GPU Architectures for Global Illumination and Beyond

Donald S. Fussell

Department of Computer Science

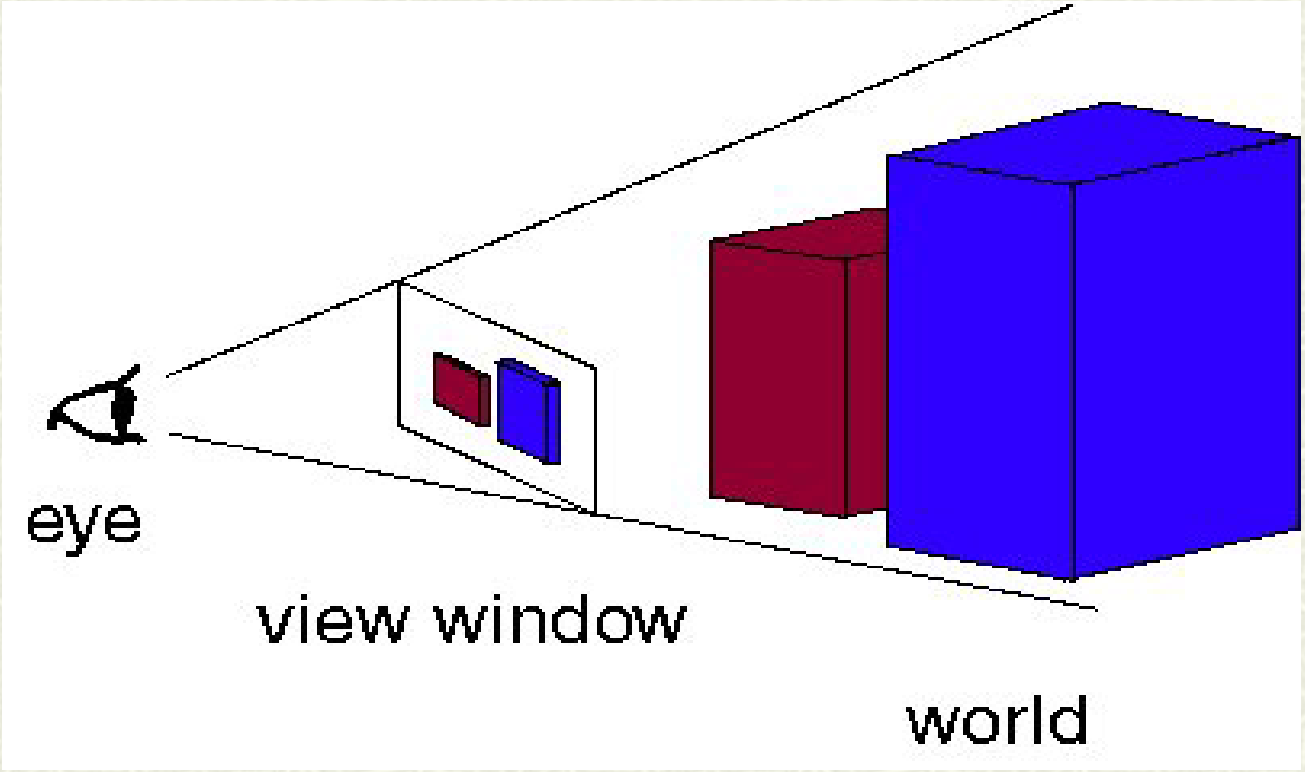
The University of Texas at Austin

UT Graphics Architecture Team

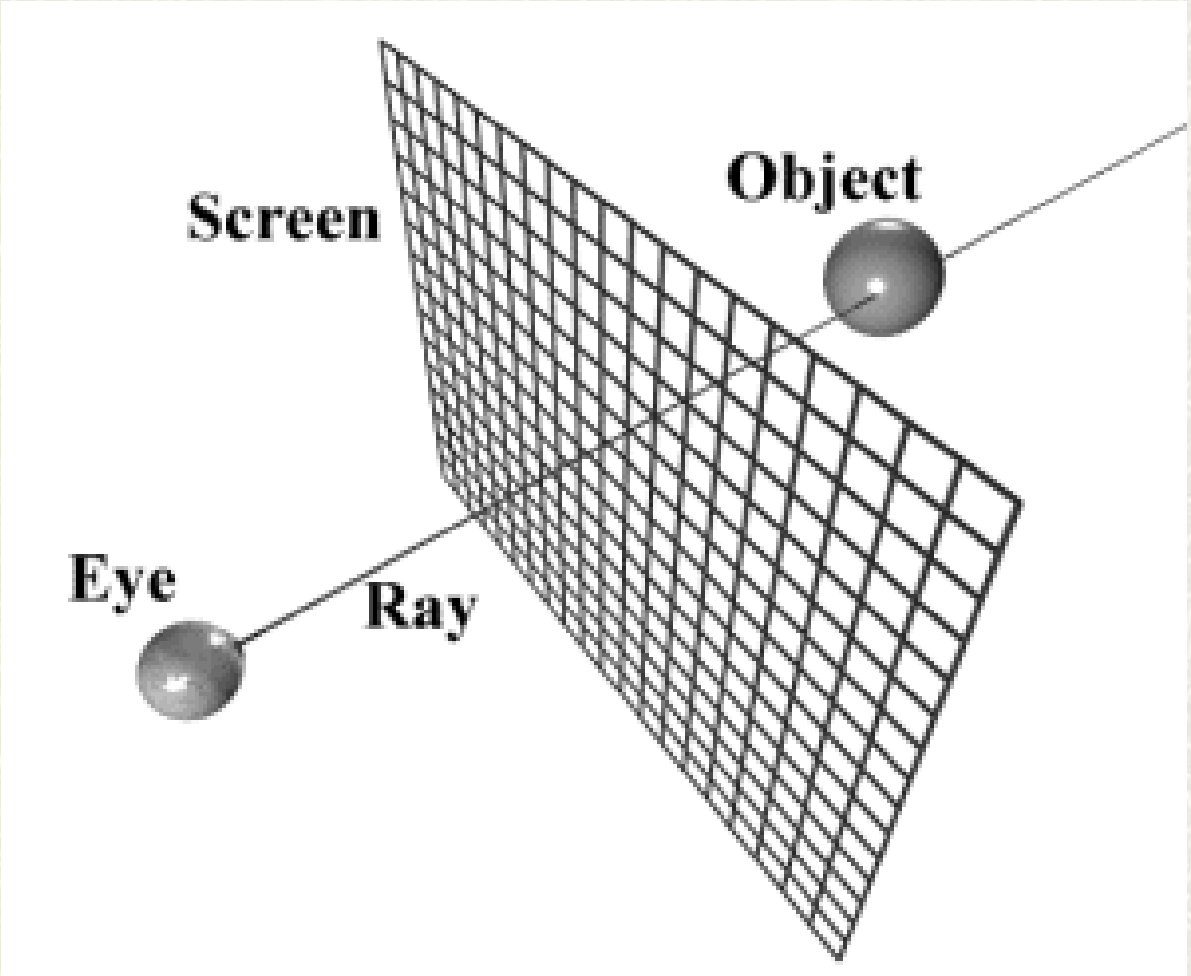
- Faculty
  - Don Fussell
  - Bill Mark
- Students
  - Chris Burns
  - Ikrima Elhassan
  - Greg Johnson
  - Juhyun Lee
  - Chris Lundberg
  - Paul Navratil
  - Chendi Zhang

