Real - Time Rendering

Ray Tracing on GPU

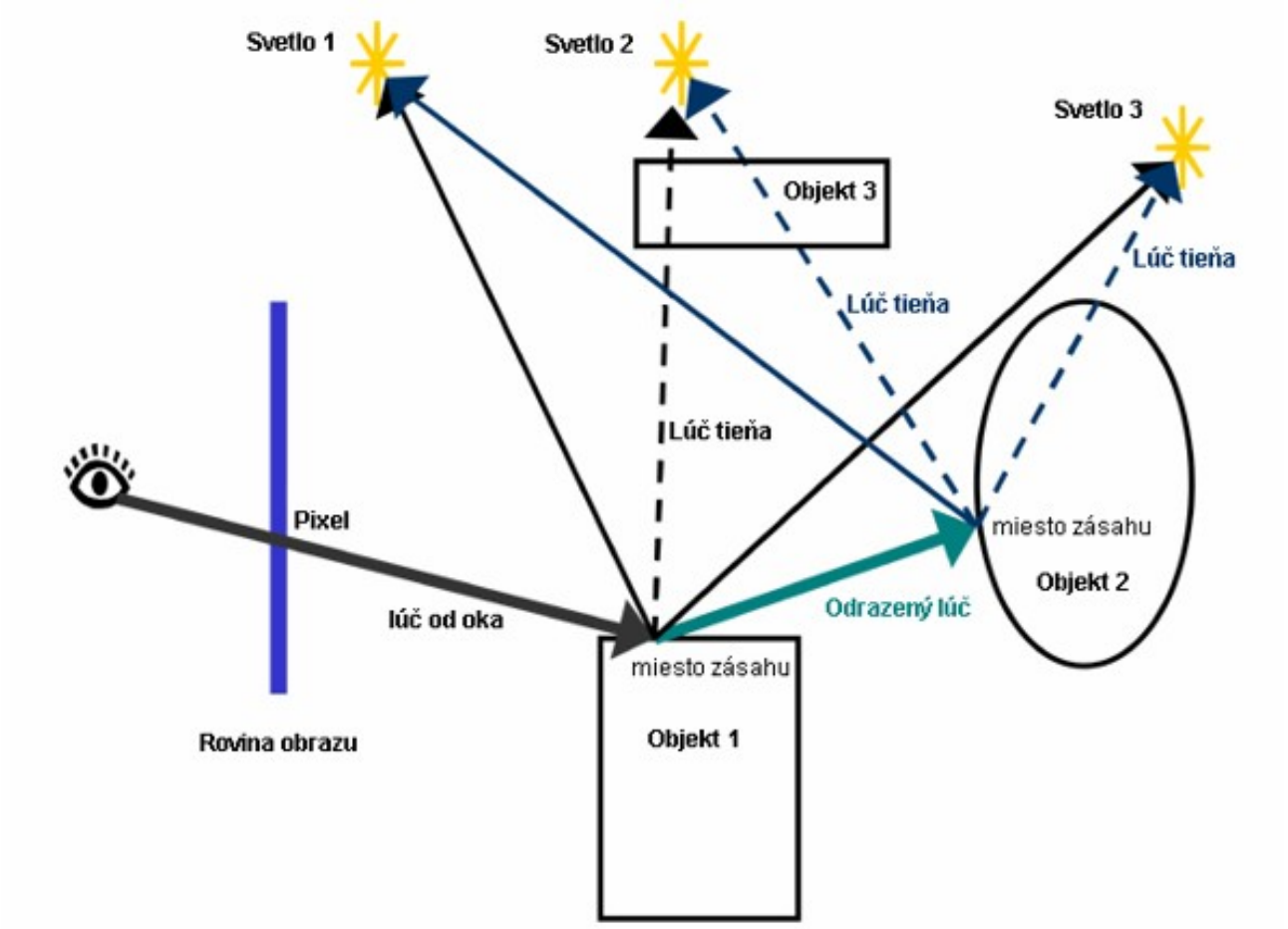
Michal Červeňanský
Juraj Starinský

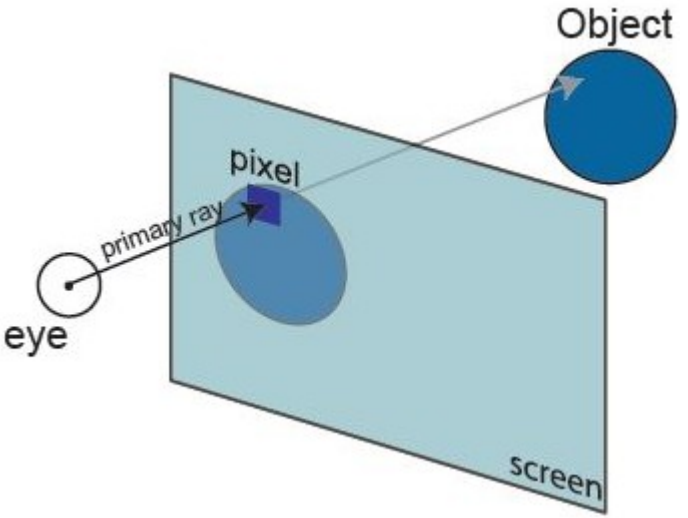
Overview

- Basics
- General method
- Data structures
- Algorithm
- Speed up techniques

Basics

- Travers ray through scene





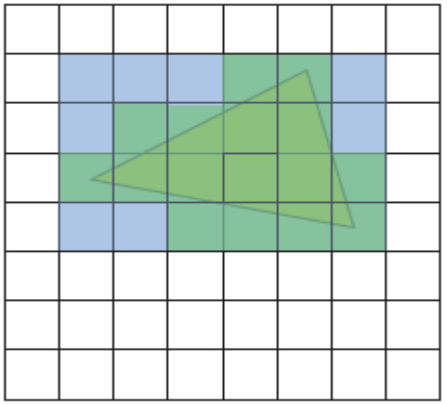
Streaming ray tracing

- GPU pipeline
  - Primary rasterization
- Problems
  - How to map ray tracing to GPU pipeline
  - Break up ray tracing into separate kernels
  - Kernels run as fragment programs
  - Uniform grid as acceleration structure
    - Various optimization

