Fast Geometric Computations using GPUs

김영준

http://graphics.ewha.ac.kr

이화여자대학교

컴퓨터학과

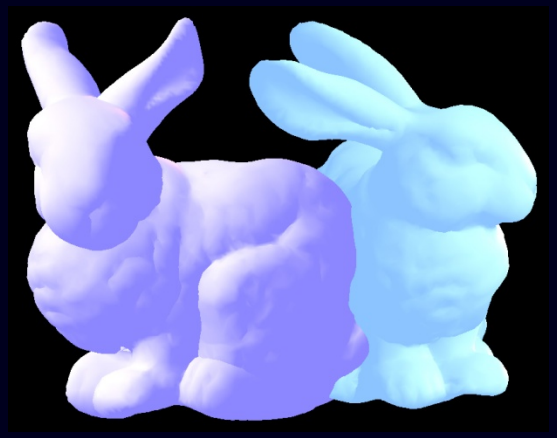
Topics

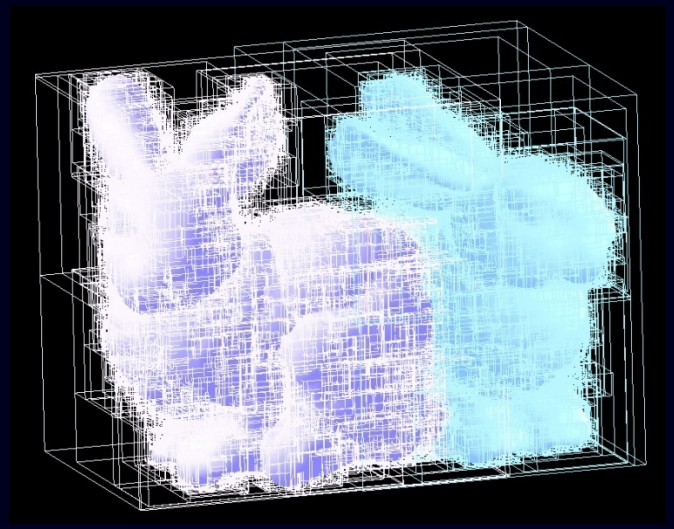
- Collision detection
- Closest point query
- Approximate arrangement computation

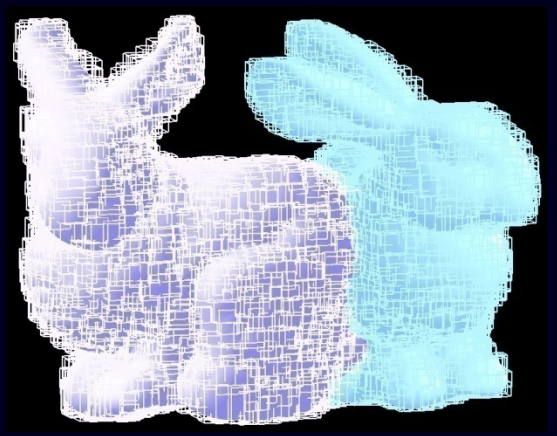
Streaming AABBs

Collision detection for deformable models

Teaser Video

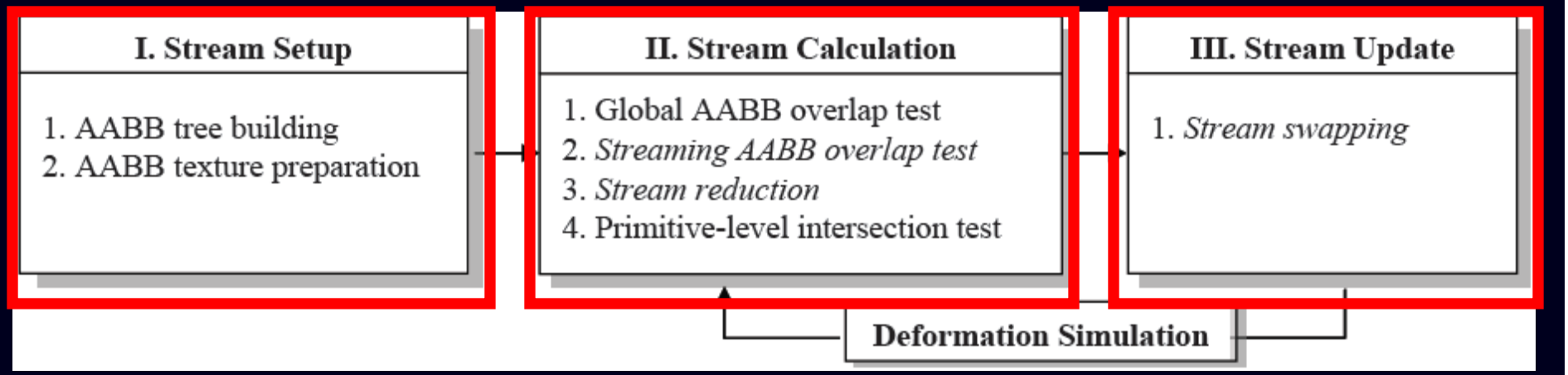




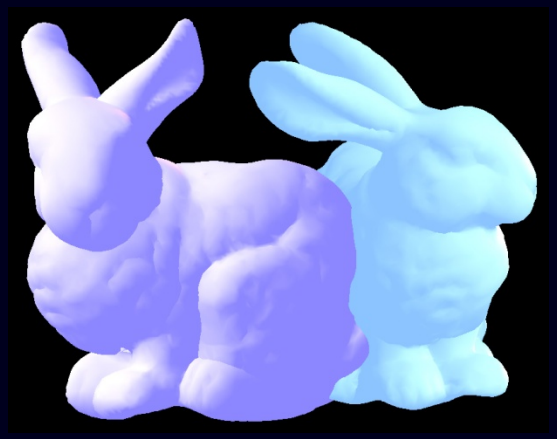


Streaming AABB Pipeline

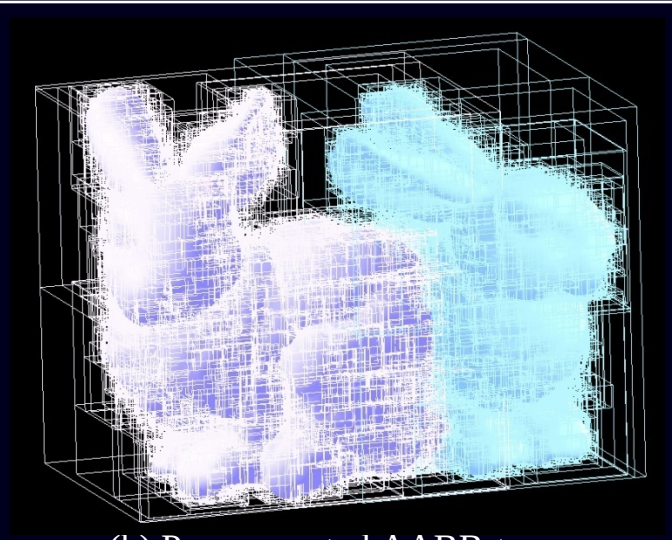
1. Stream setup
2. Stream calculation
3. Stream update



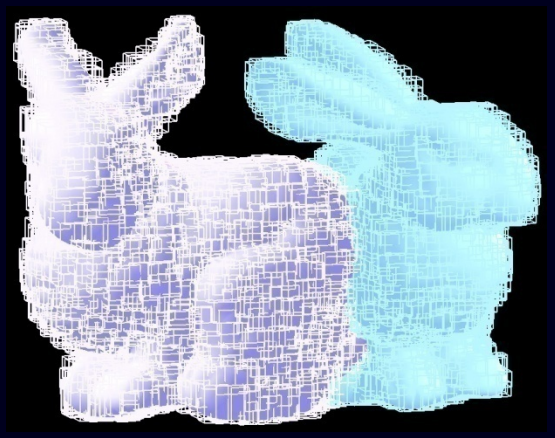
Streaming AABBs



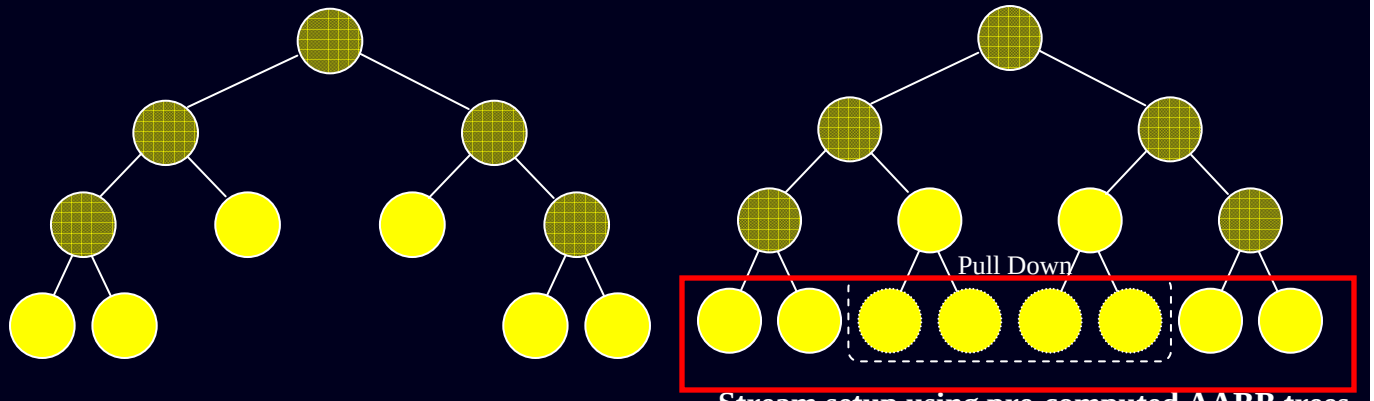
(a) Intersecting bunny models



(b) Pre-computed AABB-trees

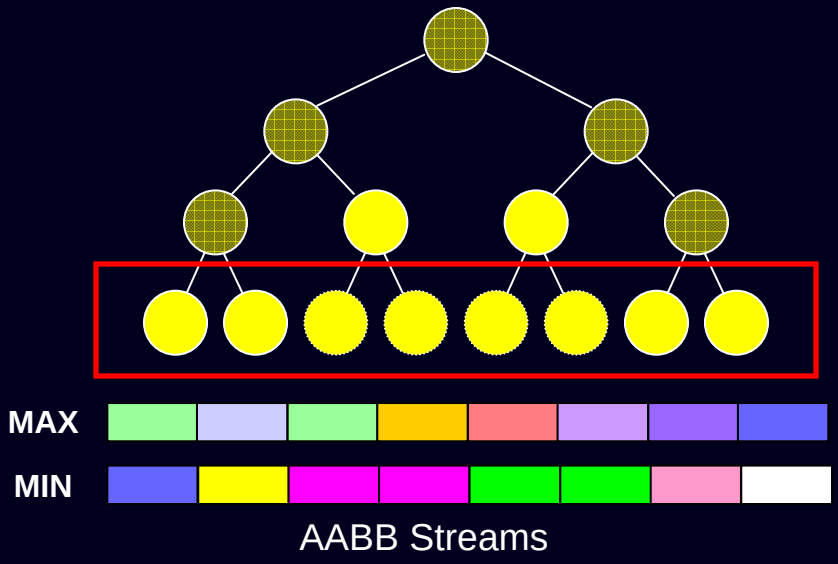


(c) Intersecting AABB streams



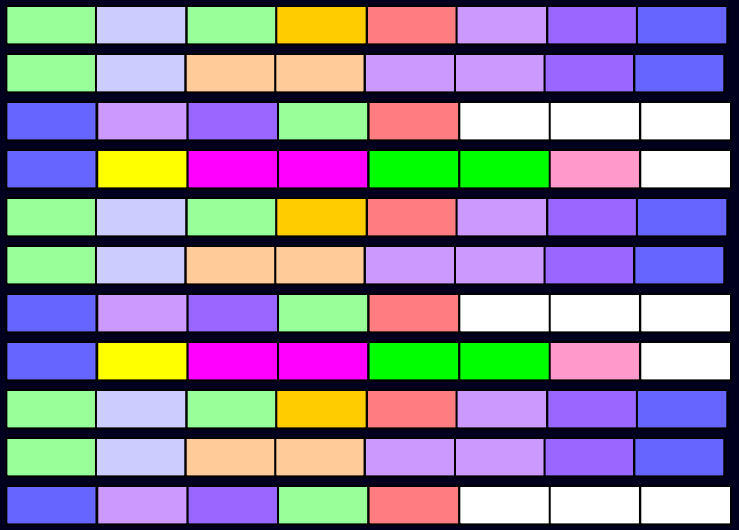
Stream setup using pre-computed AABB trees