Function of a GPU

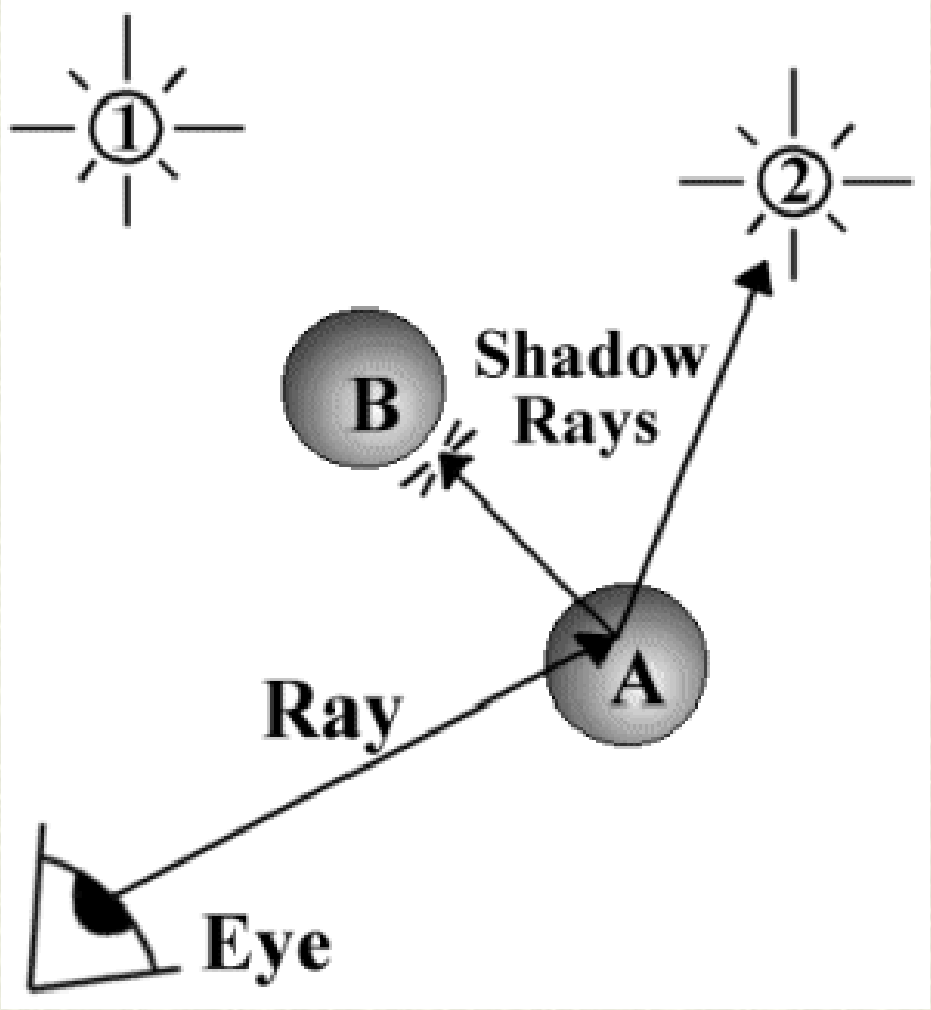
- Synthesize realistic 3-d images in real time



