

GPU Data Structures

Aaron Lefohn

Neoptica



Introduction

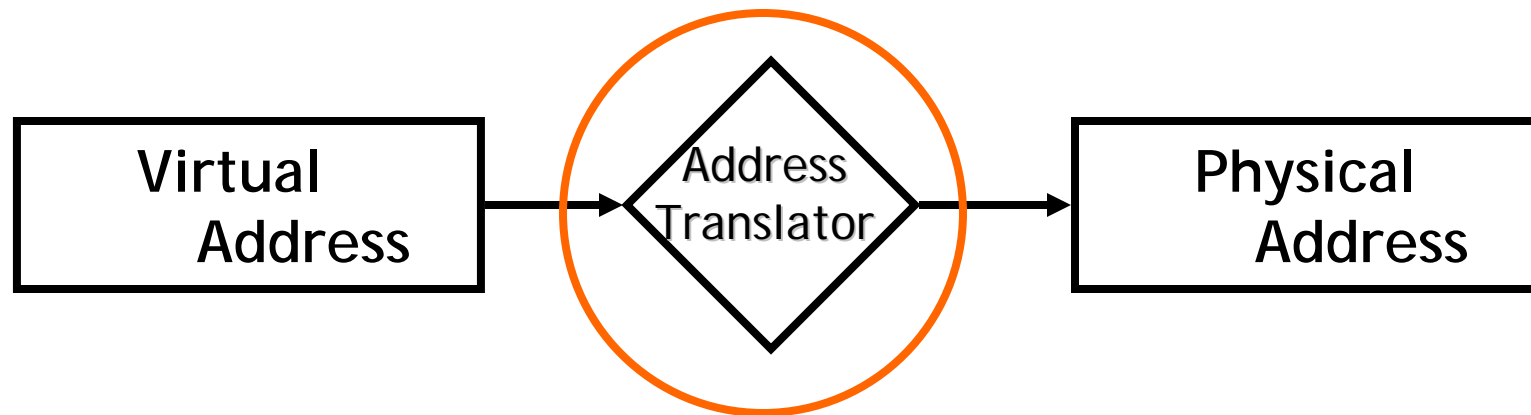
- Previous talk: GPU memory model
- This talk: GPU data structures

Properties of GPU Data Structures

- To be efficient, must support
 - Parallel read
 - Parallel write
 - Parallel iteration
- Generalized arrays fit these requirements
 - Dense (complete) arrays
 - Sparse (incomplete) arrays
 - Adaptive arrays

Address Translation

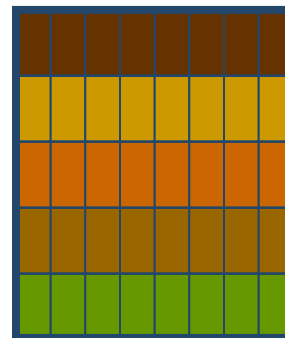
- Design pattern for generalized arrays



- Example:
 - CPU: 3D virtual memory → 1D physical memory
 - GPU: 3D virtual memory → 2D physical memory
- Address translator is core of data structure

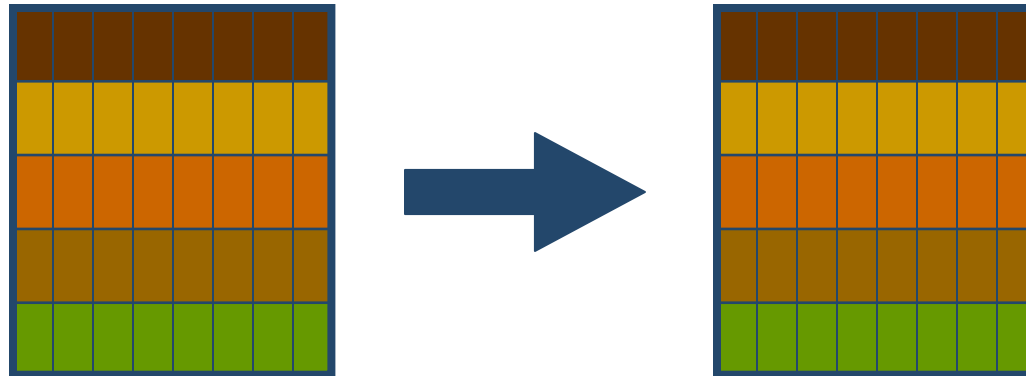
Dense (Complete) GPU Arrays

- Large 1D Arrays
 - Current GPUs limit 1D array sizes to 2048 or 4096
 - Pack into 2D memory
 - 1D-to-2D address translation