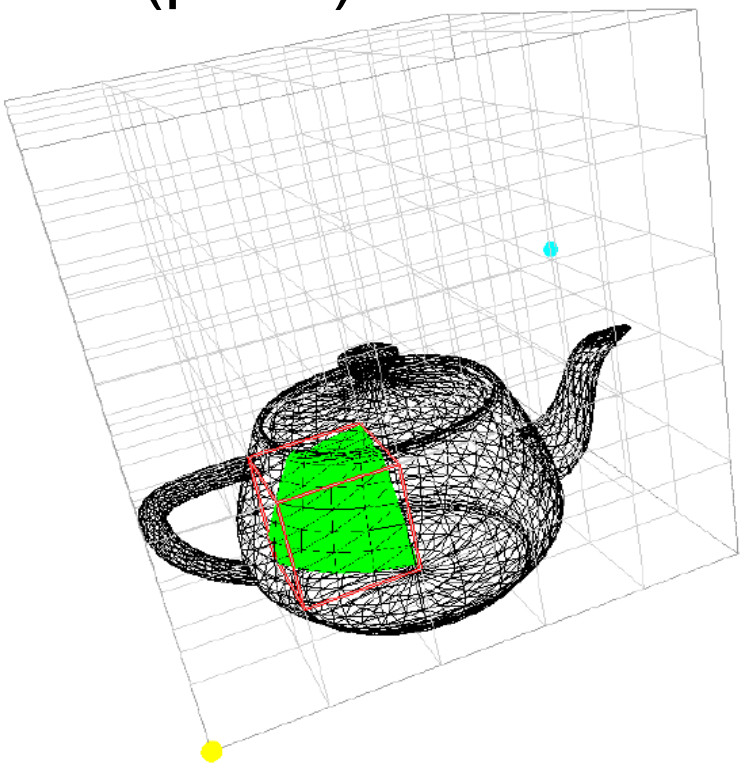
General Method

- Traverse acceleration structure
  - Cull away parts that ray cannot hit
  - Leaf nodes contain primitives
- Primitive intersection
  - Intersect ray directly
  - Return hit status to traversal
- Generate secondary rays from hit