Streaming AABBs

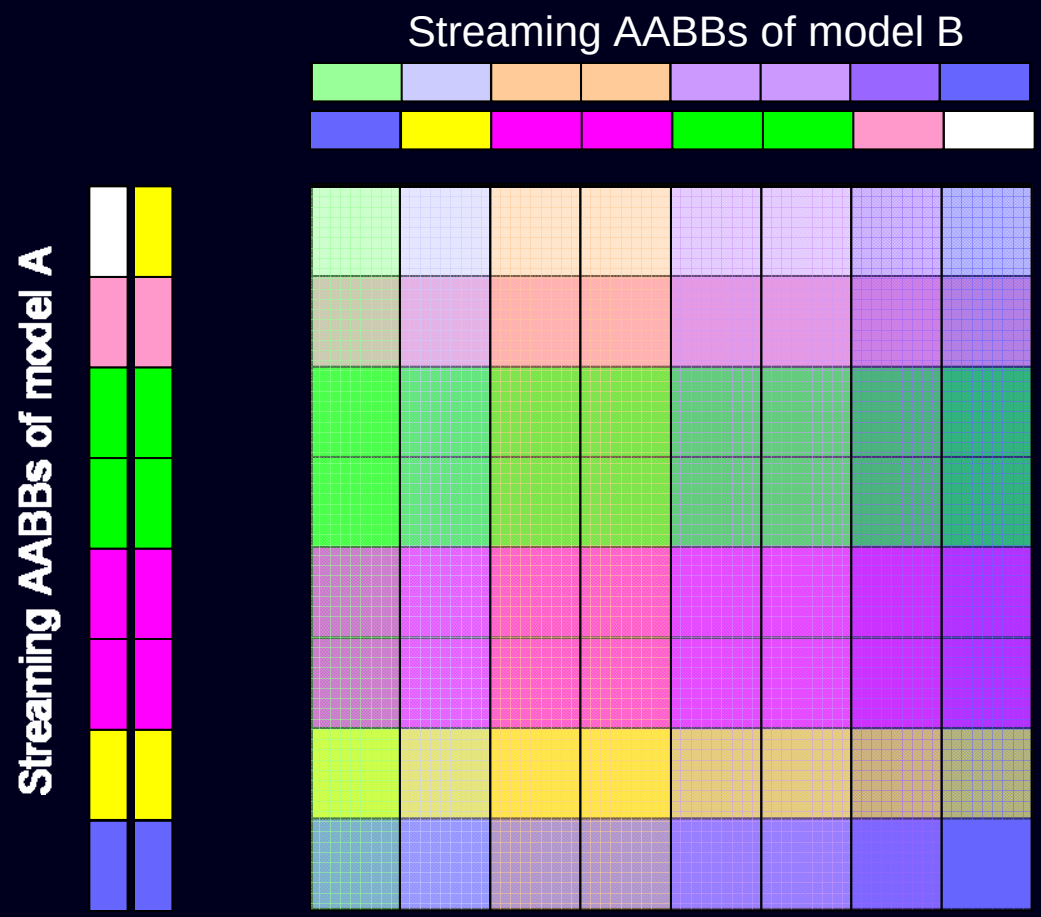


AABB Streams

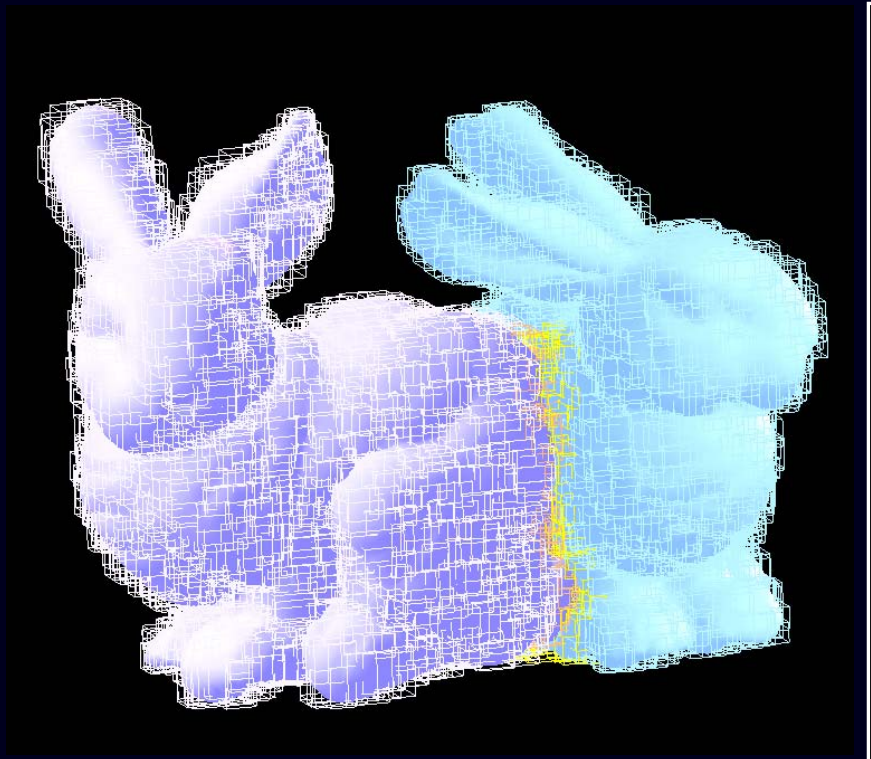
Vertex textures



Streaming AABB Overlap Test



Streaming CD Results



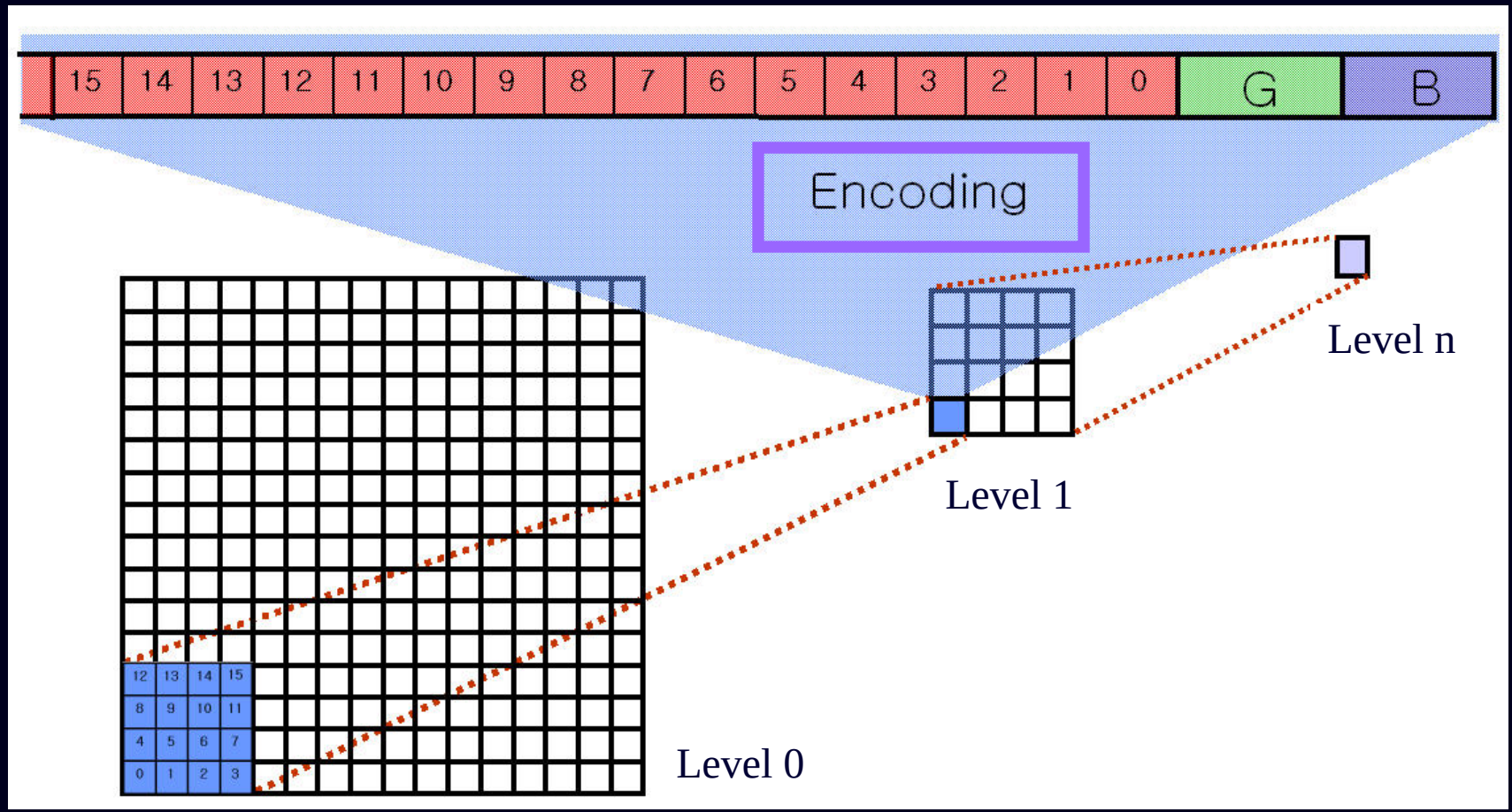
Intersected models

Collision result of AABB pairs stored at an off-screen buffer

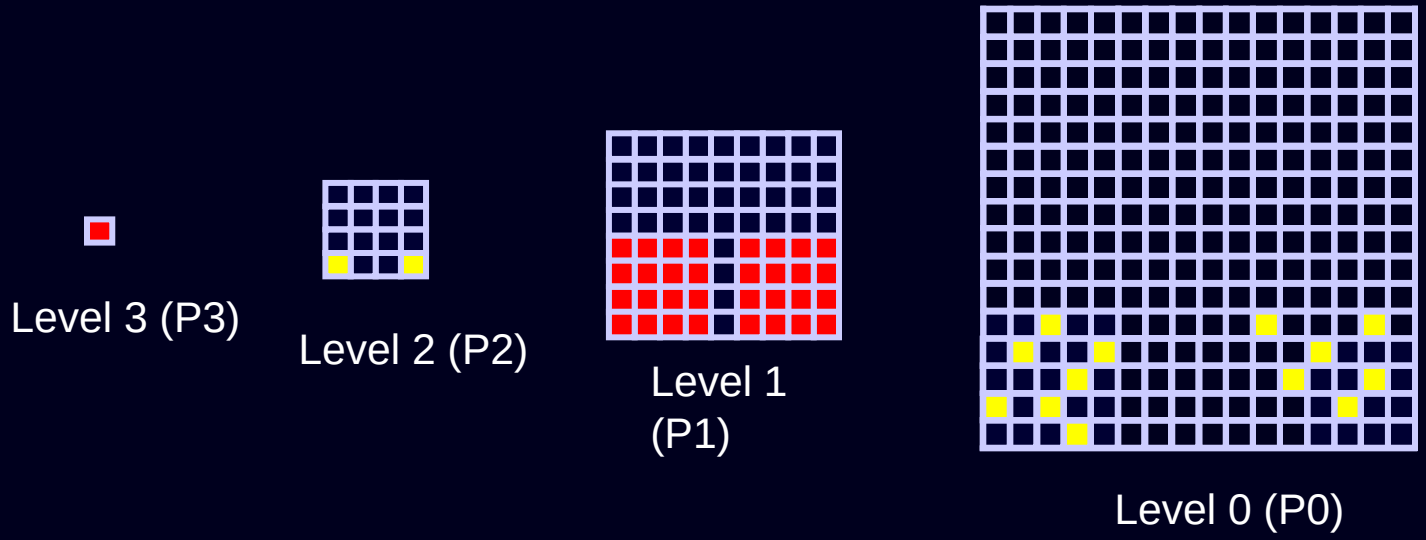
Readback from GPU to CPU

- Readback from GPU to CPU is costly
  - 50 milli-seconds for 1024x1024 frame buffer
- Hierarchical readback

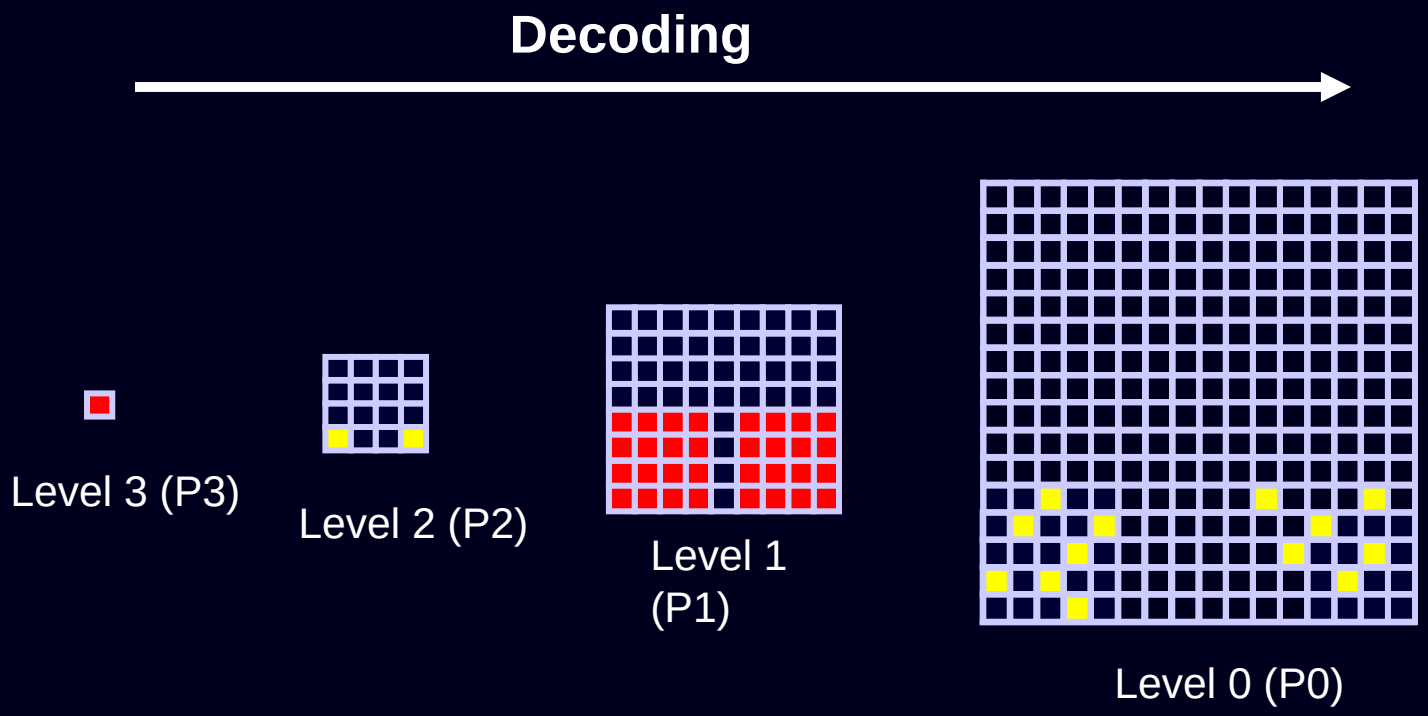
Stream Reduction



Readback



Readback

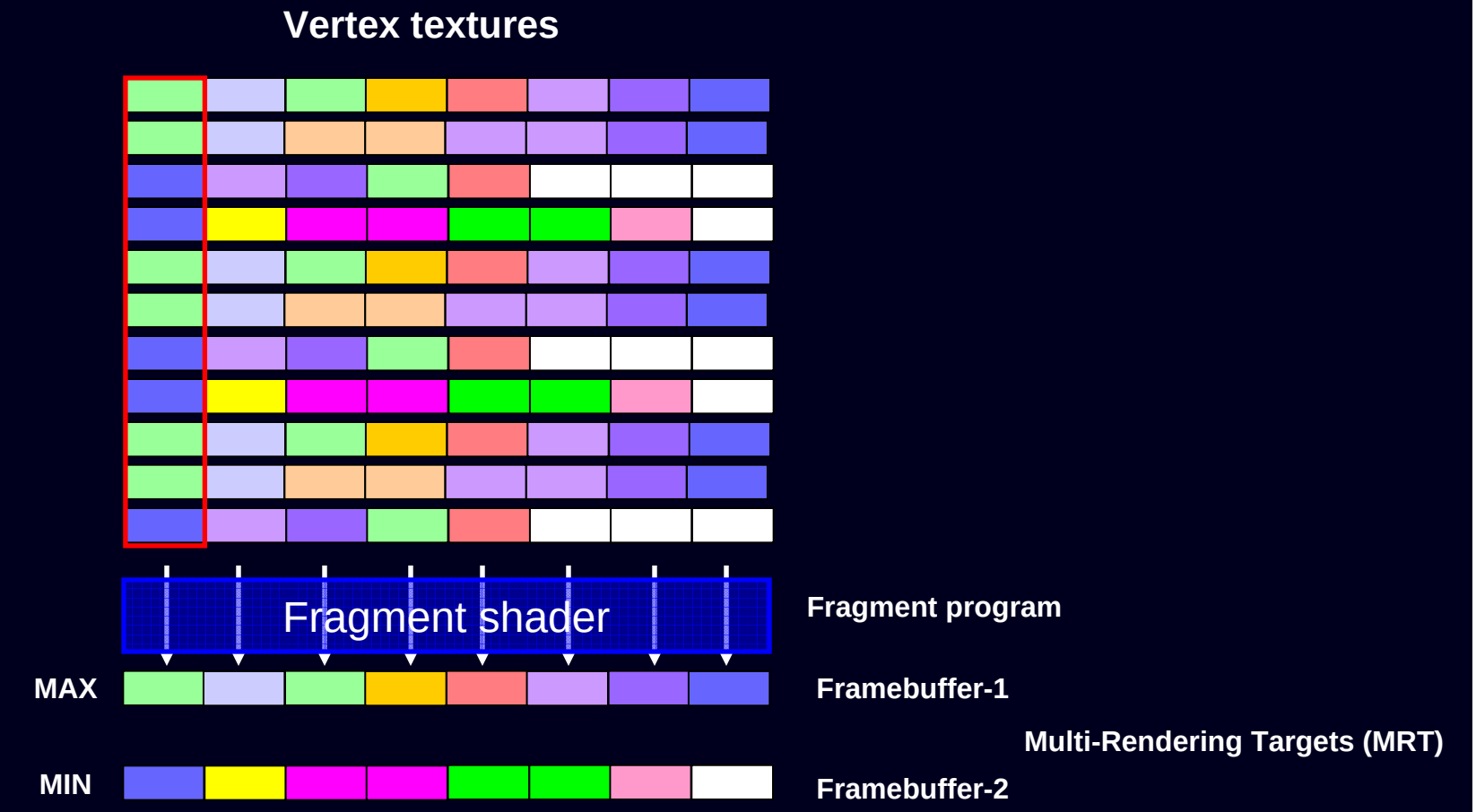


Time (encoding) + Time (decoding) << Time (direct readback) **Video**

Primitive Level Test

- Triangle-triangle overlap test using CPUs

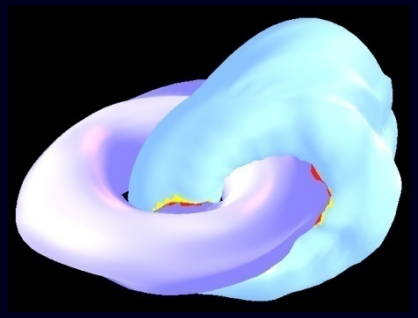
Stream Update



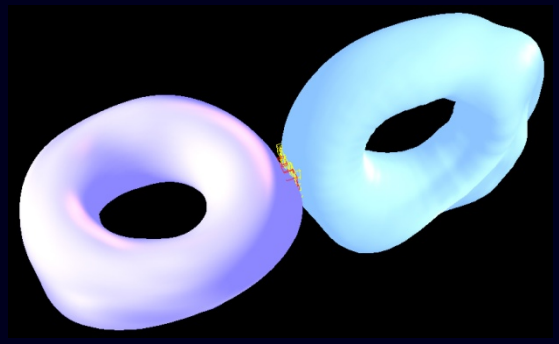
Implemetation

- Streaming computation in GPUs
  - nVIDIA GeForce 7800 GTX GPU (PCI-E)
  - 512 MB video memory
  - 32-bit floating point
- Serial computation in CPU
  - 3.4 GHz Intel Dual Core Processor,
  - 2.7 GB main memory
- Languange
  - C++ in Windows
  - nVIDIA's Cg (vp40 and fp40 profiles)
  - OpenGL 2.0

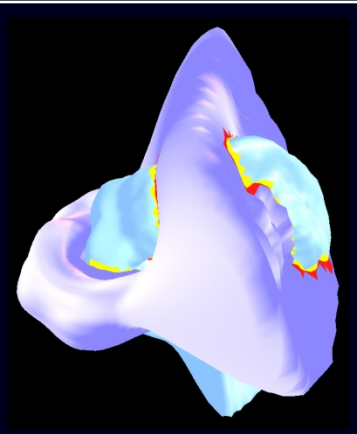
Benchmarking Scenarios



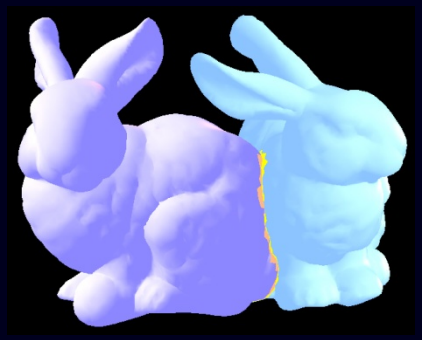
(a) Interlocking torii
(15000x2 triangles, 60-80 FPS)



