Ray Tracing and Global Illumination on Programmable Graphics Hardware

Timothy J. Purcell
Stanford University

Overview

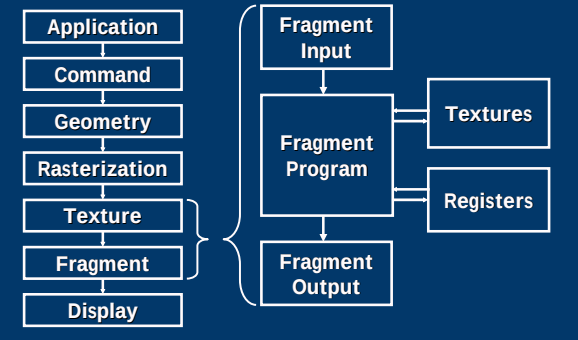
- Explore computation kernels used in ray tracing and photon mapping
  - Each kernel is applicable to more general scientific computing
- Demonstrate ray tracing and photon mapping systems utilizing these kernels
- Discuss some open issues in mapping algorithms to GPUs

Outline

- GPU abstraction
- Ordering and finding data
  - Sorting
  - Searching
  - Binning
  - Nearest neighbor queries
- Bringing it all together
  - Ray tracing and photon mapping demos
- Open issues in mapping algorithms to GPUs

GPU Abstraction

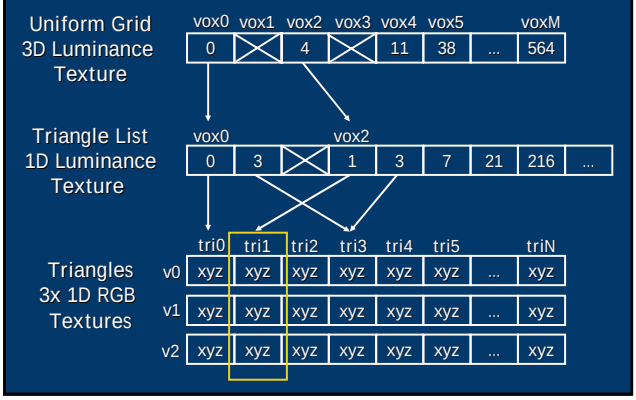
Graphics Pipeline



GPU Abstraction Basics

- **Texture memory is memory**
  - Think of dependent texture fetches as pointer dereferencing
- **Programmable GPU is a programmable stream processor**
  - Think of multipass rendering as stream and kernel programming
- **These insights were key for mapping ray tracing to a programmable GPU**

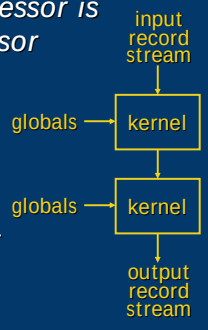
Texture Memory Organization



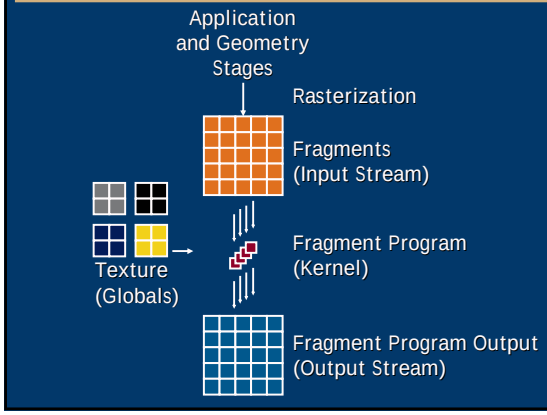
Stream Programming Model

Programmable fragment processor is essentially a stream processor

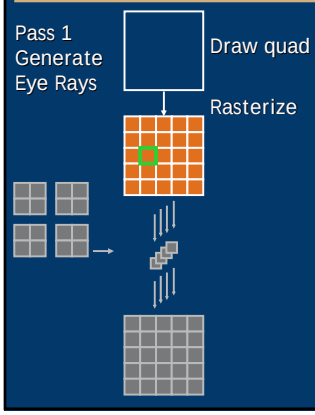
- **Kernels and streams**
  - Stream is a set of data records
  - Kernels operate on records
  - Streams connect kernels together
  - Kernels can read global memory