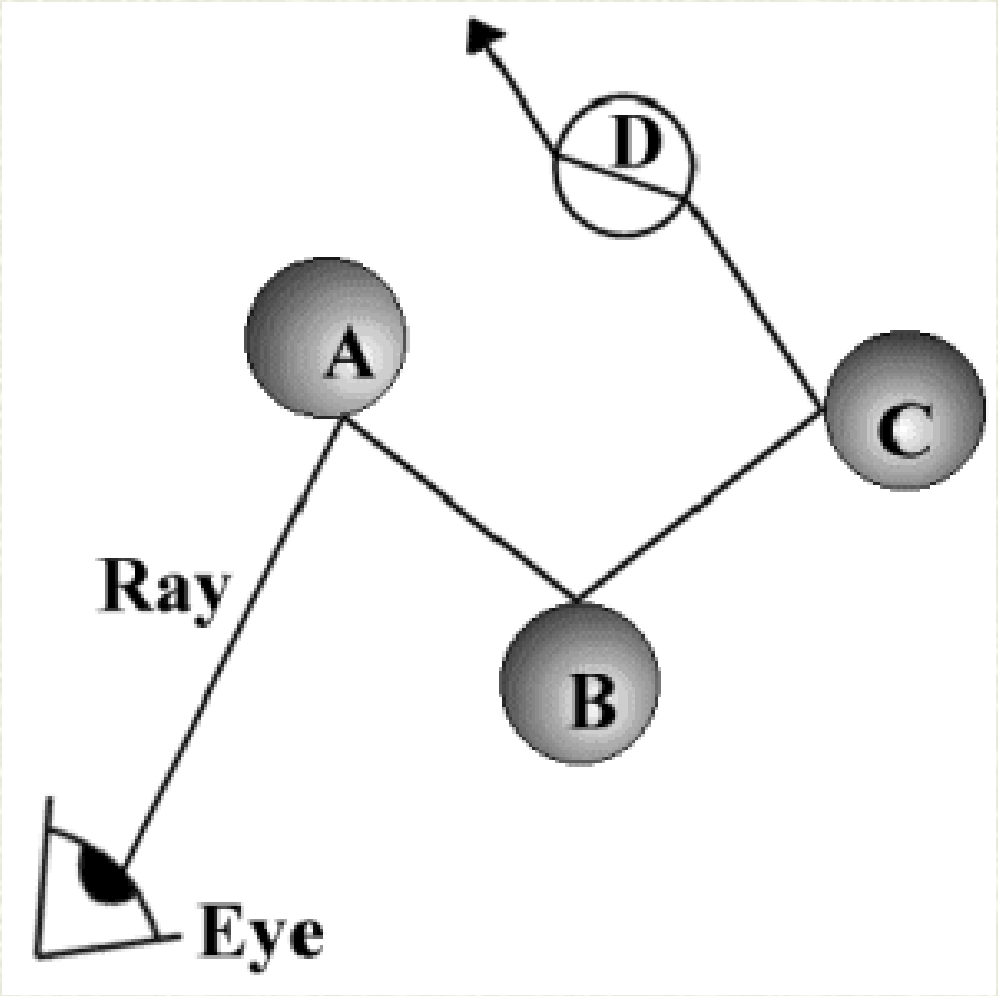
3D Rendering



Ray Tracing

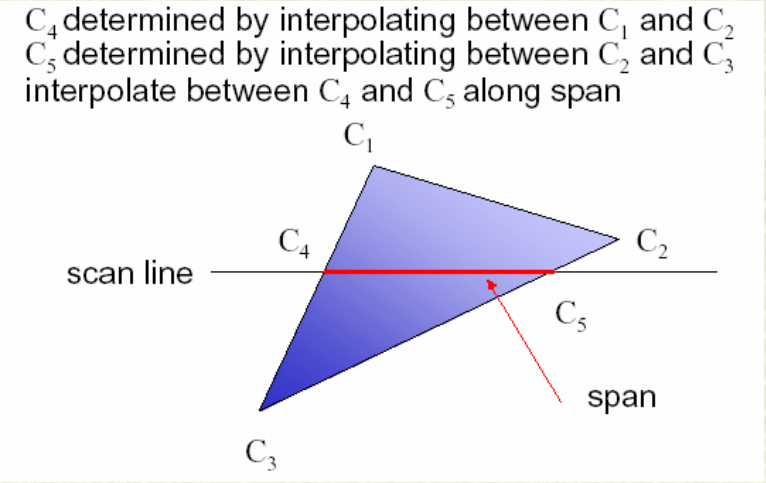


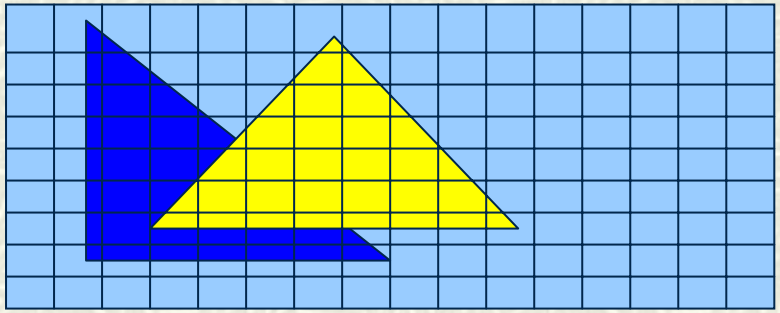
Ray Tracing



Fragment processing

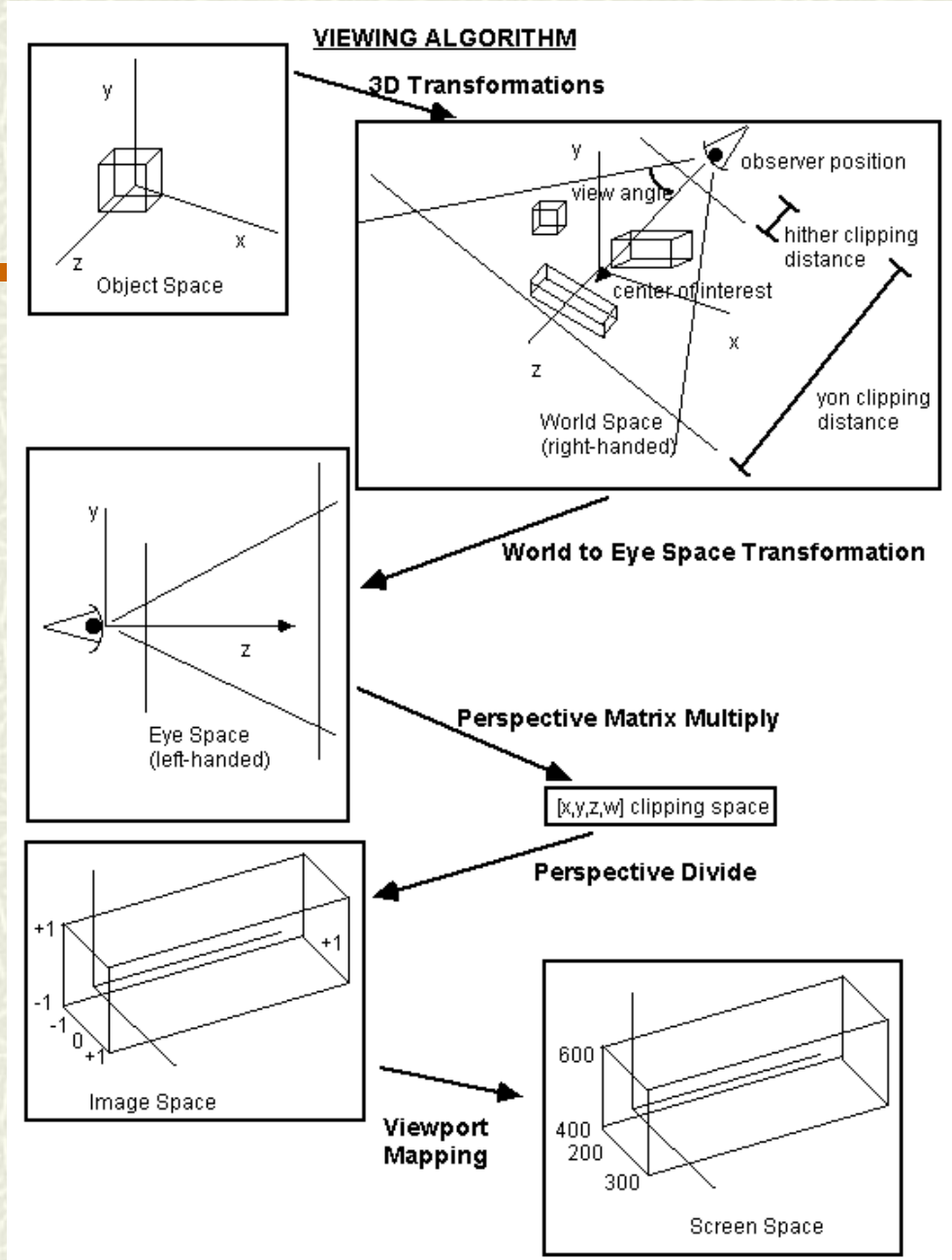
- Z-buffer pipelines originally designed to minimize computation using ***spatial coherence***
  - Interpolation for position, color, mapping etc instead of full-blown ray-object intersections and lighting computations





