Real Time Ray Tracing of Point-based Models

Sriram Kashyap | Rhushabh Goradia | Parag Chaudhuri | Sharat Chandran

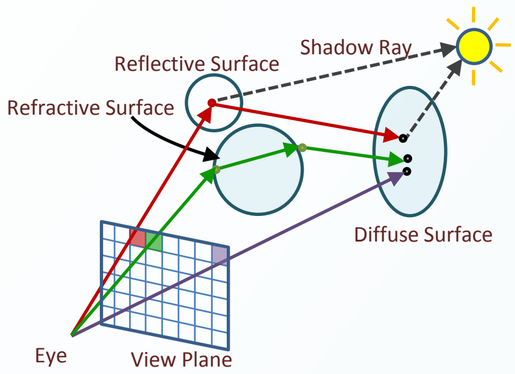
ViGIL, Department of Computer Science & Engineering, Indian Institute of Technology Bombay

{kashyap,rhushabh,paragc,sharat}@cse.iitb.ac.in; www.cse.iitb.ac.in/graphics

i3D

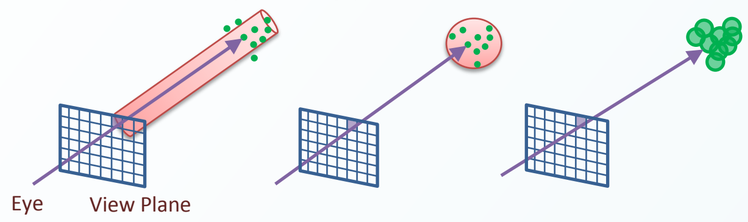
Problem Statement

Ray trace in real time a point model scene, with position, material and normal at each point



Prior Work

Techniques for ray tracing point models



Thick Rays [1] Implicit Surfaces [2] Splats [4]

Contributions

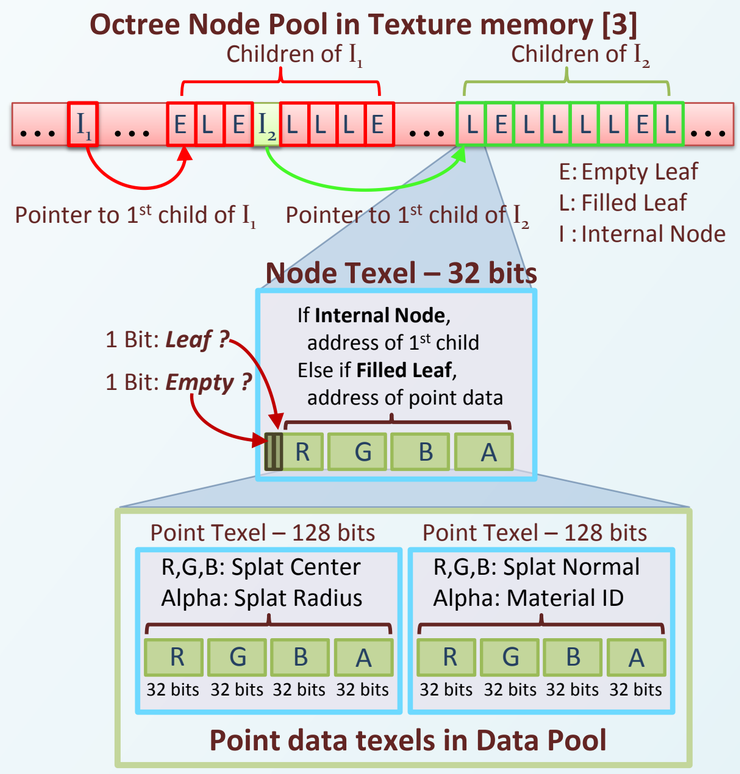
Leveraging the parallelism on the GPUs, we present a **real time point -based ray tracer** by means of

- **An efficient representation for octrees and point data as 1D textures on the GPU**
- **A fast ray traversal primitive**
- **A technique to reduce memory footprint by culling redundant splats**