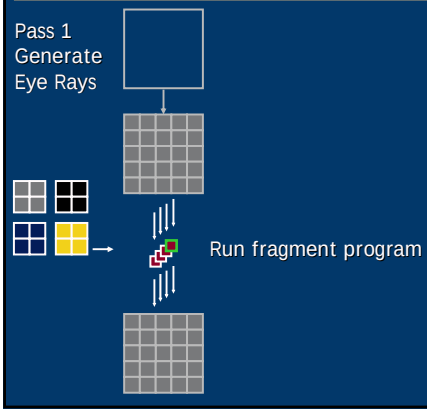
Streaming Flow Control



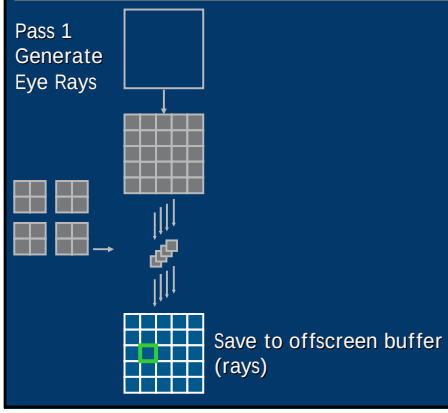
Multiple Rendering Passes



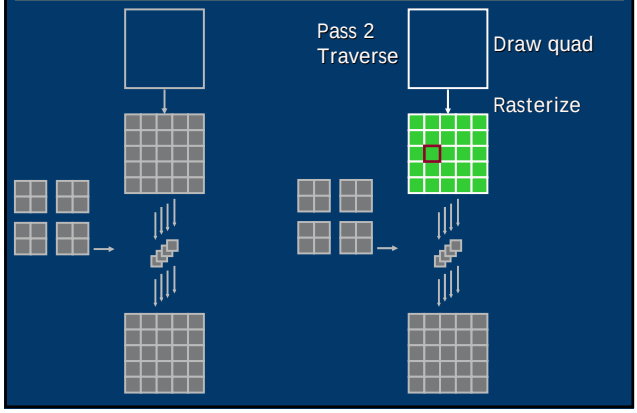
Multiple Rendering Passes



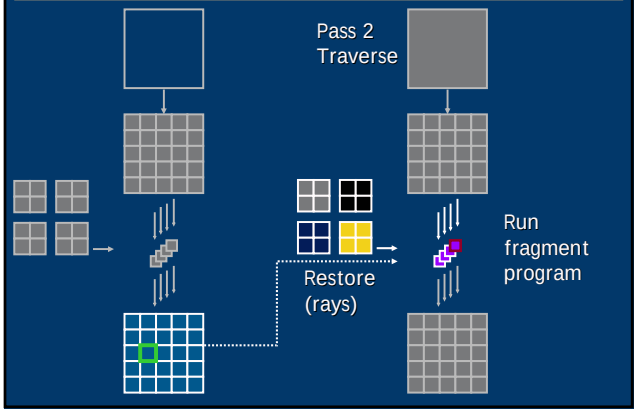
Multiple Rendering Passes



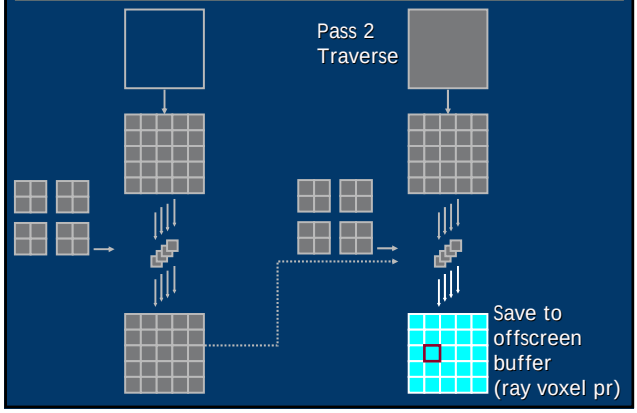
Multiple Rendering Passes



Multiple Rendering Passes



Multiple Rendering Passes

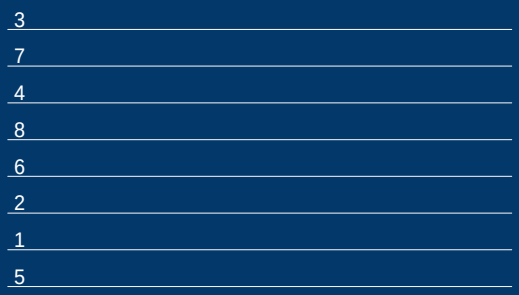


Sorting

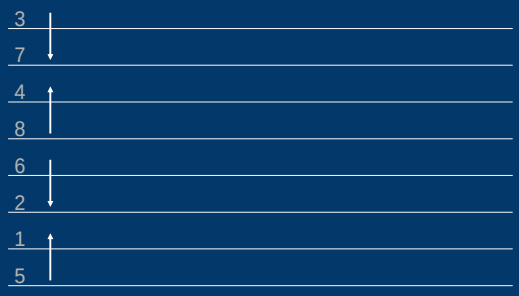
Sorting

- Given an unordered list of elements, produce list ordered by key value
- Bitonic merge sort [Batcher 68]
- Used to order photons during construction of photon map data structure