GPU Arrays

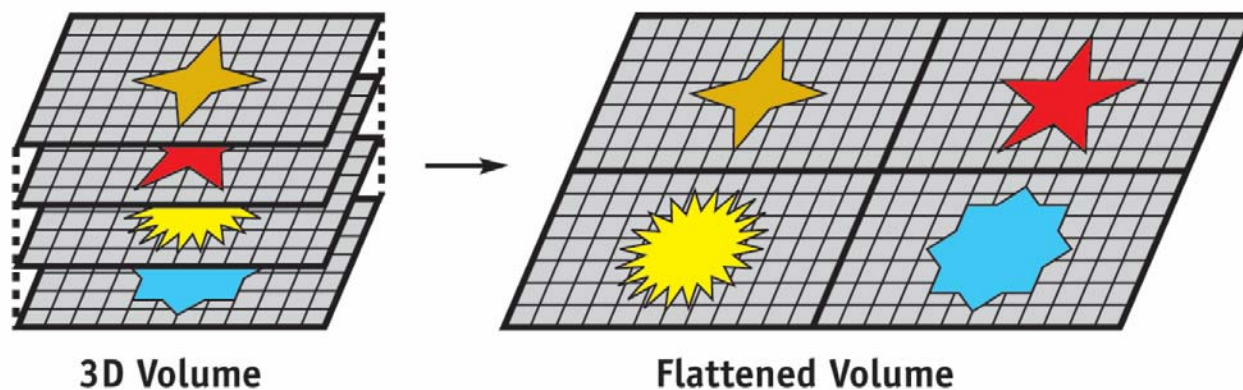
- 2D Arrays
 - Trivial, native implementation



GPU Arrays

- **3D Arrays**

- Problem: GPUs do not have 3D frame buffers
- Solutions
 1. Multiple slices per 2D buffer ("Flat 3D array")
 2. Stack of 2D slices
 3. Render-to-slice of 3D array



GPU Arrays

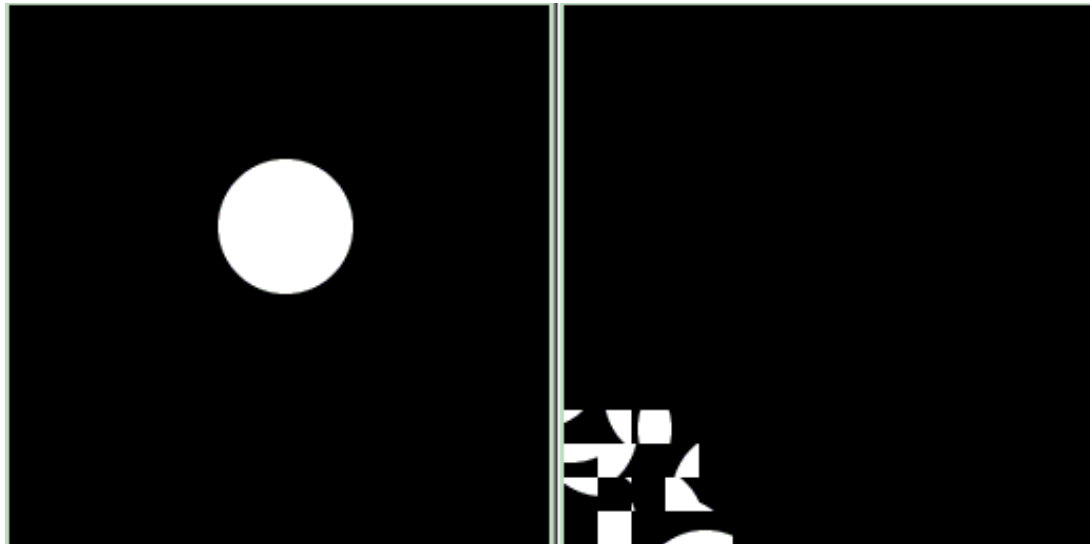
- **DX 10 memory model helps**
 - Can render to slice of 3D texture
- **“Flat 3D array” still has advantages**
 - Render entire domain in single parallel compute pass
 - More parallelism when writing (slower for reading)

