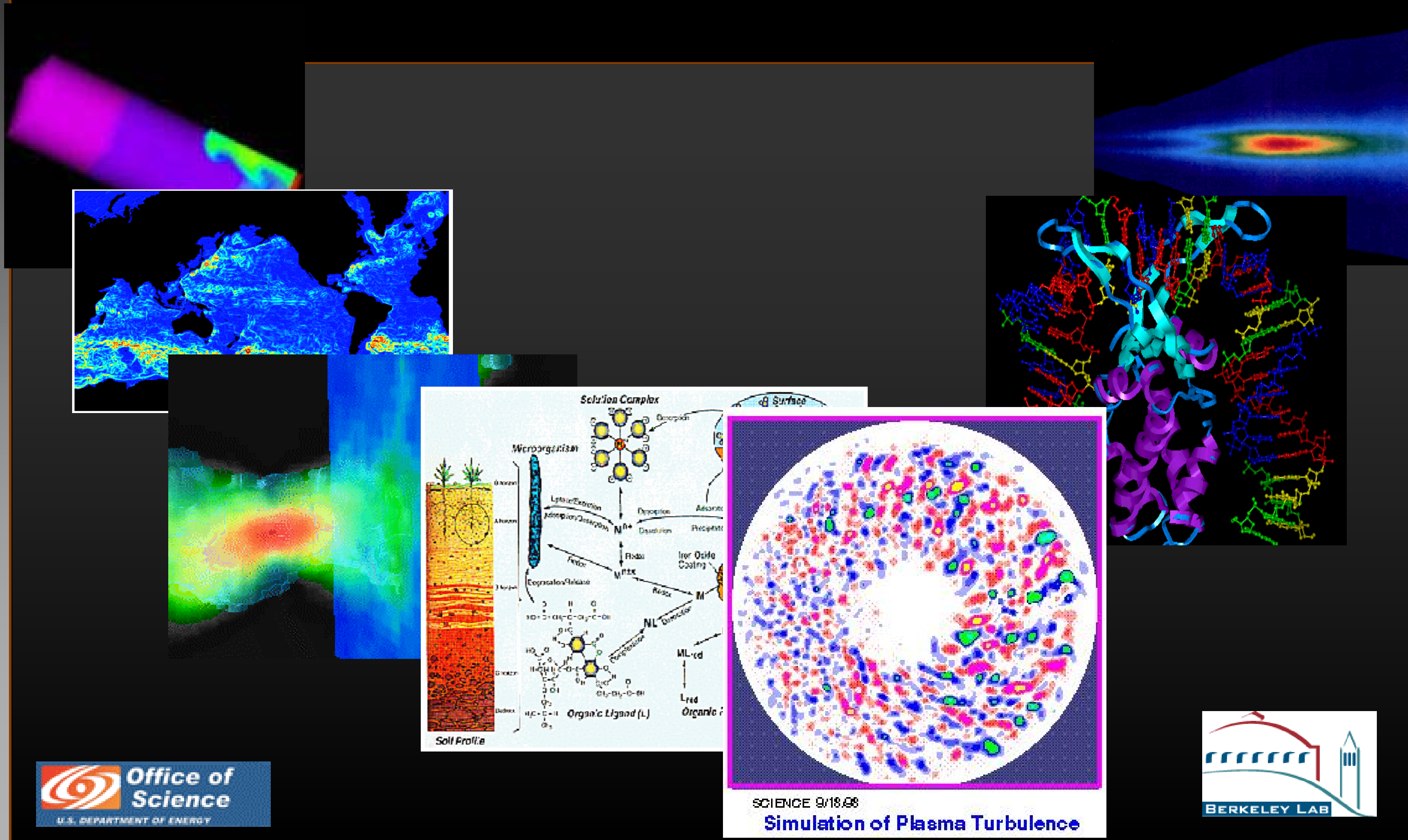
- Digital extras.
- Motion capture.



Science
U.S. DEPARTMENT OF ENERGY

BERKELEY LAB

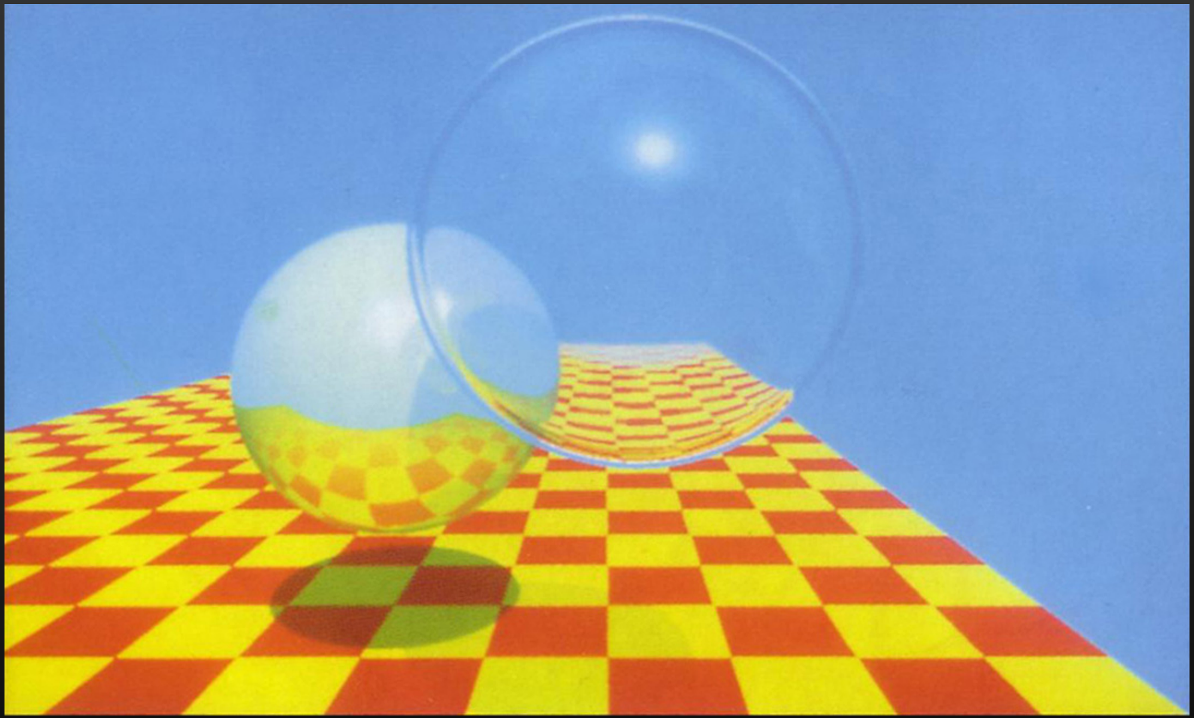
Visualization – For Gaining Insight



Office of Science
U.S. DEPARTMENT OF ENERGY

BERKELEY LAB

Ray Tracing



Ray Tracing

- An Improved Illumination Model for Shaded Display, Whitted, SIGGRAPH 1980.
- The role of the illumination model is to determine how much light is reflected to the viewer from a visible point on a surface as a function of light source direction and strength, viewer position, surface orientation, and surface properties.

Phong's Illumination Equation

$$I = I_a + k_d \sum_{j=1}^{j=ls} (\bar{N} \cdot \bar{L}_j) + k_s \sum_{j=1}^{j=ls} (\bar{N} \cdot \bar{L}'_j)^n, \qquad (1)$$

where

I = the reflected intensity,
I_a = reflection due to ambient light,
k_d = diffuse reflection constant,
$\bar{N}$ = unit surface normal,
$\bar{L}_j$ = the vector in the direction of the jth light source,
k_s = the specular reflection coefficient,
$\bar{L}'_j$ = the vector in the direction halfway between the viewer and the jth light source,
n = an exponent that depends on the glossiness of the surface.

Whitted's Improvement

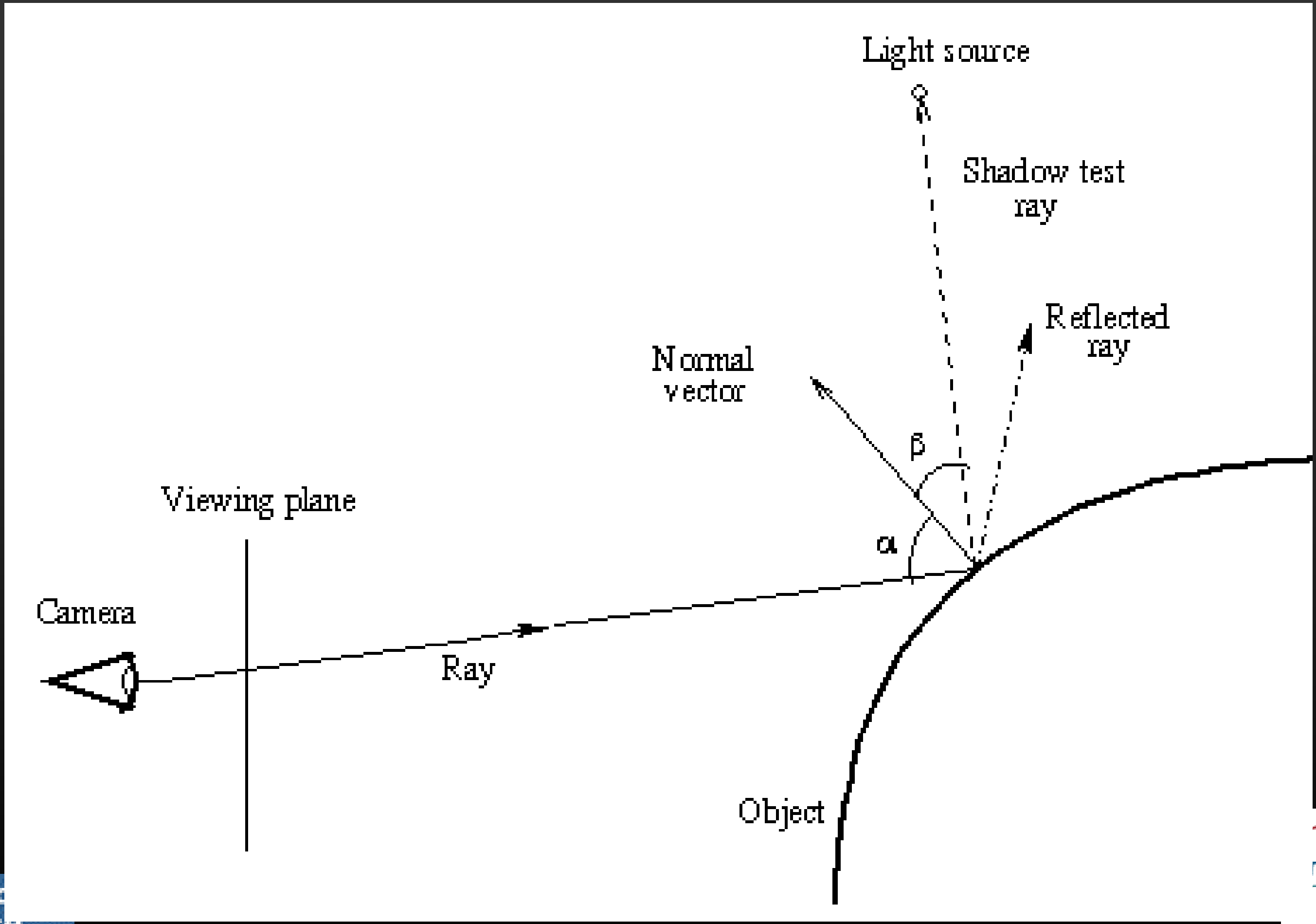
- **Retain diffuse term (computationally overwhelming to account for scene objects as light sources) from Phong Model.**
- **Refine specular term, add transmissive term.**

$$I = I_a + k_d \sum_{j=1}^{j=ls} (\bar{N} \cdot \bar{L}_j) + k_s S + k_t T, \qquad (2)$$

where

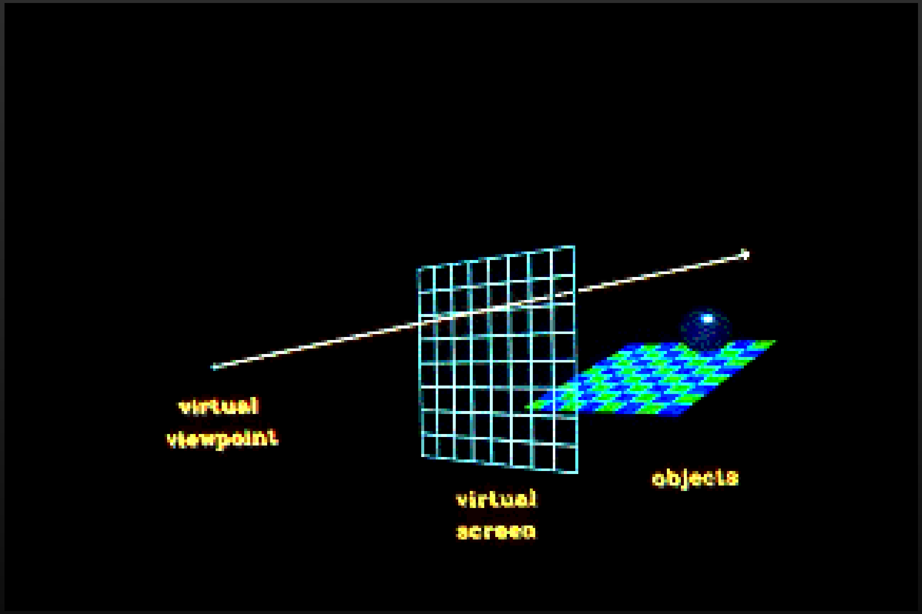
S = the intensity of light incident from the $\bar{R}$ direction,
k_t = the transmission coefficient,
T = the intensity of light from the $\bar{P}$ direction.

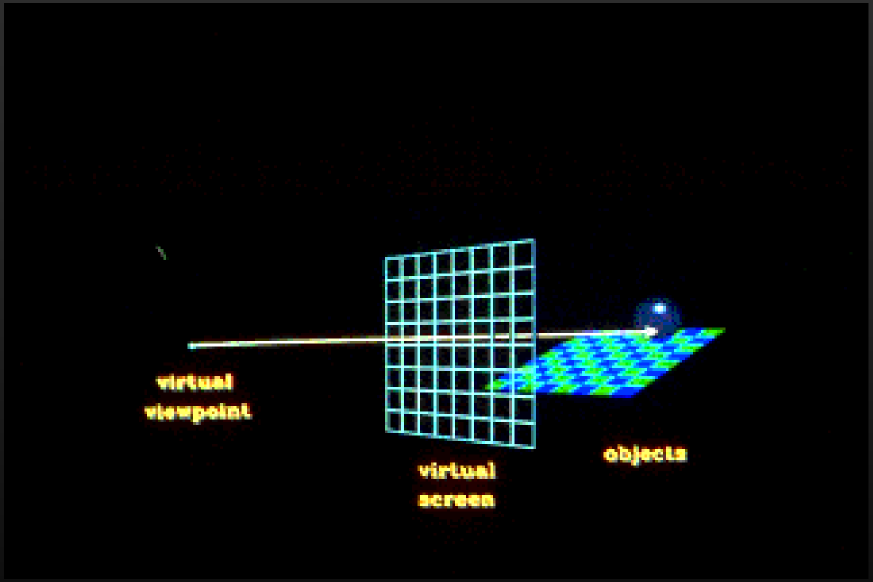
Whitted's Improvement, ctd.



