

Motion Planning & Physically-based Modeling Using GPUs

Ming C. Lin

University of North Carolina at Chapel Hill

<http://gamma.cs.unc.edu/voronoi/planner>

<http://gamma.cs.unc.edu/PIVOT>

<http://gamma.cs.unc.edu/PD>

<http://gamma.cs.unc.edu/CULLIDE>

<http://gamma.cs.unc.edu/ICE>



SIGGRAPH'03

M. C. Lin

Organization

- Motion Planning
- Proximity Queries
- Physically-based Simulation

SIGGRAPH'03

M. C. Lin

Organization

- **Motion Planning**
- Proximity Queries
- Penetration Depth Estimation

SIGGRAPH'03

M. C. Lin

Motion Planning

Given the initial and goal configurations, the geometric description of the robot/agent and the environment, find a viable path if one exists.



SIGGRAPH'03

M. C. Lin

Applications

- Assembly Planning
- Car Painting
- Waste Cleaning

- Character animation
- Virtual human or artificial life
- Molecular docking
- Surgical Planning
- Virtual prototyping
- Pipe layout

SIGGRAPH'93

M. C. Lin

Two Classes of Planning Algorithms

- Criticality-based
 - Complete
 - Computational expensive
 - Difficult to implement

- Random Sampling
 - Probabilistically complete
 - Simple to implement
 - Fast in practice for most scenarios
 - Performance degrading on narrow passages

SIGGRAPH'93

M. C. Lin

GVD: Maximally Clear Points

Points with the largest distance values to nearby obstacles



SIGGRAPH'93

M. C. Lin

Planning Based on GVD & MAT

- O'Dunlaing, Sharir and Yap [1983]
- Canny and Donald [1987]
- Latombe [1991]
- Choset, Burdick, et al. [1994-1999]
- Vleugels and Overmars [1996,1997]
- Guibas, Holleman and Kavraki [1999]
- Wilmarth, Amato and Stiller [1999]
- and others

SIGGRAPH'93

M. C. Lin

Basic Ideas

- Use of rasterization graphics hardware for real-time motion planning in dynamic environments
- Discrete GVD of any primitive types and approximate distance function with bounded errors
- Simple and “implementation-friendly” acceleration techniques for Voronoi-based robot motion planning
 - Construct Voronoi roadmap
 - Biasing path selection
 - Quick collision rejection test
 - Sampling near medial axis
 - Reduce search space
 - Establish milestones

SIGGRAPH'93

M. C. Lin

Outline

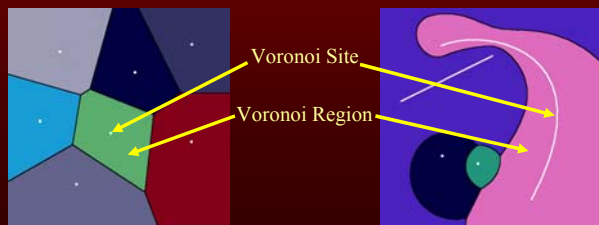
- **Review: Computing GVD with Graphics Hardware**
- Motion Planning with GVD
- Implementation & Results
- Summary

SIGGRAPH'93

M. C. Lin

Voronoi Diagram

Given a collection of geometric primitives, it is a subdivision of space into cells such that all points in a cell are *closer* to one primitive than to any other

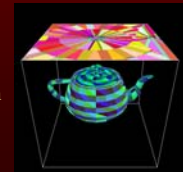


SIGGRAPH'93

M. C. Lin

Accelerated Computation of GVD [Hoff et al. SIGGRAPH'99]

- Compute a discrete Voronoi diagram by rendering a three dimensional distance mesh for each Voronoi site.
- The polygonal mesh is a bounded-error approximation of a distance functions btw a site and a 2D planar grid of points.
- Each site is assigned a unique color & distance mesh rendered in that color.
- For each sample point, the graphics system computes the closest site and the distance to that site using polygon scan-conversion and the Z-buffer depth comparison.