GPU Arrays

- Higher Dimensional Arrays
 - Pack into 2D buffers
 - N-D to 2D address translation

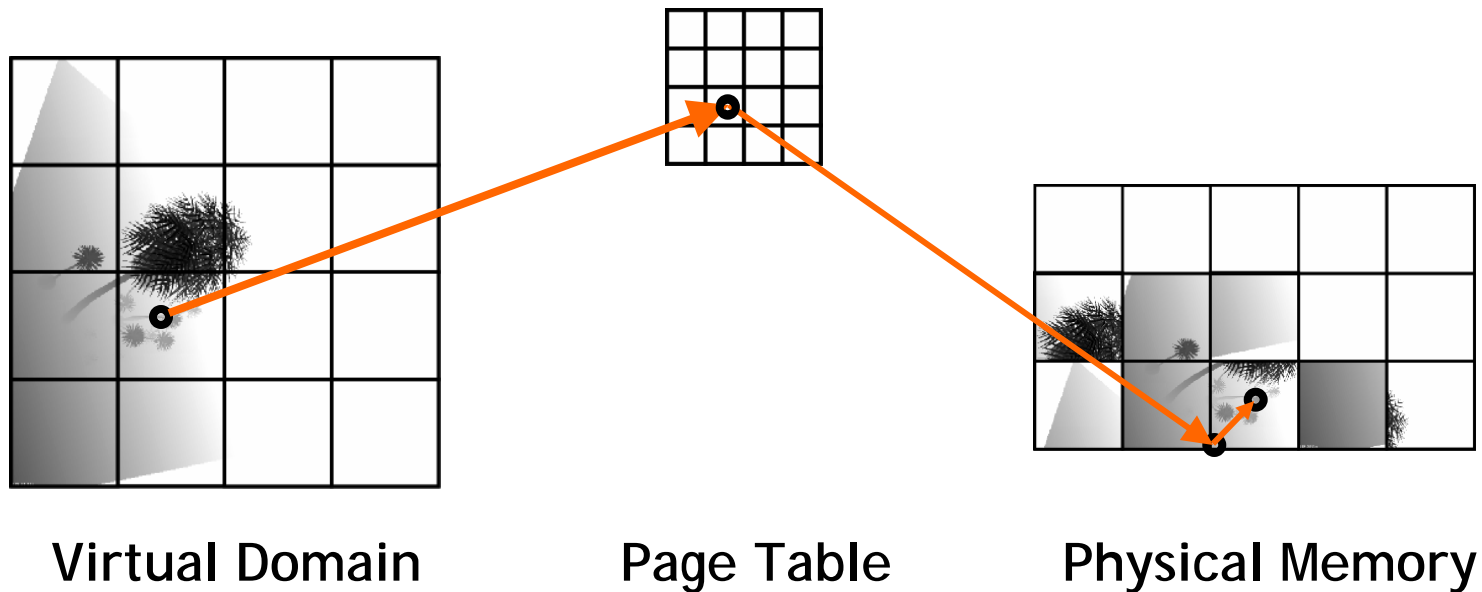
Sparse/Adaptive Data Structures

- **Why?**
 - Reduce memory pressure
 - Reduce computational workload
- **Basic Idea**
 - Pack “active” data elements into GPU memory