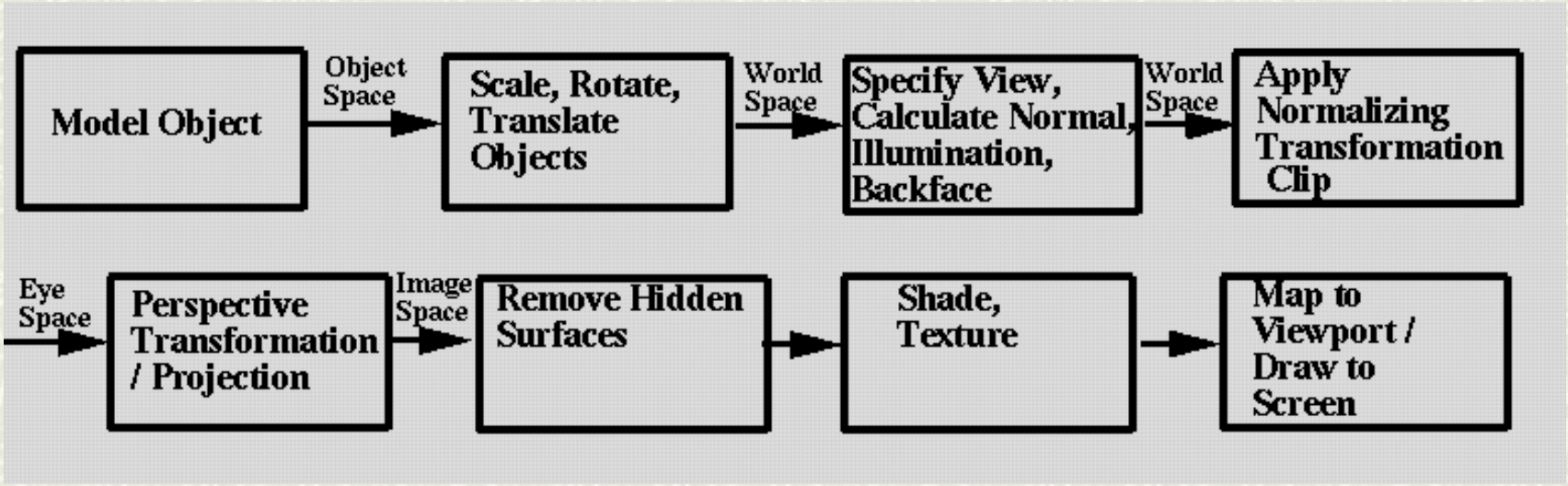
Vertex processing

- Floating point processing of vertex position, normal, color, etc
- 4-element matrix-vector operations



Basic GPU Organization

- To date this has been done using a *z-buffer pipeline*



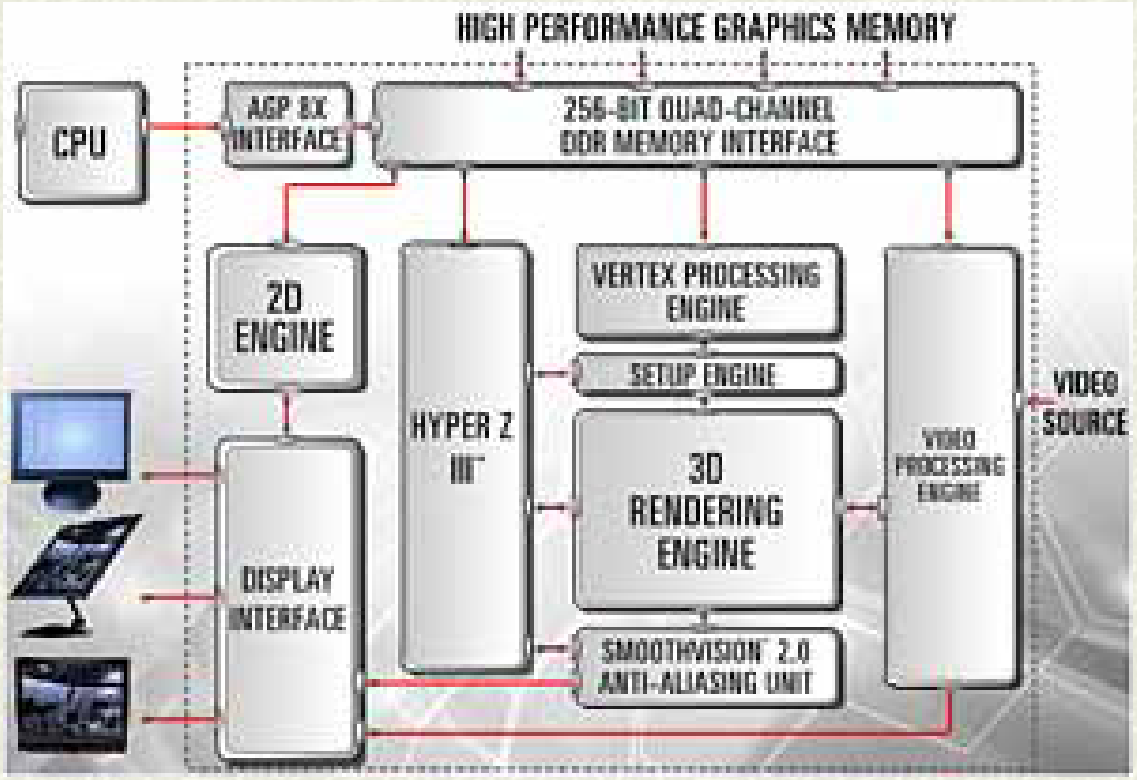
Key GPU performance issues

- In current technology, computation is cheap but the memory interface is slow
  - DRAM is typically around 200 clock cycles from the processor
- Fortunately, spatial coherence also leads to locality of memory references and thus good memory access behavior
- So, z-buffer pipelines are still well-suited to modern technology

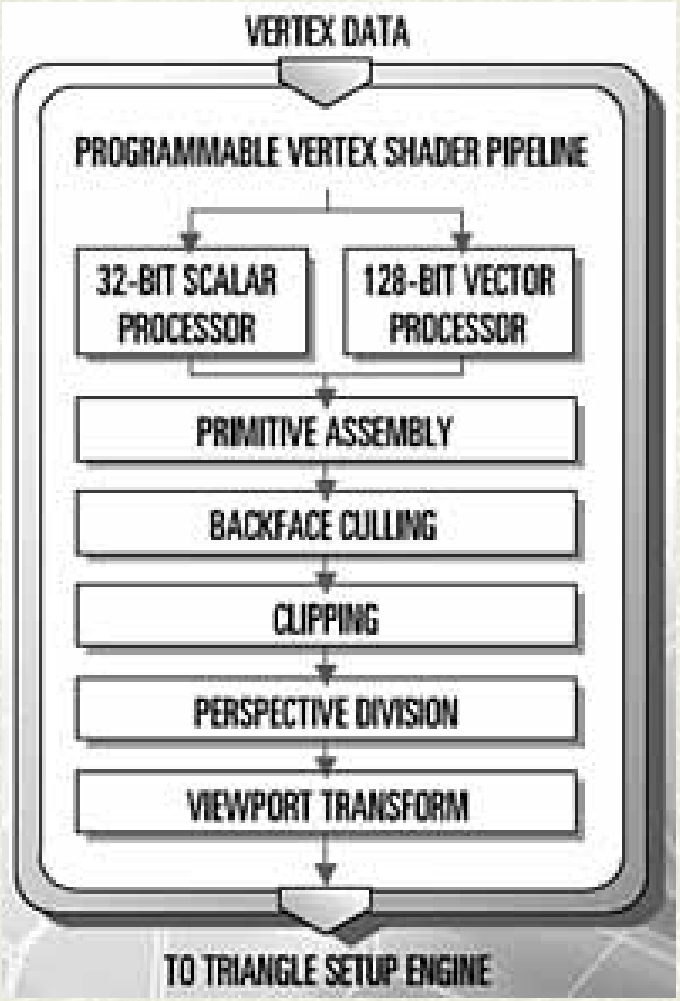
Recent trends

- Programmability
- Original z-buffer based systems streamed data through fixed pipelines
- Now, fragment processing supports programmable shading
- Vertex processing also becoming more programmable
- Still streaming through programmable processors