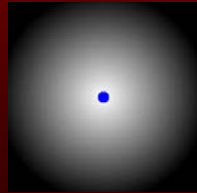


SIGGRAPH'93

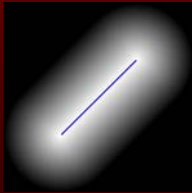
M. C. Lin

The Distance Functions for 2D Primitives

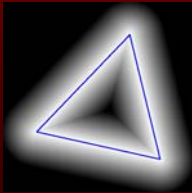
Evaluate distance at each pixel for all sites
Accelerate using graphics hardware



Point



Line

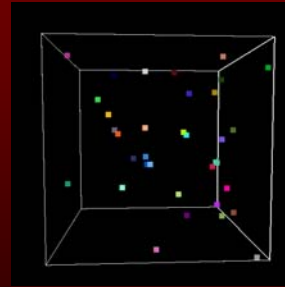


Triangle

SIGGRAPH'03

M. C. Lin

3D Voronoi Diagrams



Point sites

Graphics hardware can
generate one 2D slice at a
time

SIGGRAPH'03

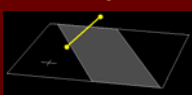
M. C. Lin

3D Distance Functions

Point



Line segment



Triangle



1 sheet of a
hyperboloid



Elliptical cone



Plane

SIGGRAPH'03

M. C. Lin

Outline

- Review: Computing GVD with Graphics Hardware
- **Motion Planning with GVD**
- Implementation & Results
- Summary

SIGGRAPH'03

M. C. Lin

Basic Approaches

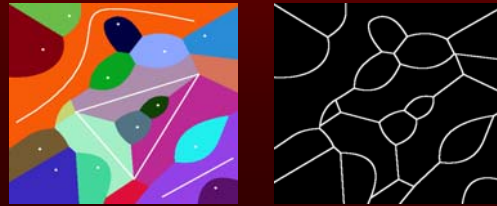
- **Depth Buffer** – providing distance function and distance gradient (finite difference)
- **Color Buffer** – building Voronoi graphs
- **Combination of Both**
 - Compute weighted Voronoi graphs
 - Voronoi vertices used for milestones
 - Weighted edges used for selecting paths
 - Distance values for quick rejection

SIGGRAPH'03

M. C. Lin

Voronoi Boundary

- For each voxel in discrete GVD, associate a color corresponding to an object ID and a distance value to this obstacle.
- To extract the boundary, use continuation method similar to iso-surface extraction -- starting from a seed point and step to next by bracketing boundary curves in a 2x2x2 region of sampled points.



SIGGRAPH'03

M. C. Lin

Constructing Voronoi Roadmap

- Identify Voronoi vertices
- Extract Voronoi boundary
- Build Voronoi graph
- Select path based on edge weights
- Incrementally construct Voronoi roadmap
 - local planner (PFF) between milestones
 - quick collision rejection test & exact CD

SIGGRAPH'03

M. C. Lin

Voronoi Roadmap/Graph for Dynamic Environments



SIGGRAPH'03

M. C. Lin

Outline

- Computing GVD with Graphics Hardware
- Motion Planning with GVD
- **Implementation & Results**
- Summary

SIGGRAPH'03

M. C. Lin

Planning Using Voronoi Diagrams

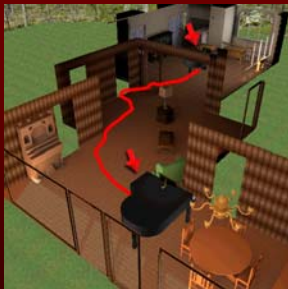
- Use the Voronoi graphs to find the near optimal path (in terms of both path length and clearance to nearby obstacles)
- The graphs are updated on the fly using GPU
- Use the potential field approach to orient the robot
- Applicable to both static and dynamic environments

[Hoff, et al, ICRA'00]