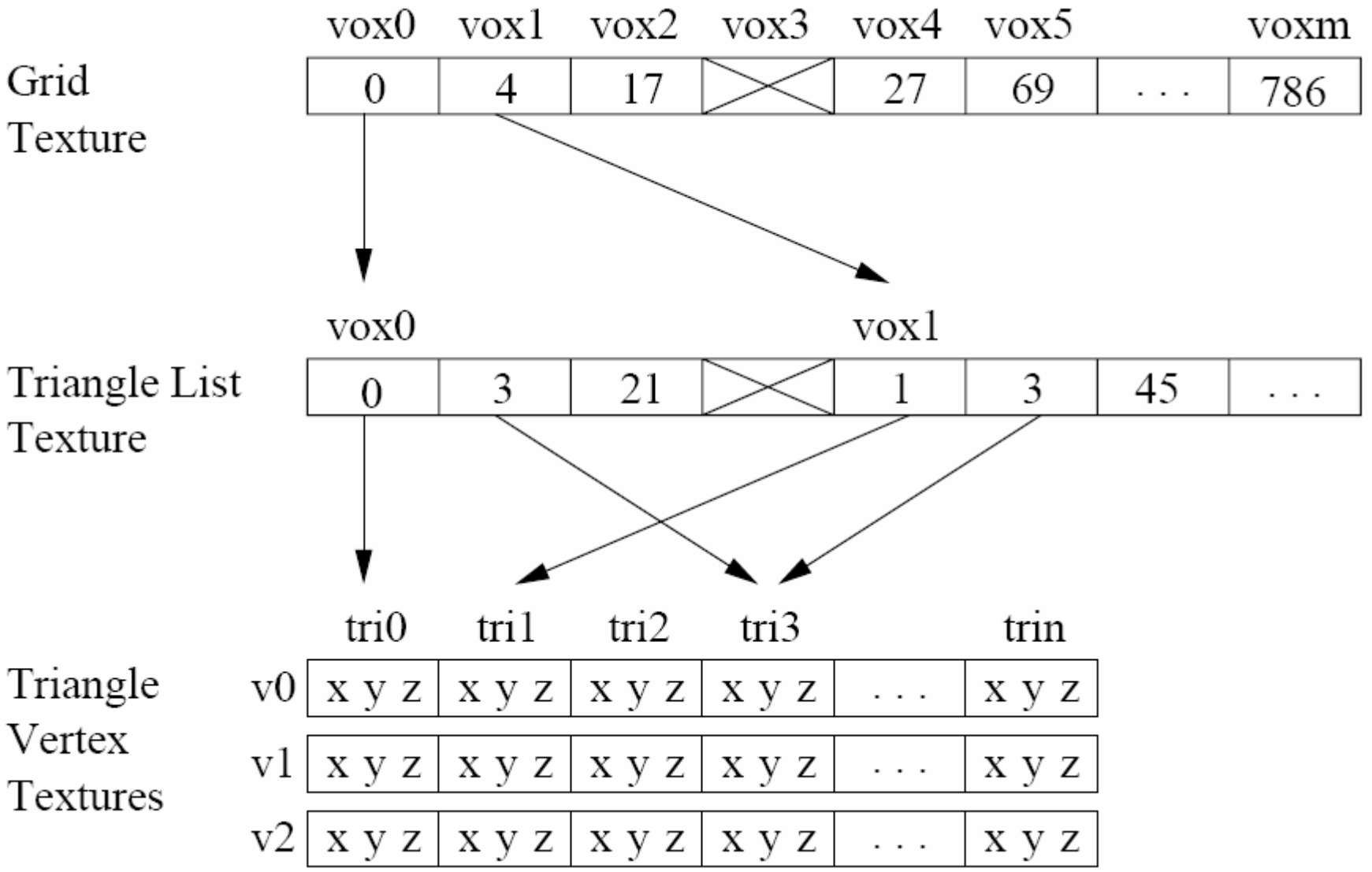
Acceleration structure



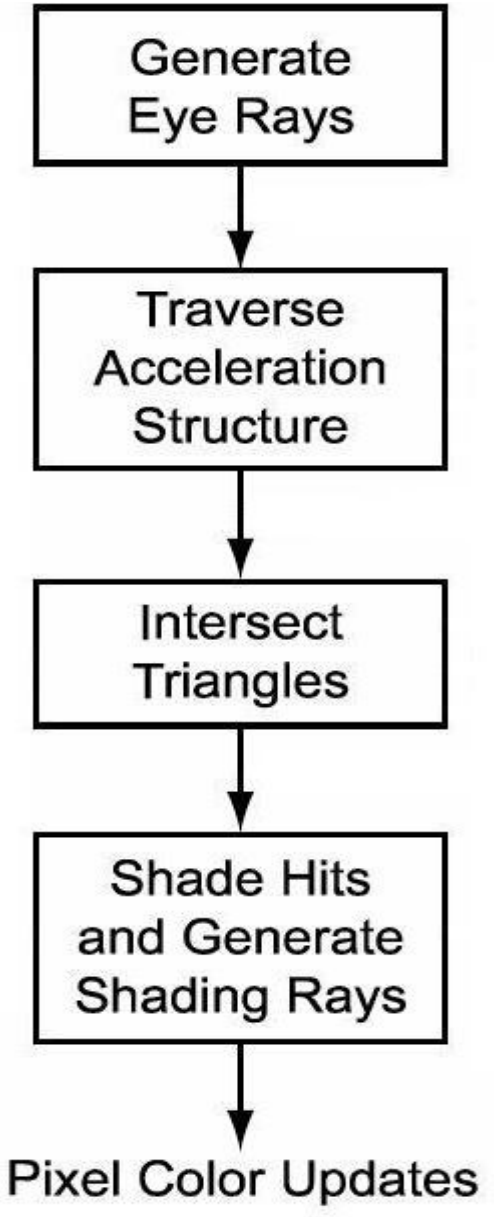
- Uniform grid
  - GPU friendly → 3D texture
  - Dependent fetches for lookup
  - Each voxel → several primitives (parts)
- Precomputed on CPU
  - Slow
  - Static scene
  - Triangles
  - #Triangles (resolution)