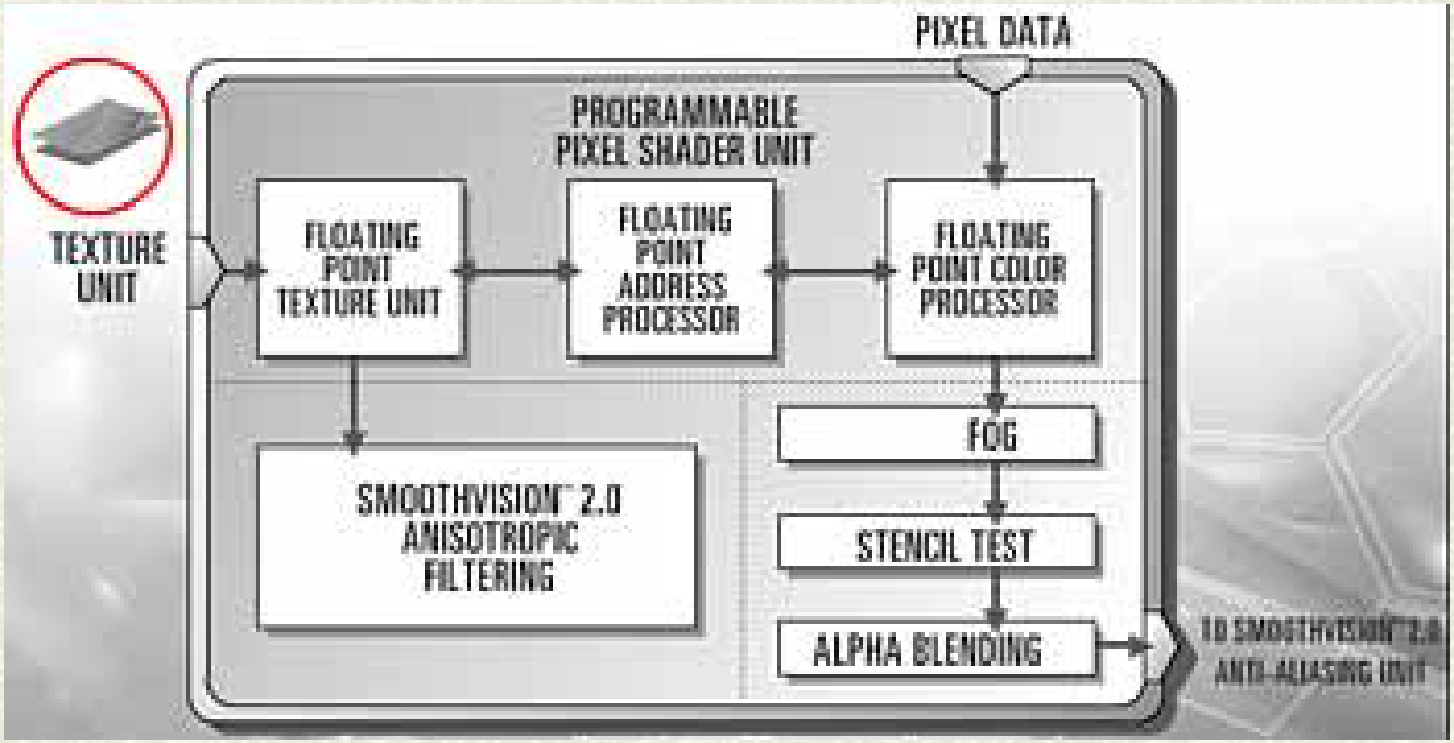
Current generation example – ATI Radeon 9700



Programmable vertex processing



Programmable fragment processing



Where to go from here?

- More of the same- more programmability, more parallelism?
- How about higher quality imagery-global illumination
- The fundamental compromise in a z-buffer pipeline is the use of local illumination to take maximum advantage of coherence
- Can we get beyond the need for this compromise in next-generation machines?

Hardware supported global illumination

- How about real-time ray tracing?
  - Already done for static scenes of limited size on existing GPUs
  - Already done with fewer limitations on large-scale parallel machines
- Our aim is to do this for fully dynamic scenes of comparable complexity to those handled by modern z-buffer pipelines

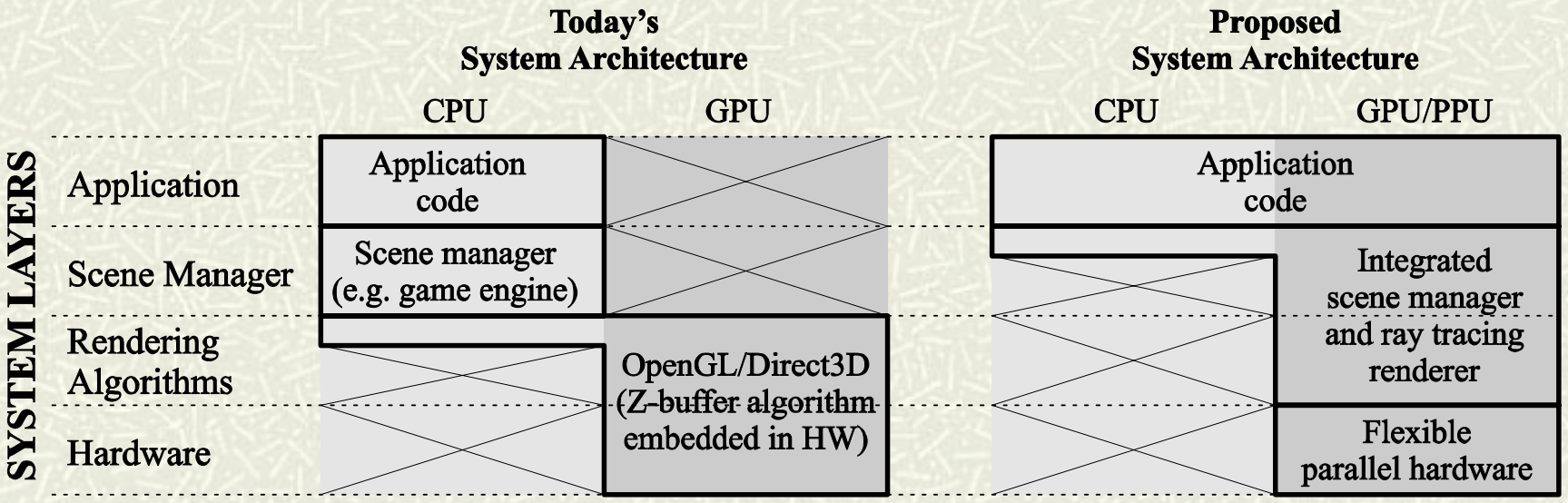
Algorithmic requirements

- We will have to start by changing the way graphics systems are constructed
  - Key feature – separation of modeling functions from rendering functions, OpenGL and DirectX APIs provide the interface
  - Today's chips support rendering but not modeling
  - CPU has to support modeling, but this is becoming ever more important to overall system performance
    - Occlusion culling
    - Scene graph management