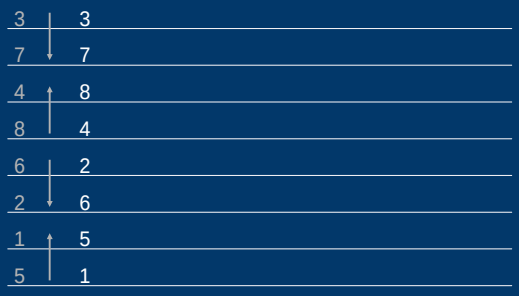
Bitonic Merge Sort



Bitonic Merge Sort

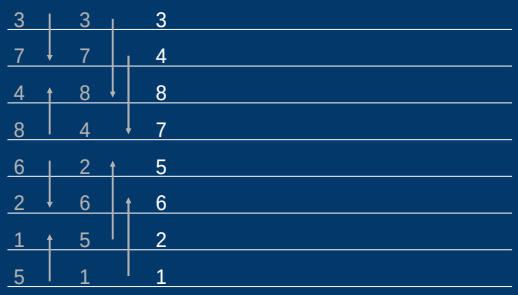


Bitonic Merge Sort

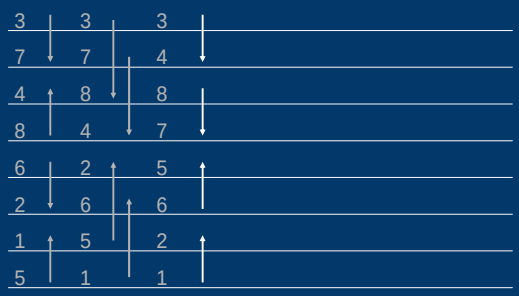


Bitonic Merge Sort

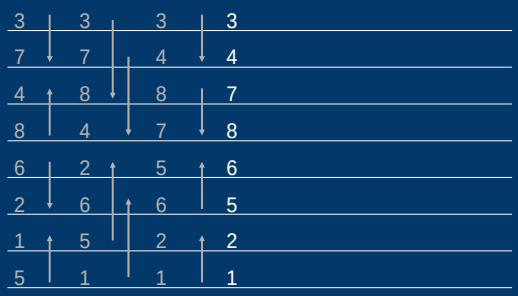
Bitonic Merge Sort



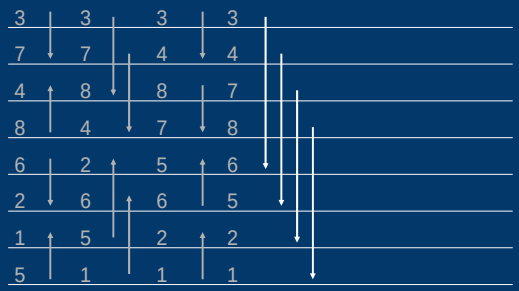
Bitonic Merge Sort



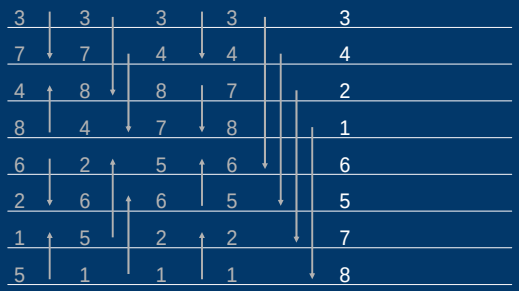
Bitonic Merge Sort



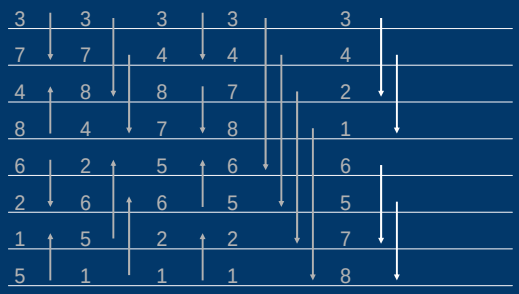
Bitonic Merge Sort



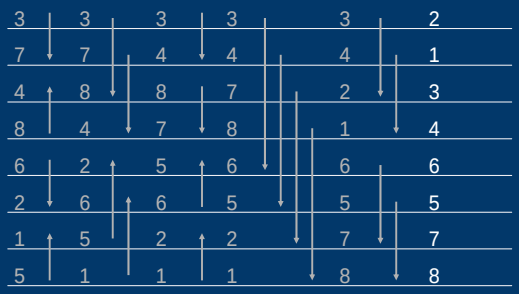
Bitonic Merge Sort



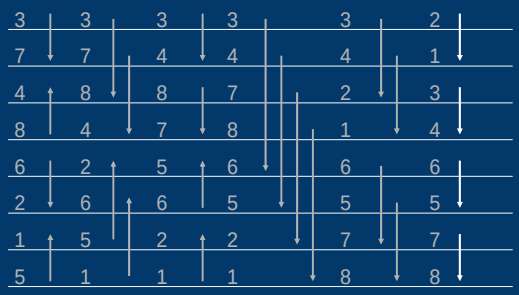
Bitonic Merge Sort



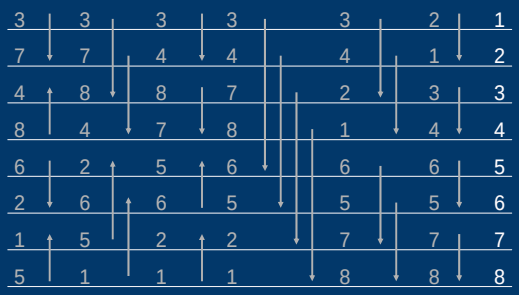
Bitonic Merge Sort



Bitonic Merge Sort



Bitonic Merge Sort



Bitonic Merge Sort Summary

- **Separate rendering pass for each swap**
  - $O(\log^2 n)$ passes
- **Hand coded to around 20 instructions**
  - 512x512 elements - 3060 instructions per pixel

Searching

Searching

- **Find a specific element in an ordered list**
- **Binary search**
- **Used to build uniform grid structure for photon map**

Binary Search

- **Find the first element in each grid cell**
  - If none, find first element in next cell

Binary Search

- **Find the first element in each grid cell**
  - If none, find first element in next cell

Binary Search

- **Find the first element in each grid cell**
  - If none, find first element in next cell

Binary Search

- **Find the first element in each grid cell**
  - If none, find first element in next cell

Grid	v0	v1	v2	v3	v4	v5	
	4	4	4	4	4	4	initialize
	2	2	2	6	6	6	step 1
	1	3	3	5	5	5	step 2