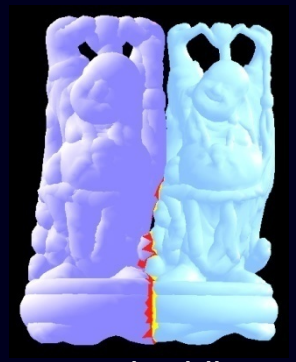
(b) Touching torii
(15000x2 triangles, 90-100 FPS)



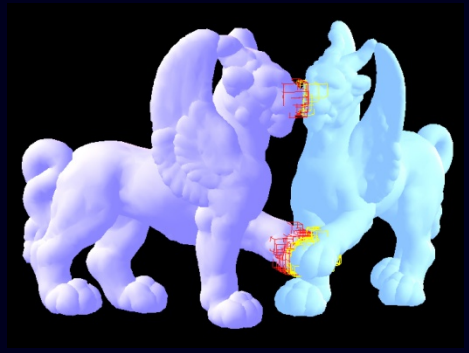
(c) Merging torii
(15000x2 triangles, 25-30 FPS)



(d) Bump bunnies
(15000x2 triangles, 50-60 FPS)



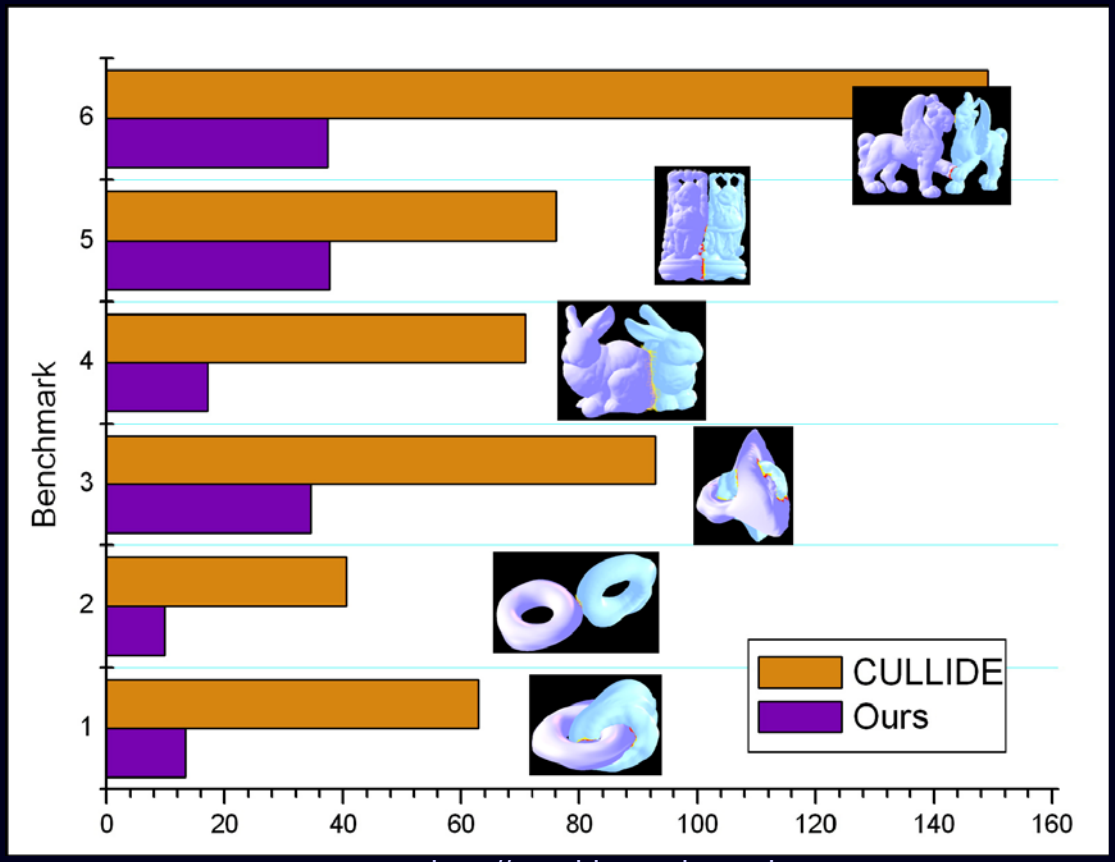
(e) Happy buddhas
(20000x2 triangles, 25-40 FPS)



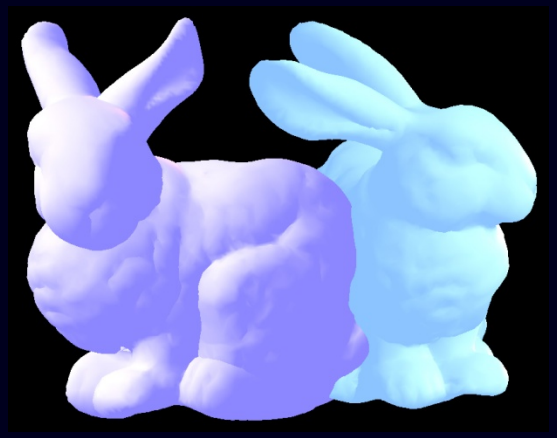
(f) Intimate animals
(50000x2 triangles, 20-30 FPS)

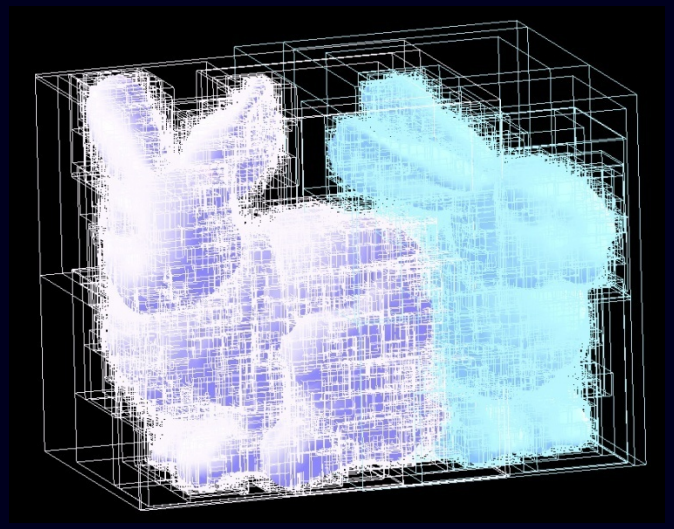
Performance Comparison

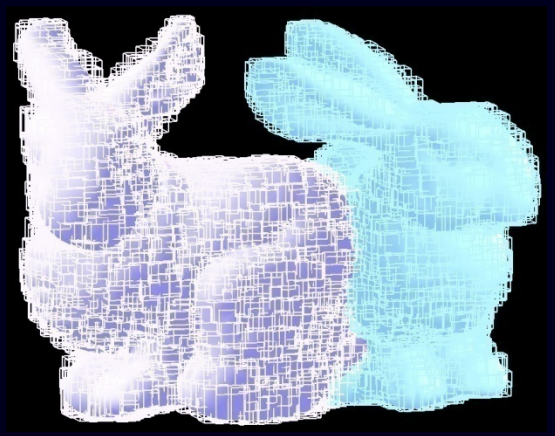
- Three times performance improvement over CULLIDE
- No collision misses up to floating point precision



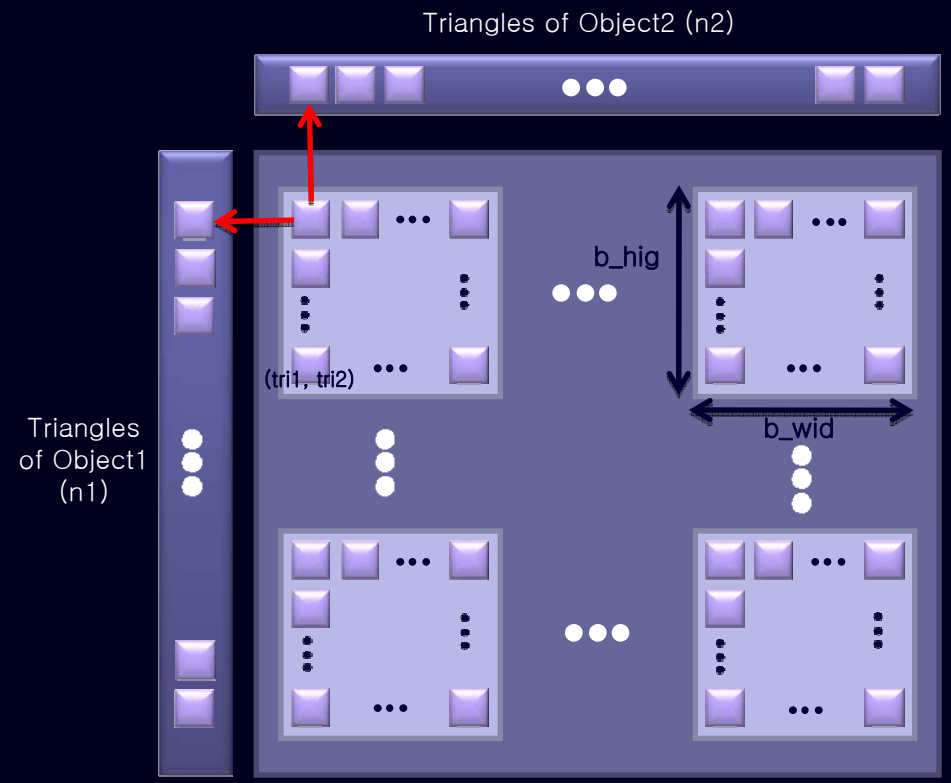
Demo Video







Parallel Triangle Intersection Test Using CUDA



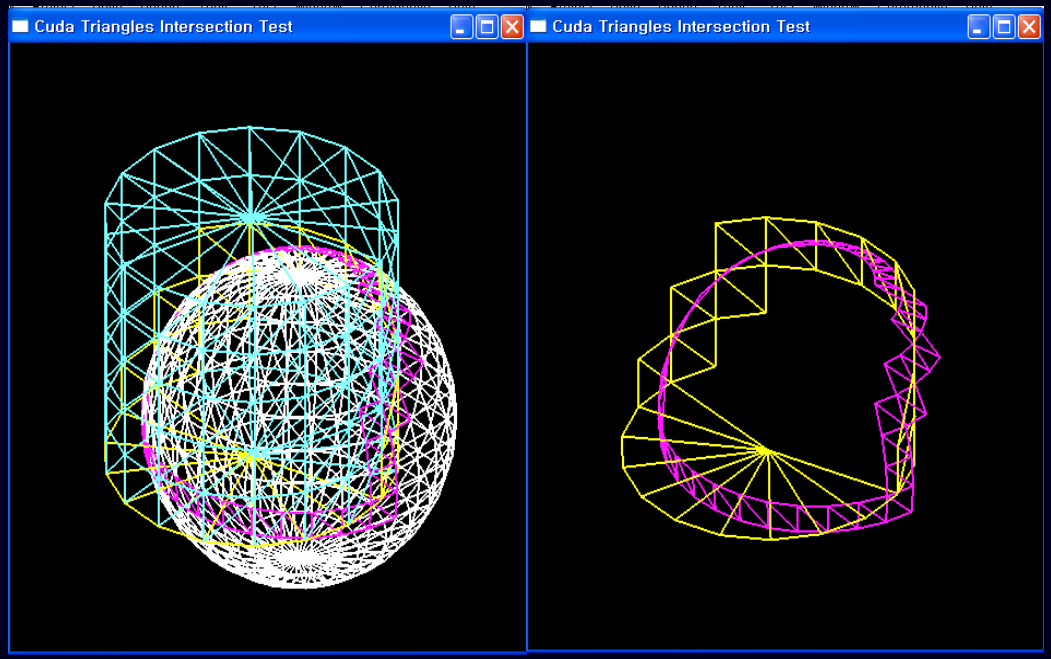
Implementation

- Environment
  - Microsoft Visual Studio 2005
  - CUDA Toolkit/SDK 1.1
  - OpenGL Library
  - Intel Core2Duo E6550
  - nVIDIA GeForce 8800 GTX
- Benchmarks

	Model 1	Model 2
Scenario 1	Sphere(960)	Cylinder(216)
Scenario 2	Cup(7580)	Azucar(5250)
Scenario 3	Cup(7580)	Spoon(26012)

Intersection Results

- Sphere & Cylinder



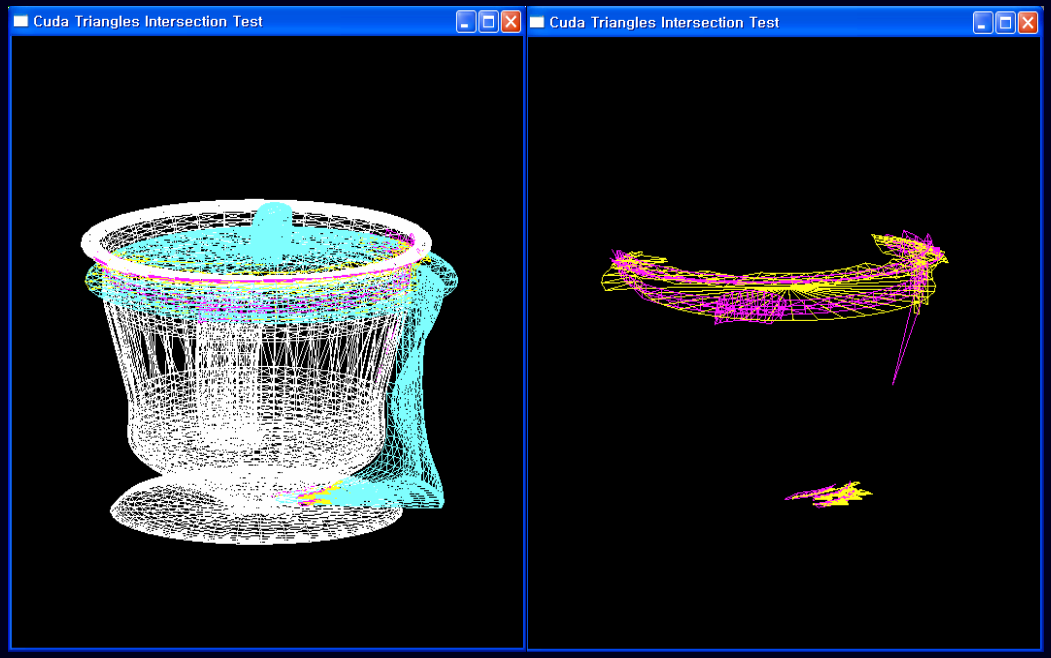
- CPU vs. GPU(CUDA)

	Time (ms)
CPU	11.84
GPU(CUDA)	2.34

5 times faster!

Intersection Results

- Cup & Azucar



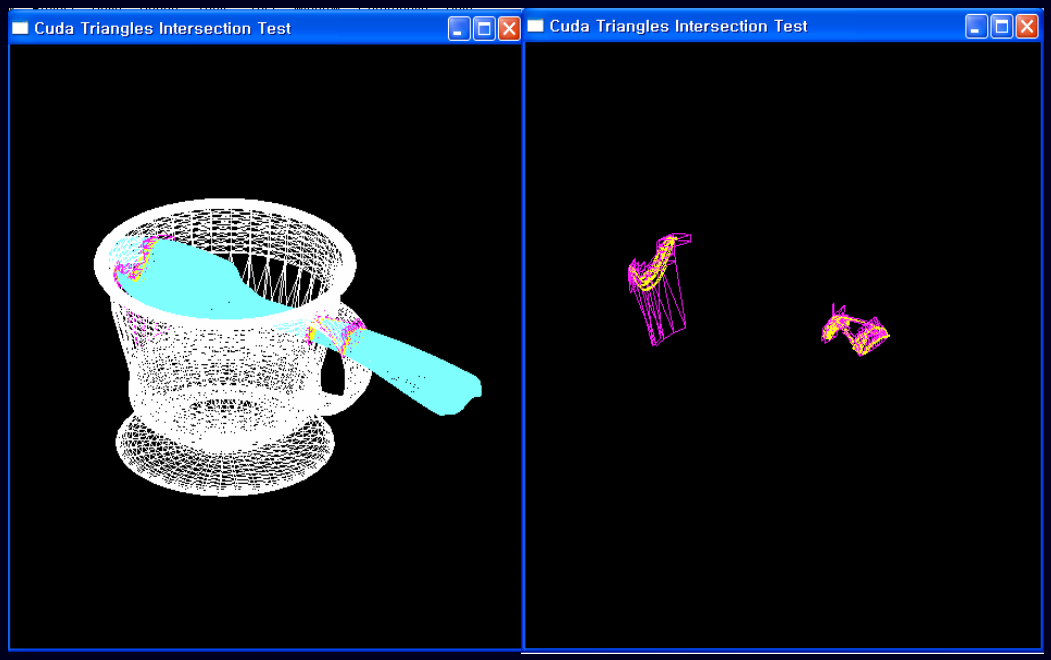
- CPU vs. GPU(CUDA)

	Time (ms)
CPU	2099.5
GPU(CUDA)	183.9

11 times faster!