SIGGRAPH'03

M. C. Lin

Real-time Motion Planning : Static Scene



Plan motion of piano (arrow) through 100K triangle model



Distance buffer of floorplan used as potential field

SIGGRAPH'03

M. C. Lin

Real-time Motion Planning : Dynamic Scene



Plan motion of music stand around moving furniture



Distance buffer of floor-plan used as potential field

SIGGRAPH'03

M. C. Lin

Voronoi-Based Sampling

- Use hardware accelerated computation of generalized Voronoi diagram (GVD) for PRM with Voronoi-based sampling
- Classify narrow passages to select different sampling strategies
- Quick collision rejection tests using hardware computed distance functions
- 25 to 1000% speedup over uniform sampling on preliminary benchmarks

[Pisula, et al, WAFR'00]

SIGGRAPH'03

M. C. Lin

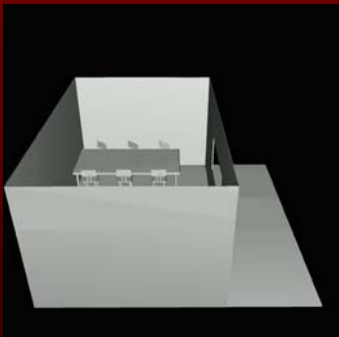
3D Benchmark (I)



SIGGRAPH'03

M. C. Lin

3D Benchmark (II)



Model Courtesy of J-P Laumond at LAAS, Toulouse

SIGGRAPH'03

M. C. Lin

Voronoi-Based Hybrid Planner

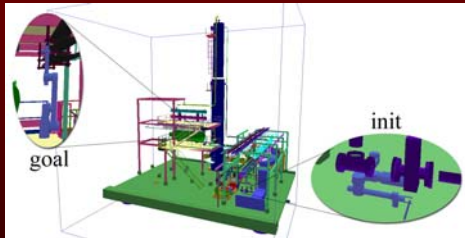
- Use Voronoi Graphs as the estimated path
- Use the curvature of the path to orient the robot
- When a collision occur, use random sampling to correct the estimated path

[Foskey et al, IROS'01]

SIGGRAPH'03

M. C. Lin

Articulated Robots: Crane Complex



GVG: 138.7s

Query: 334.7s

Model courtesy Jean-Paul Laumond

SIGGRAPH'03

M. C. Lin

Constraint-Based Motion Planning

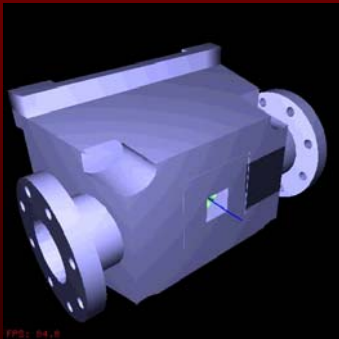
- Transform a motion planning problem into a constraint-based dynamic simulation
- Use hardware accelerated computation of distance fields for potential functions and quick collision rejection
- Applicable to both dynamic and static environments with moving obstacles and multiple collaborative agents

[Garber & Lin'02]

SIGGRAPH'03

M. C. Lin

Collaborating Agents

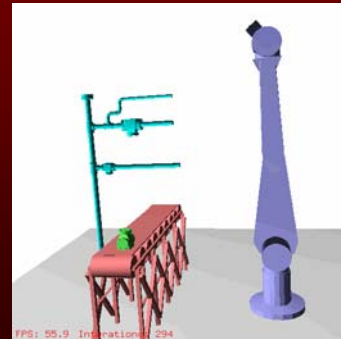


FPS: 64.8

SIGGRAPH'03

M. C. Lin

Virtual Prototyping



FPS: 55.9 Interactions: 294

SIGGRAPH'03

M. C. Lin

Outline

- Review: Computing GVD with Graphics Hardware
- Motion Planning with GVD
- Implementation & Results
- Summary