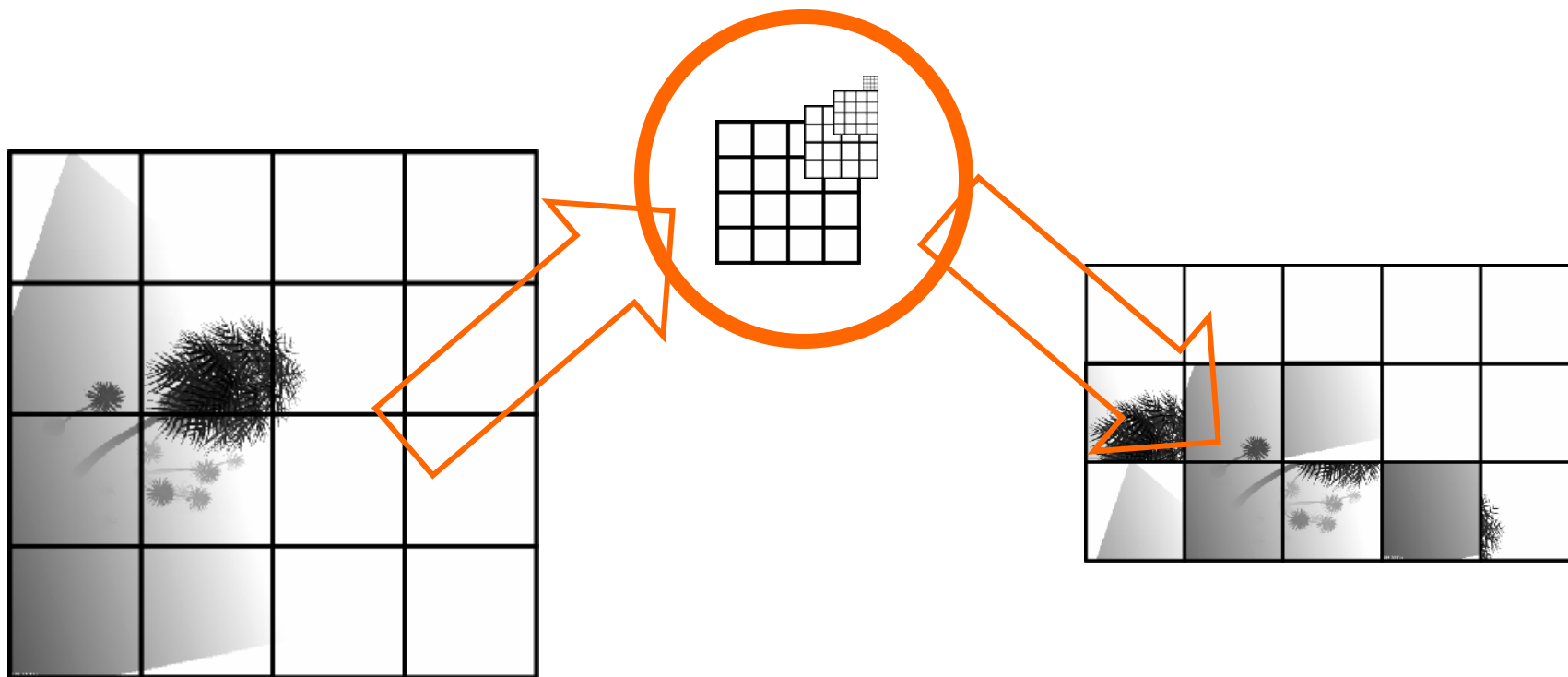
Page Table Sparse/Adaptive Arrays

- Dynamic sparse/adaptive N-D array
(Lefohn et al. 2003/2006)
 - Block-based sparse N-D arrays, octrees, quadtrees