Intersection Results

- Cup & Spoon



- CPU vs. GPU(CUDA)

	Time (ms)
CPU	10377.62
GPU(CUDA)	1084.22

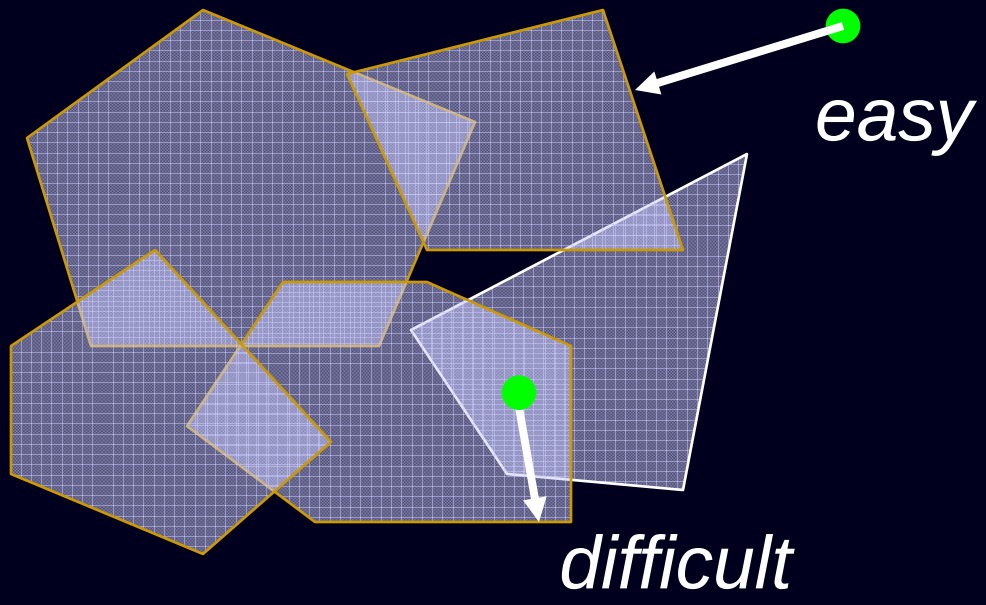
10 times faster!

Closest Point Query

Application to penetration depth computation

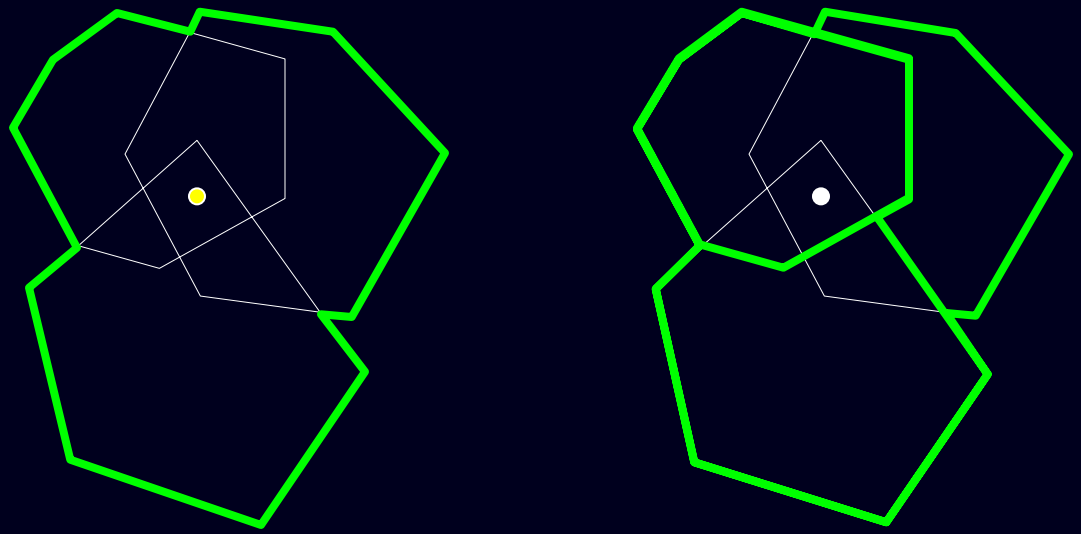
Closest Point Query

- Shortest distance from a point to the surface of the union of convex polytopes
- Approximate the query using GPUs



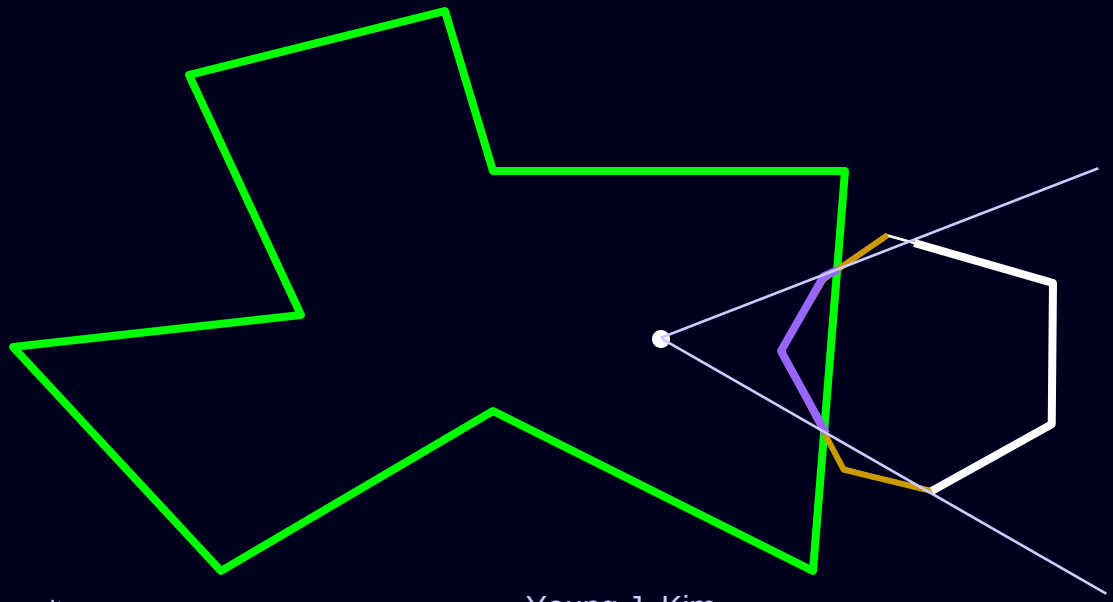
Closet Point Query

- Main Idea
  - Incrementally expand the current front of the boundary



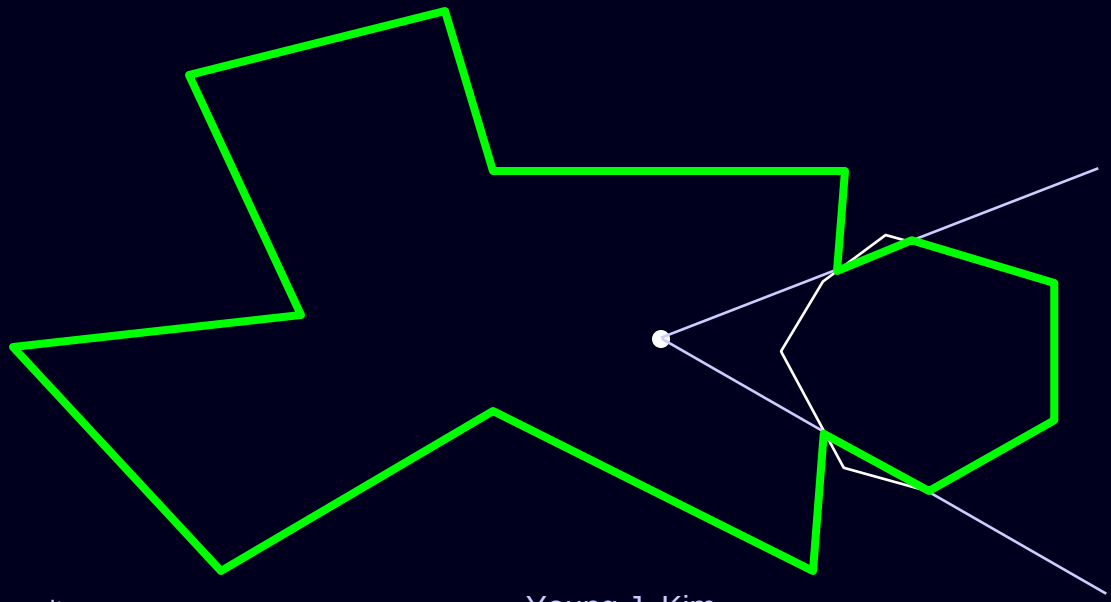
Closest Point Query

1. Render front faces, and open up a window where z-value is less than the current front
2. Render back faces w/ z-greater-than test
3. Repeat the above *m* times, where m := # of obj's

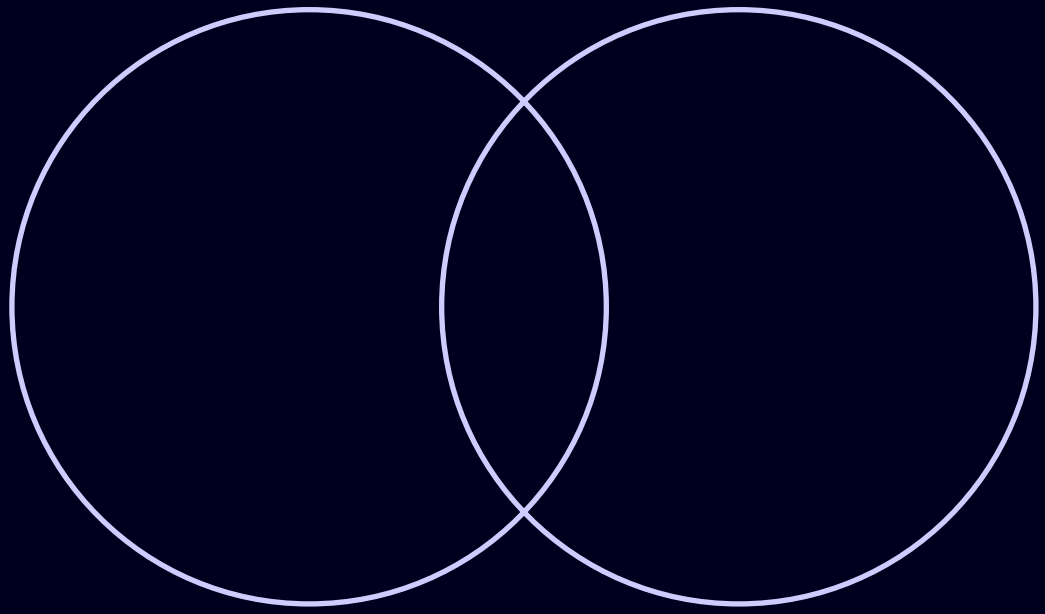


Closest Point Query

1. Render front faces, and open up a window where z-value is less than the current front
2. Render back faces w/ z-greater-than test
3. Repeat the above *m* times, where m := # of obj's

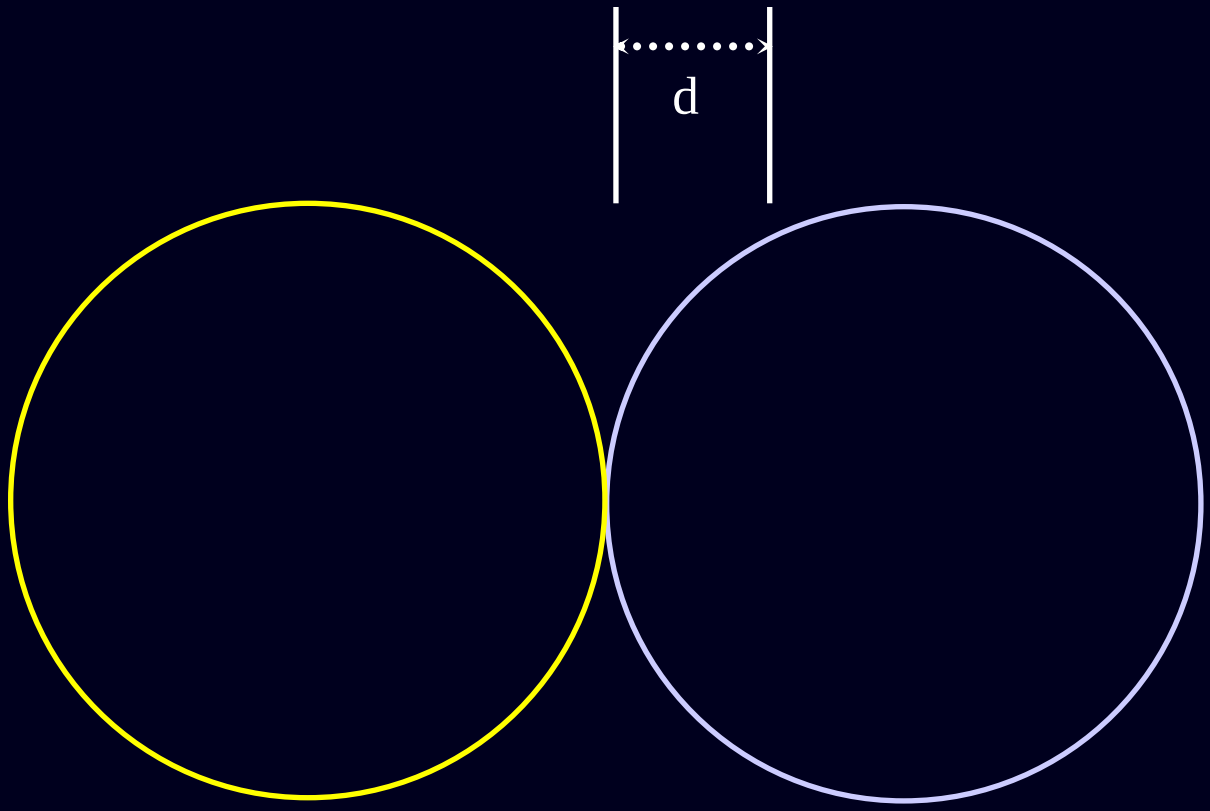


Penetration Depth (PD)



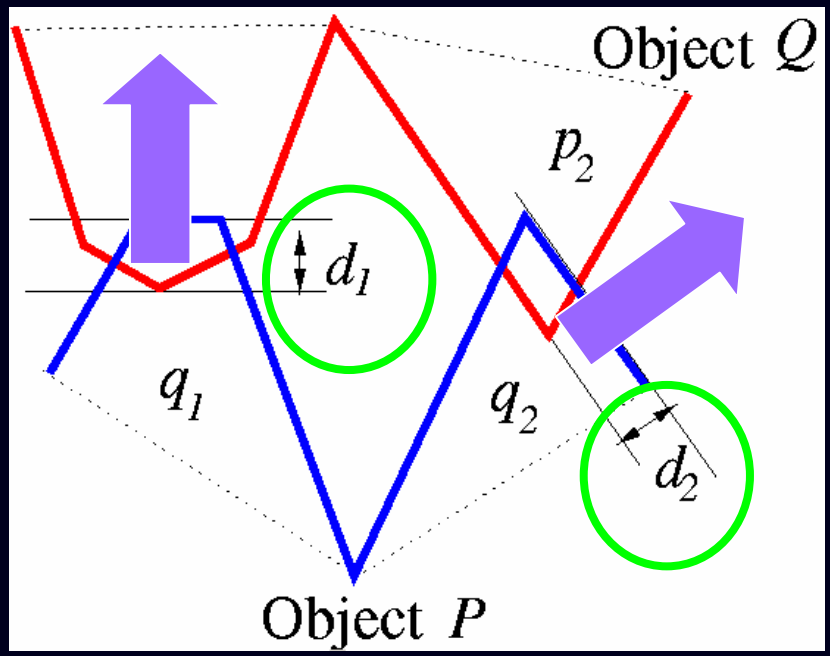
Minimum translational distance needed to separate objects

Penetration Depth (PD)



Minimum translational distance needed to separate objects

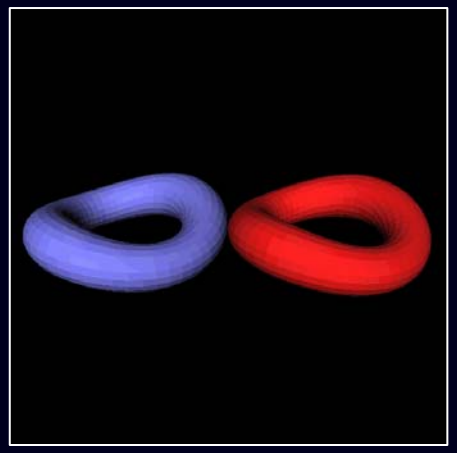
Penetration Depth



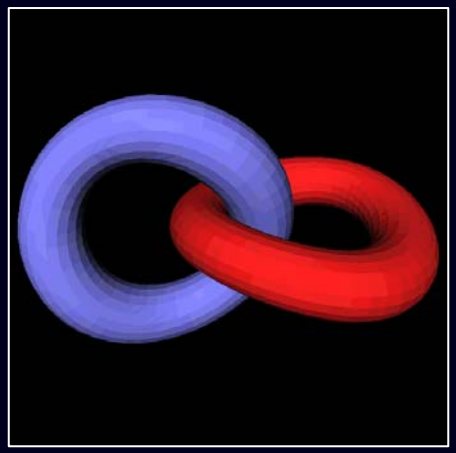
PD determines the amount of repulsive forces
in penalty-based, 6DOF haptic rendering