SIGGRAPH'03

M. C. Lin

Summary

- Techniques to exploit graphics hardware for *real-time* motion planning of a rigid robot in 3D
- Simple and easy to implement
- Applicable to both static and dynamic environments
- Extended to Voronoi-based sampling for PRM
- Core of the Voronoi-based hybrid-planner
- Central to Constrained Motion Planning

SIGGRAPH'03

M. C. Lin

Organization

- Motion Planning
- Proximity Queries
- Physically-based Simulation

SIGGRAPH'03

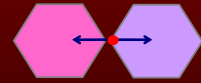
M. C. Lin

Proximity Queries

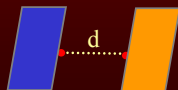
Geometric reasoning of spatial relationships among objects (in a dynamic environment)



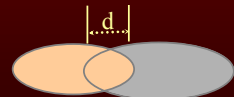
Collision Detection



Contact Points & Normals



Closest Points & Separation Distance



Penetration Depth

SIGGRAPH'03

M. C. Lin

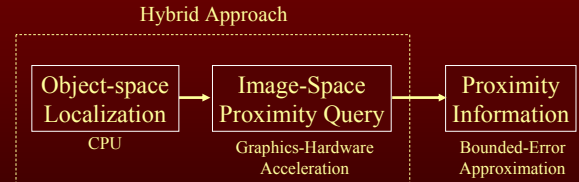
Applications

- Dynamic simulation – *contact force calculation*
- Haptic Rendering -- *restoring force computation*
- Computer Animation – *motion control*
- Motion Planning – *distance computation*
- Rapid Prototyping – *tolerance verification*
- Virtual Environments -- *interactive manipulation*
- Simulation-Based Design – *interference detection*
- Engineering analysis – *testing & validation*
- Medical Training – *contact analysis and handling*
- Education – *simulating physics & mechanics*

SIGGRAPH'03

M. C. Lin

Our Approach



[Hoff, Zaferakis, Lin & Manocha, I3D01]

SIGGRAPH'03

M. C. Lin

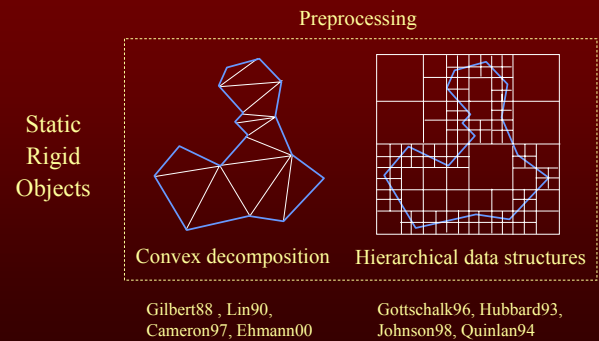
Related Work

- Proximity query algorithms
 - Exact, object-space
 - Approximate, image-space
- Distance fields

SIGGRAPH'03

M. C. Lin

Exact, Object-Space Algorithms



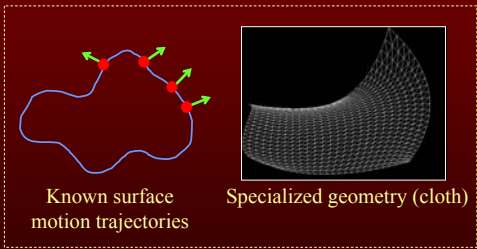
SIGGRAPH'03

M. C. Lin

Exact, Object-Space Algorithms

Restricted Cases

Dynamic Deformable Objects



Snyder93

Baraff92, Baraff98

SIGGRAPH93

M. C. Lin

Approximate, Image-Space Algorithms

Algorithms simplified and accelerated using graphics hardware:

- Visibility: Sutherland74, Catmull75
- Intersections: Rossignac92, Myskowski95, Baci98
- CSG: Goldfeather86, Rossignac90, Stewart00
- Robot motion-planning: Lengyel90, Hoff00
- Voronoi diagrams: Hoff99

Advantages:

- Simplicity
- Bounded error
- Linear complexity
- Robustness

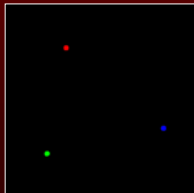
SIGGRAPH93

M. C. Lin

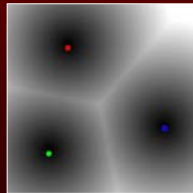
Distance Fields

Distance fields give distance to nearest object from any point:

- Fast marching method Sethian96
- Generalized Voronoi diagrams Hoff99
- Adaptively-sampled distance fields Frisken00
- Distance fields for penetration computation Fisher01



3 point objects



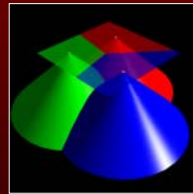
Distance field

SIGGRAPH93

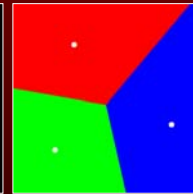
M. C. Lin

Distance Fields

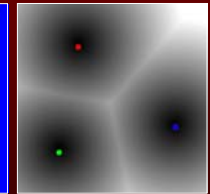
Voronoi diagram computation using graphics hardware (Hoff99)



Render polygonal mesh approximations of object distance fields



Color buffer



Depth buffer

Result after compositing distance fields using minimum depth test

SIGGRAPH93

M. C. Lin

Our Goal

Proximity query algorithm with the following properties:

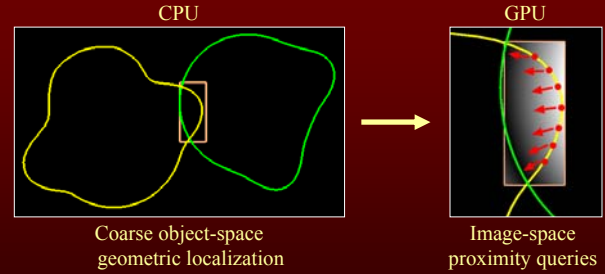
- Performs all proximity queries
- Handles non-convex primitives
- Requires no precomputation or complex data structures
- Fast and efficient
- Robust
- Portable
- Bounded error

No previous algorithm found, even in 2D!

SIGGRAPH'03

M. C. Lin

Our Hybrid Approach



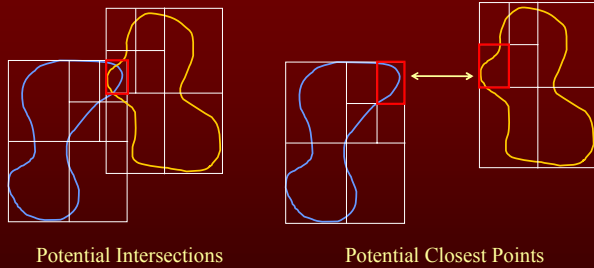
Balance load by varying localization coarseness and error bound

[Hoff, Zaferakis, Lin & Manocha, I3D01]

SIGGRAPH'03

M. C. Lin

Geometric Localization



- Reduces the number of pixels processed per image-space query
- Vary max depth in hierarchy to balance CPU/graphics loads

SIGGRAPH'03

M. C. Lin

Image-Space Proximity Queries

- Intersections
- Distance field and gradients



- Contact points and normals
- Closest points
- Separation distance and direction
- Penetration depth and direction

SIGGRAPH'03

M. C. Lin

Intersections

Boundary-boundary Boundary-volume Volume-volume

Draw A → Draw B → Intersection = Overwritten Pixels

SIGGRAPH'03 M. C. Lin

Distance Field and Gradients

Draw polygon to encode negative sign in a buffer Central difference to compute gradient at a pixel

SIGGRAPH'03 M. C. Lin

Image-Space Proximity Queries

- Intersections
- Distance field and gradients

↓

- Contact points and normals
- Closest points
- Separation distance and direction
- Penetration depth and direction