Uniting modeling with rendering

- Ray tracing requires spatial data structures for acceleration that are quite similar to occlusion culling structures
- Z-buffers systems now deal awkwardly with occulusion culling support
- GPU can be made much faster than CPU at this critical function
- GPU will support both modeling and rendering

System architecture

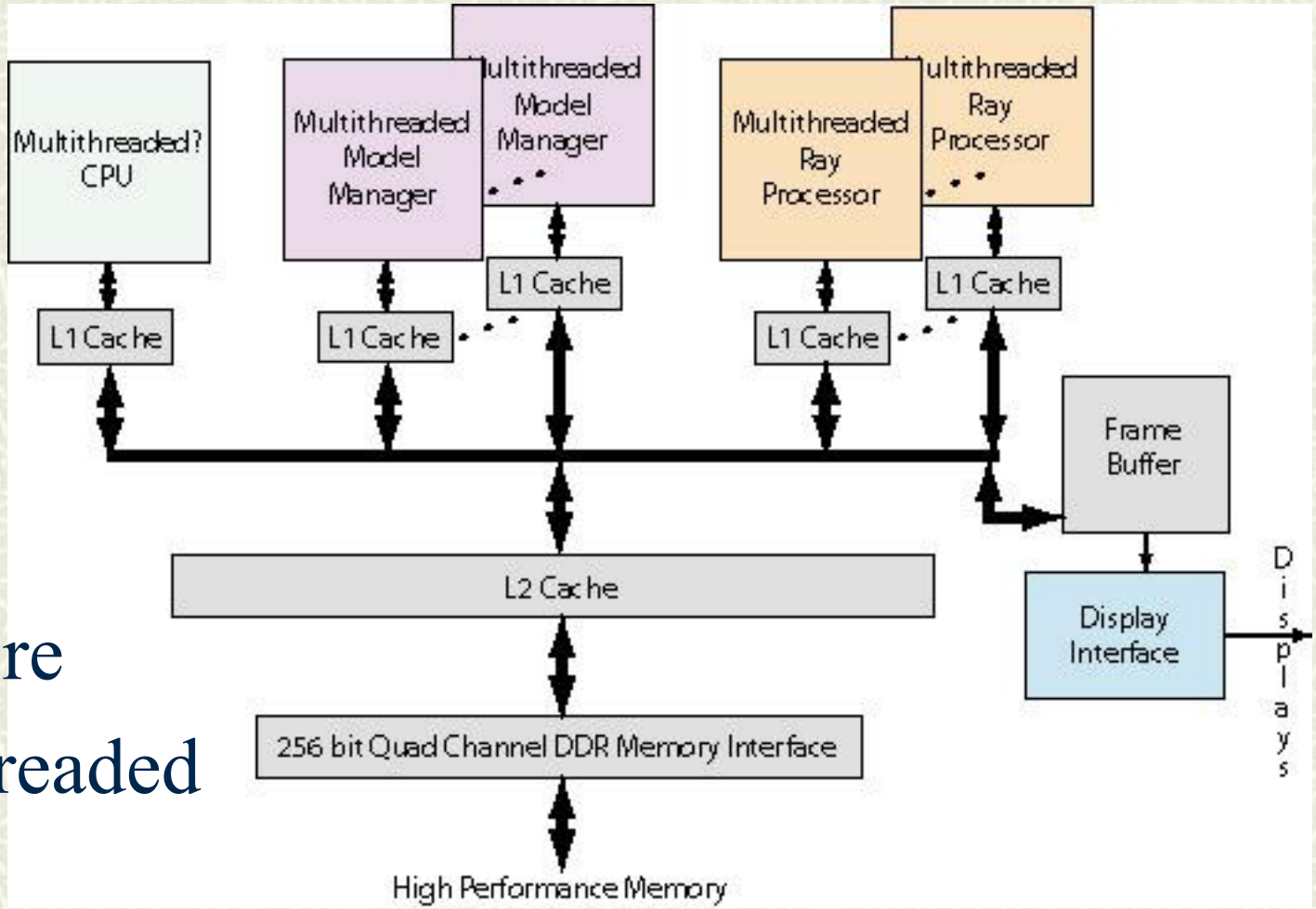


Future GPU characteristics

- Must support irregular computations
  - Today we are just getting data dependent branching into these systems
- Must be highly programmable
  - Applications are more diverse than rendering
- Must be bandwidth – not latency – optimized
- Must continue to be highly parallel for performance