Penetration Depth

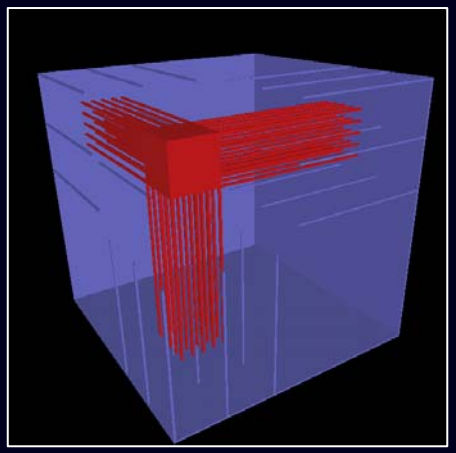
- Shortest distance from the origin to the surface of convex *Minkowski sum*



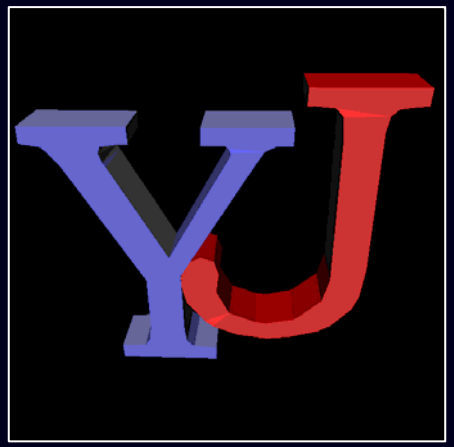
0.3 sec



3.7 sec



1.9 sec

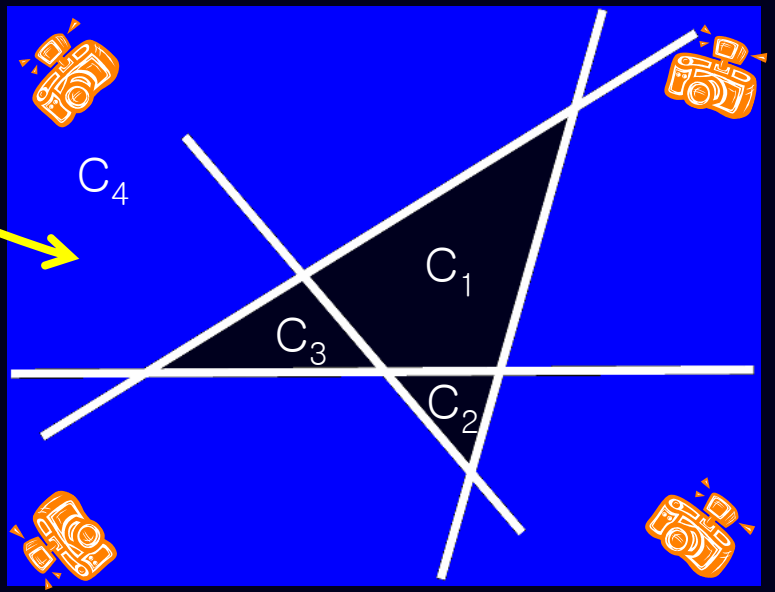


0.4 sec

Approximate Arrangement Computation

Application to global visibility and swept volume

Arrangement

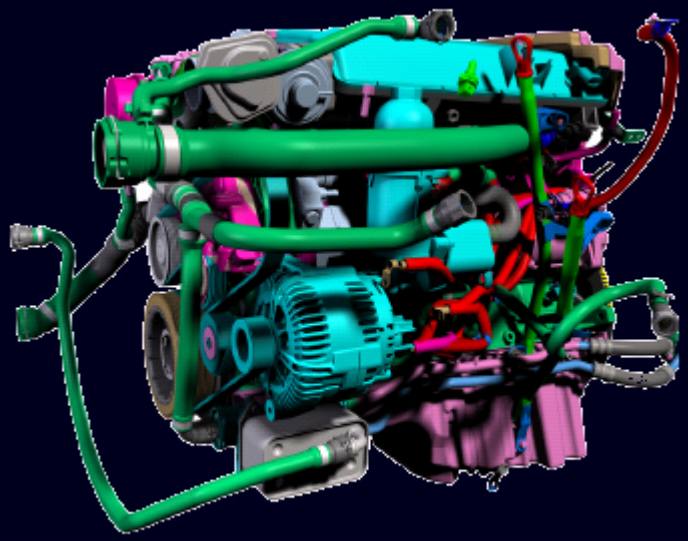


A simple arrangement of 5 lines

- An *arrangement* of geometric objects is the decomposition of space induced by given geometric primitives
- The *outer envelope* is defined as a boundary of the cell in an arrangement, which can be reached from the infinity

[Halperin 97]

Motivations



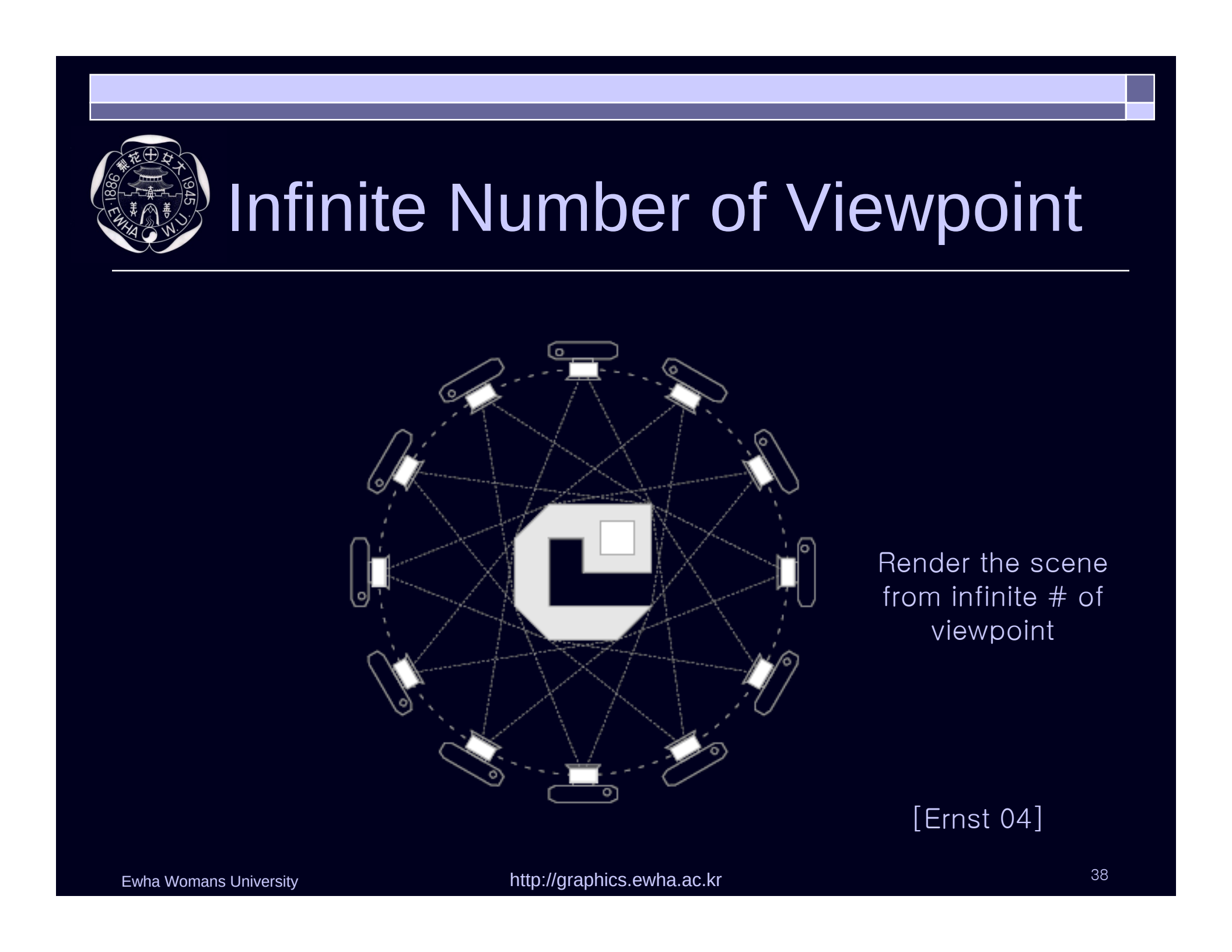
Engine of a BMW 5
with 3,741,833 triangles

Car door
with 782,018 triangles

[Ernst 04]

Goal

- Remove invisible geometric elements when the inspection is performed from outside only
  - Design reviews
  - Visual inspection
  - Training simulations



Infinite Number of Viewpoint

Render the scene from infinite # of viewpoint

[Ernst 04]

Main Contributions

- Robust and practical approach to find invisible elements from a given scene database
- Pose the problem as envelope computation
  - Use of GPUs to accelerate the computation