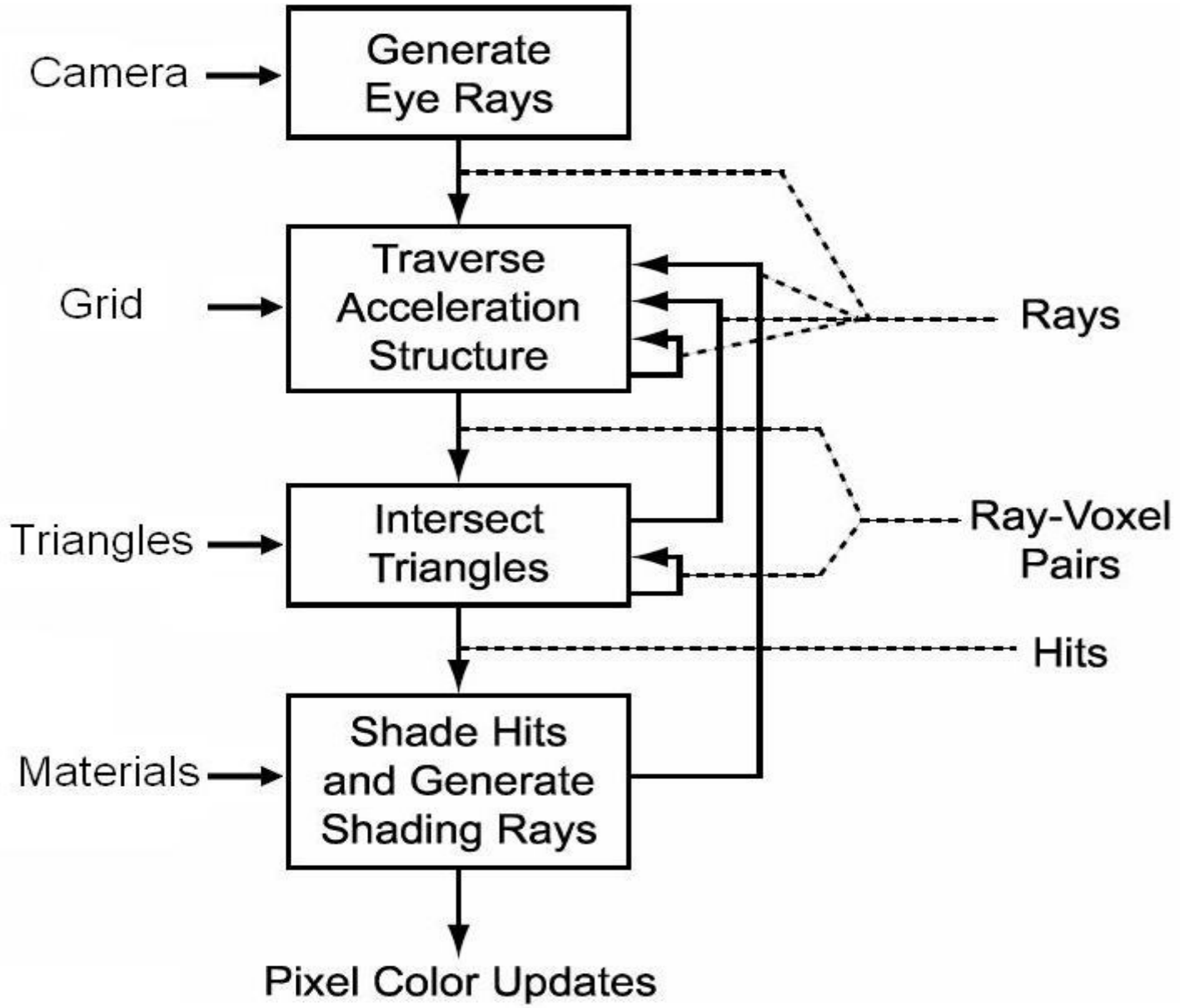
Uniform grid



Streaming scheme

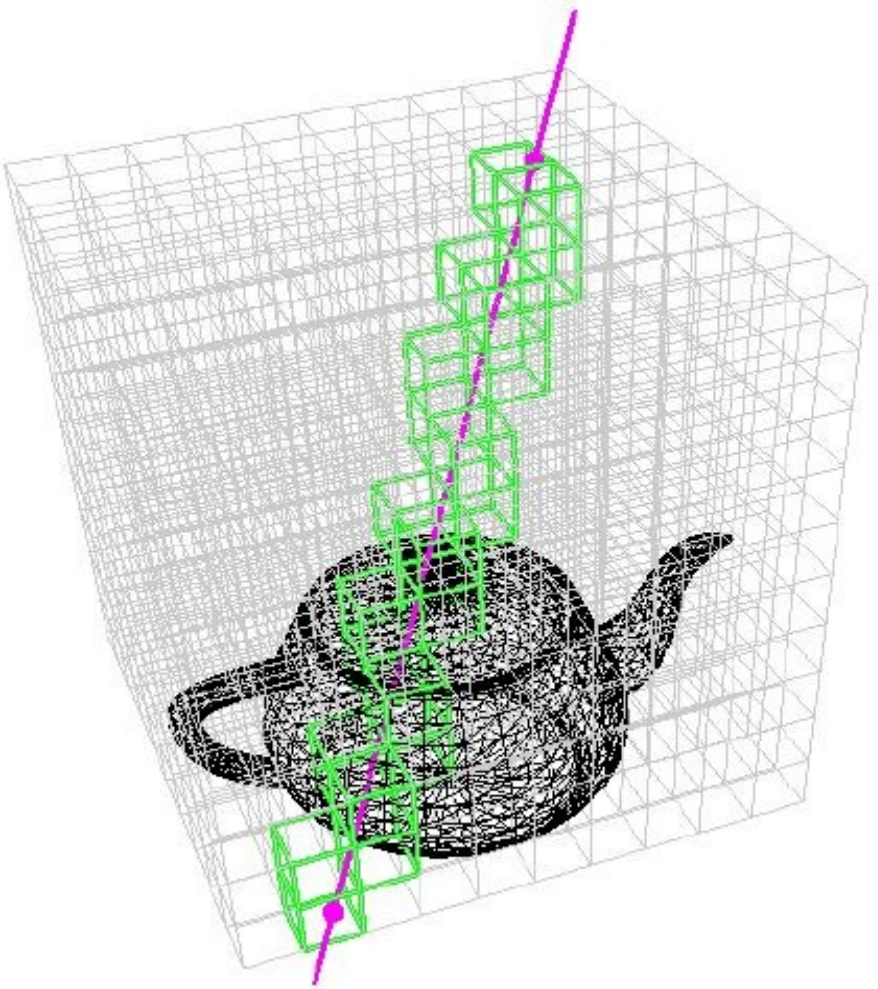


Streaming scheme

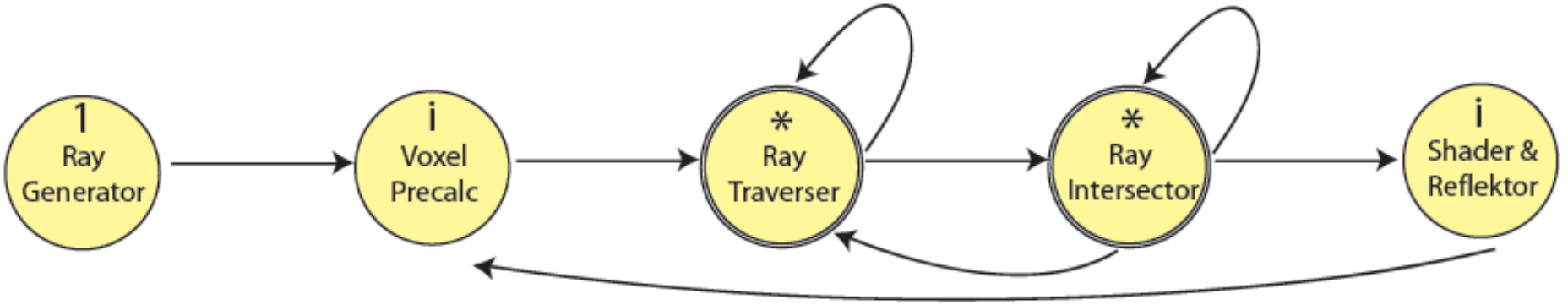


Traverse uniform grid

- 3D-DDA algorithm
  - Initialization
  - Incremental traversal
    - Non empty voxel
    - Out of grid