Example Rays – Simple Intersections

- (See http://www.siggraph.org/education/materials/HyperGraph/raytrace/rtrace1.htm)

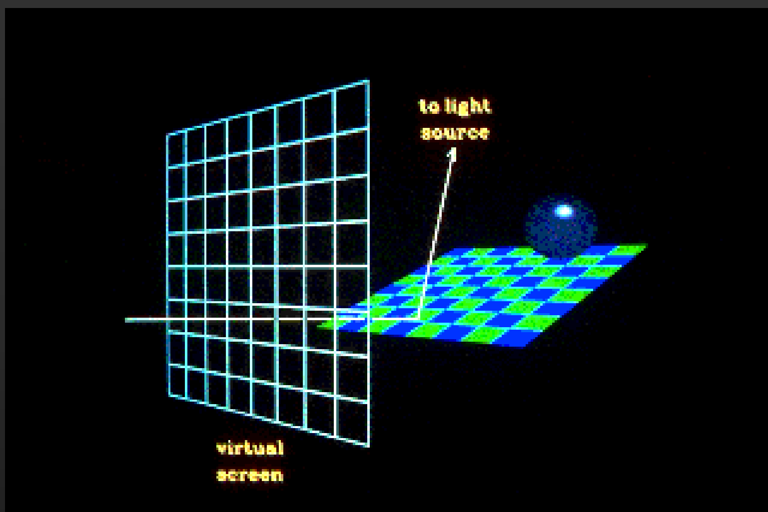


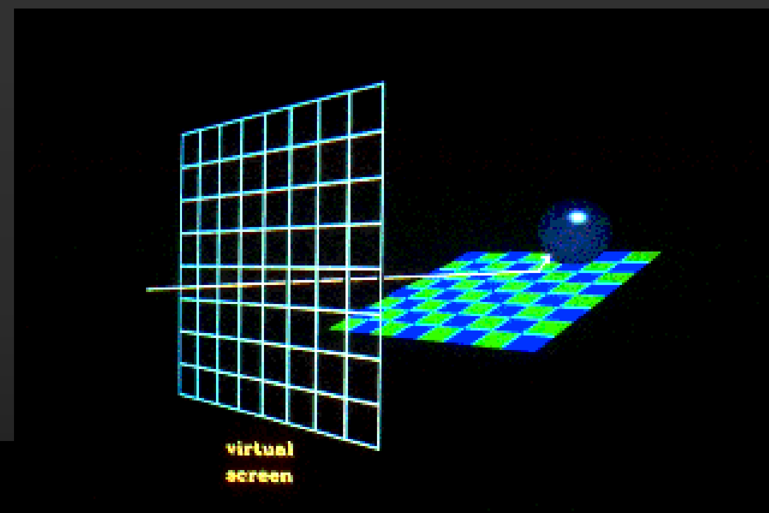


Office of Science
U.S. DEPARTMENT OF ENERGY

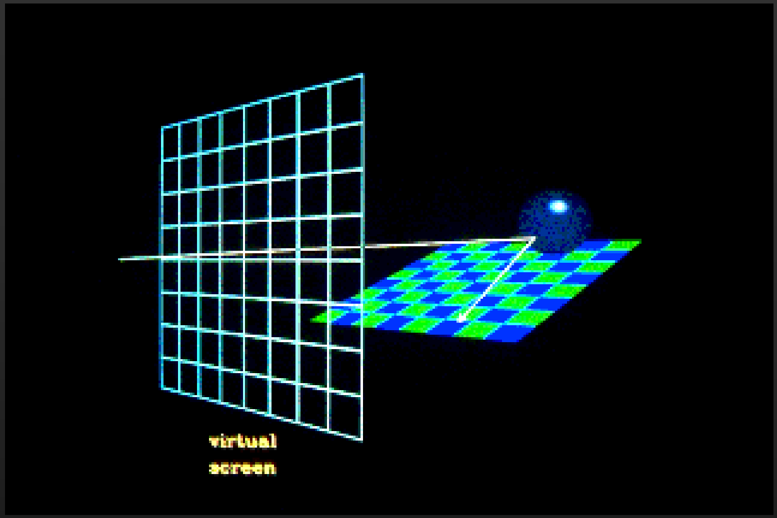
BERKELEY LAB

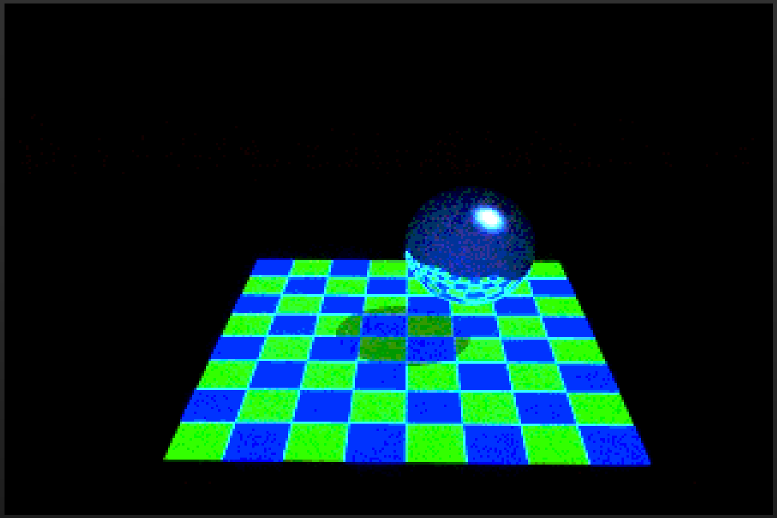
Shadow Rays





Reflected Rays

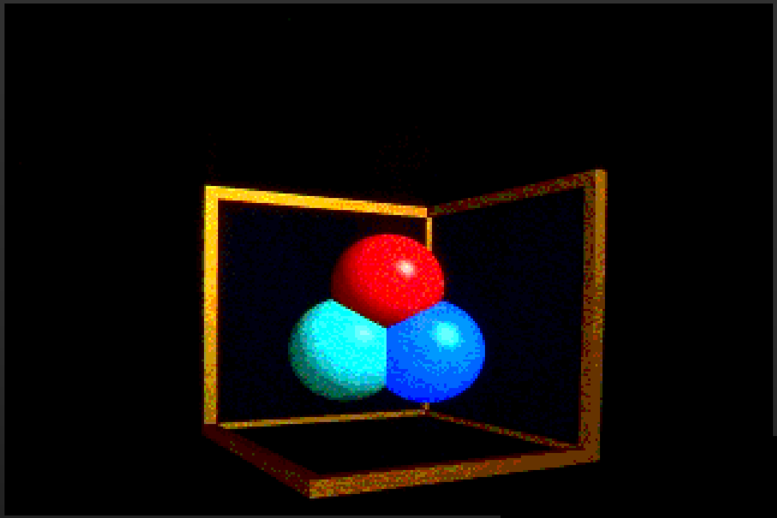


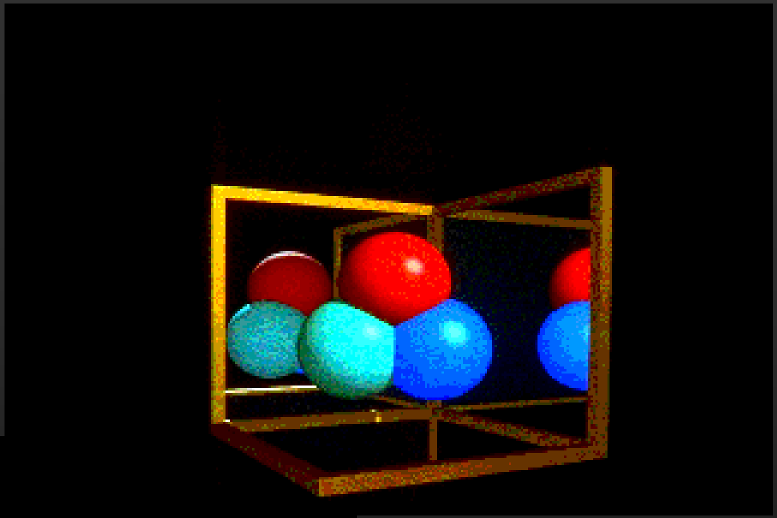


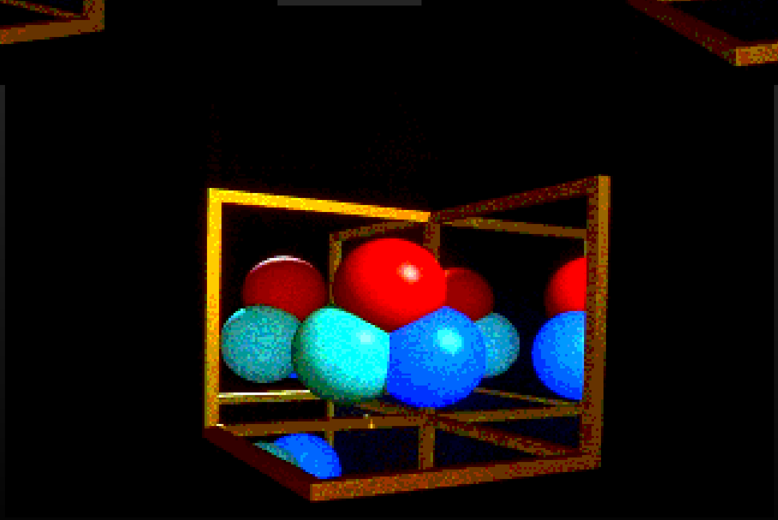
Office of Science
U.S. DEPARTMENT OF ENERGY

BERKELEY LAB

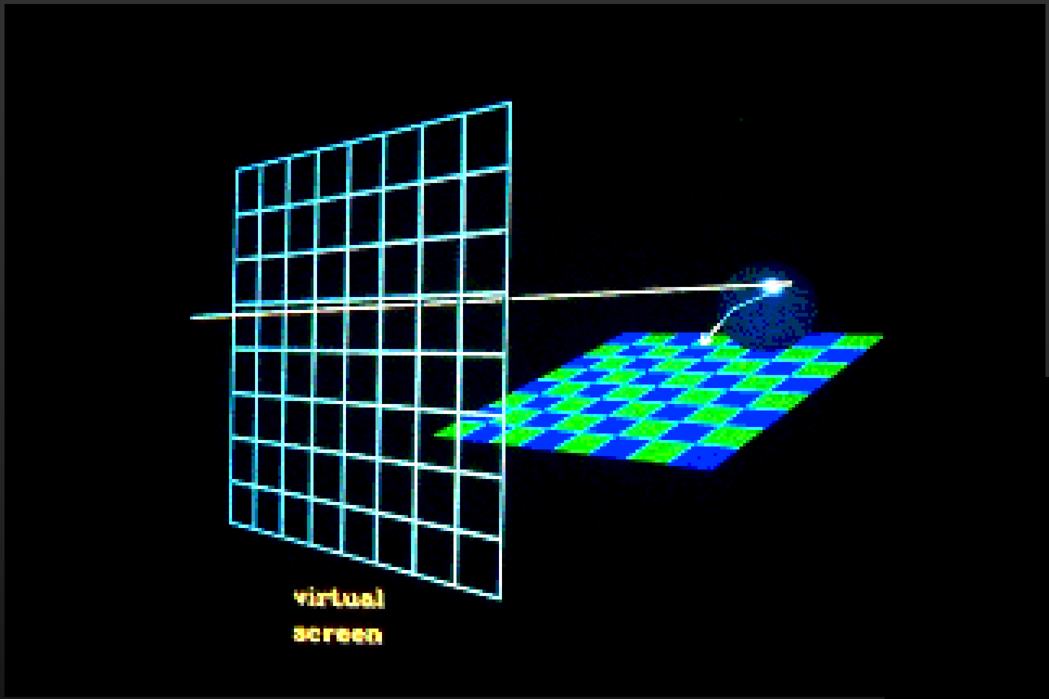
More Reflected Rays

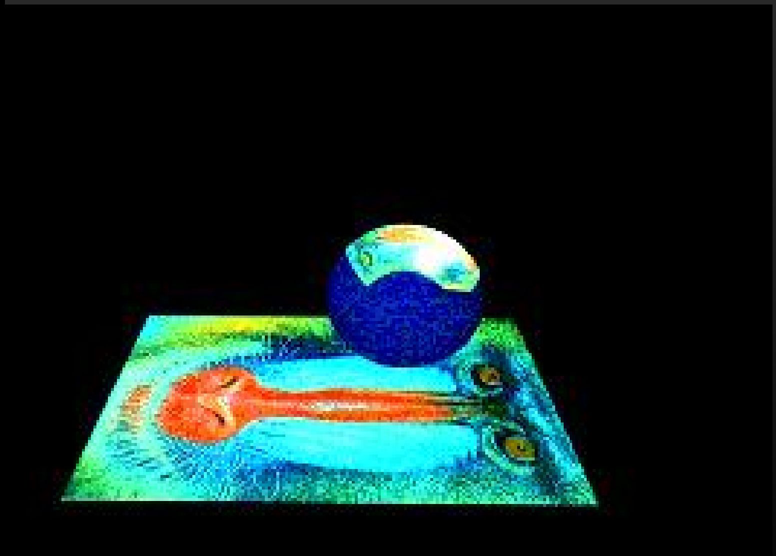






Refracted Rays





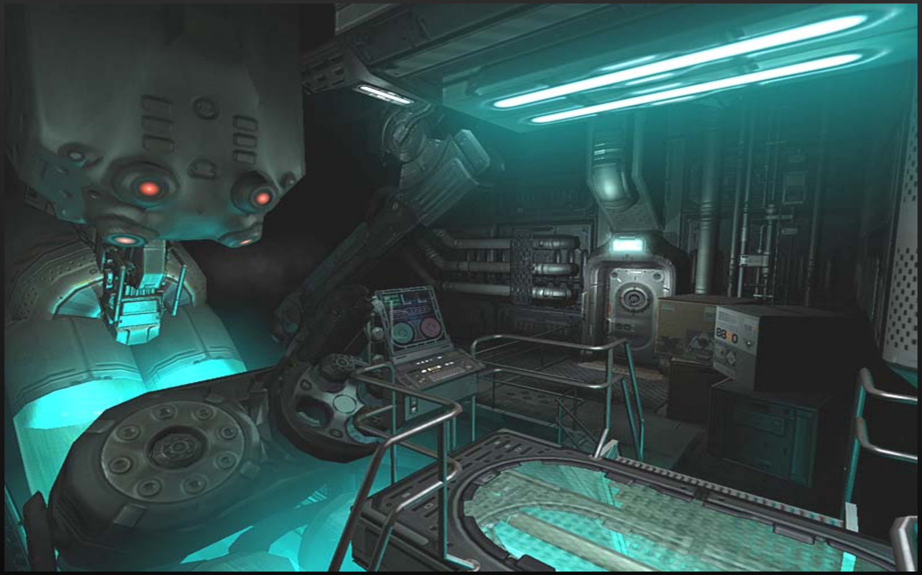
Office of Science
U.S. DEPARTMENT OF ENERGY

Parallelizing Ray Tracing

- **Computation of color at each pixel is relatively independent.**
  - Image-order parallelization.
  - Requires (potential) duplication of the scene on each node.
- **Challenges**
  - Load balancing.
  - Data distribution.
- **vs. Accelerating Ray Tracing**

Graphics APIs and the Graphics Pipeline

(Images from Doom3)



Office of Science
U.S. DEPARTMENT OF ENERGY

BERKELEY LAB

The Question You Might Be Asking

- What does a discussion of graphics APIs have to do with parallelism?
- Answer: you will better understand the breadth and depth of parallelism in graphics and visualization with some background information about how it all works.

What is a Graphics API?

- **Declarative: what, not how.**
  - Scene description: viewer here, trees there, lights just so.
  - Can use a variety of rendering systems: Renderman, POVRay, Performer, OpenRM Scene Graph, etc.
- **Imperative: how, not what.**
  - A series of draw commands.
  - OpenGL, DirectX, D3D, Xlib, etc.