Sorted Photon List (v# is key)	0	1	2	3	4	5	6	7
	v0	v0	v0	v2	v2	v2	v5	v5

Binary Search

- **Find the first element in each grid cell**
  - If none, find first element in next cell

Grid	v0	v1	v2	v3	v4	v5	
	4	4	4	4	4	4	initialize
	2	2	2	6	6	6	step 1
	1	3	3	5	5	5	step 2
	0	2	2	6	6	6	step 3

Sorted Photon List (v# is key)	0	1	2	3	4	5	6	7
	v0	v0	v0	v2	v2	v2	v5	v5

Binary Search

- **Find the first element in each grid cell**
  - If none, find first element in next cell

Grid	v0	v1	v2	v3	v4	v5	
	4	4	4	4	4	4	initialize
	2	2	2	6	6	6	step 1
	1	3	3	5	5	5	step 2
	0	2	2	6	6	6	step 3
	0	3	3	6	6	6	step 4

Sorted Photon List (v# is key)	0	1	2	3	4	5	6	7
	v0	v0	v0	v2	v2	v2	v5	v5

Binary Search Summary

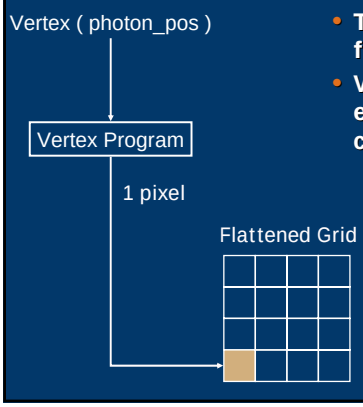
- **Single rendering pass**
- **O(log n) steps**
  - 18 instructions per step (Cg 1.1)
  - 512x512 elements – 342 instructions per pixel

Binning

Binning

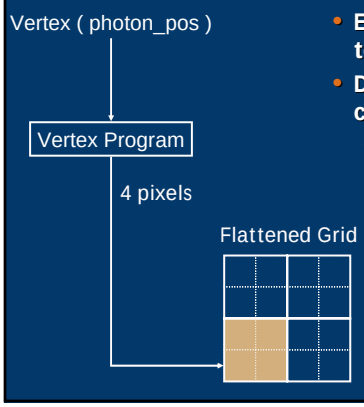
- **Given an unordered list of elements, produce list ordered by key value**
  - Multiple elements with same key are placed in the same bin since they have no ordering amongst themselves
- **Stencil routing**
- **Alternate approach to building the photon map data structure**

Stencil Routing



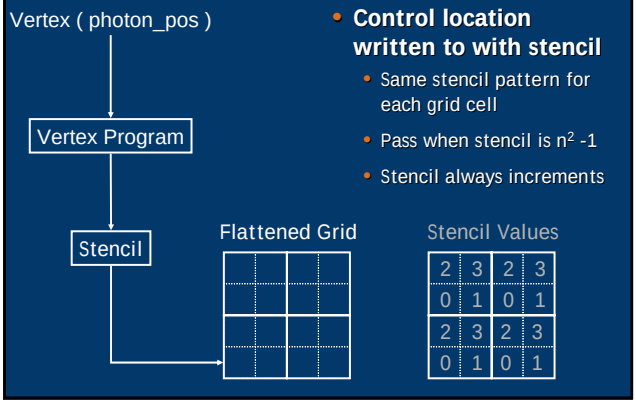
- **Treat framebuffer as a flattened grid**
- **Vertex program sends each photon to its grid cell**
  - Only the last photon in each cell is saved

Stencil Routing



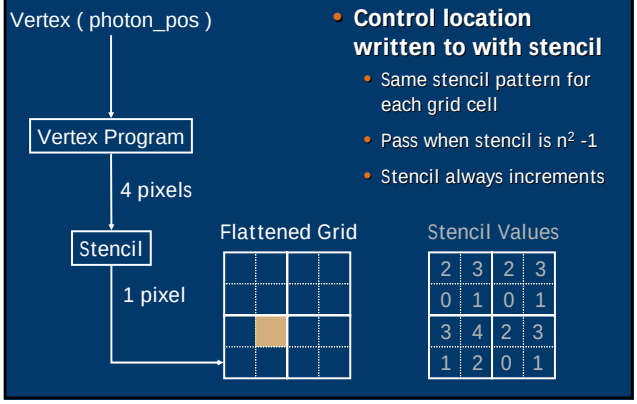
- **Enlarge each grid cell to n^2 pixels**
- **Draw fat points to cover each fat cell**
  - glPointSize(n)