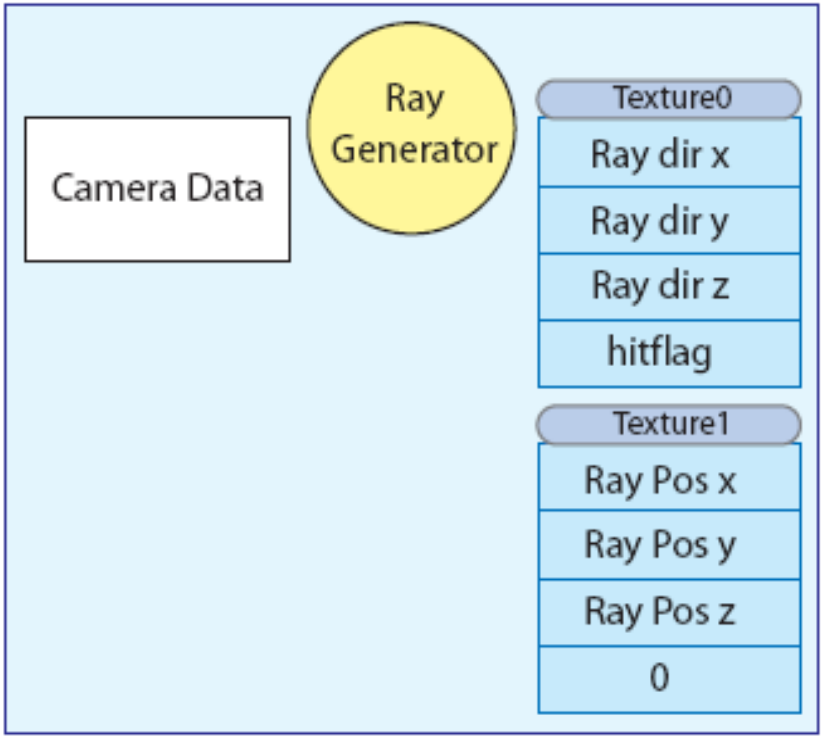


Algorithm


1
Ray
Generator
i
Voxel
Precalc
*
Ray
Traverser
*
Ray
Intersector
i
Shader &
Reflektor

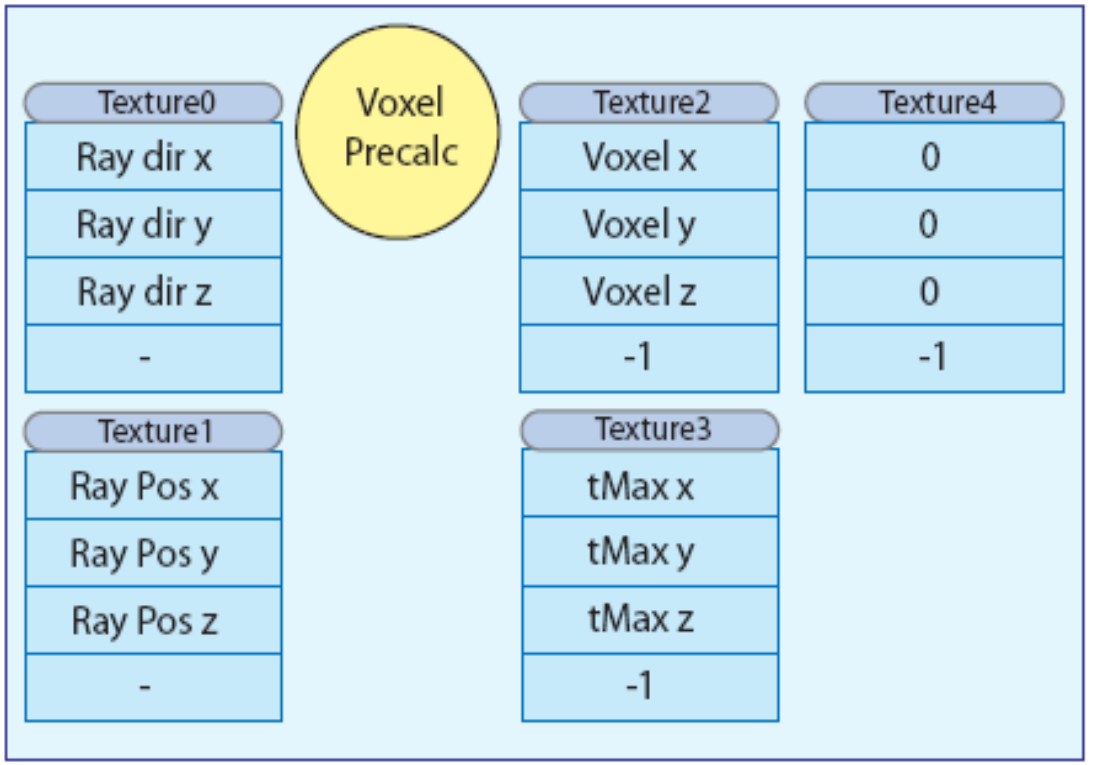
Algorithm
Ray Generation

- In – camera data
- Out – ray vector, Bbox intersection

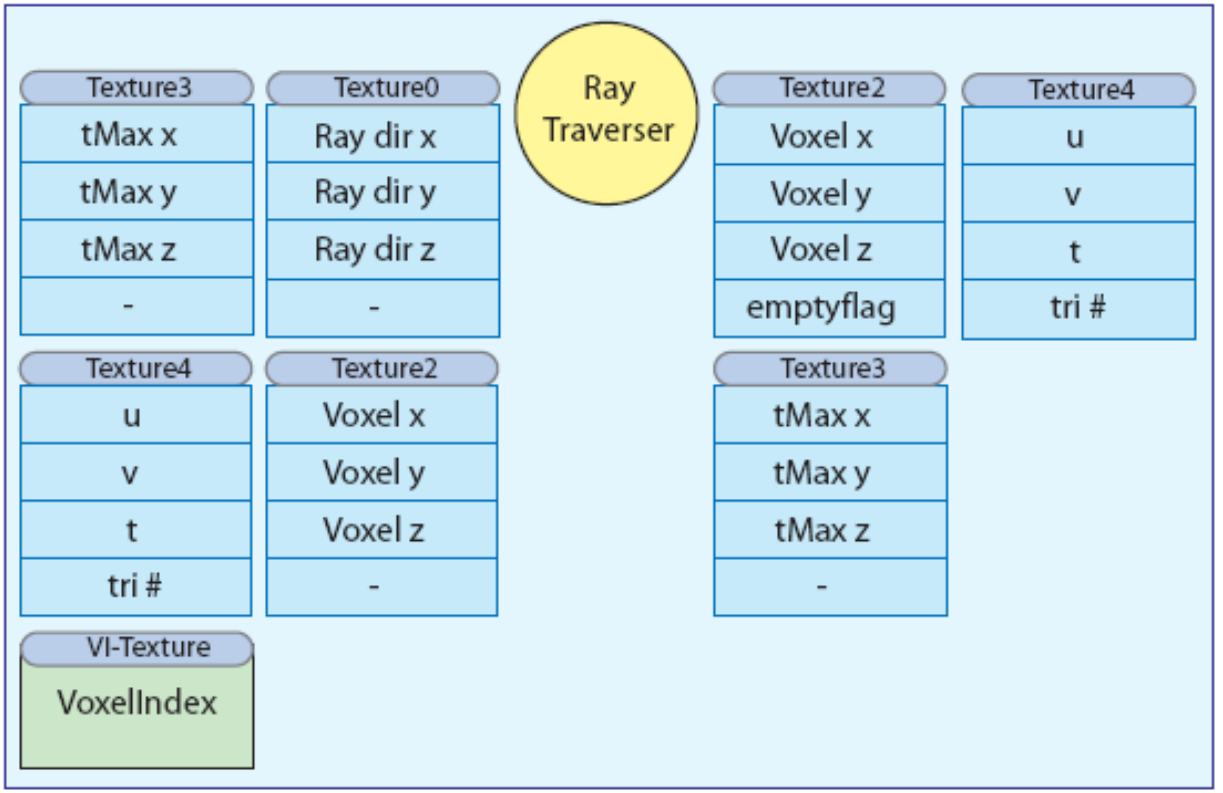


Algorithm
Voxel Precalc

- In – ray vector, position (world coord)
- Out – in/out position (grid coord)