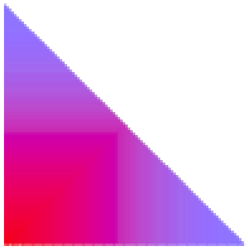
Office of Science
U.S. DEPARTMENT OF ENERGY
BERKELEY LAB

OpenGL

- **Industry-standard graphics API.**
  - Supported on all major platforms.
  - Relationship between OpenGL and window system.
- **Basic primitives: lines, triangles, points.**
  - Higher level primitives (NURBS, quadrics) built using fundamental geometry.
- **Phong lighting model, but Gouroud shading.**

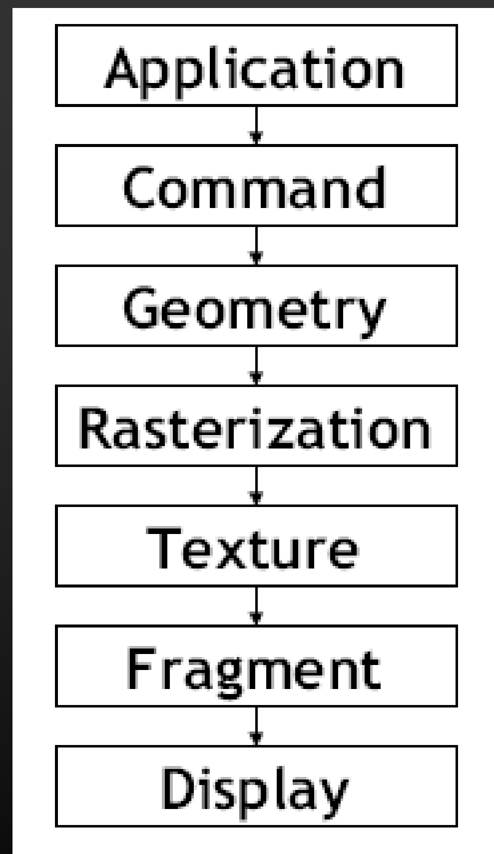
Example OpenGL Code

```
glBegin(GL_POLYGON);
  glColor(RED);
  glVertex3i(0,0,0);
  glVertex3i(1,0,0);
  glVertex3i(0,1,0);
glEnd()
```



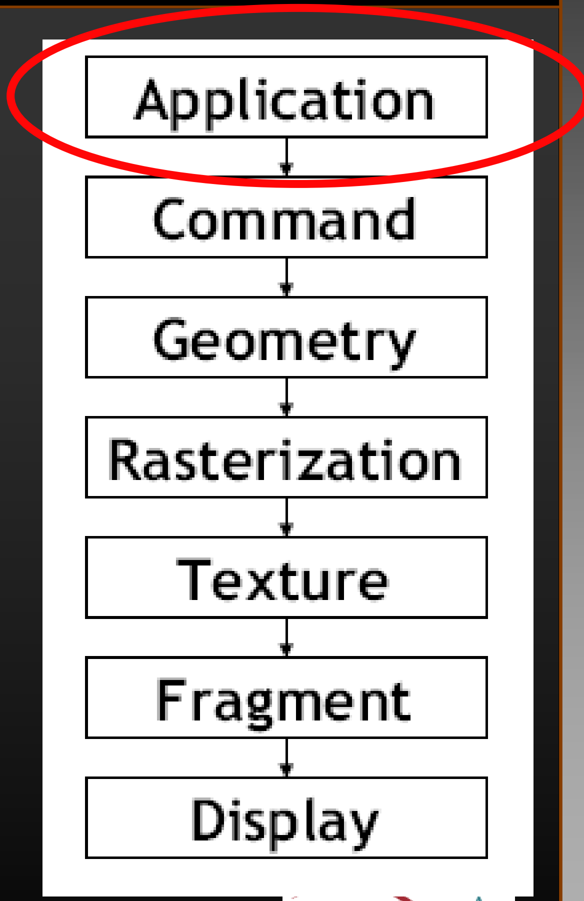
```
glBegin(GL_POLYGON);
  glColor(RED);
  glVertex3i(0,0,0);
  glColor(BLUE);
  glVertex3i(1,0,0);
  glColor(BLUE);
  glVertex3i(0,1,0);
glEnd()
```

General: Graphics Pipeline



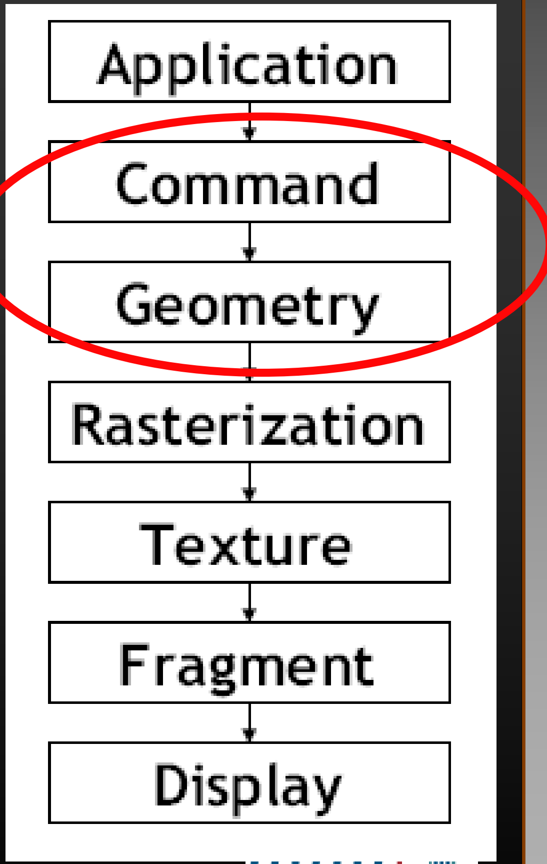
The Graphics Application

- **Application**
  - Simulation
  - Visualization
  - Database Traversal
  - User interaction (games!)



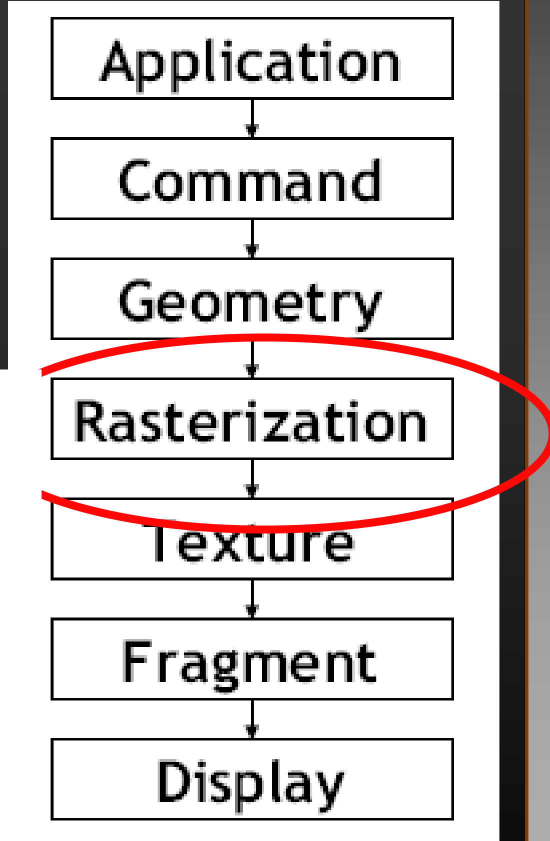
Commands and Geometry

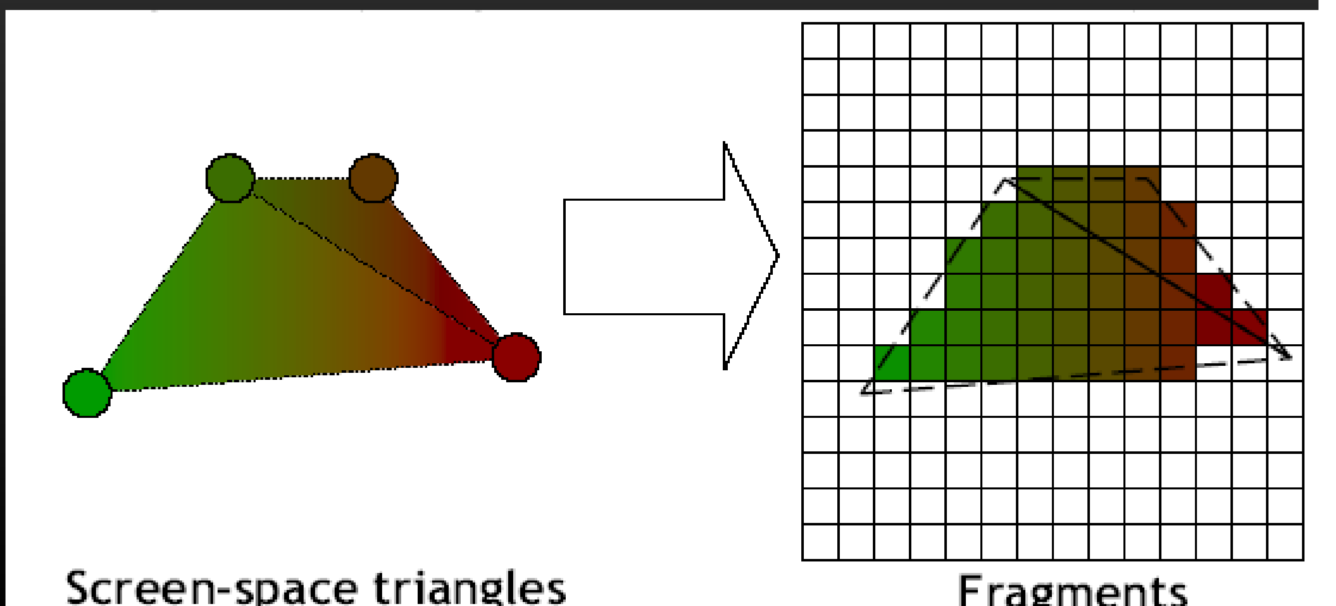
- **Command**
  - (What are commands?)
  - Buffering, parsing.
- **Geometry**
  - Transform, light.
  - Automatic operations.
  - Culling, clipping.



Rasterization

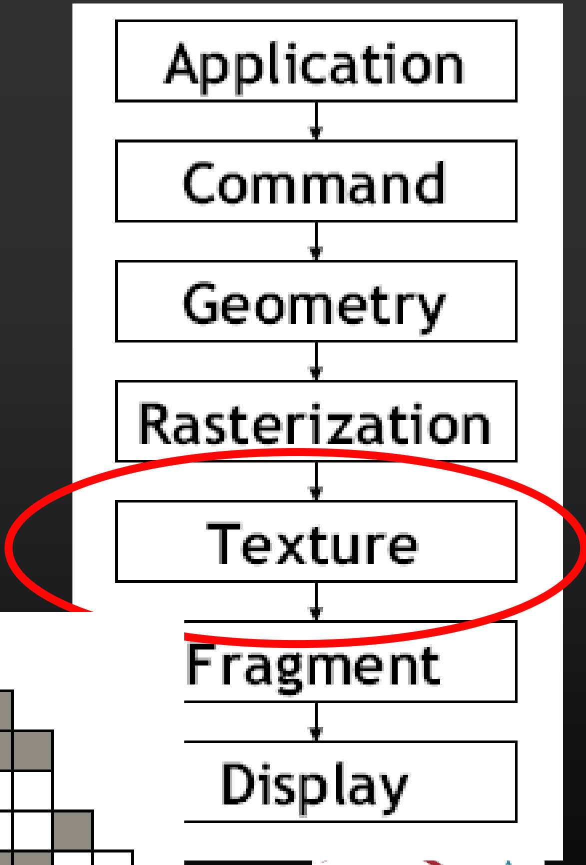
- Setup, sampling (produces "fragments", interpolation (color, depth).

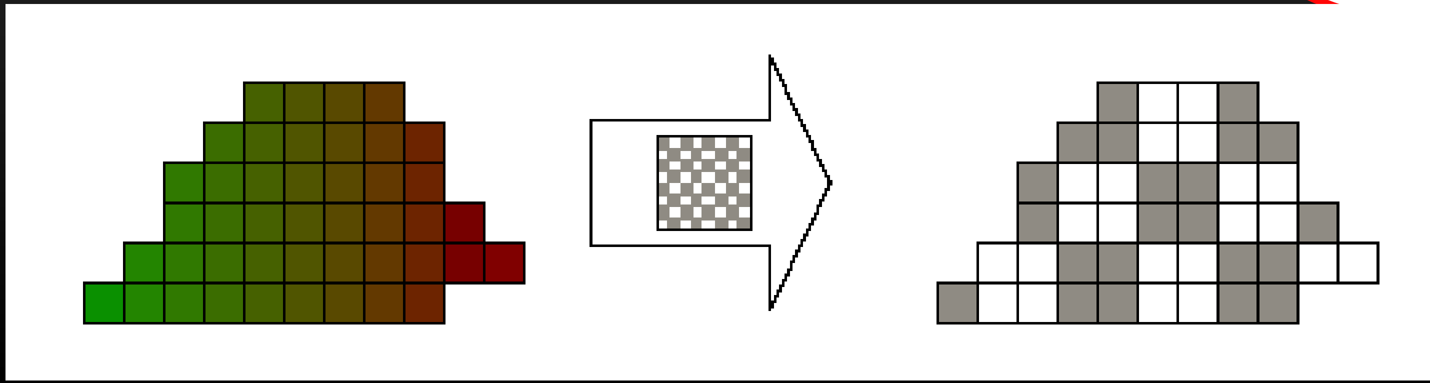




Texturing

- Texture transformation and projection.
- Texture address calculation.
- Texture filtering.