Stencil Routing



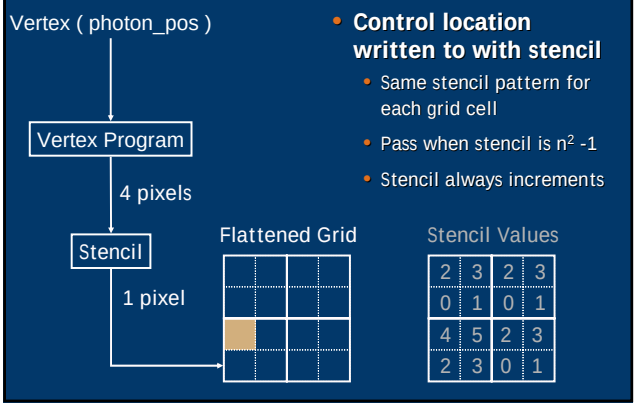
- **Control location written to with stencil**
  - Same stencil pattern for each grid cell
  - Pass when stencil is n^2 -1
  - Stencil always increments

Stencil Routing



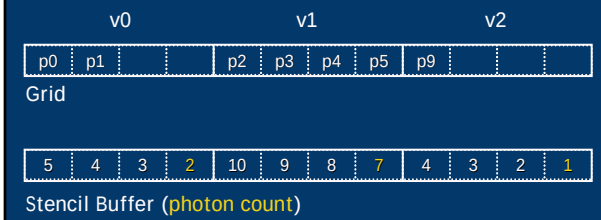
- **Control location written to with stencil**
  - Same stencil pattern for each grid cell
  - Pass when stencil is n^2 -1
  - Stencil always increments

Stencil Routing



- **Control location written to with stencil**
  - Same stencil pattern for each grid cell
  - Pass when stencil is n^2 -1
  - Stencil always increments

Photon Map Structure



Assuming p6 – p8 all land in v1

Stencil Routing Summary

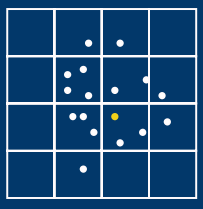
- Single rendering pass
- Vertex program is fast
  - 42 instructions (Cg 1.1)
- Resulting structure is sparse
- Fixed number of entries per bin may be unacceptable
- Requires render to vertex array (readback)
- Requires stencil buffer readback

Nearest Neighbor Queries

Nearest Neighbor Queries

- Given a sample point *p*, find the *k* points nearest *p* within a data set
- kNN-grid
- Used when computing radiance estimate of a given sample point during photon mapping

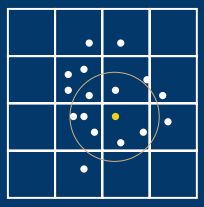
kNN-grid Algorithm



- sample point
- candidate photon
- photons in estimate

Want a 4 photon estimate

kNN-grid Algorithm

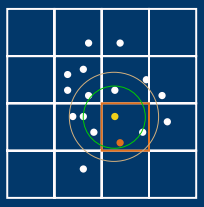


- sample point
- candidate photon
- photons in estimate

Want a 4 photon estimate

- **Candidate photons must be within max search radius**
- **Visit voxels in order of distance to sample point**

kNN-grid Algorithm

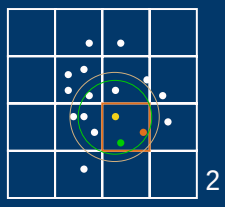


- sample point
- candidate photon
- photons in estimate

Want a 4 photon estimate

- **If current number of photons in estimate is less than number requested, grow search radius**