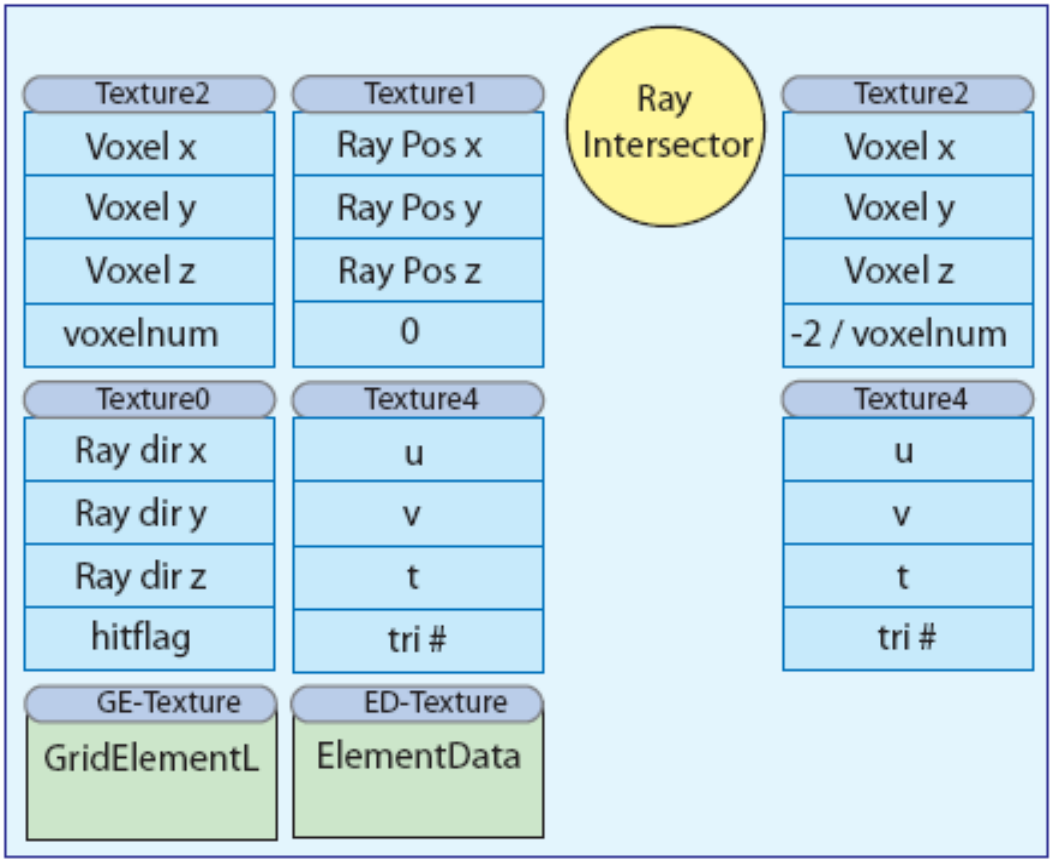


Algorithm
Ray Traverser

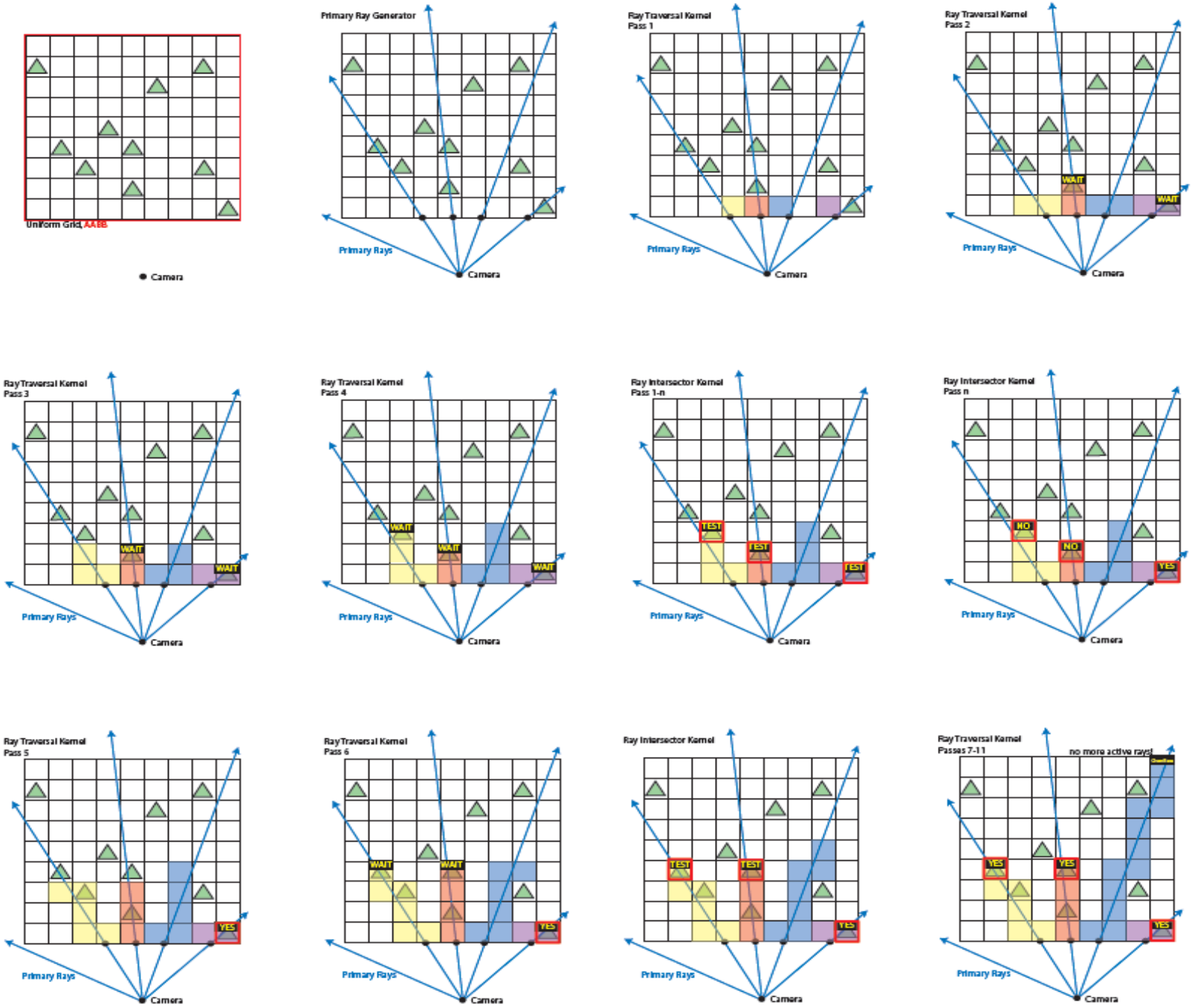


active	traverse Grid
wait	ready to check intersections
dead	ray doesn't hit grid (was already rejected in voxel precalculation)
inactive	a valid hit point was found
overflow	traversal left voxel space (no valid hits)