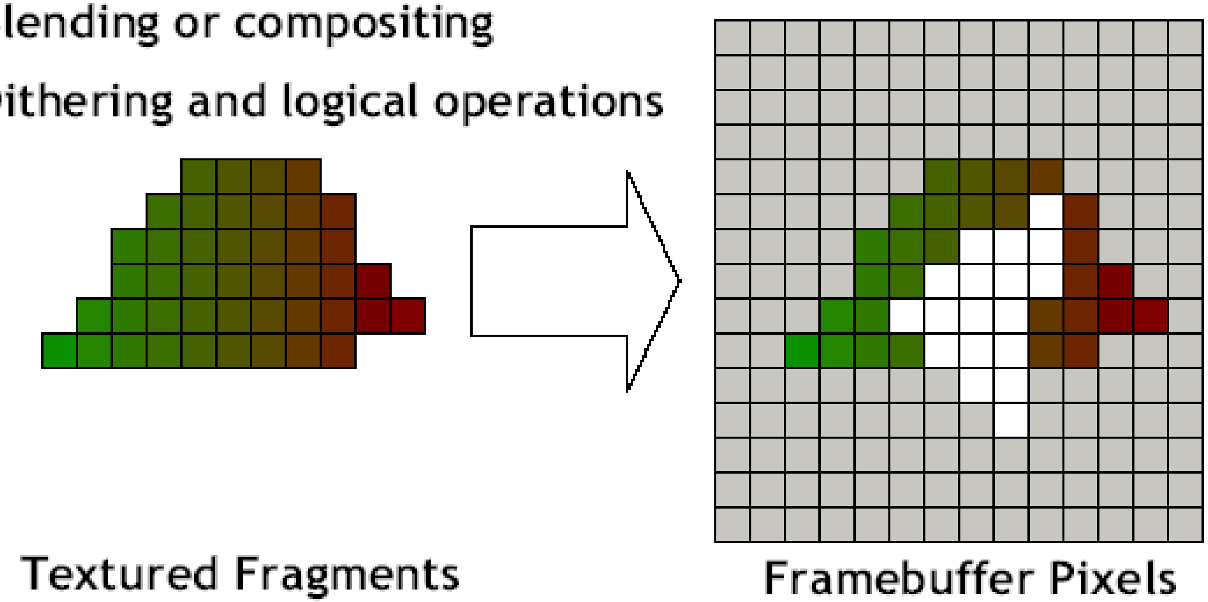
Fragment Operations

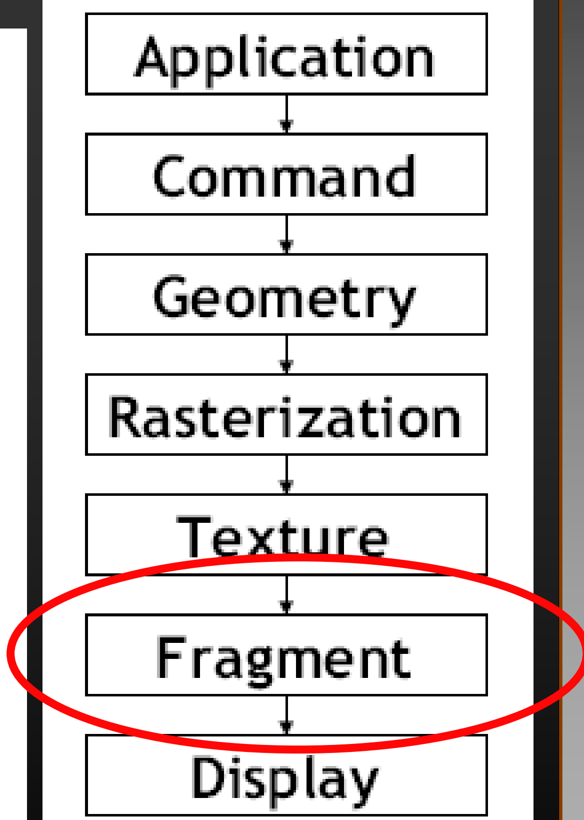
Texture combiners and fog

Owner, scissor, depth, alpha and stencil tests

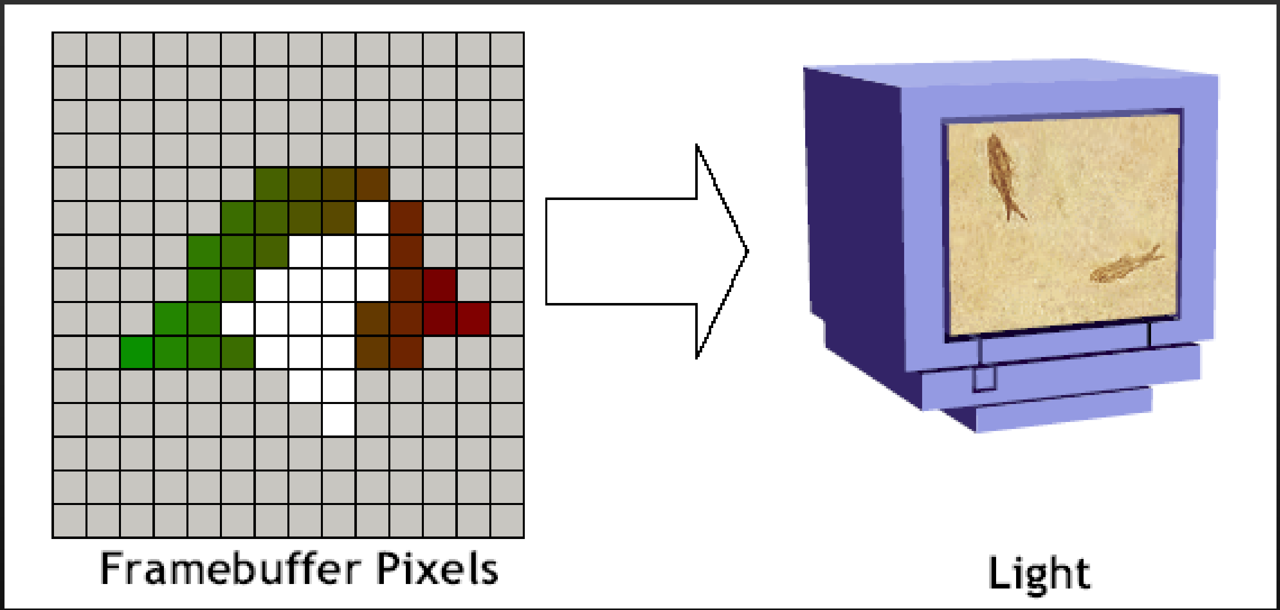
Blending or compositing

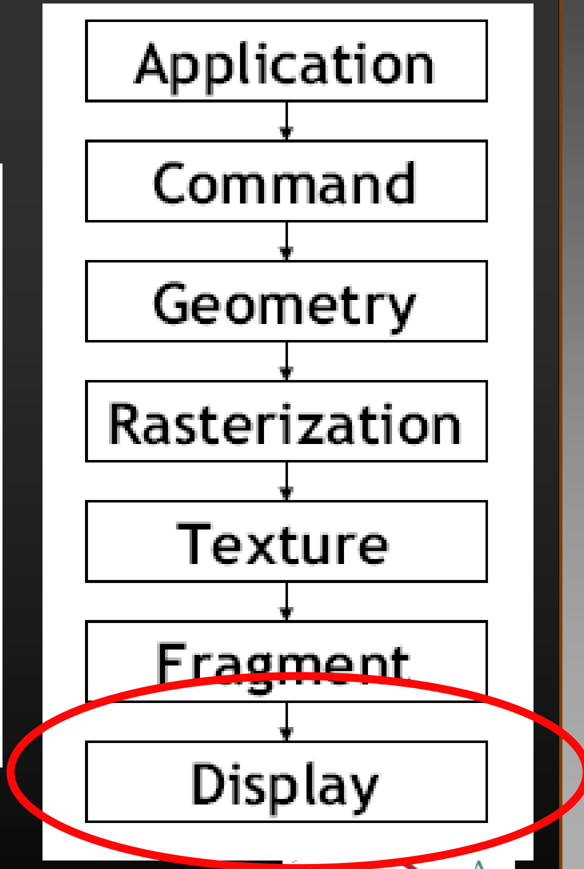
Dithering and logical operations



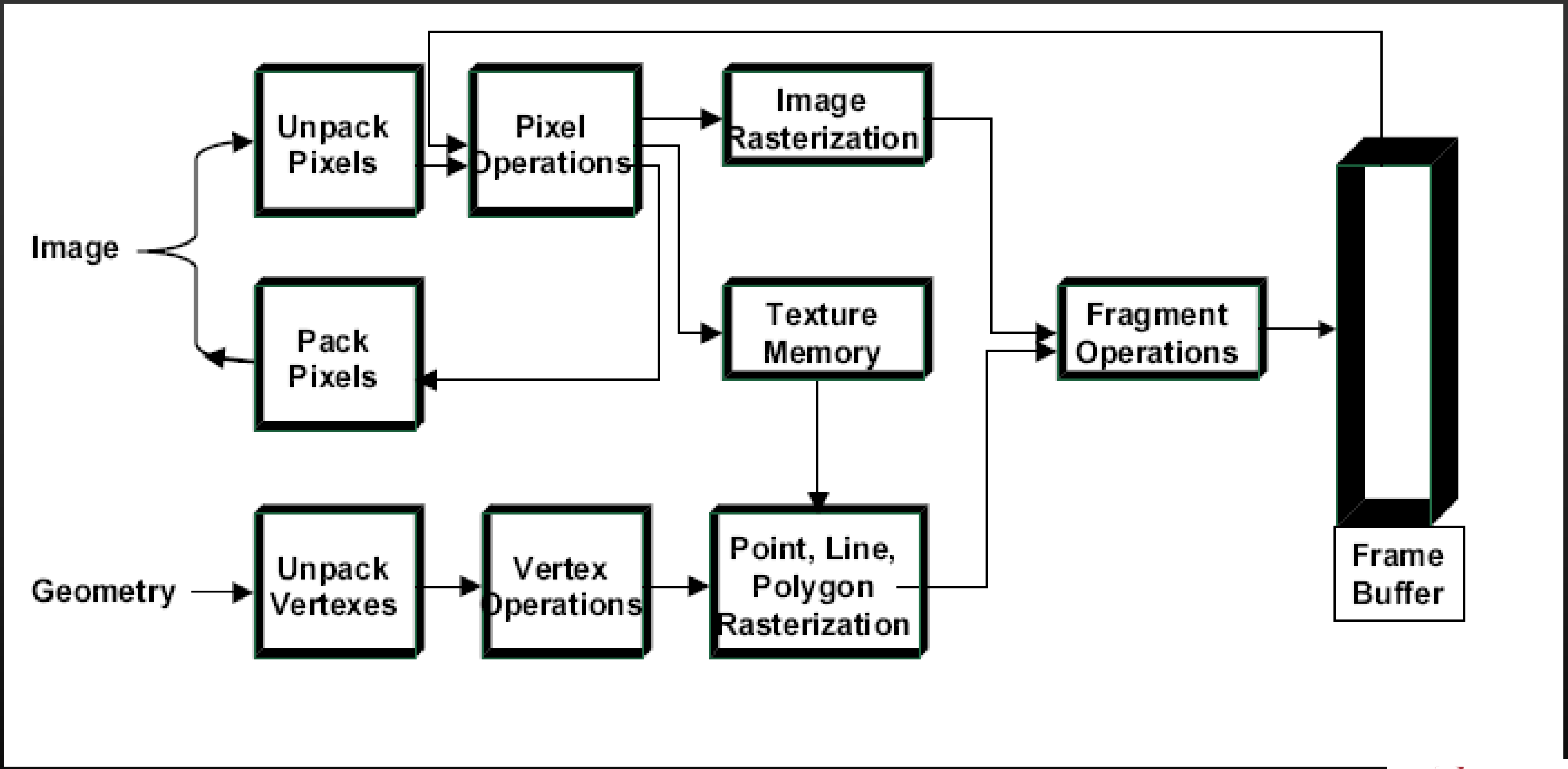


Display

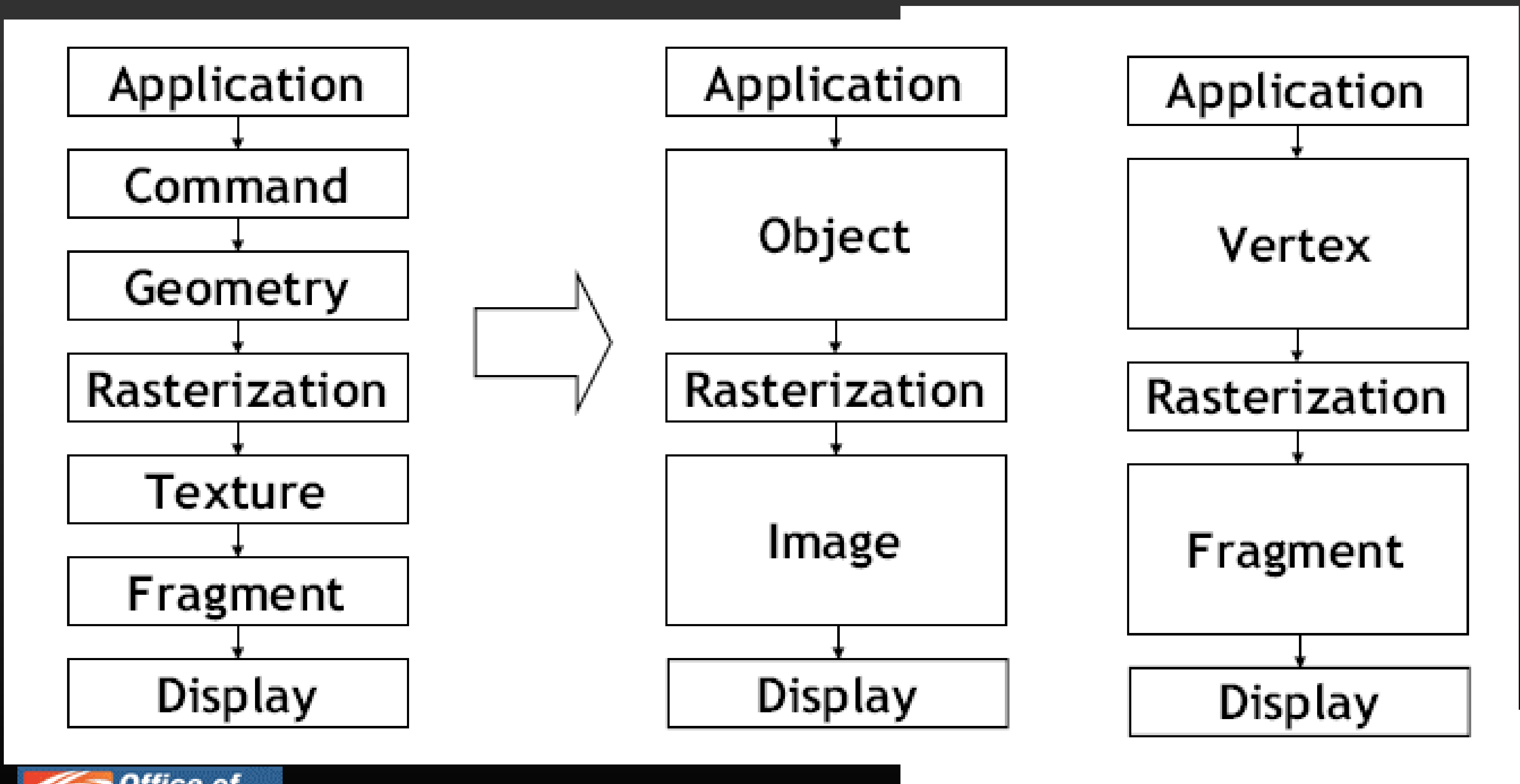

Framebuffer Pixels
Light


Application
Command
Geometry
Rasterization
Texture
Fragment
Display

OpenGL Processing Pipeline



Another Pipeline View



Performance: Frequency of Operations

Geometry processing = per-vertex

- Transformation and Lighting (T & L)
- Floating point; complex operations
- 10 million vertices

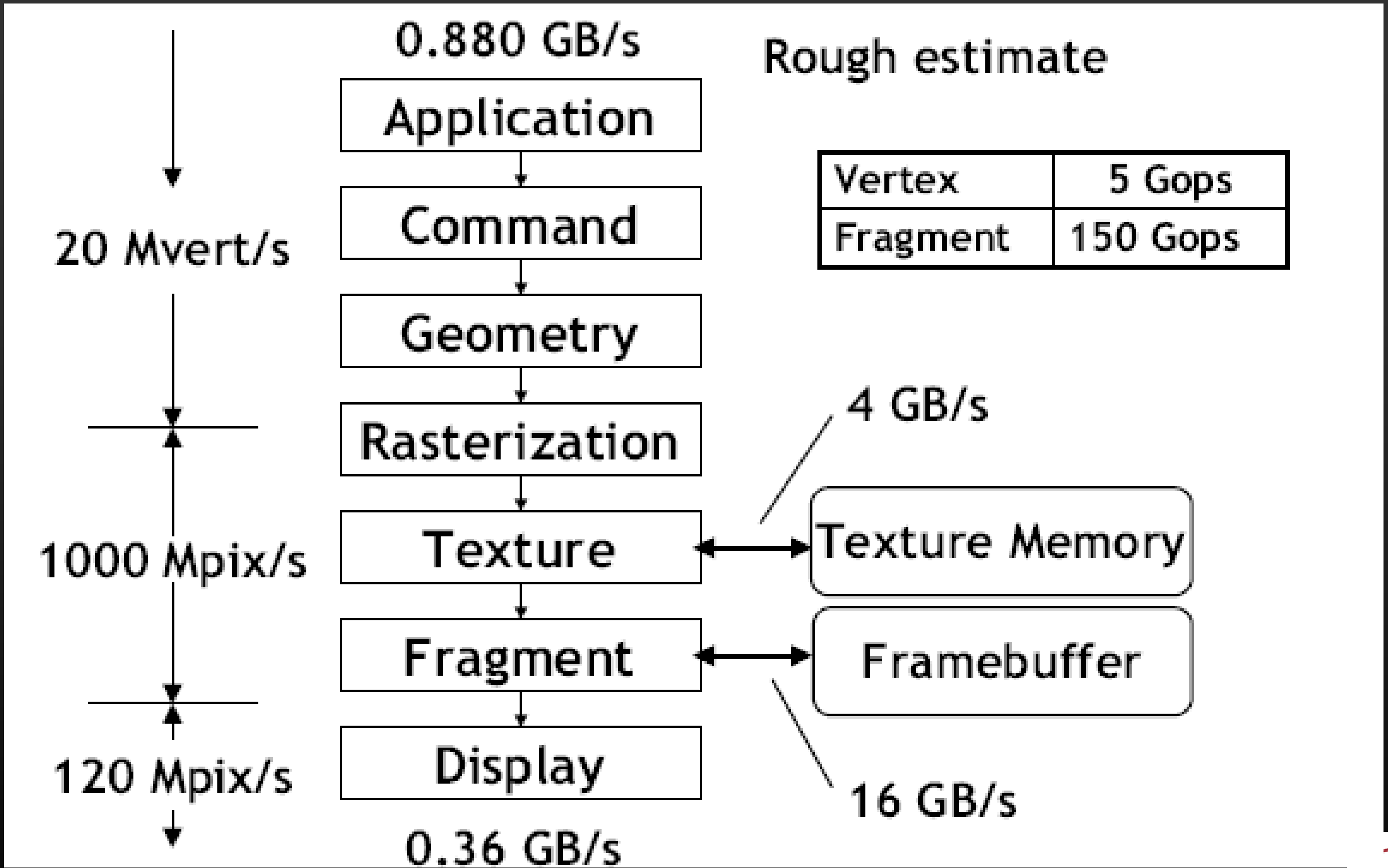
Fragment processing = per-fragment

- Blending and texture combination
- Fixed point; limited operations
- 1 billion fragments