Our proposed architecture

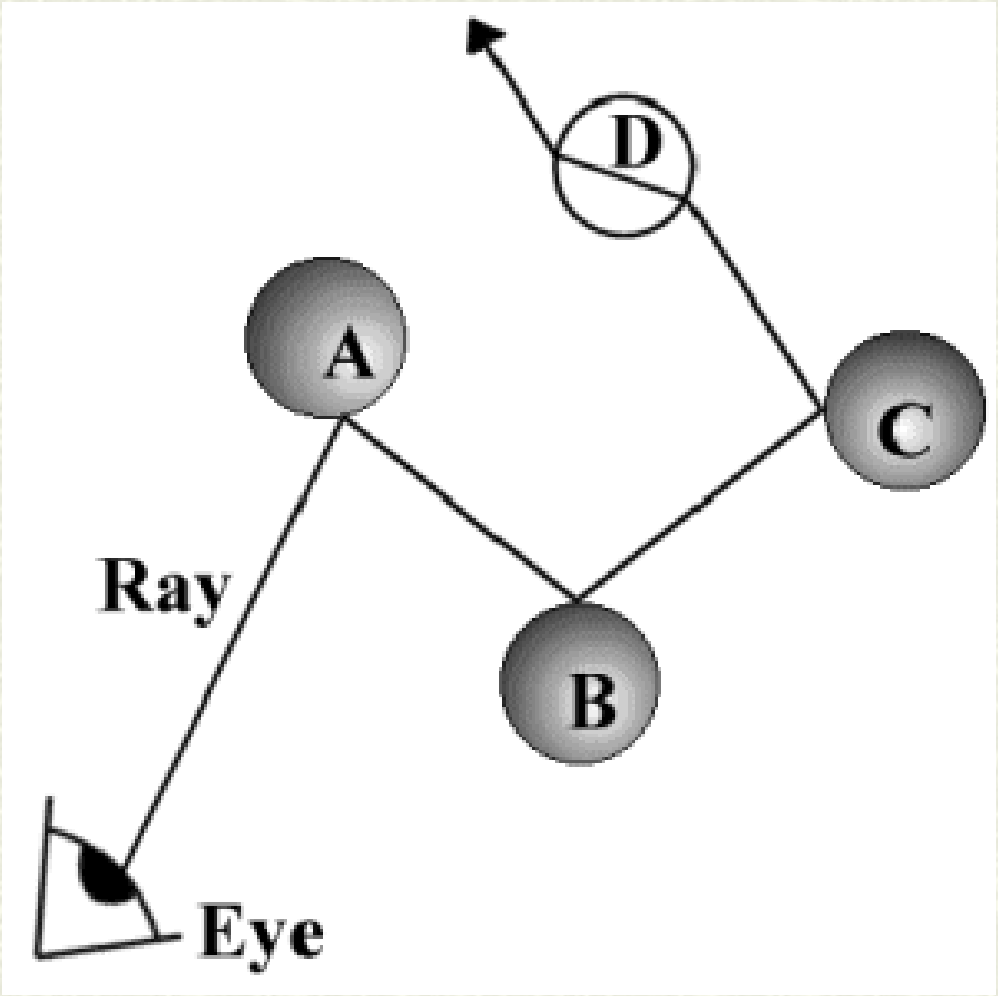
- MIMD
- Multicore
- Multithreaded



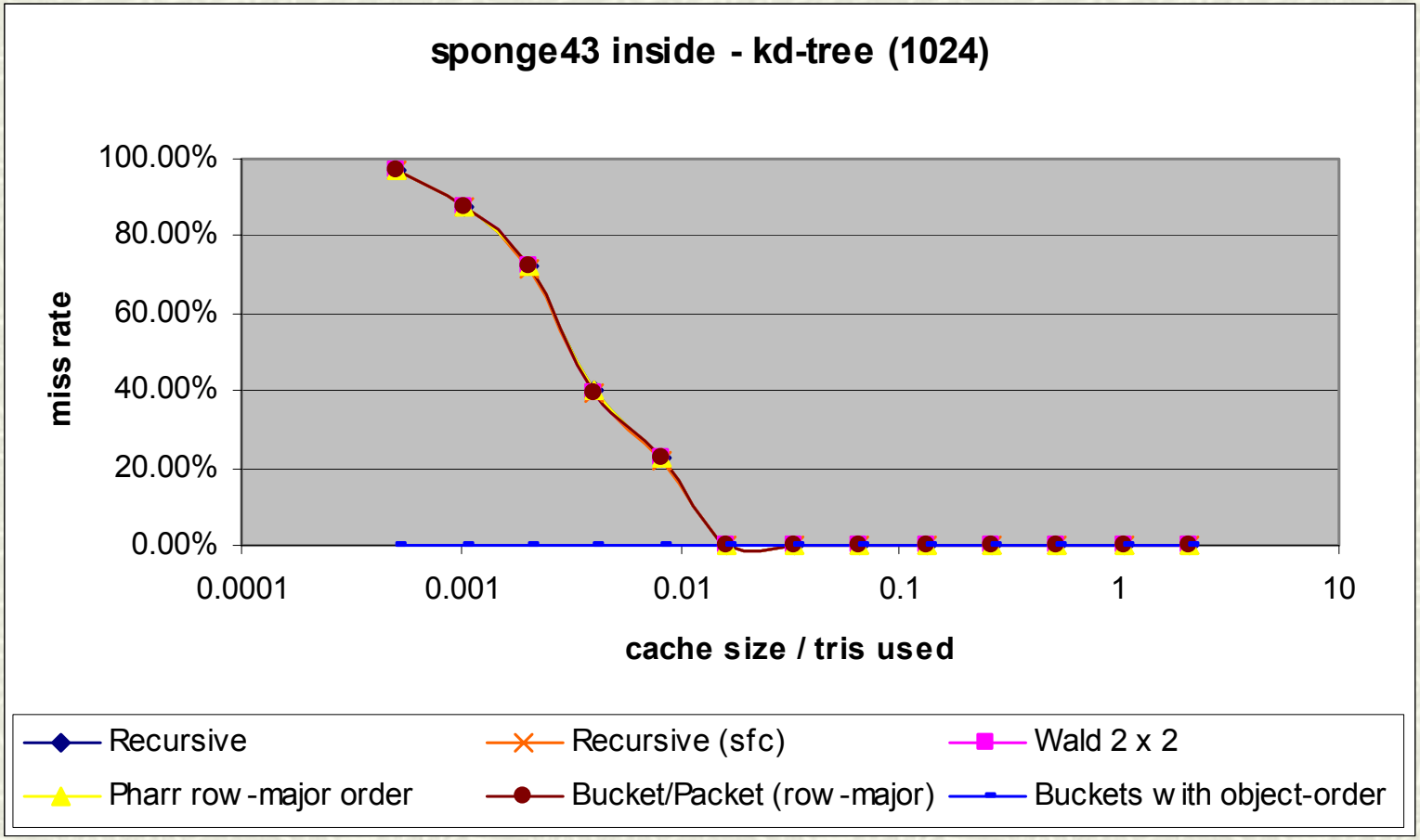
Approaching the efficiency of z-buffers

- We have been working to adapt ray tracing algorithms to perform well on the 2 level on chip cache scheme envisioned above
  - K-d tree as both acceleration structure and L2 cache management structure
  - New ray ordering to maximize cache utilization