Algorithm Overview

Distance field representations

Use GPUs

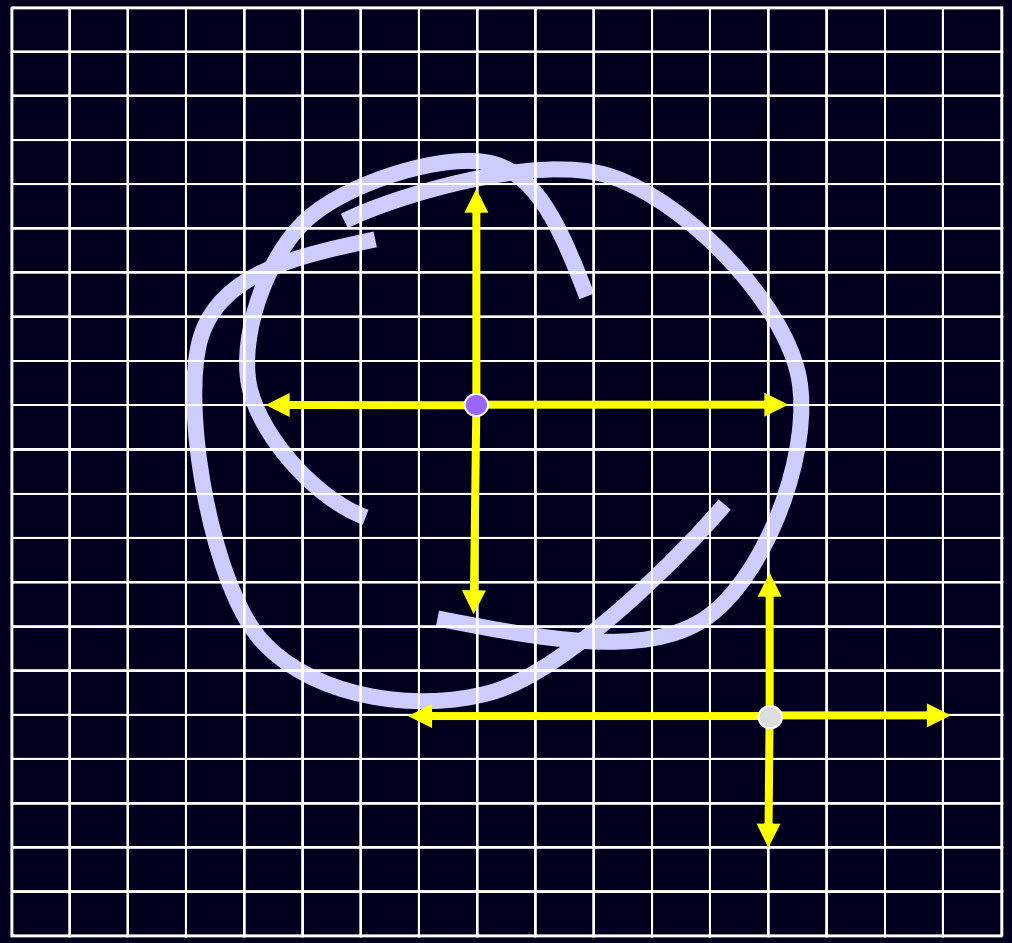
Fast marching method

Use CPUs

Extract visible surface

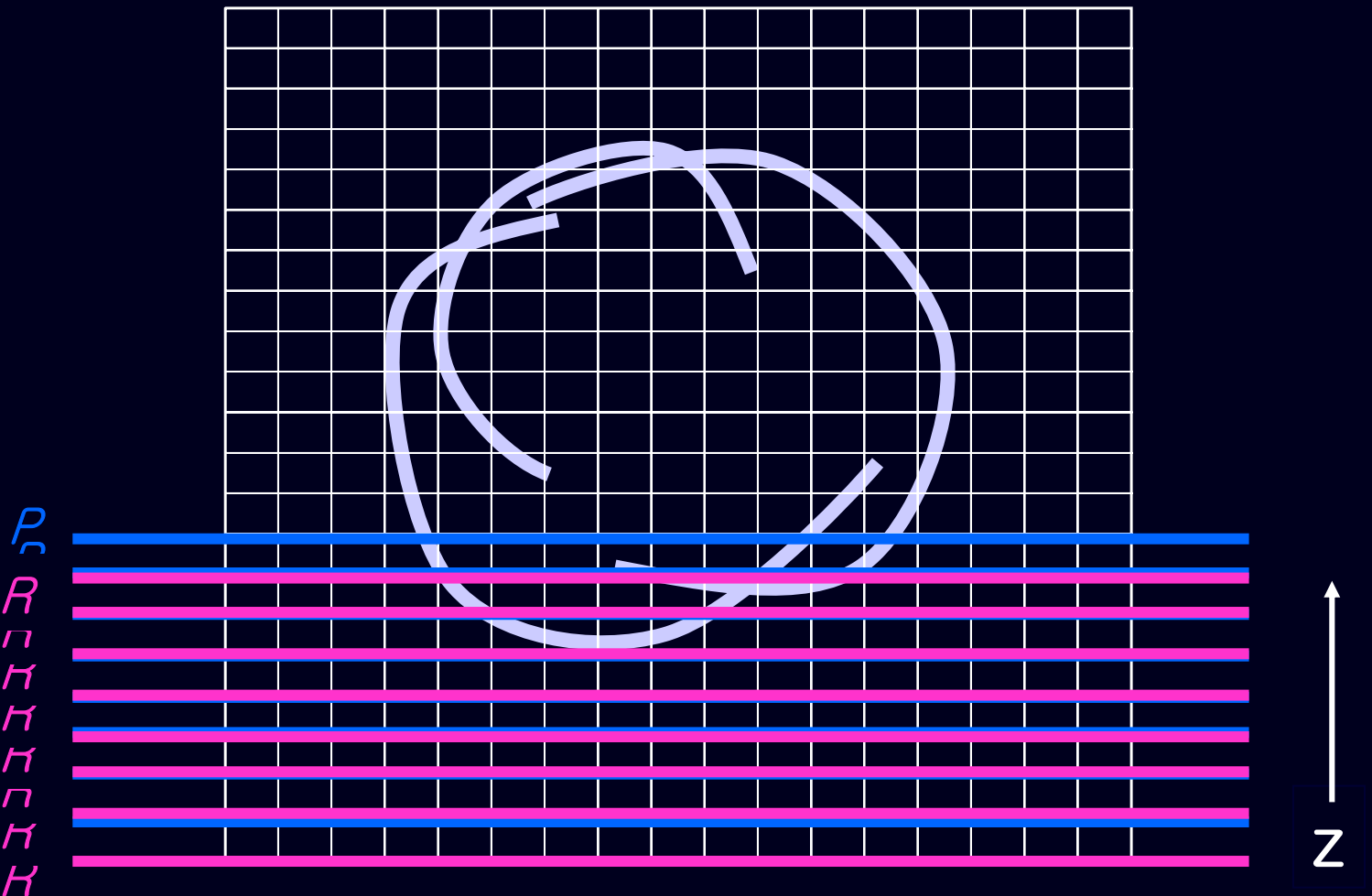
Use CPUs

Directional Distance Fields

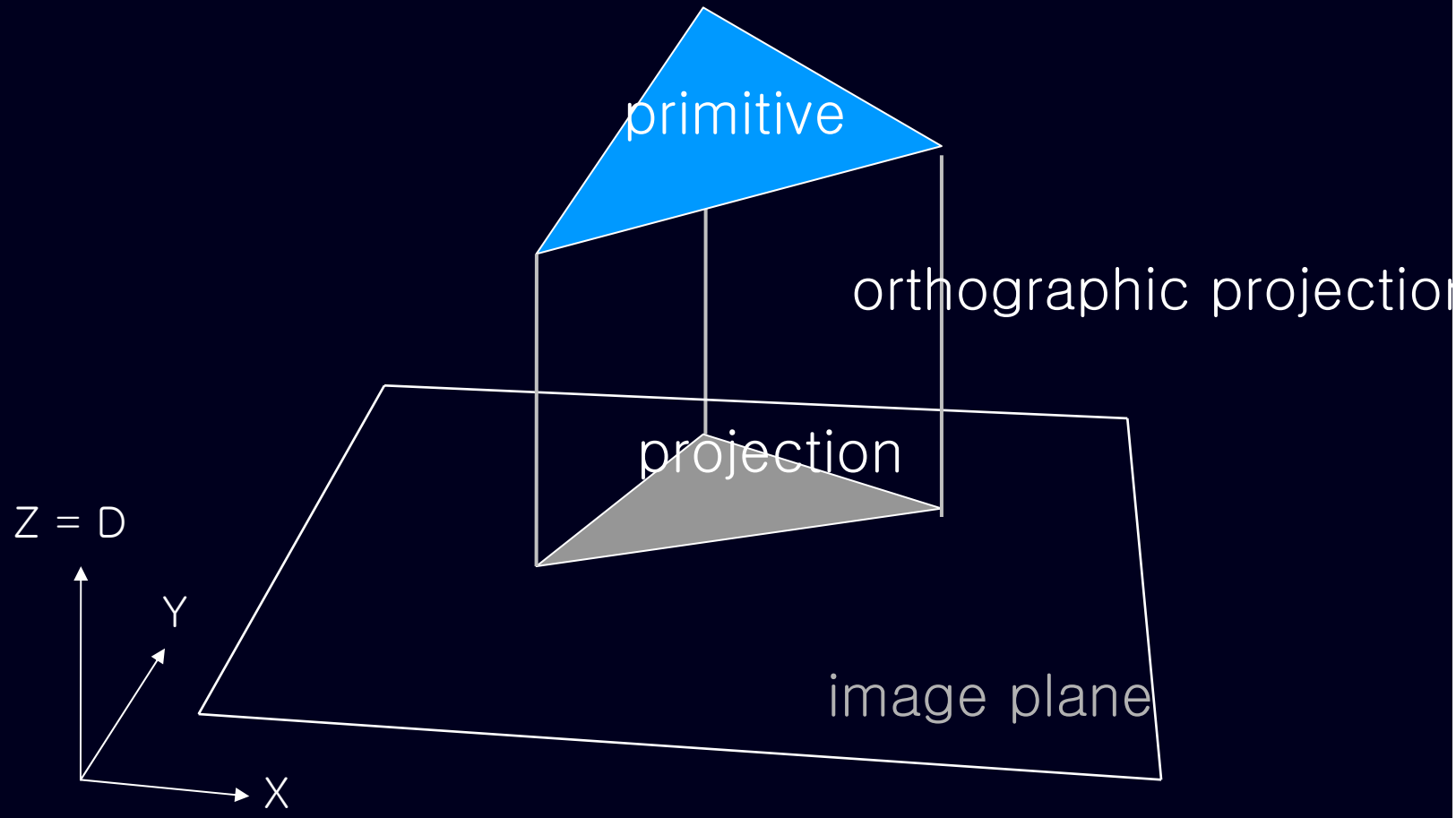


- 3D voxel grid where each voxel contains a value of shortest distance to the surface
- We compute directional distance fields for six principal directions (+x, -x, +y, -y, +z, -z)

Distance Fields Computations

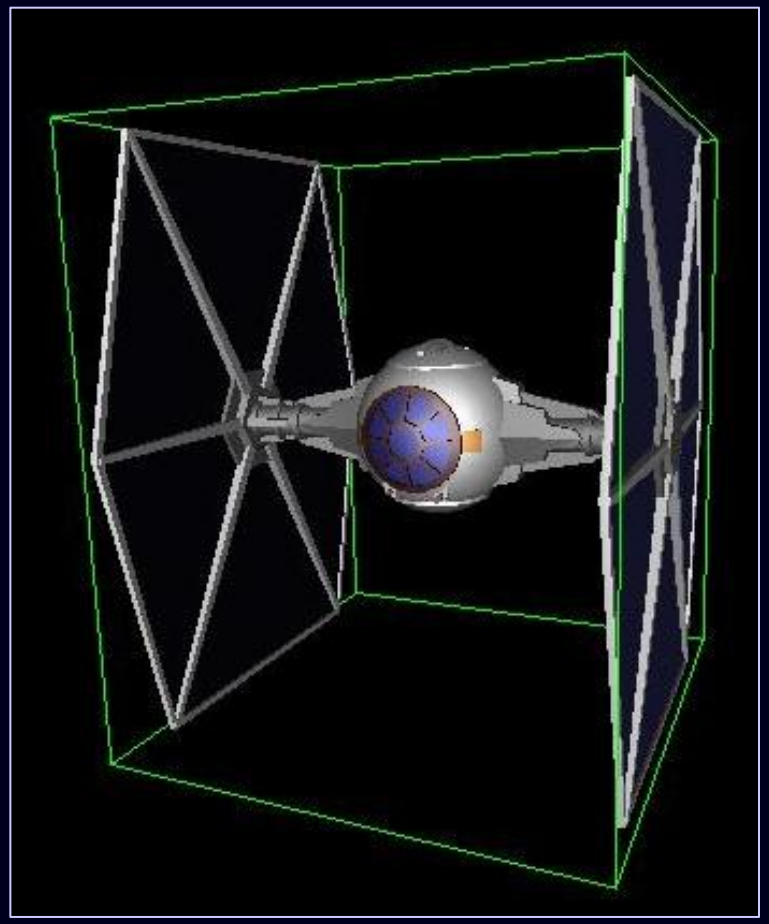


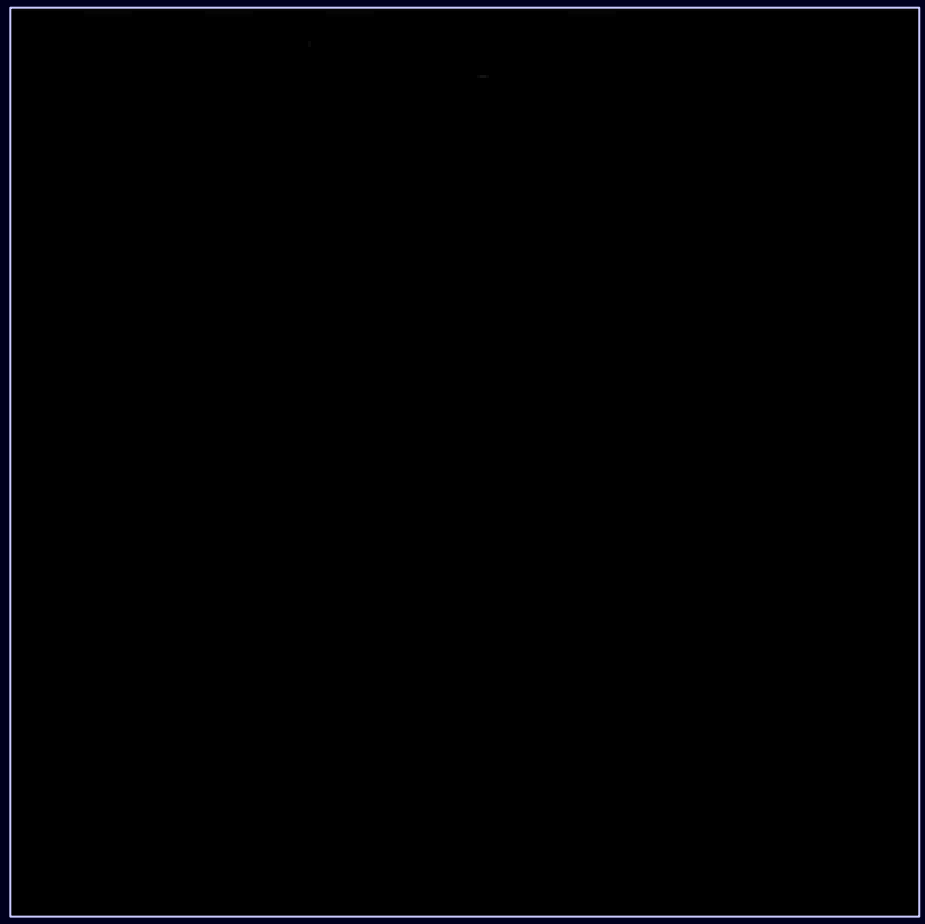
Distance Fields Computations



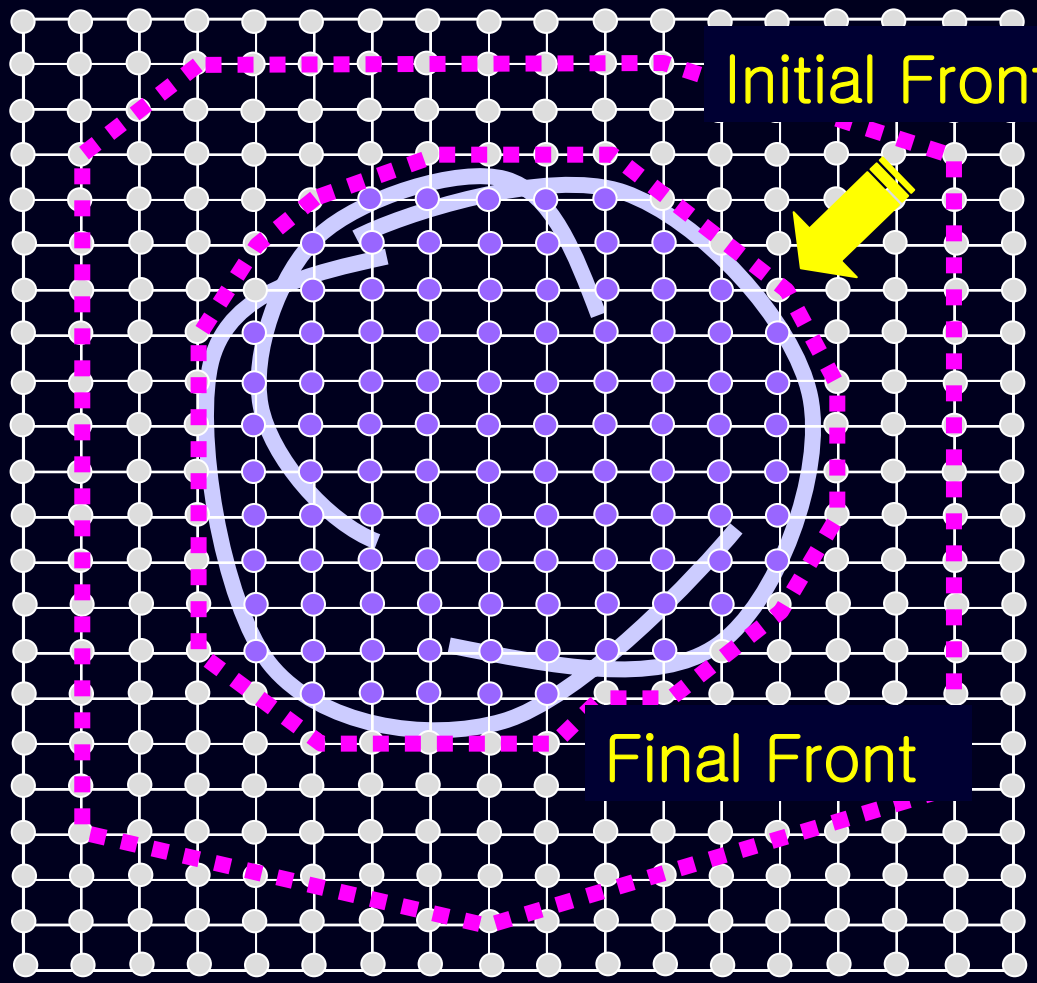
Z-buffer holds the directed distance values