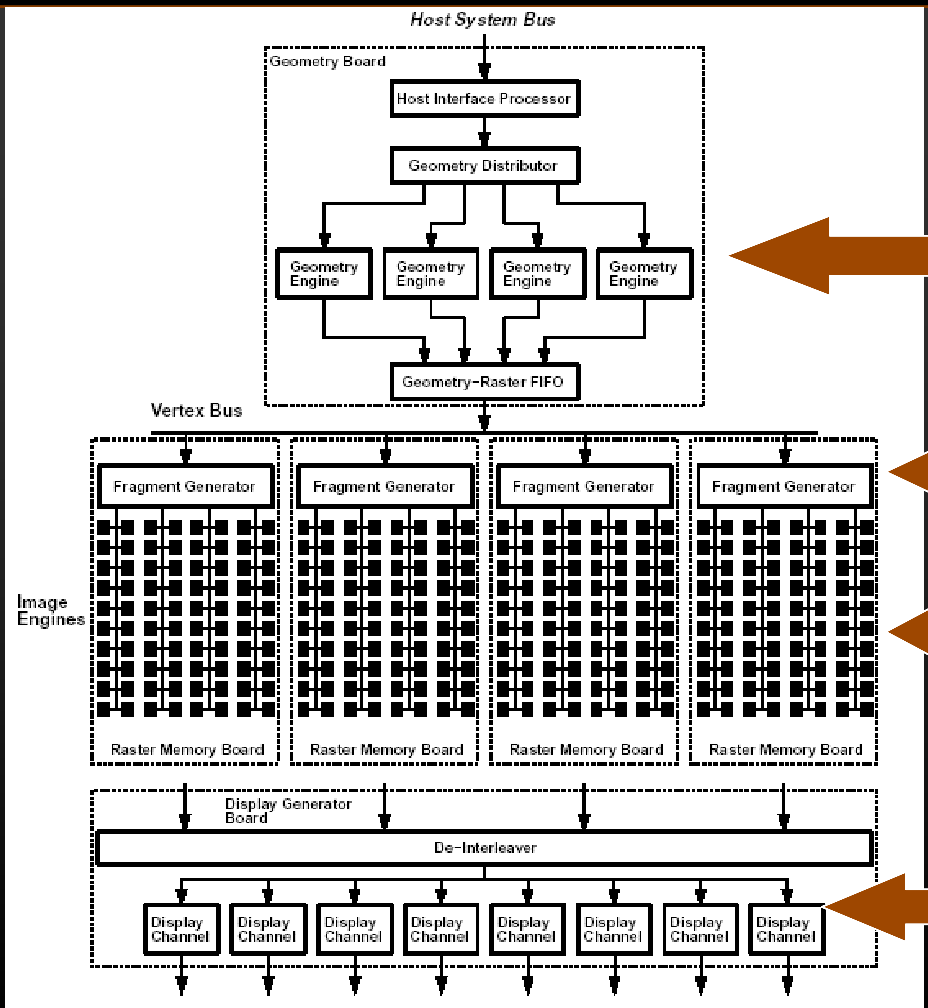
Office of Science
U.S. DEPARTMENT OF ENERGY

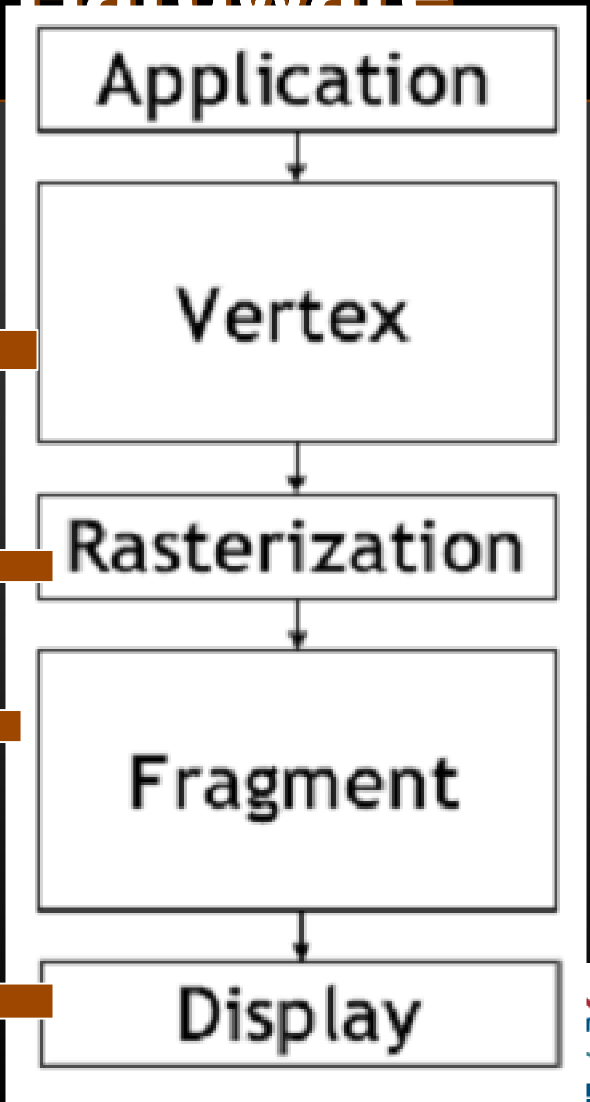
BERKELEY LAB

Processing and Communications



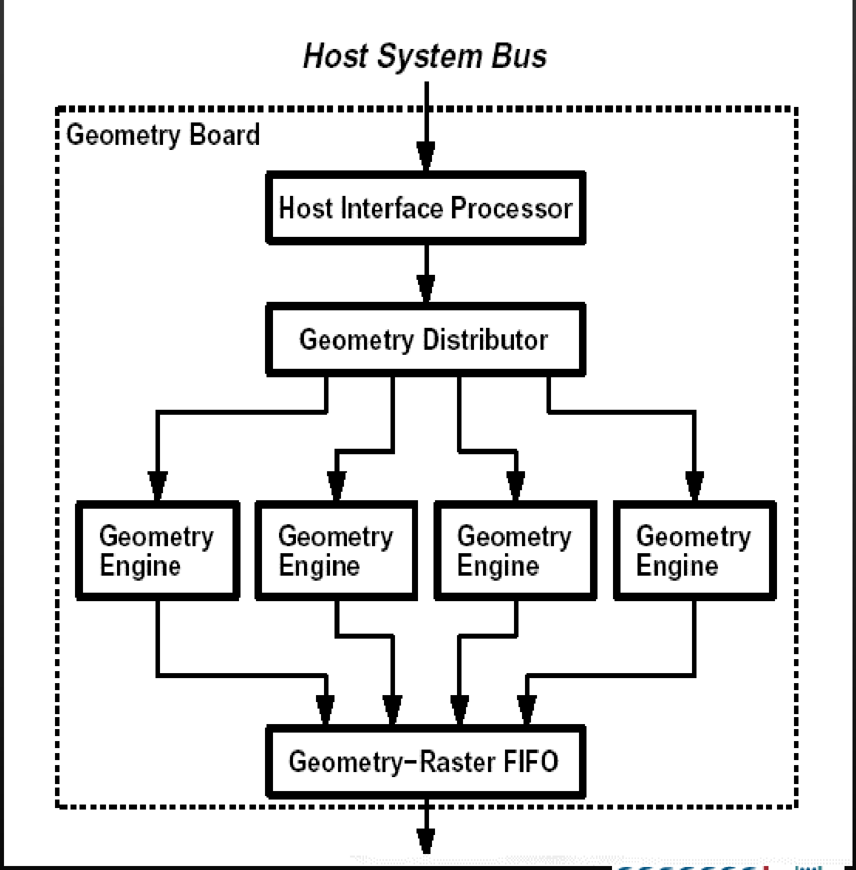
SGI's Reality Engine and InfiniteReality Graphics Hardware





The Geometry Board

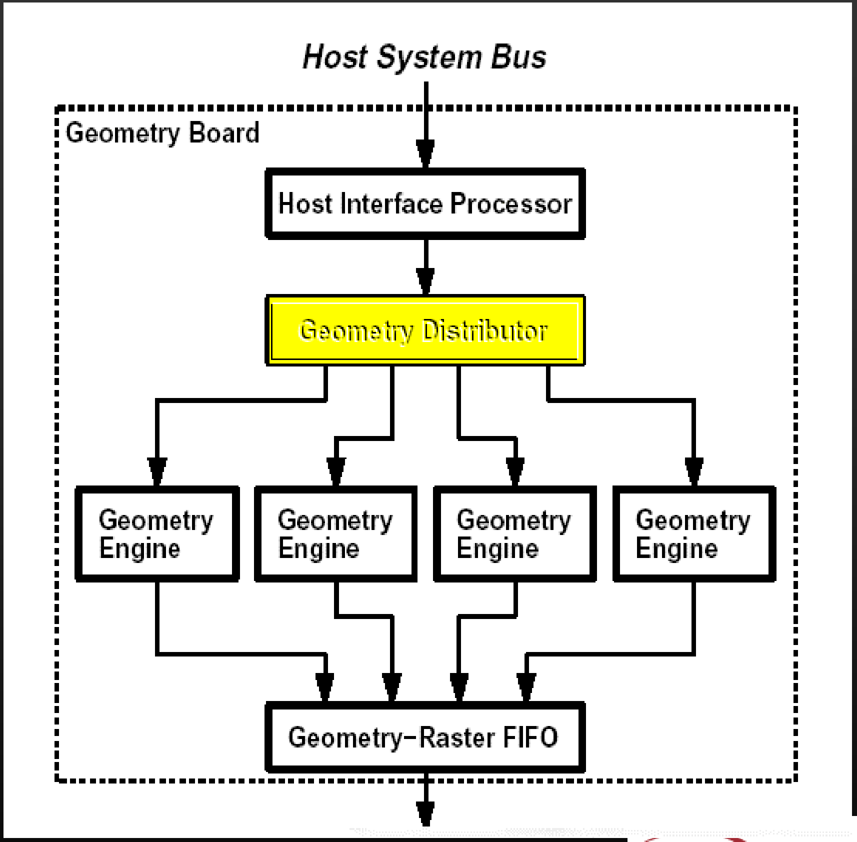
- Host Computer Interface
- Command Interpretation and Geometry Distribution Logic
- Four Geometry Engine processors in a MIMD arrangement.



Geometry Distributor

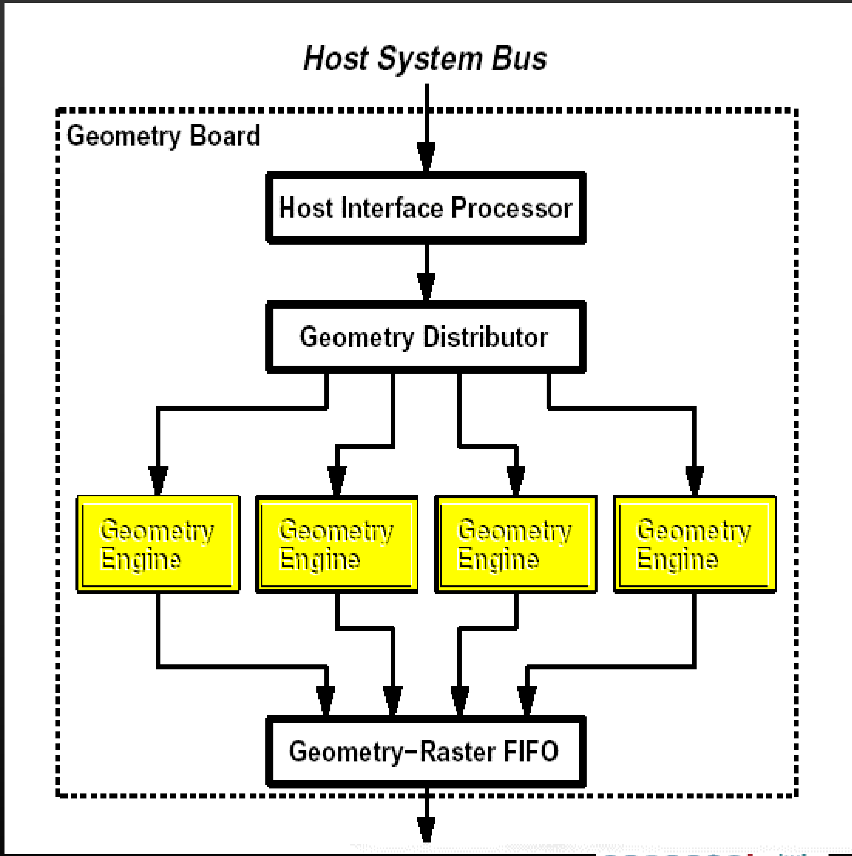
The Geometry Distributor passes incoming data and commands from the Host Interface Processor to individual Geometry Engines for further processing.

Least-busy distribution scheme.



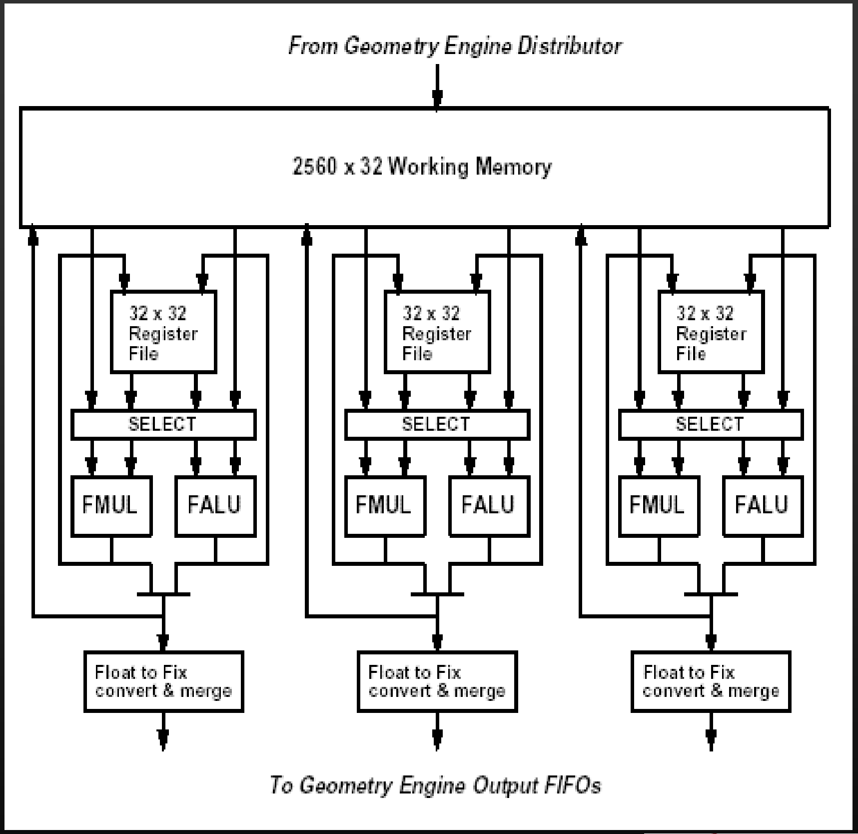
Geometry Engine

- The Geometry Engine is a single instruction multiple datapath (SIMD) arrangement of three floating point cores, each of which comprises an ALU and a multiplier plus a 32 word register.
- A 2560 word on-chip memory holds elements of OpenGL state.