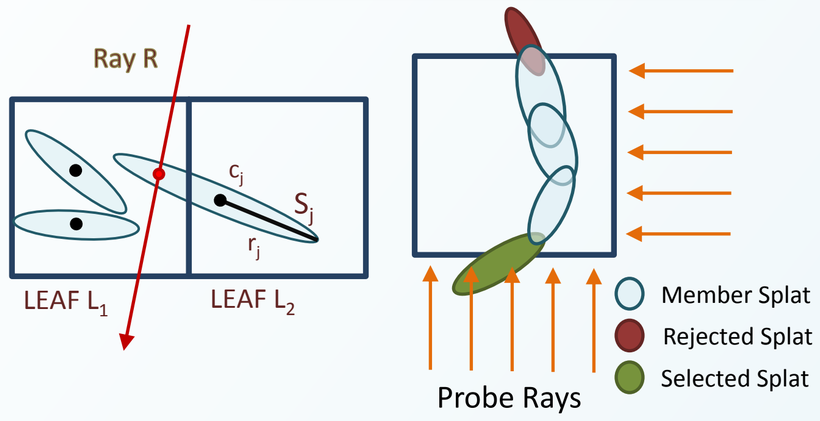
Overview

- Construct octree for the point model (on CPU)
- With primary rays in parallel (on GPU) :
  - Traverse octree & check for ray-splat intersections
  - Send shadow rays and perform local shading
  - Send secondary rays (reflect/refract)

Representation



Memory Footprint



Problem: Ray R intersects splat S_j, but center of S_j in not in leaf L_1

Solution: *Splat Replication* [4]

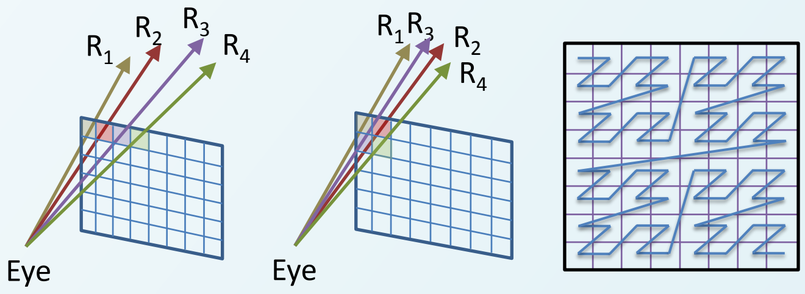
a. Initially, each leaf contains only member splats (splats whose centers intersect the given leaf)
b. Subsequently, each leaf contains additional external splats (splats whose extents intersect the leaf)

Problem: Splat replication increases the memory footprint

Solution: *Memory Footprint Reduction*

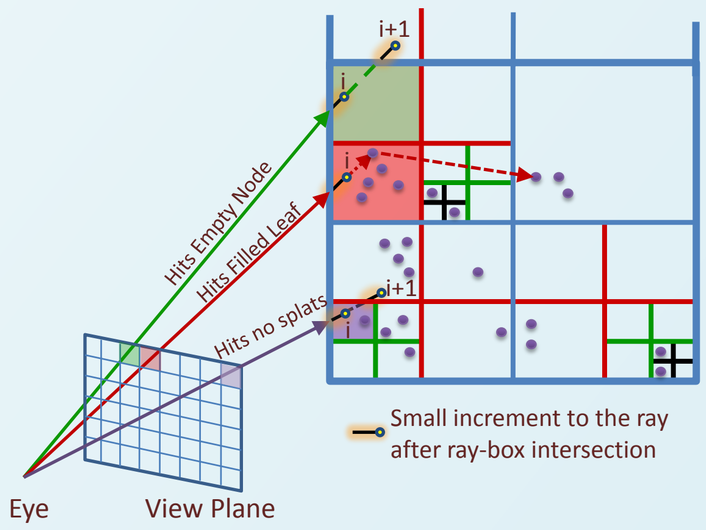
a. Send probe rays in various directions
b. If a probe ray does not intersect member splats, check intersection with external splats
c. Retain only those external splats that are hit by probe rays

Ray Coherence



- Ray coherence is achieved using Z-order space filling curve
- Better than linear scaling with increase in resolution due to reduced warp divergence on the GPU