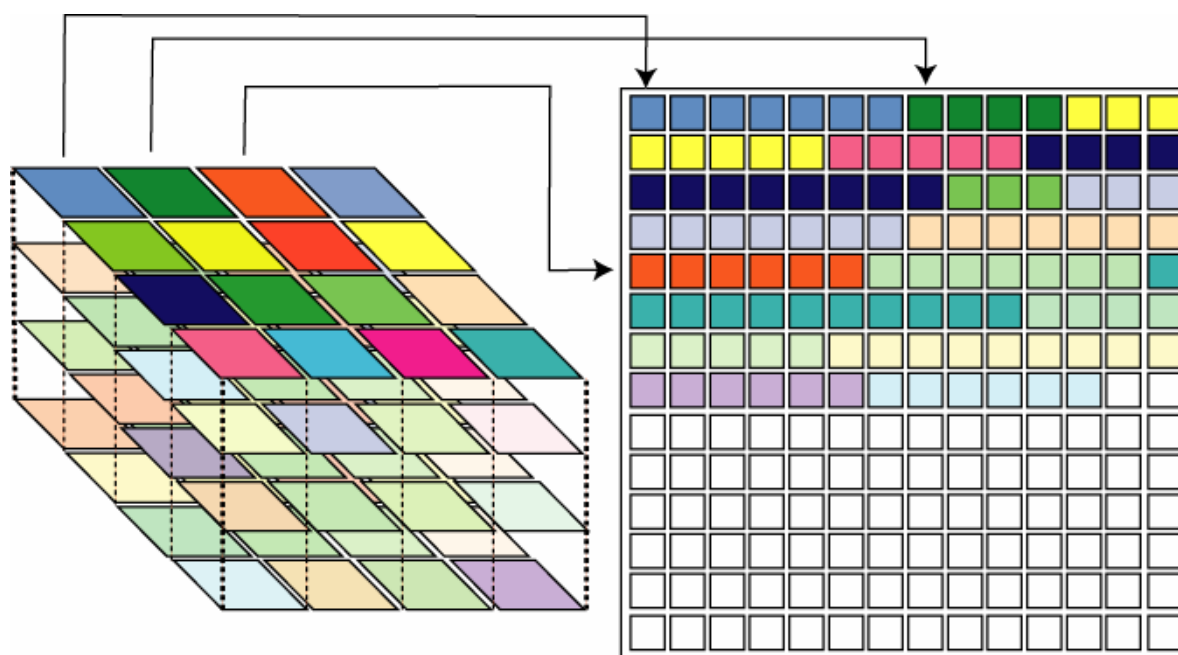
Page Table Sparse/Adaptive Arrays

- Hierarchical page table



Dynamic Adaptive Data Structure

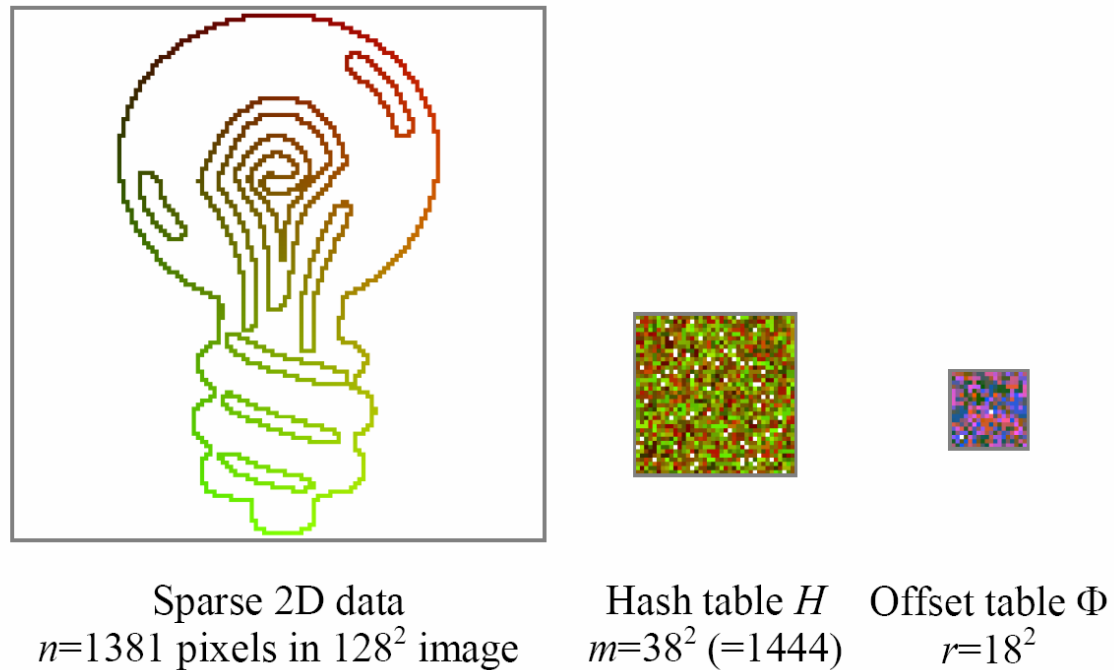
- Photon map (kNN-grid) (Purcell et al. 2003)