Distance Fields Generation

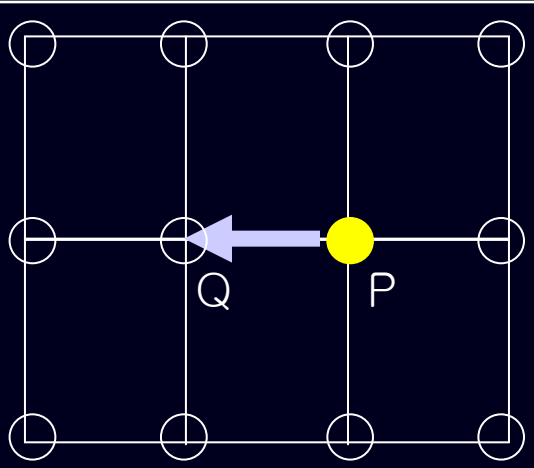




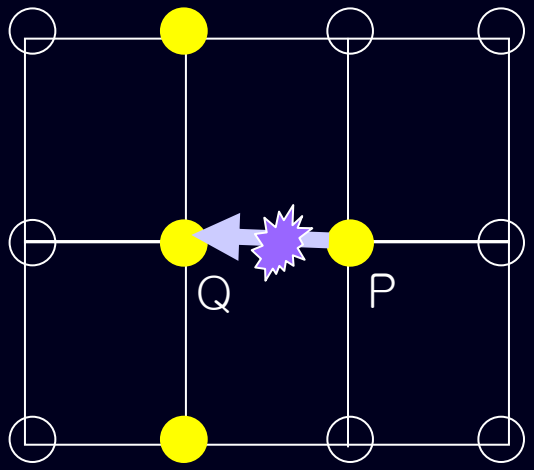
Front Propagation



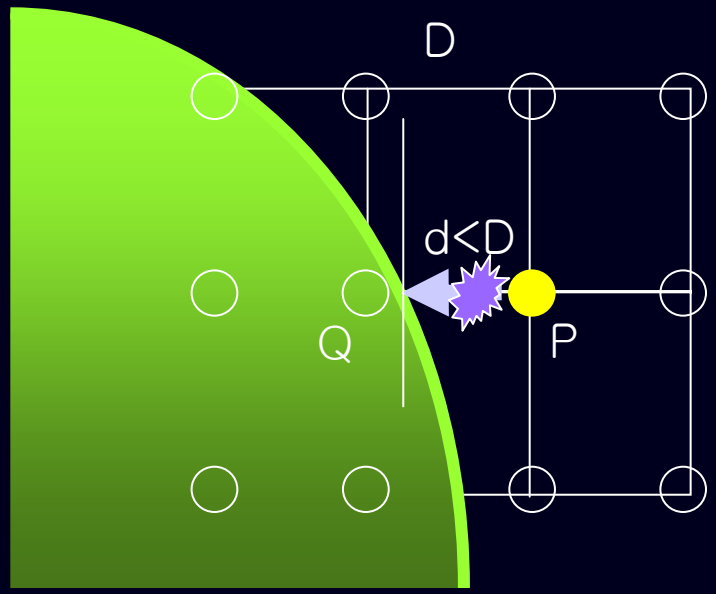
Front Propagation Rules



1. Propagate

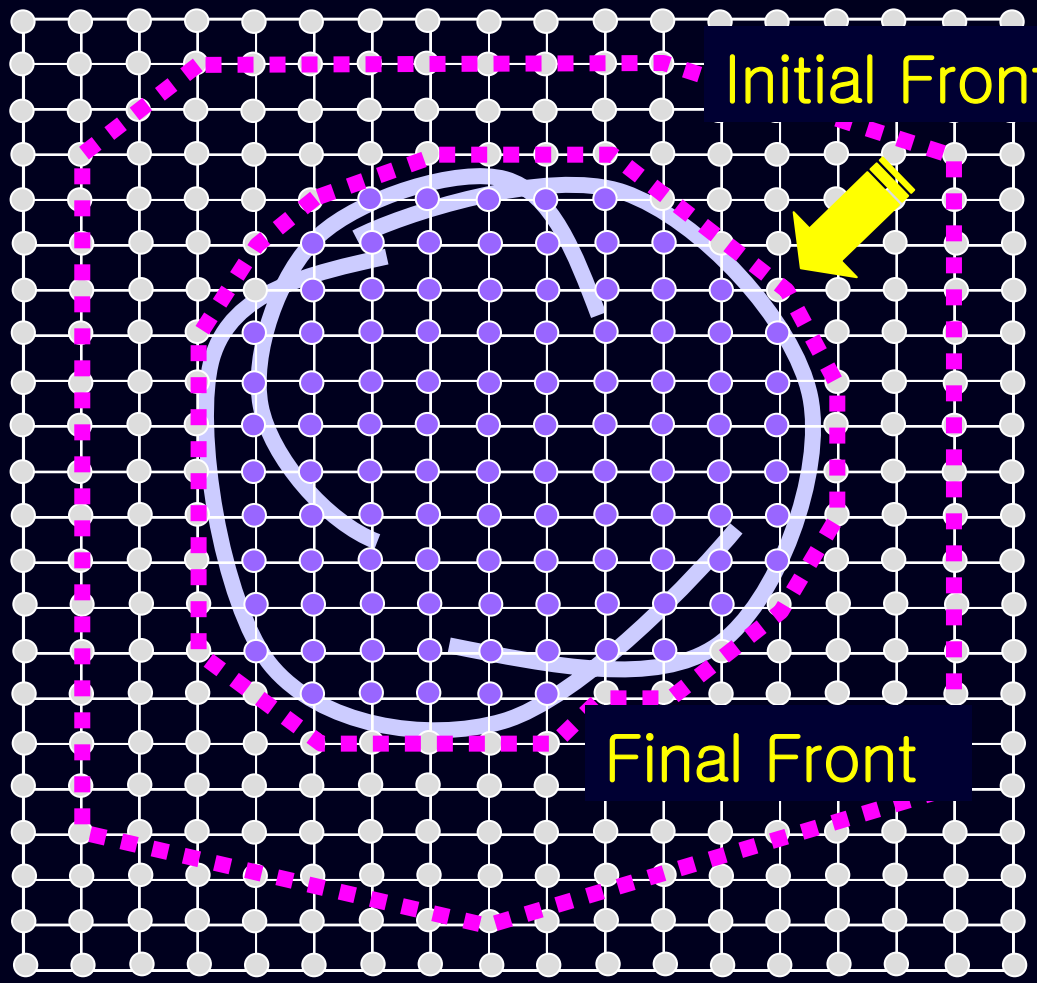


2. Don't Propagate

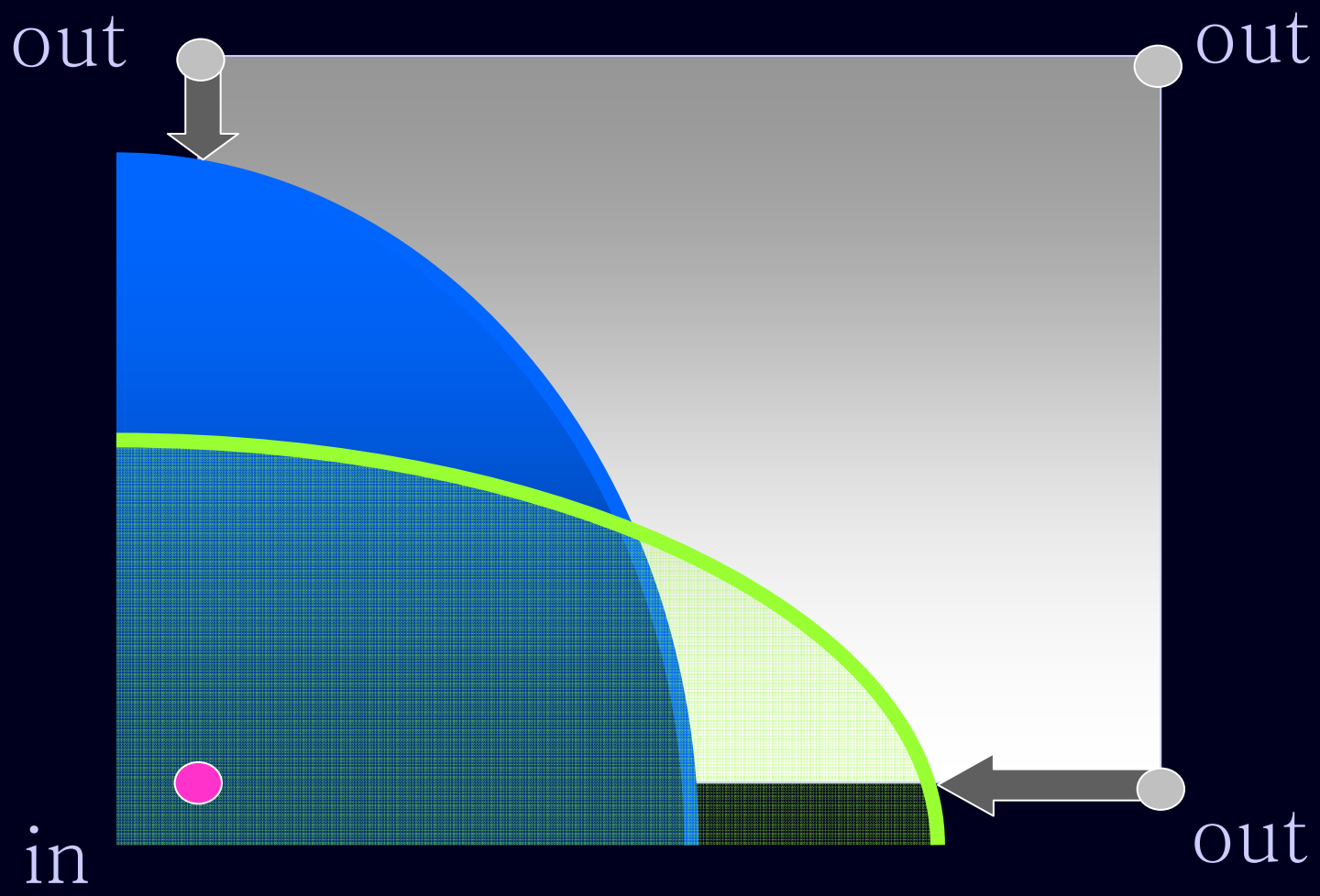


3. Find envelope and don't propagate

Front Propagation



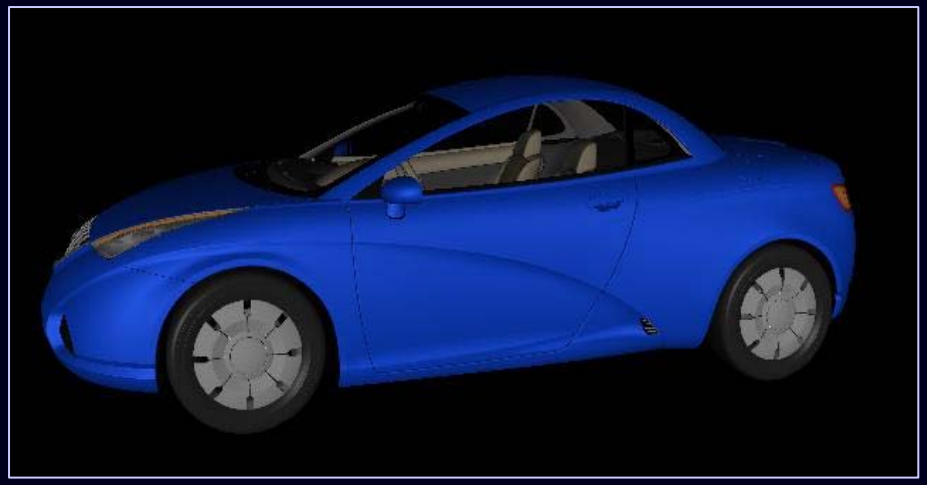
Extract Visible Surface

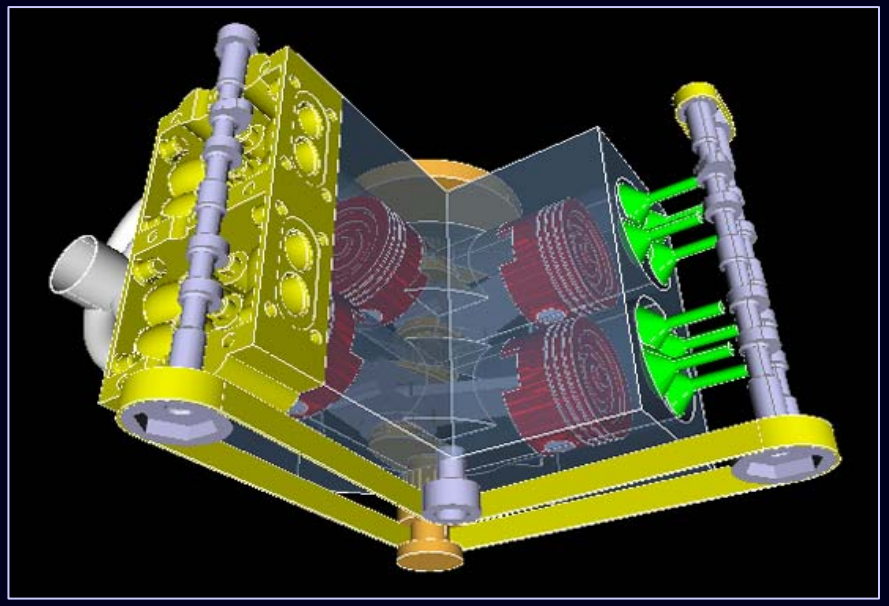


Implement Environment

- Microsoft Visual Studio .Net
- OpenGL, OpenSG library
- Dual AMD Athlon 64 2.6GHz PC with nVIDIA GeForce 7900 GX2

Benchmarking Models

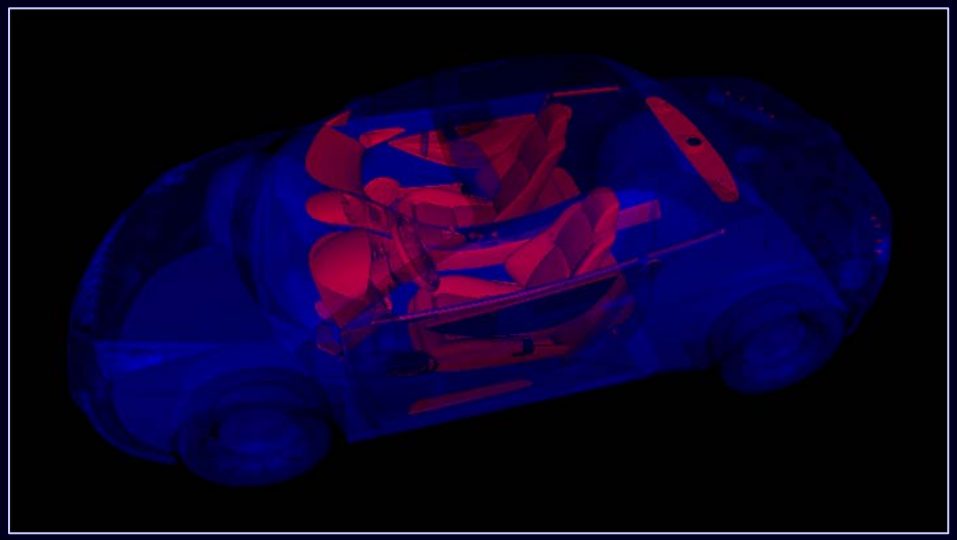


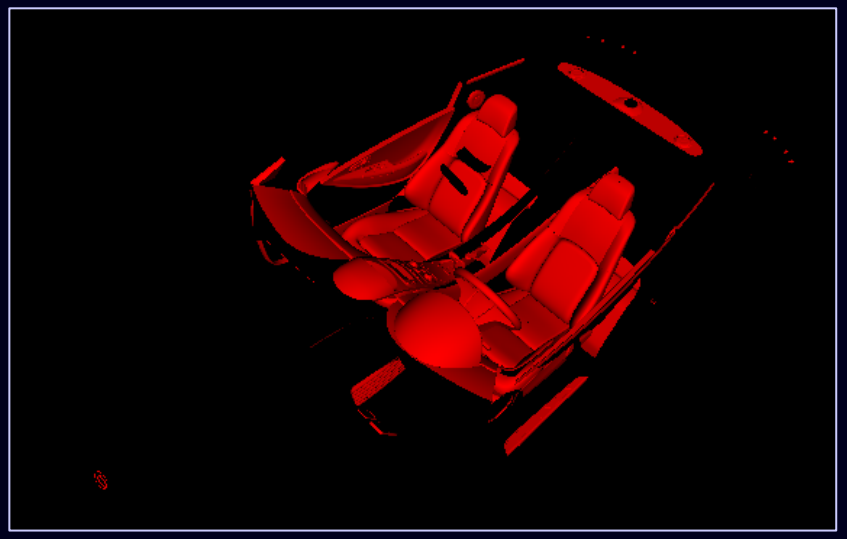


Car Model
3M triangles
0.25K scene graph nodes

Engine Model
0.3 M triangles
7K scene graph nodes

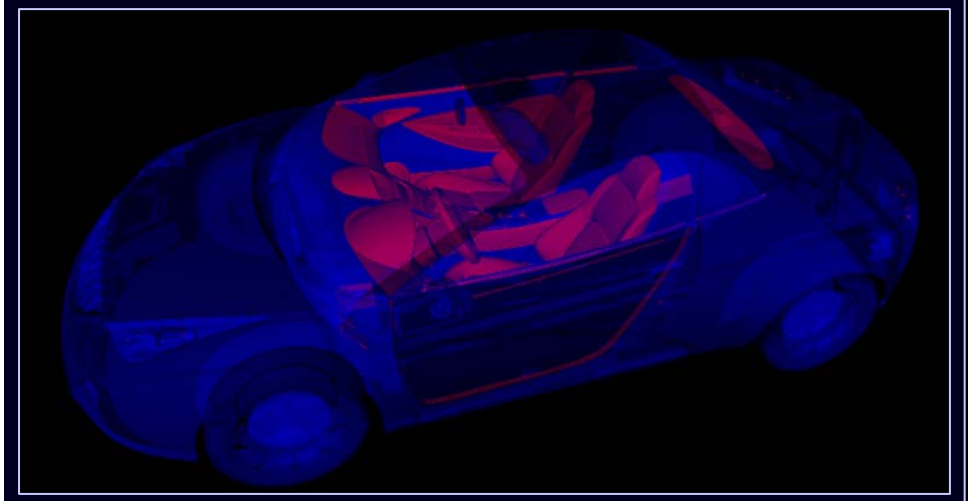
Results





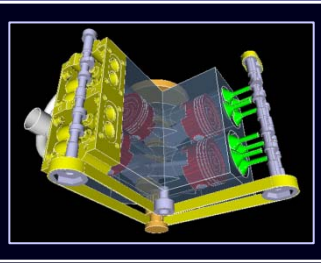
Grid Resolution: 128^3, Removal Rate: 71%