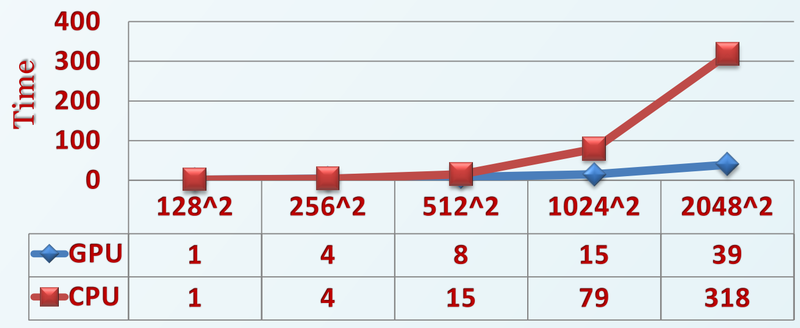
Ray Traversal



Results

System : 1.86GHz Intel Core 2 Duo, nVidia GTX 275
Models: David, Dragon, Buddha (chosen to enable quality comparisons)

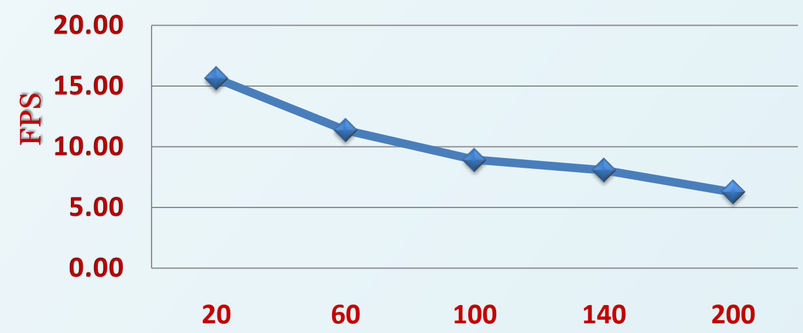
Variation in Time with Resolution (David, 10^6 Points)



	128^2	256^2	512^2	1024^2	2048^2
GPU	1	4	8	15	39
CPU	1	4	15	79	318

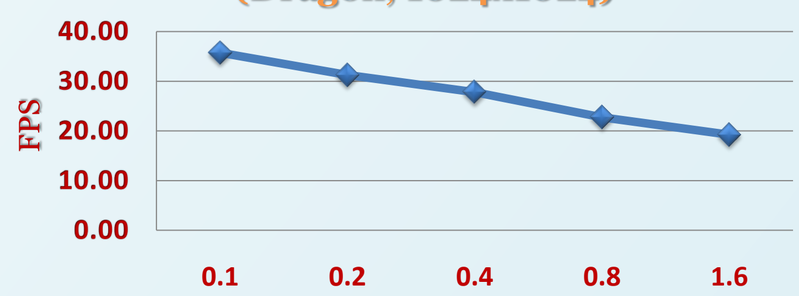
Y-axis: Render time scaling baseline: 128x128

Variation in FPS with Octree Depth (David, 10^6 Points, 1024x1024)



X-axis: Maximum number of points per octree leaf

Variation in FPS with Model Size (Dragon, 1024x1024)



X-axis: Input model size in millions

FPS comparison with [2] (at 512x512)

	Model	Size	FPS			
		millions of points	Local Illumination	With Shadows	Reflection (4 bounce)	Refraction (8 bounce)
Our Results	David	3	80	55	20	17
	Dragon	1.3	114	85	30	21
Wald and Seidel [2]	David	1	10.6	4.1	N/A	N/A
	Dragon	1.3	75	5.7	N/A	N/A

Memory Footprint Reduction

Model	Size (millions of points)	Linsen et al. [4] After Replication	[Our Method] After Replication	Additional Preprocess Time (secs)
Museum	3.75	39	8	30
Buddha	1.3	13	2.4	10
David	1	13	2	9
Dragon	0.4	3.3	1.3	4

References

1. Schaufler G., and Jensen, H.-W. 2000. Ray Tracing Point Sampled Geometry. In *Proceedings of the Eurographics Workshop on Rendering Techniques*, 319-328
2. Wald, I., and Seidel, H.-P. 2005. Interactive Ray Tracing of Point Based Models. In *Proceedings of Symposium of Point Based Graphics,* 9-16
3. Lefebvre S., Hornus S., and Neyret F. 2005. *GPU Gems 2.* Addison Wesley, Ch. Octree Textures on GPU, 595-614
4. Linsen, L., Muller, K., and Rosenthal, P. 2007. Splat-based Ray Tracing of Point Clouds. In *Journal of WSCG*, 51-58