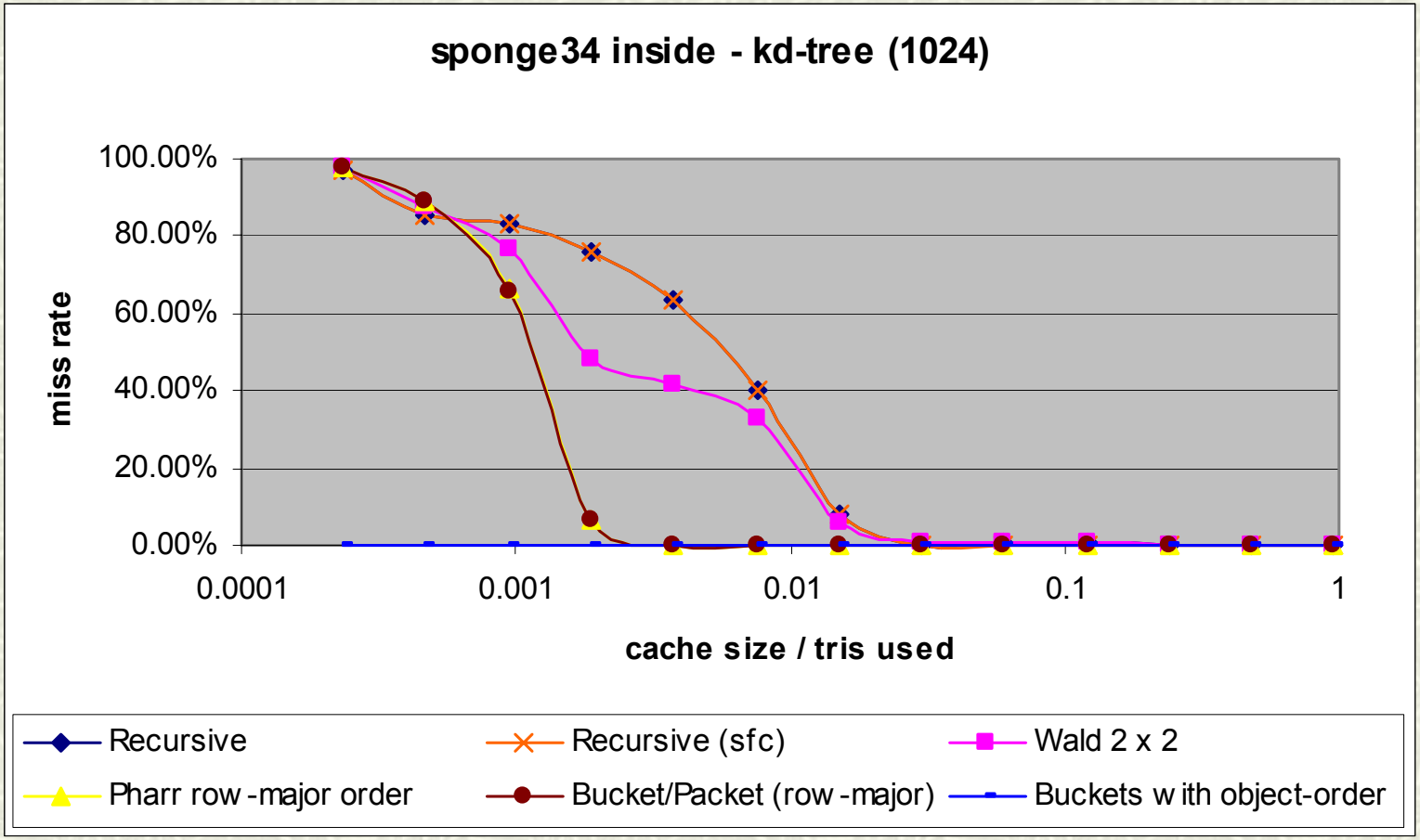
Ray Tracing



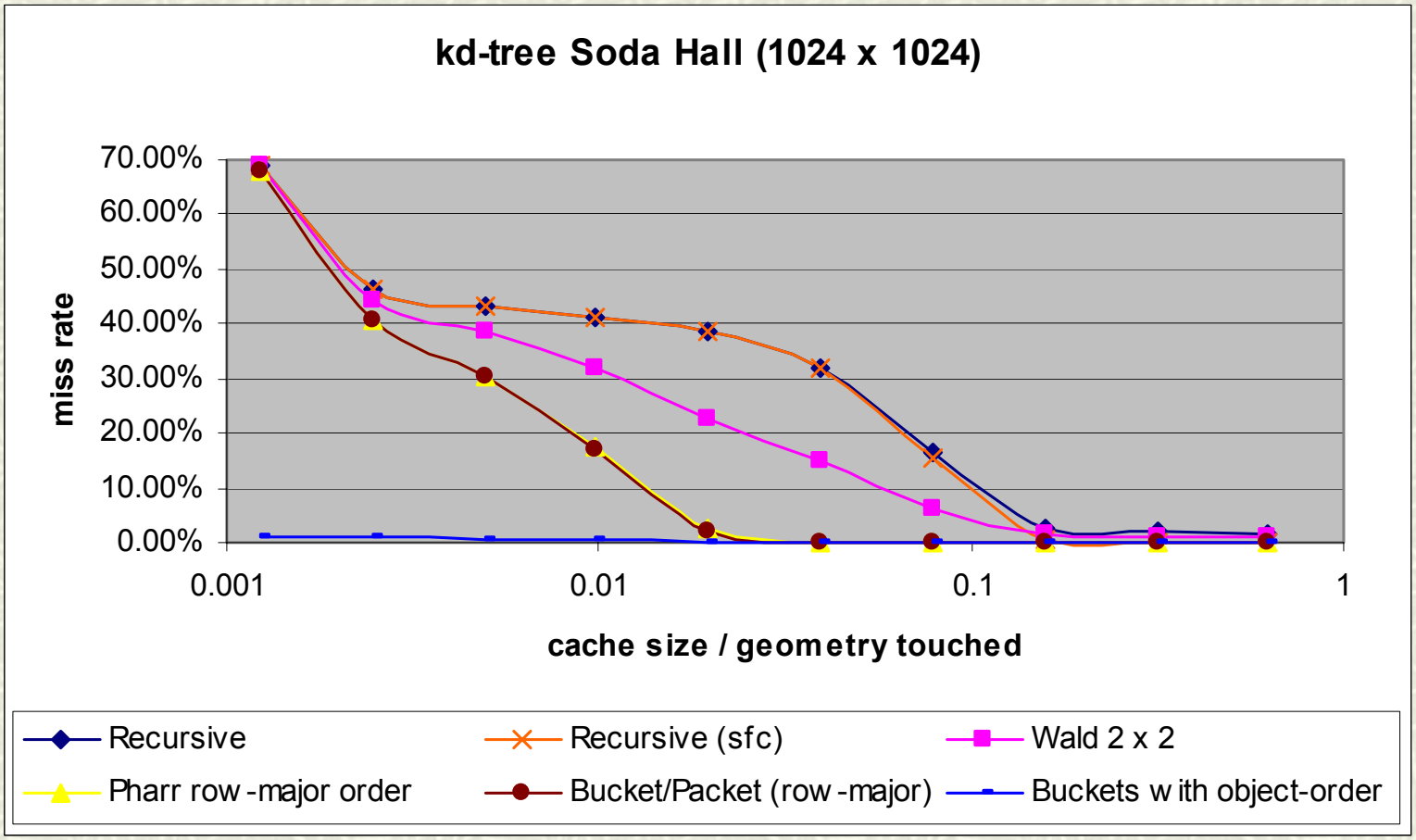
Test scene results



Test scene results



Real scene results



How general purpose will these be?

- GPUs are more powerful than CPUs, if they become more generally useful, the capacity of networks of PCs for scientific computation will dwarf those of today
- Since GPUs are as necessary to modern PCs as CPUs, we envision the evolution of single-die systems containing both the serial CPU and the parallel GPU

Conclusions

- Global illumination – the next (and near) frontier for GPUs
- This will require a finer-grained version of the multithreaded, multicore architectures emerging today
- Tight coupling of modeling and rendering
- New algorithmic approaches
- New substrate for general scientific computing