*Image from "Implementing Efficient Parallel Data Structures on GPUs,"
Lefohn et al., GPU Gems II, ch. 33, 2005*

GPU Perfect Hash Table

- Static, sparse N-D array (Lefebvre et al. 2006)



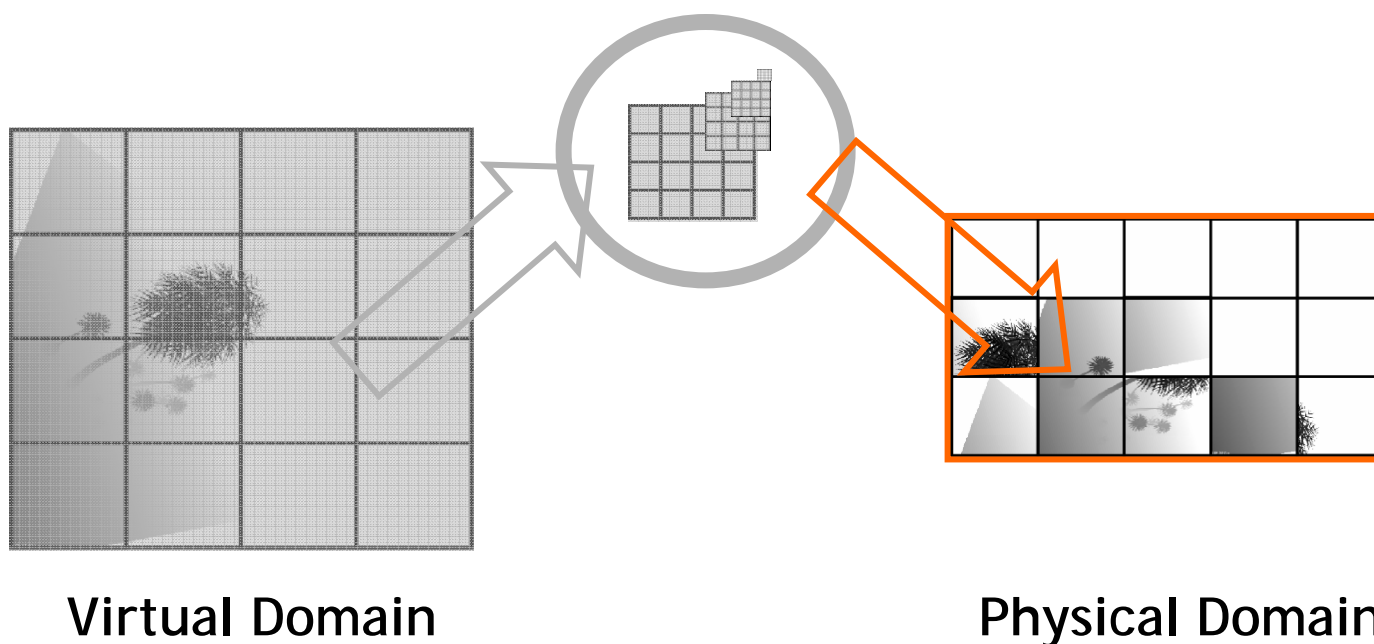
*Figure from Lefebvre, Hoppe, "Perfect Spatial Hashing,"
ACM SIGGRAPH, 2006*