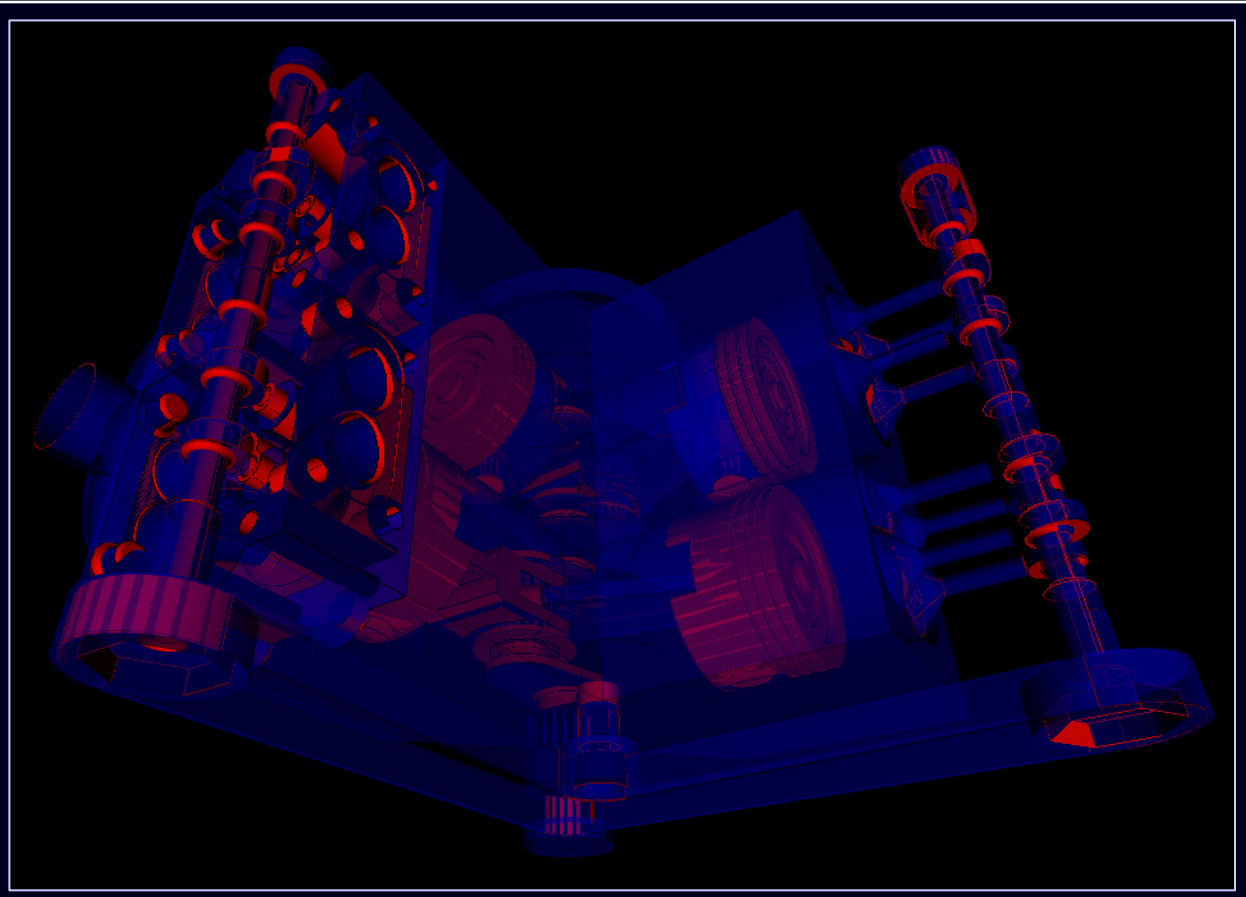
Results



Grid Resolution: 256^3, Removal Rate: 69%

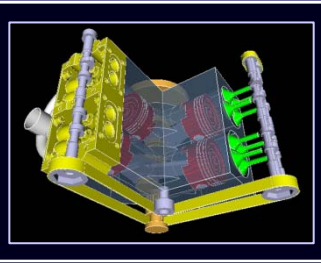
Results

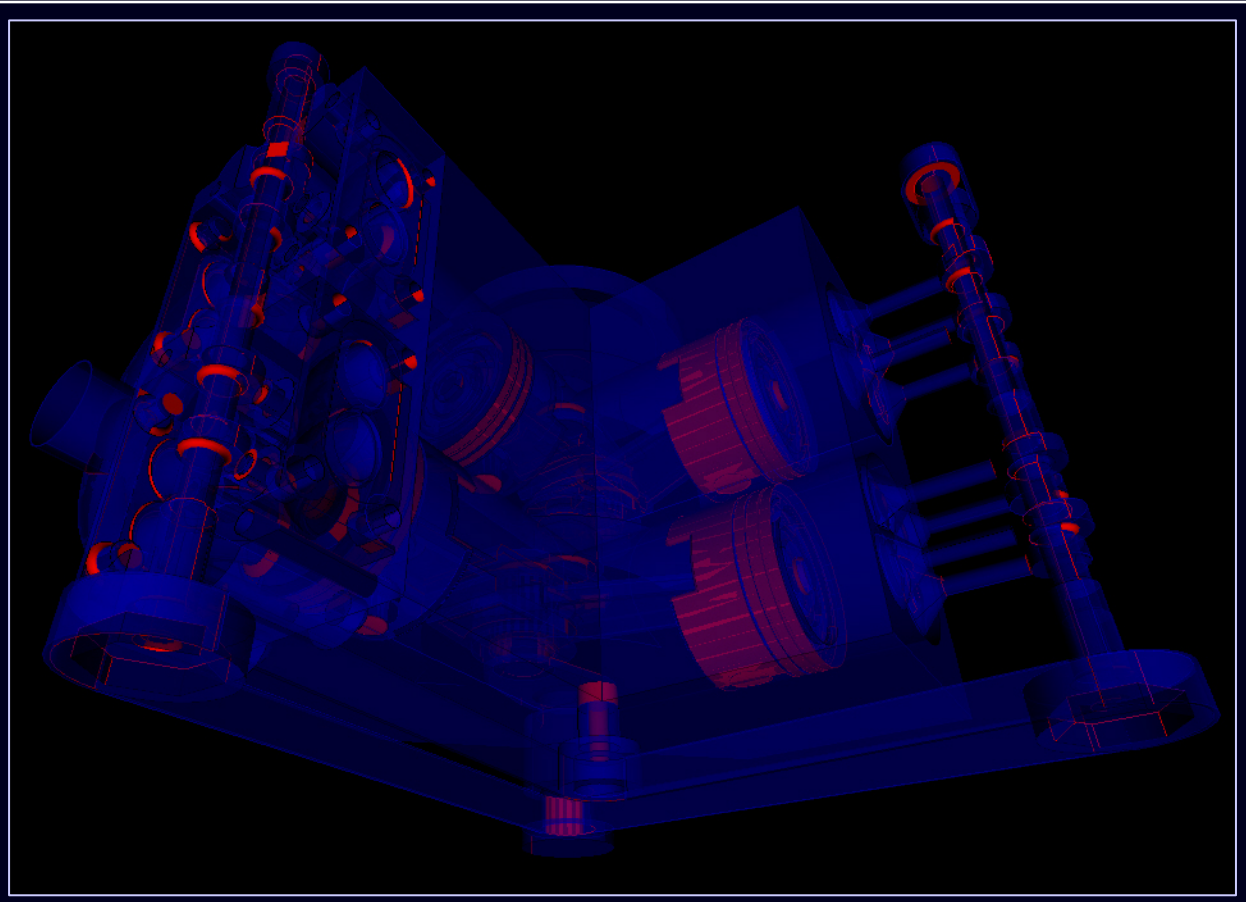




Grid Resolution: 64^3 Removal Rate: 82%

Results





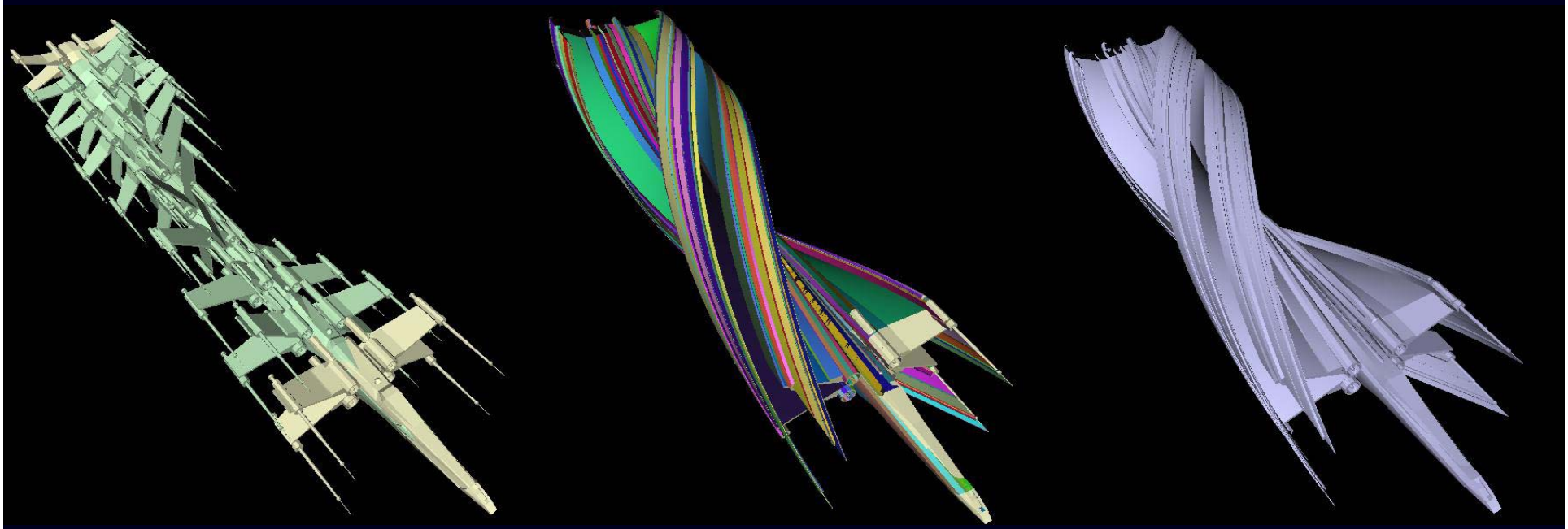
Grid Resolution: 128^3 Removal Rate: 64%

Computation Time

Models	Grid resolution	directed distance field computation time	Front propagation time
Car (247 nodes, 3M Tri.)	128	3.42sec	0.05sec
	256	6.59sec	0.10sec
Engine (7177 nodes, 270K Tri.)	64	1.85sec	0.05sec
	128	4.26sec	0.40sec

Average : 4.18sec

Application to Swept Volume



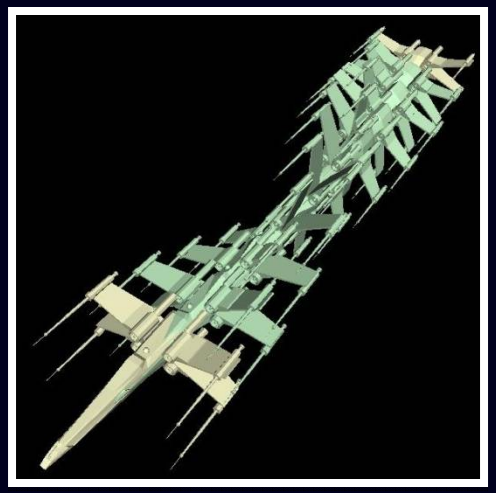
Sweep Trajectory

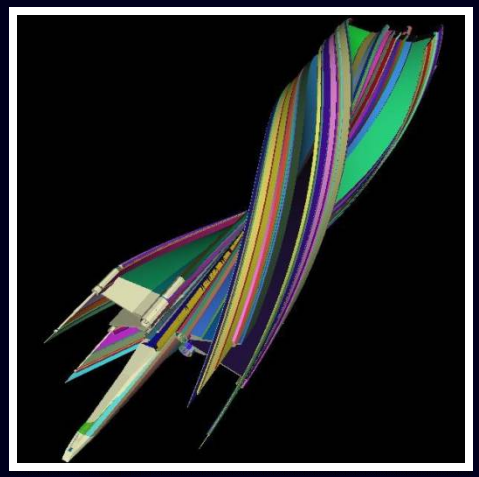
Arrangement

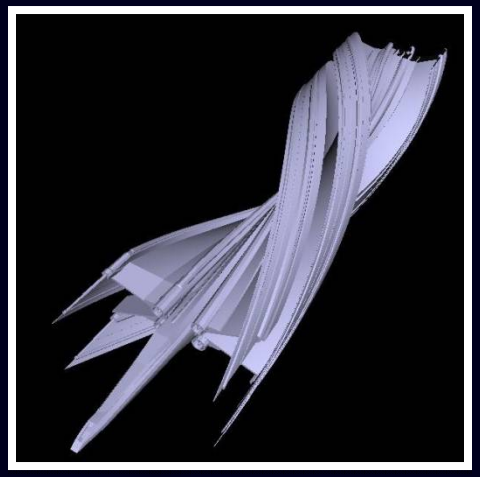
Boundary of SV

Computational Pipeline

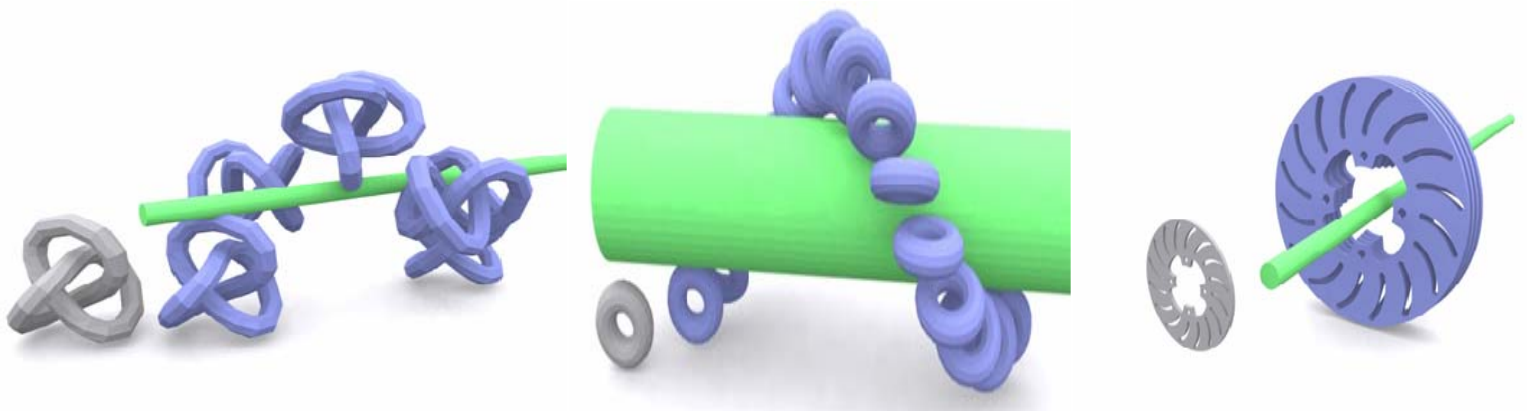
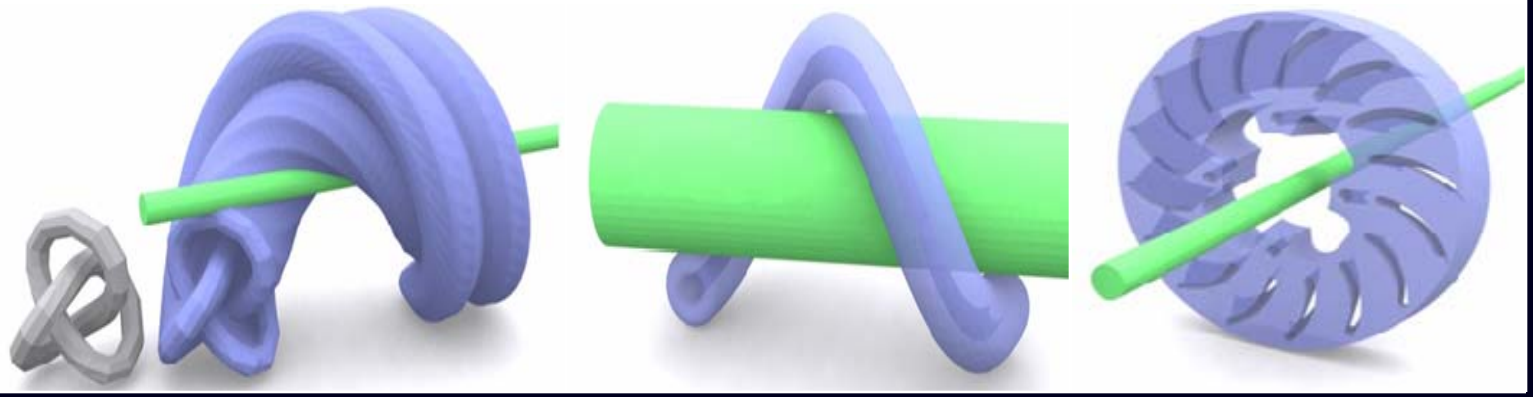
1. Enumerate surface primitives
2. Compute their arrangement
3. Traverse the arrangement and extract the *outermost boundary*



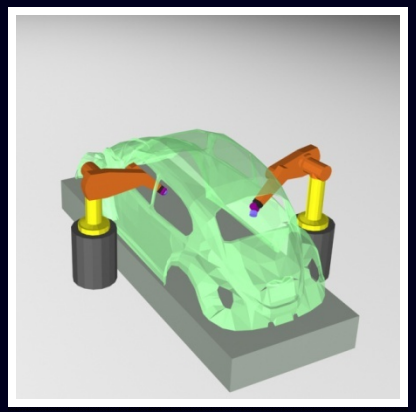


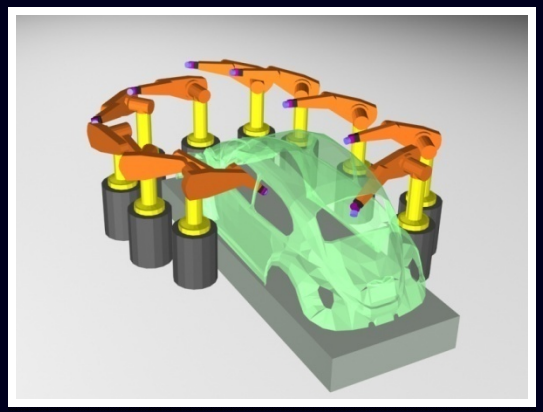


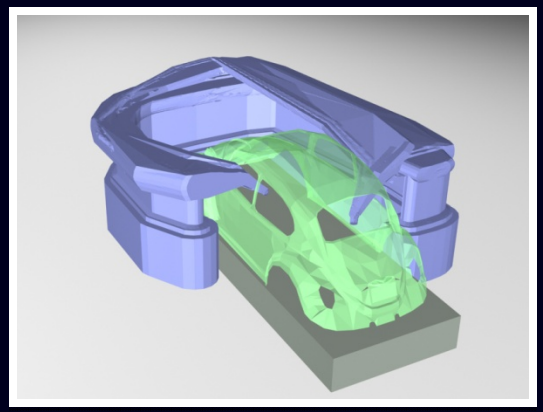
Results

Trajectory	
Sweep	

Results







References

- Xinyu Zhang, Young J. Kim, Interactive collision detection for deformable models using Streaming AABBs, *IEEE Transactions on Visualization and Computer Graphics*, 13(2), Mar/Apr,2007

- Y. J. Kim, K. Hoff, M. C. Lin and D. Manocha, Closest point query among the union of convex polytopes using rasterization hardware, *Journal of Graphics Tools*, 7.4, 2003

- 민혜정, 이민경, 김영준, 대용량 모델 렌더링을 위한 전역적 가시화 컬링 알고리즘, *한국 그래픽스 학회*, Nov. 2006

- Y. J. Kim, G. Varadhan, M. C. Lin and D. Manocha, Fast Swept Volume Approximation of Complex Polyhedral Models, *Computer Aided Design* , 36(11), Sep 2004.

Thank you

Questions to kimy@ewha.ac.kr