kNN-grid Algorithm

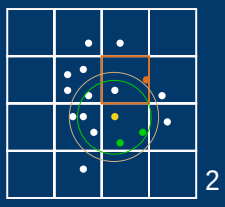


- sample point
- candidate photon
- photons in estimate

Want a 4 photon estimate

- **If current number of photons in estimate is less than number requested, grow search radius**

kNN-grid Algorithm

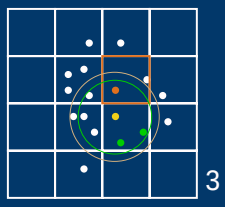


- sample point
- candidate photon
- photons in estimate

Want a 4 photon estimate

- **Don't add photons outside maximum search radius**
- **Don't grow search radius when photon is outside maximum radius**

kNN-grid Algorithm

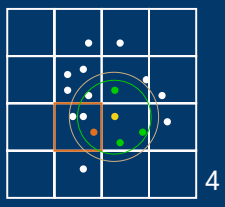


- **Add photons within search radius**

- sample point
- candidate photon
- photons in estimate

Want a 4 photon estimate

kNN-grid Algorithm

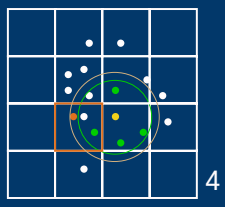


- **Add photons within search radius**

- sample point
- candidate photon
- photons in estimate

Want a 4 photon estimate

kNN-grid Algorithm

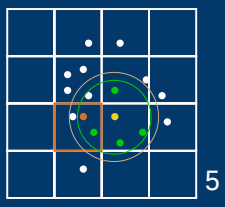


- **Don't expand search radius if enough photons already found**

- sample point
- candidate photon
- photons in estimate

Want a 4 photon estimate

kNN-grid Algorithm

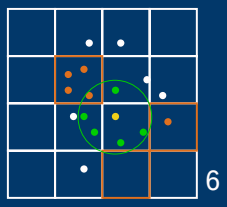


- **Add photons within search radius**

- sample point
- candidate photon
- photons in estimate

Want a 4 photon estimate

kNN-grid Algorithm

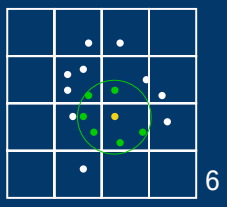


- sample point
- candidate photon
- photons in estimate

Want a 4 photon estimate

- **Visit all other voxels accessible within determined search radius**
- **Add photons within search radius**

kNN-grid Summary



- sample point
- candidate photon
- photons in estimate

Want a 4 photon estimate

- **Finds all photons within a sphere centered about sample point**
- **May locate more than requested *k*-nearest neighbors**
- **202 instructions per candidate neighbor (Cg 1.1)**

Bringing It All Together

Ray Tracing and Photon Mapping Demos

http://graphics.stanford.edu/papers/rtongfx
http://graphics.stanford.edu/papers/photongfx

Open Issues in Mapping Algorithms to GPUs

Open Issues

- **Compute mask, branching, or stream buffer?**
  - Need some way to prevent execution of expensive programs over programmed subset of pixels
- **Scatter**
  - Expensive to implement
    - Not unsolved problem
  - Many algorithms benefit
    - Sorting gains log(n) with native scatter
    - Building photon map improves by $\log^2 n$

Open Issues

- **Read-modify-write buffers**
  - Efficient save and restore over multiple passes
- **Addressing modes**
  - 2D textures vs. 1D textures
- **Integer computation**
  - Sometimes you need a mod or a div
- **Readback speed**

Acknowledgments

- **Pat Hanrahan**
- **Ian Buck, Bill Mark, Craig Donner, Mike Cammarano, Henrik Wann Jensen, Pradeep Sen, Eric Chan**
- **ATI - Bob Drebin, James Percy**
- **Nvidia - Kurt Akeley, David Kirk, Matt Papakipos, Nick Triantos**
- **Funding**
  - Nvidia Graduate Research Fellowship
  - ATI, DARPA, MERL, Nvidia, Sony, Sun