GPU Iteration

- GPU good at random-access read
 - Often do not need to define input iterators
- GPU (mostly) performs only streaming writes
 - Every GPGPU data structure must support an output iterator compatible with GPU rasterization
 - GPU iterator defines contiguous (2)-D range

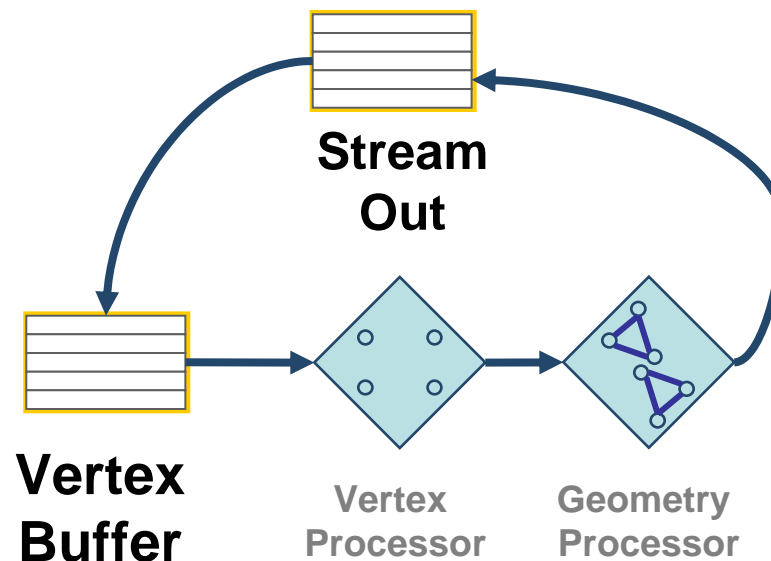
GPU Iteration

- Traverse physical data layout
 - Example
 - Draw single quad over all voxels in flattened volume
 - Draw one quad per-page over page-table-based array



GPU Iteration

- GPUs can now build data-structure iterators
 - DX10 "StreamOut" from geometry processor
 - Now easier to build dynamic GPU data structures



GPU Iteration

- **Optimizing input iterators**
 - Galoppo et al. optimized input and output iterators for their matrix representation
 - Match 2D GPU memory layout
 - Match memory access pattern of algorithm
 - Difficult because memory layouts are unspecified and proprietary
 - More on this for GPU sorting in next talk...

Galoppo, Govindaraju, Henson, Manocha,

"LU-GPU: Efficient Algorithms for Solving Dense Linear Systems on Graphics Hardware,"

ACM/IEEE Supercomputing, 2005

GPU Data Structure Conclusions

- Fundamental GPU memory primitive is a fixed-size 2D array (when building atop graphics APIs)
- GPU data structures require parallel read/write
- GPGPU needs more general memory model
 - Low-level interfaces such as ATI's CTM and NVIDIA's CUDA (!!)
 - Iterator access patterns must match storage layout
 - Commonplace for CPU programmers, but...
 - Need low-level information from hardware vendors

GPU Data Structure Conclusions

- **Building complex data structures on GPU still hard**
 - Can use data-parallel algorithms (sort, scan, etc.)
 - DX10 geometry processor “StreamOut” helps
 - Is less parallelism more efficient for building data structures?
- **Next step is having GPU build iterators for dynamic, complex data structures**

More Information

- **Overview (with code snippets)**
 - Lefohn, Kniss, Owens, "Implementing Efficient Parallel Data Structures on GPUs," Chapter 33, GPU Gems II
- **High-level GPU data structures**
 - Lefohn, Kniss, Strzodka, Sengupta, Owens, "Glift: Generic, Efficient, Random-Access GPU Data Structures," ACM Transactions on Graphics, 2006
- **GPGPU State-of-the-art report**
 - Owens, Luebke, Govindaraju, Harris, Krüger, Lefohn, Purcell, "A Survey of General-Purpose Computation on Graphics Hardware," Eurographics STAR report, 2005