Geometry Engine, ctd.

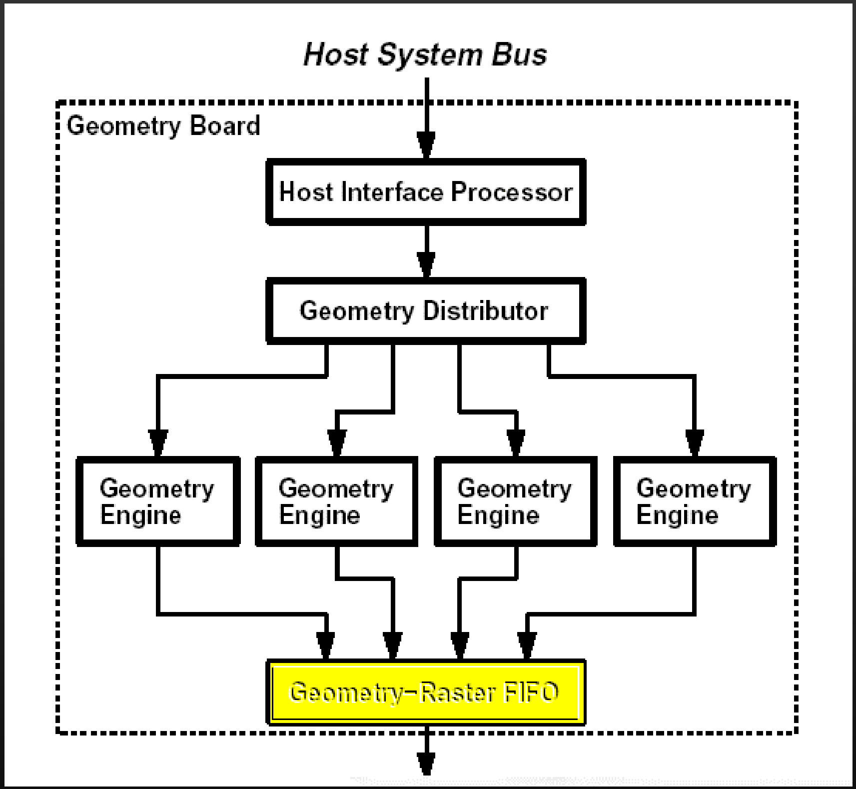
- Each of the three cores can perform two reads and one write per instruction to working memory.
- The working memory allows data to be shared easily among cores.



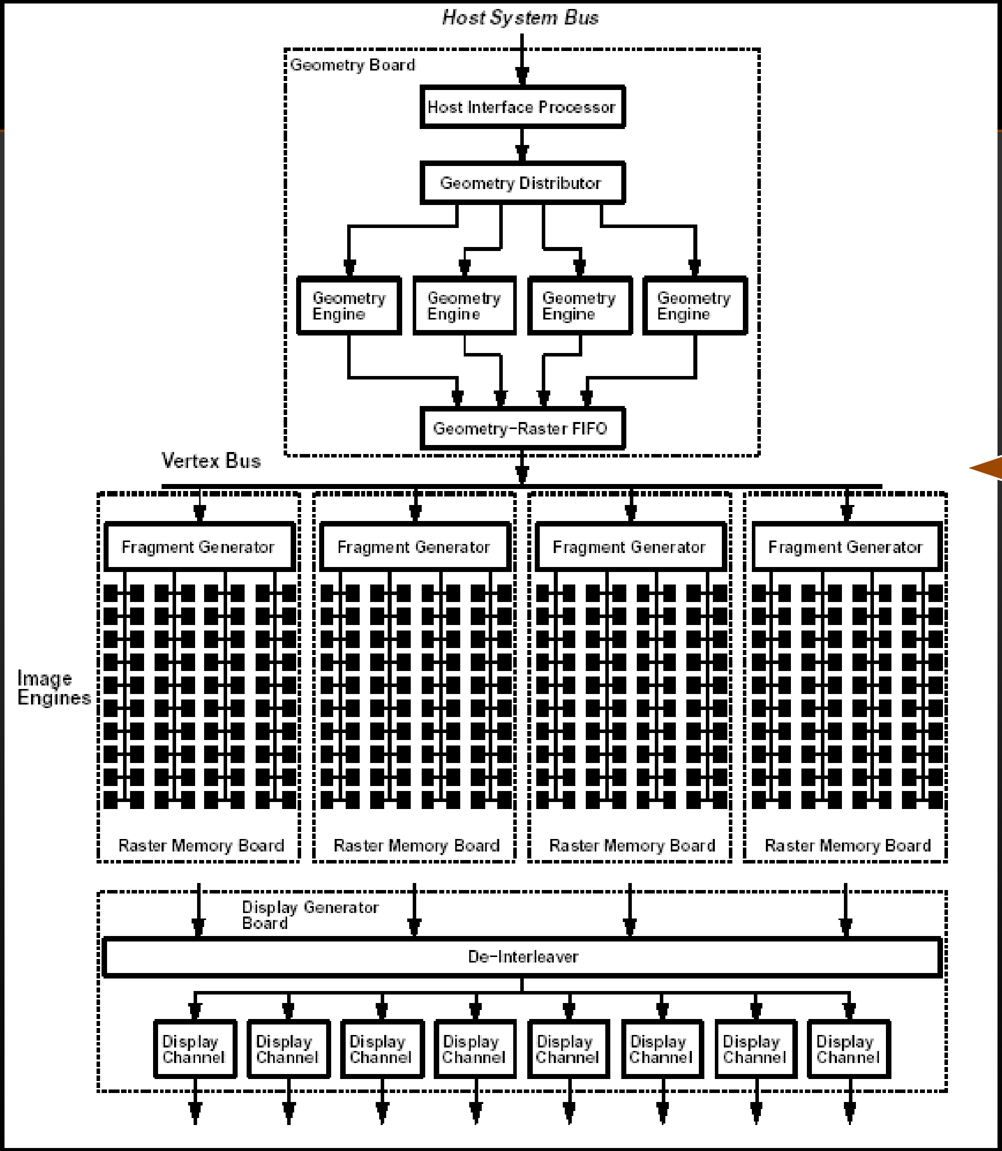
Geometry FIFO

- A FIFO large enough to hold 65536 vertexes is implemented in SDRAM.

- The merged geometry engine output is written, through the SDRAM FIFO, to the Vertex Bus.



Where We Are



Office of Science
U.S. DEPARTMENT OF ENERGY

BERKELEY LAB

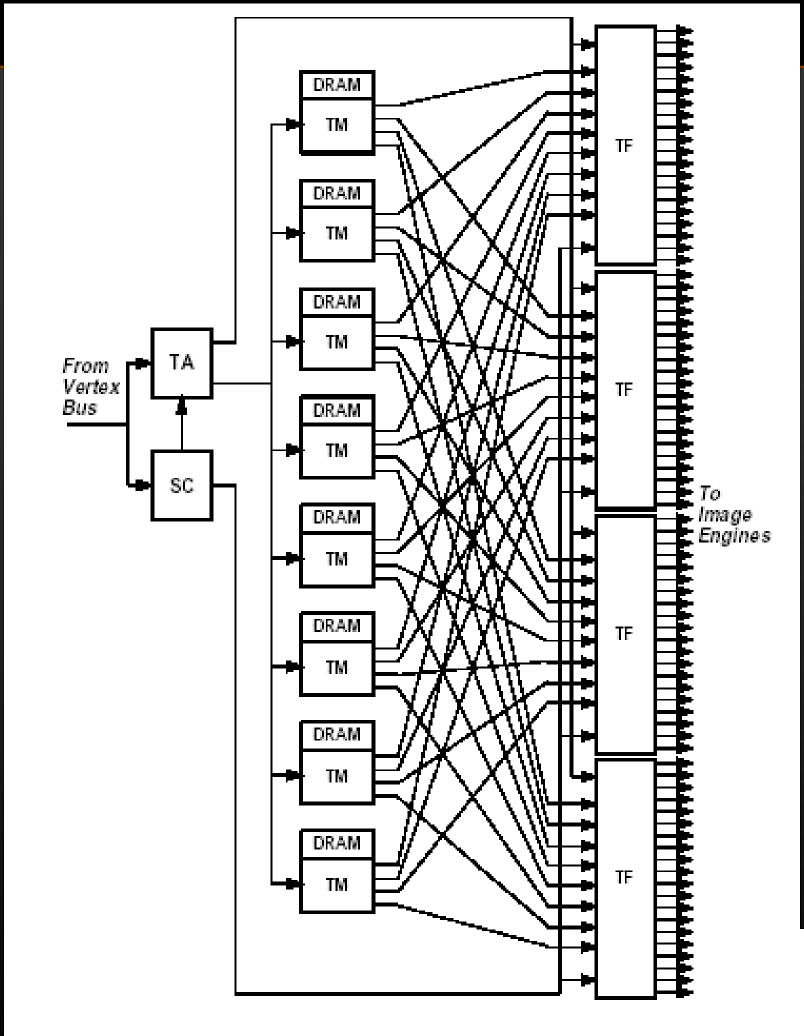
Vertex Bus

- The InfiniteReality system employs a Vertex Bus to transfer only screen space vertex information.

- Supports the OpenGL triangle strip and triangle fan constructs, so the Vertex Bus load corresponds closely to the load on the host-to-graphics bus.

A Fragment Generator

- The Scan Converter (SC) and Texel Address Calculator (TA) perform scan conversion, color and depth interpolation, perspective correct texture coordinate interpolation and LOD computation.



Office of Science
U.S. DEPARTMENT OF ENERGY

BERKELEY LAB

Image Engines

- Fragments output by a single Fragment Generator are distributed equally among the 80 Image Engines owned by that generator.
- Each Image Engine controls a single 256K x 32 SDRAM that comprises its portion of the framebuffer.

Display Hardware

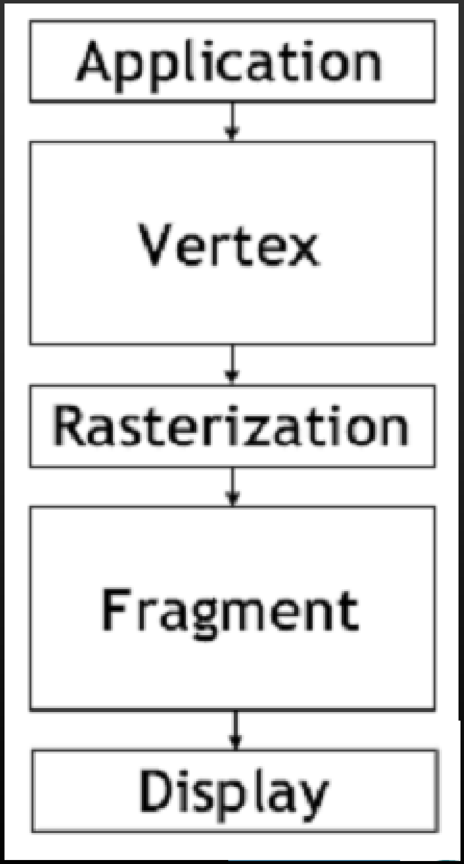
- Each of the 80 Image Engines on the Raster Memory boards drives one or two bit serial signals to the Display Generator board.

- The base display system consists of two channels, expendable to eight.

Parallelism in Graphics

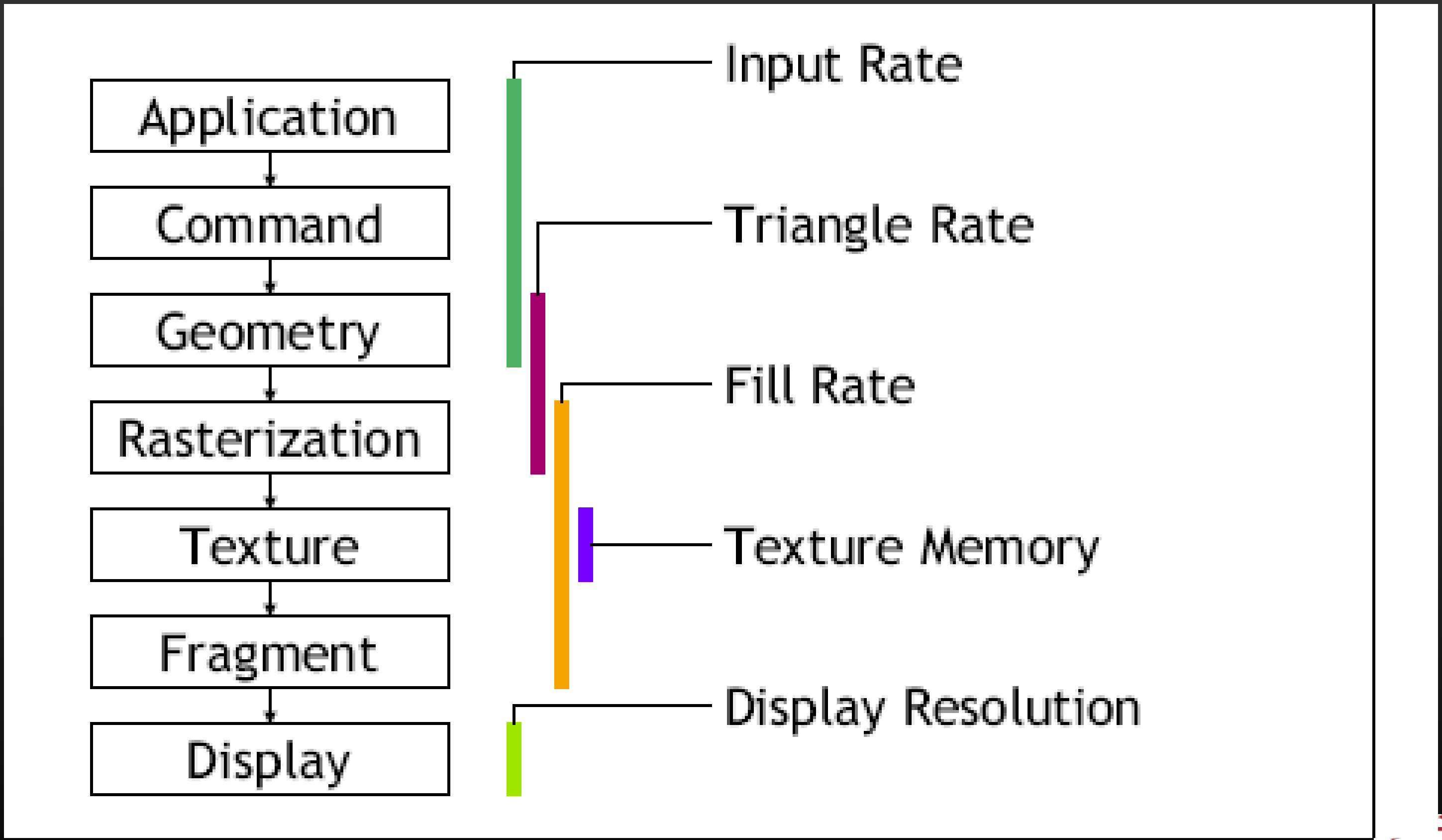
Parallelism in Graphics

- **Parallelism**
  - Design a single component (either a single stage or a complete graphics pipeline) and replicate it to increase performance.
  - Pipelining (work overlap).
- **Communication**
  - Connects components, allowing parallel work to be load balanced.
- **Considerations: dependencies and ordering.**