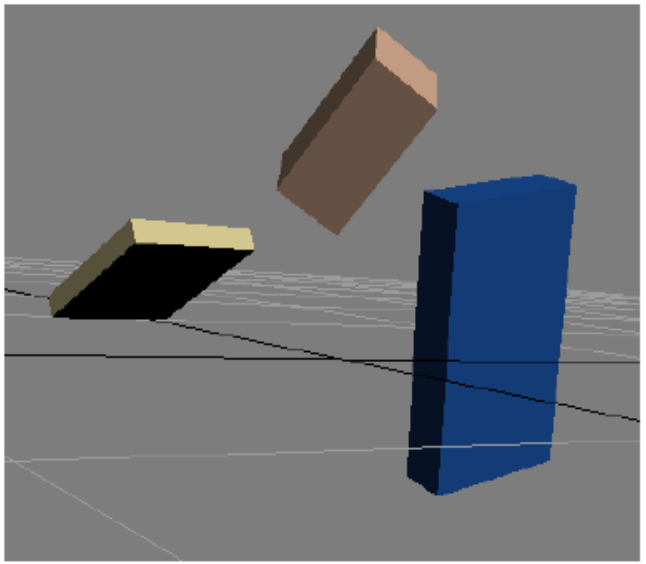
Algorithm
Ray Intersector

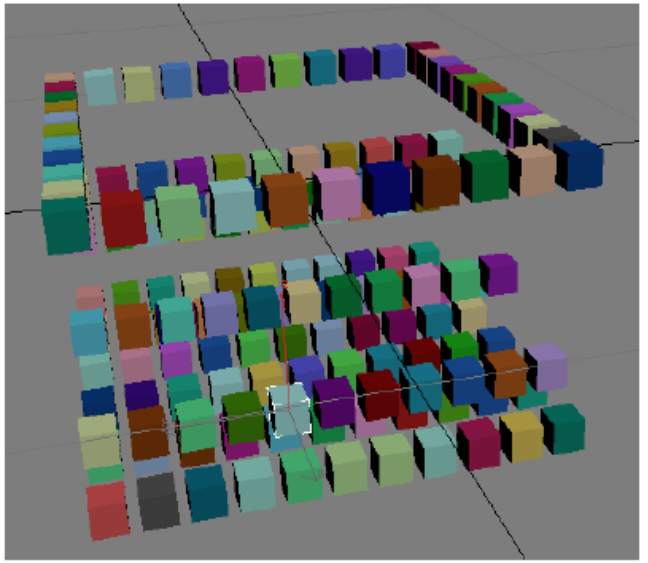


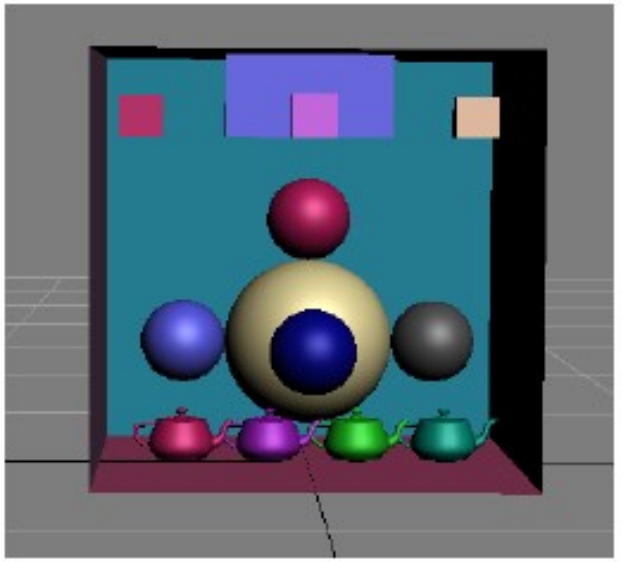
Traversing / Intersection loop

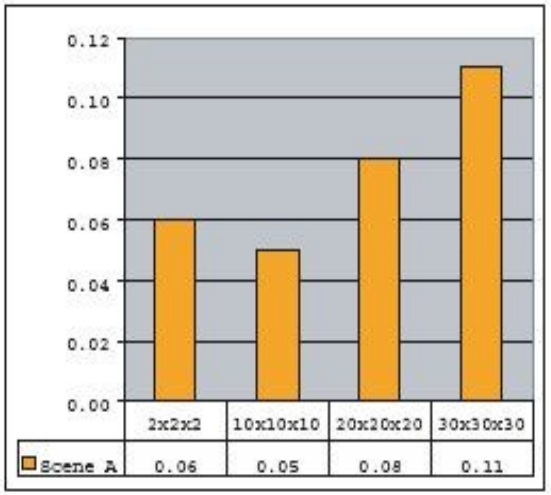


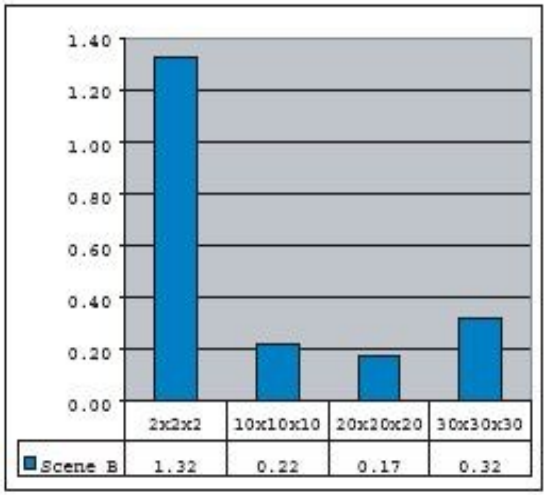
Grid size

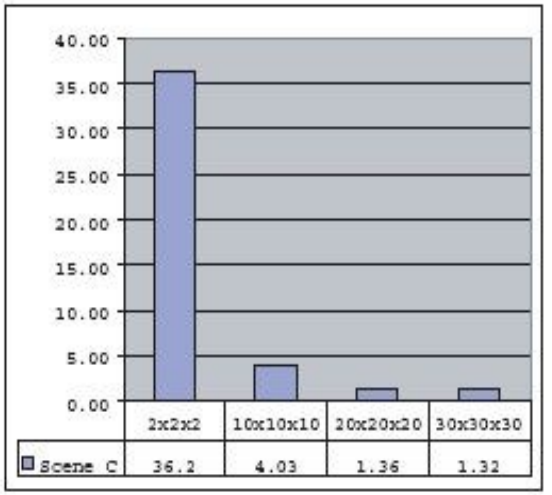







0.12
0.10
0.08
0.06
0.04
0.02
0.00
2x2x2
10x10x10
20x20x20
30x30x30
Scene A
0.06
0.05
0.08
0.11


1.40
1.20
1.00
0.80
0.60
0.40
0.20
0.00
2x2x2
10x10x10
20x20x20
30x30x30
Scene B
1.32
0.22
0.17
0.32

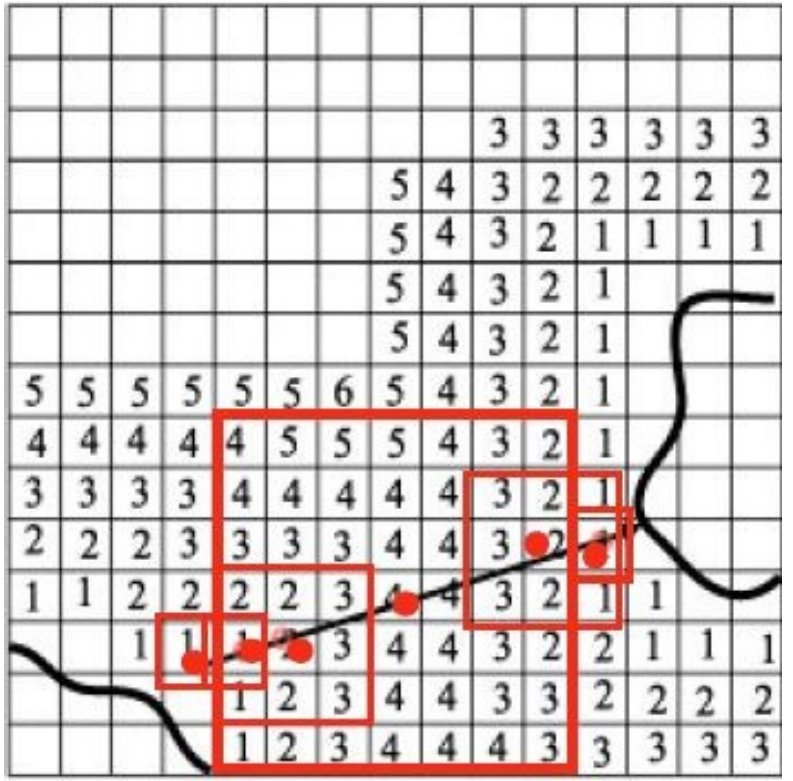

40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.00
0.00
2x2x2
10x10x10
20x20x20
30x30x30
Scene C
36.2
4.03
1.36
1.32

Disadvantages

- Multipass
  - Inefficient
  - #Texture read / write