Office of Science
U.S. DEPARTMENT OF ENERGY

Scaling Performance Areas

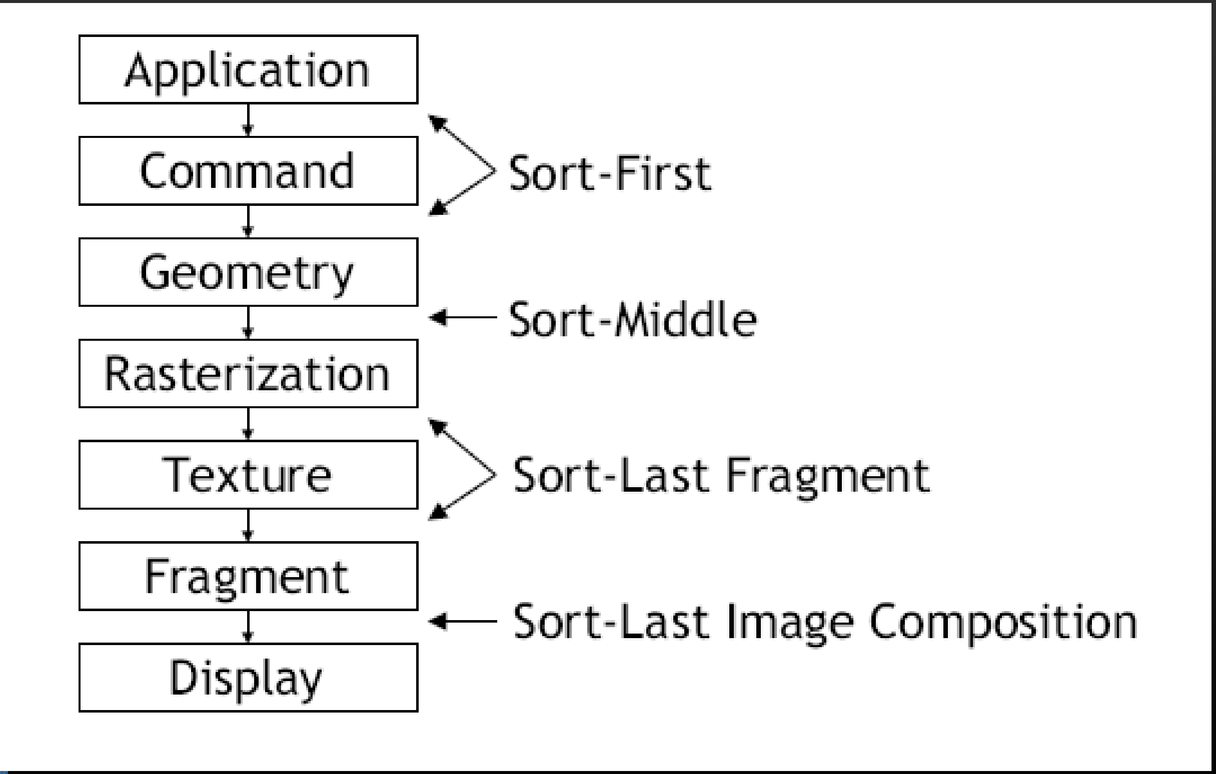


Sources of Parallelism

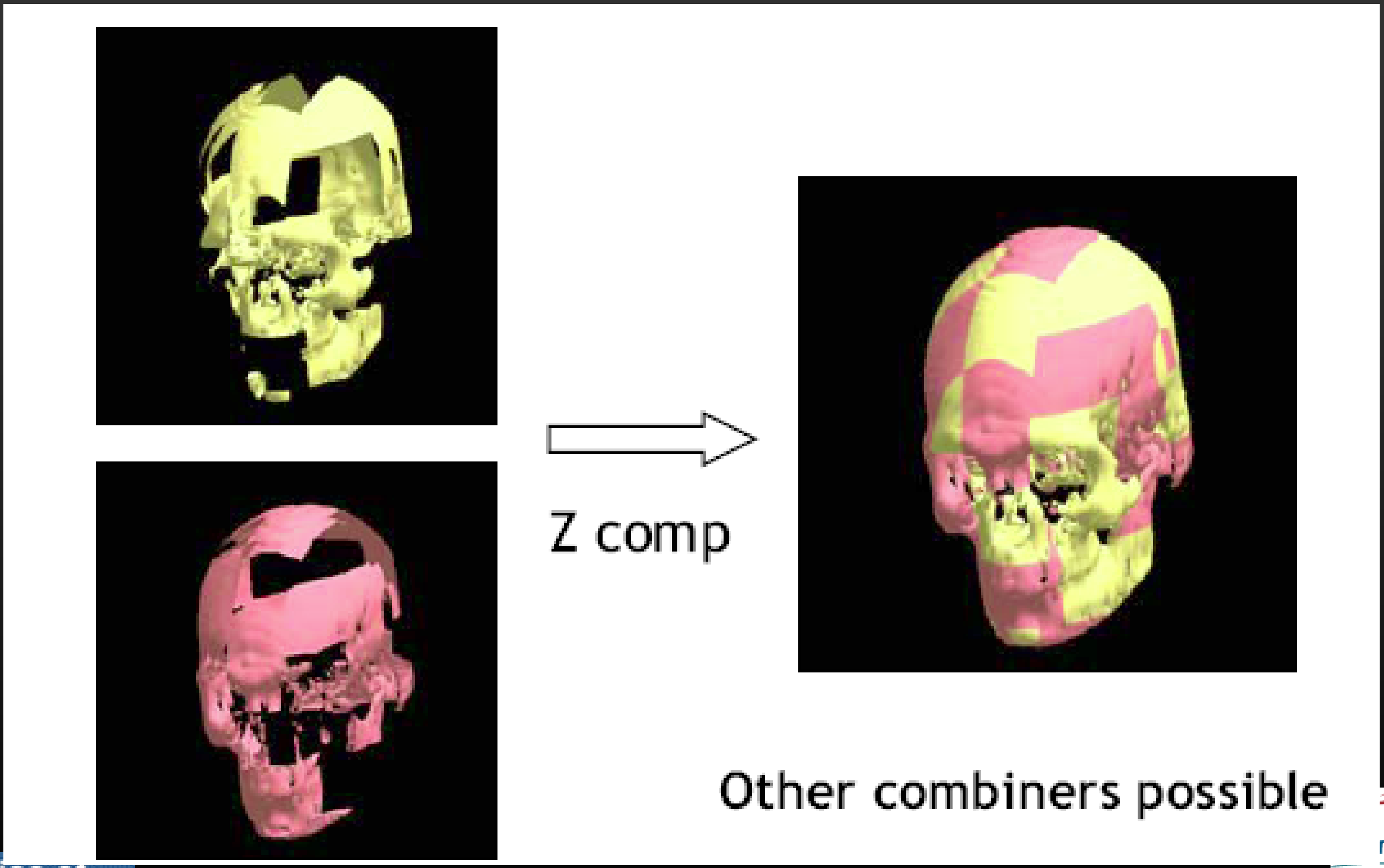
- **Task parallelism**
  - Graphics Pipeline
- **Data Parallelism**
  - Frame & image parallel.
  - Object (geometry) parallel.

Sorting Taxonomy

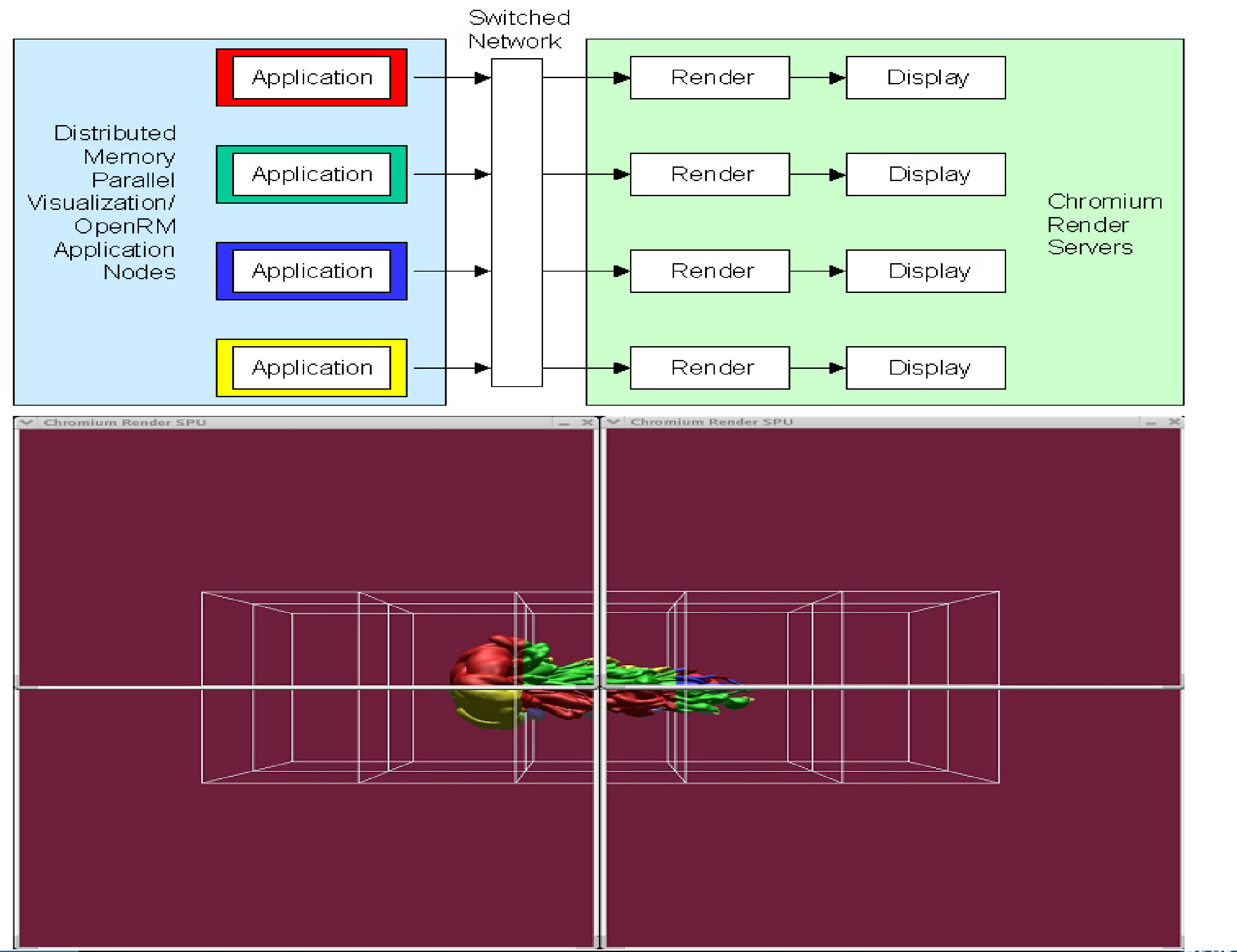
Further reading: S. Molnar, M. Cox, D. Ellsworth, H. Fuchs, A sorting classification of parallel rendering.



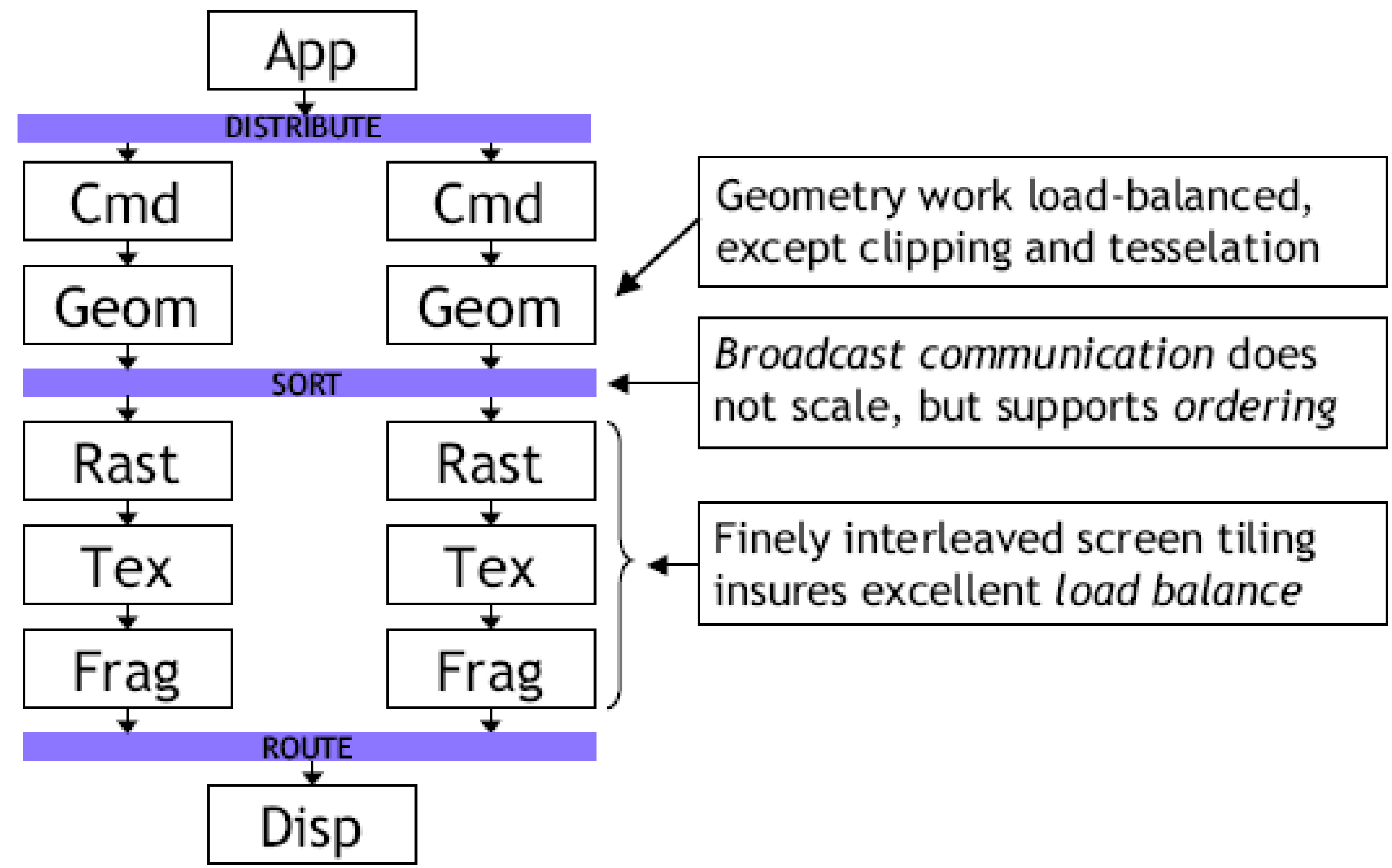
Sort Last – Pixels/Images



Sort First – Geometry

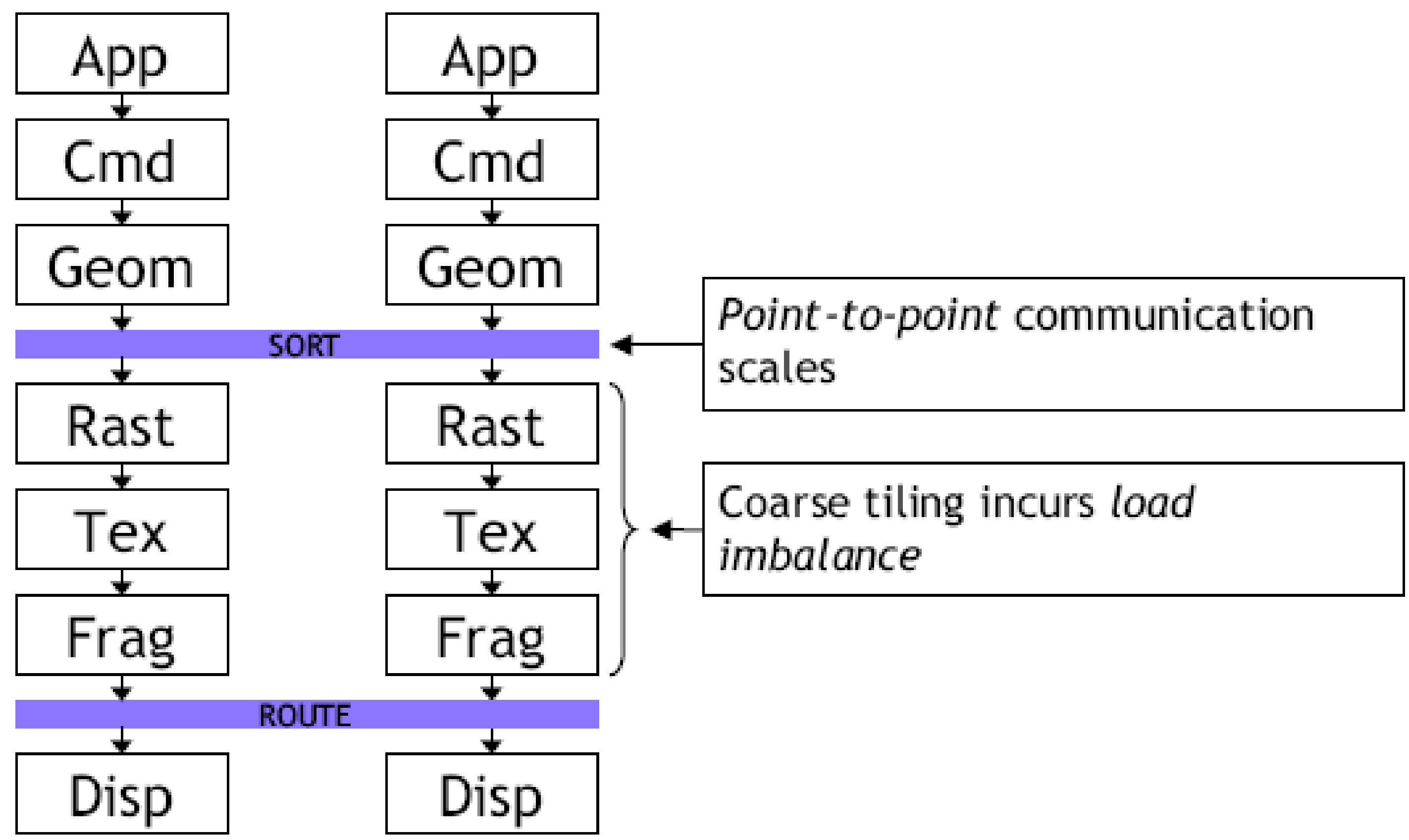


Sort Middle – Broadcast



SGI Graphics Workstations: RealityEngine, InfiniteReality

Sort Middle – Point-to-Point



UNC PixelPlanes, Stanford Argus

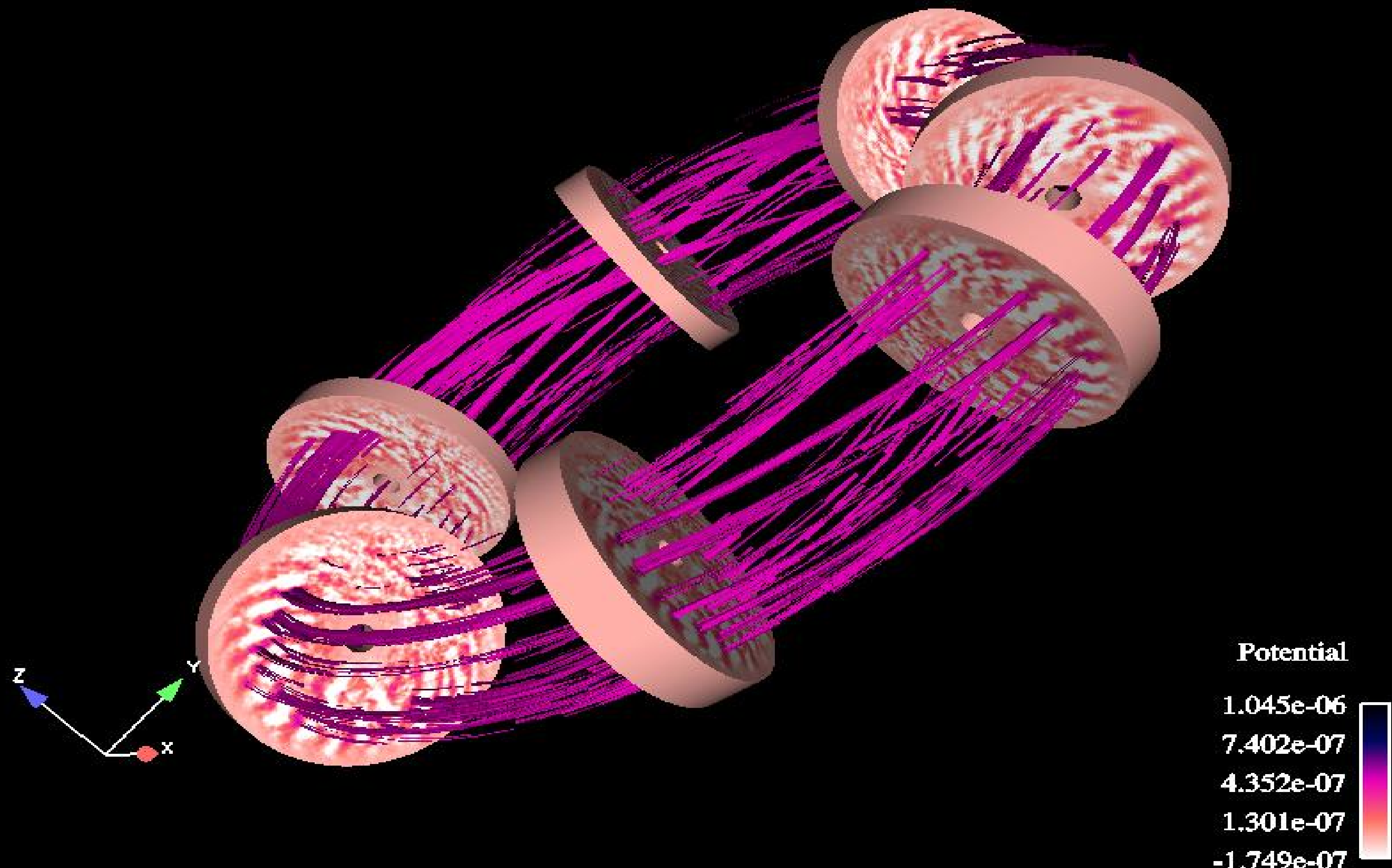
Parallelism in Graphics: Observations

- Task & Data parallelism implemented in silicon in InfiniteReality and modern GPUs.
- Parallelism not an intrinsic part of graphics APIs.
- Chromium: Parallel, stream-based OpenGL "replacement." See chromium.sf.net.
  - Parallel synchronization constructs implemented as OpenGL "extensions."
  - Supports sort-first, sort-last.

Office of Science
U.S. DEPARTMENT OF ENERGY

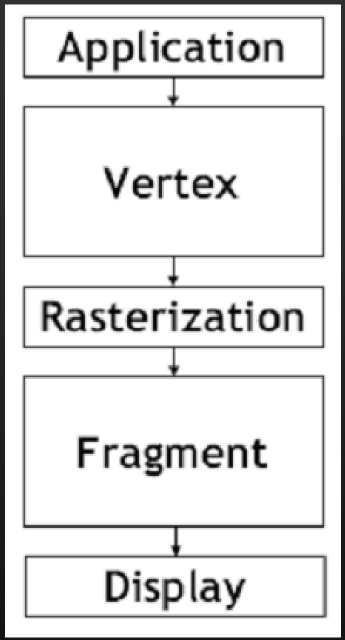
BERKELEY LAB

Visualization



Visualization

- **Visualization is the art/science of transforming abstract data into images.**
- **Visualization algorithms:**
  - Produce data that can be then processed by a traditional graphics pipeline, or
  - Are rendering algorithms that produce images.
  - Are highly data intensive (opportunity for performance gain through parallelism!!)



Parallel Visualization Issues

- **What, exactly, is being optimized?**
  - Raw data access, manipulation or movement.
  - Visualization task performance?
  - Rendering performance?
- **Architecturally, much overlap with parallel graphics algorithms.**

Parallel Visualization Algorithms

- **Data Parallel.**
  - Divide data amongst PEs.
- **Image Parallel.**
  - Divide work to correspond to screen space.
- **Hybrid Approaches.**

Data Parallel, Serial Rendering

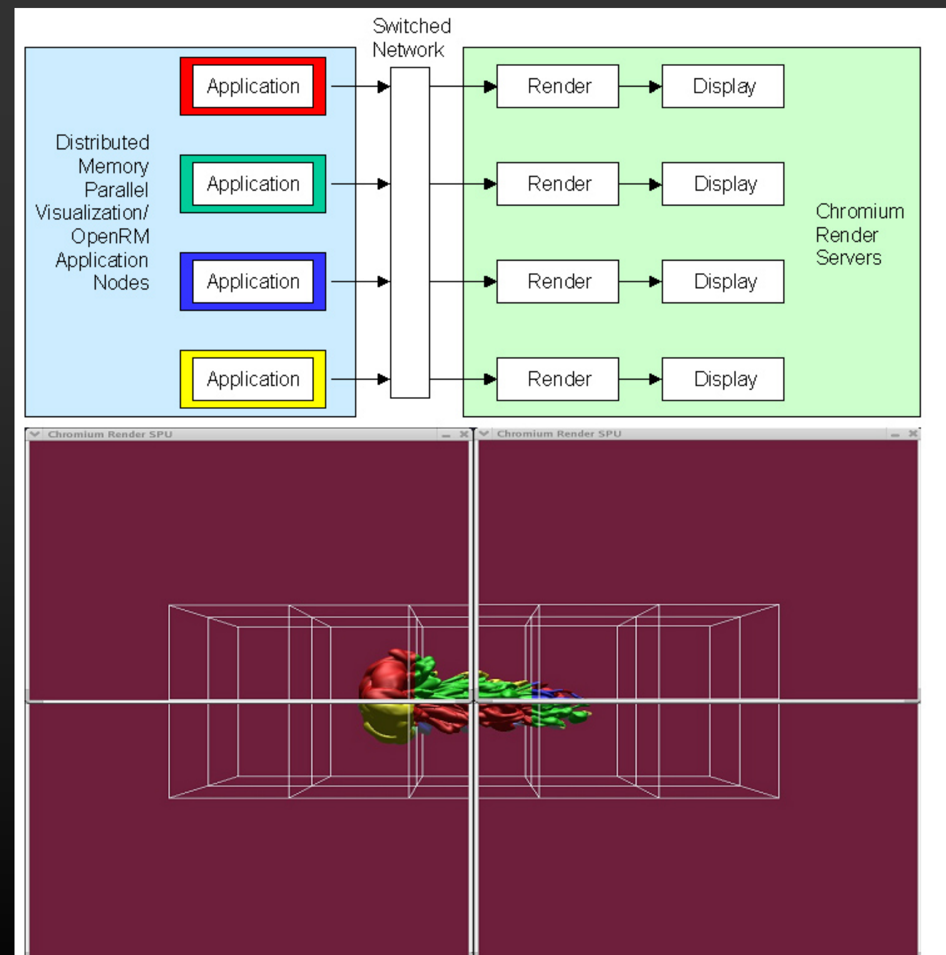
- **Data distributed amongst PEs (scatter), then rendered on a single host (gather).**
- **Considerations:**
  - Visualization load balance – evenly distributing the data doesn't necessarily result in equal work. (E.g., isosurface).
  - Cost of data processing (vis) often outweighs any concerns about algorithm load imbalance or rendering costs.

Office of Science
U.S. DEPARTMENT OF ENERGY

BERKELEY LAB

Data Parallel, Sort-First Parallel

- Data divided evenly amongst PEs.
- Resulting geometry routed to rendering PE as a function of tile coordinates.

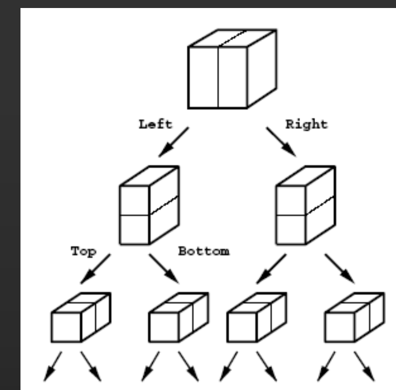


Office of Science
U.S. DEPARTMENT OF ENERGY

BERKELEY LAB

Data Parallel, Sort-Last Parallel

- Data distributed using k-d partitioning.
- Ray-casting volume renderer produces image of data subset on each PE.
- Images from each PE combined using binary swap compositing.
- Binary swap is a specialized form of a reduction operator, but all PEs participate at each stage of the reduction.



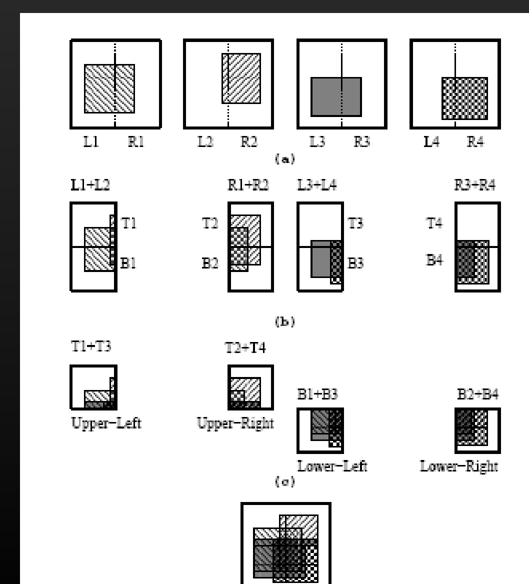


Image Parallel

- **Partition work (data) as a function of screen-space projection.**
- **Considerations**
  - Cost of moving data during interactive transformation.
  - Cost of combining image tiles.
- **(Ray tracing techniques)**

Parallel Visualization

- **Sort-last.**
  - Predictable communications costs, but performance dominated by number of pixels in final display.
- **Sort-first.**
  - Scales well with increasing data size, but communication costs not easily predictable.
- **Sort-middle.**
  - Not commonly used – high intermediate bandwidth not well supported on modern architectures.

Remote Visualization

- **Sort-first: send geometry to remote desktop.**
  - Offers possibility of retained-mode frame rates on remote desktop, requires one-time performance “hit” (for static scenes). Good approach to hide network latency.
- **Sort-last: send images to remote desktop.**
  - Most flexible solution, but no chance of hiding latency or network performance from remote user.
  - People have grown accustomed to 60fps, and don’t readily accept lesser performance.

Office of Science
U.S. DEPARTMENT OF ENERGY

BERKELEY LAB

Summary

- Computer graphics and visualization are problem-rich environments for parallelization.
- There are many different types of parallelization possible: data parallel, image parallel, pipelining.
- Implementations of parallelism exist in both hardware and software.
- We've just scratched the surface in this presentation.

Office of Science
U.S. DEPARTMENT OF ENERGY
BERKELEY LAB

Acknowledgement

- **Material in these slides was gratuitously borrowed from other sources. These include:**
  - Pat Hanrahan, Stanford. http://graphics.stanford.edu/courses/cs448a-01-fall/
  - John van Rosendale, William & Mary.
  - Silicon Graphics Computer Systems.

The End