- Uniform grid
  - Memory consumption
  - Geometry distribution

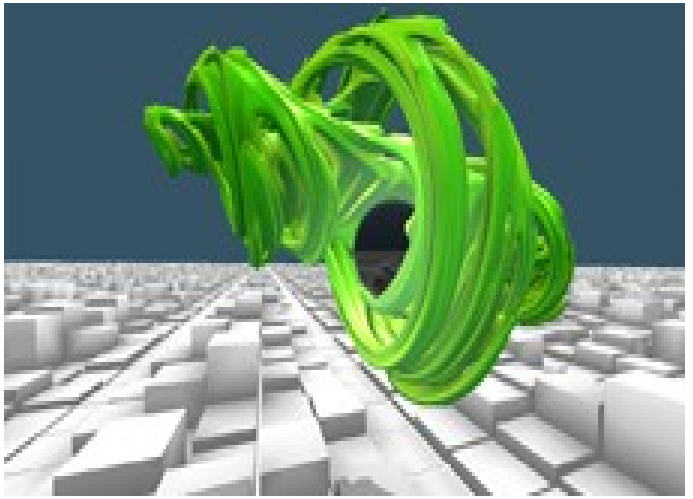
Speed up techniques

- Proximity clouds/distance fields
- BVH implementations
- kd-tree implementations



OptiX

- NVIDIA Optix
  - Interactive ray tracing
  - http://www.nvidia.co.uk/object/optix_uk.html



References

- Real-time rendering slides
- http://www.itnews.sk/buxus_dev/generate_page.php?page_id=50072
- http://news.cnet.com/crave/?keyword=Ray+Tracing
- http://www.clockworkcoders.com/oglsl/rt/
- http://graphics.stanford.edu/papers/rtongfx/rtongfx.pdf
- http://gamma.cs.unc.edu/GPGP/lectures/RTGPU.ppt
- http://graphics.stanford.edu/papers/i3dkdtree
- http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.103.3319&rep=rep1&type=pdf

Questions ?