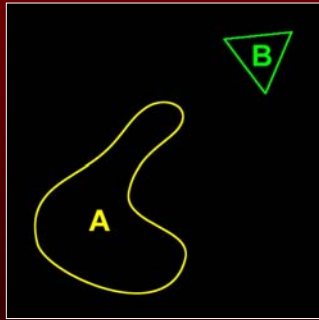
SIGGRAPH'03 M. C. Lin

Contact Points and Normals

Close contact points as intersection of enlarged boundary Normals from gradient of distance field

SIGGRAPH'03 M. C. Lin

Closest Points

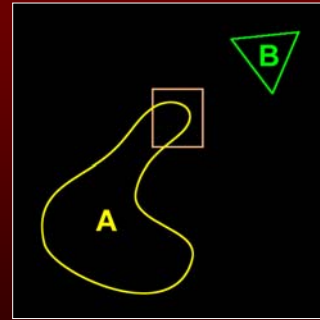


Find closest point on object A to object B

SIGGRAPH'03

M. C. Lin

Closest Points

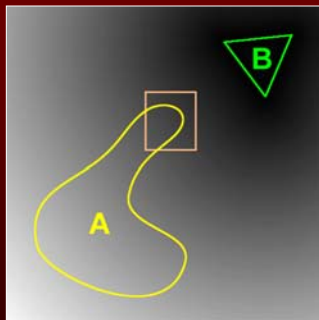


Localize region on A containing the closest point

SIGGRAPH'03

M. C. Lin

Closest Points

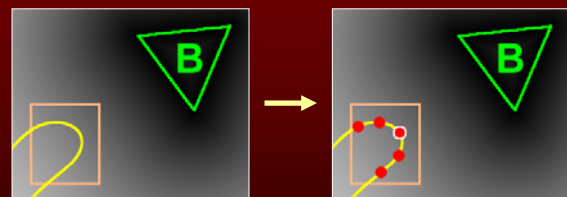


Compute distance field of B in A's localized region

SIGGRAPH'03

M. C. Lin

Closest Points

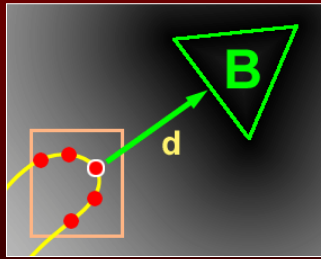


- Render boundary of A to find set of potential closest points
- Check distances at each point to find the closest

SIGGRAPH'03

M. C. Lin

Separation Distance and Direction

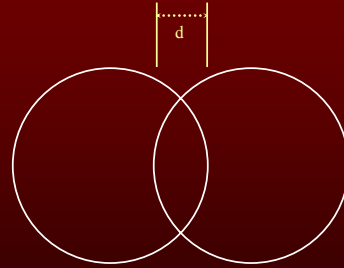


Distance at closest point = min separation distance
Gradient at closest point = separating axis direction

SIGGRAPH'03

M. C. Lin

Penetration Depth

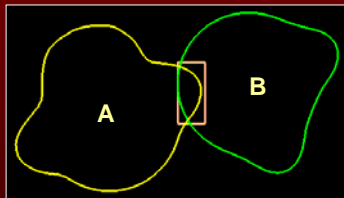


Minimum translational distance needed to separate objects

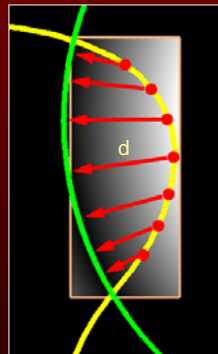
SIGGRAPH'03

M. C. Lin

Penetration Depth and Direction for a Point



Compute penetration depth and direction for A's boundary points that intersect B's interior

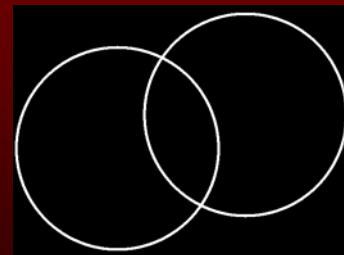


SIGGRAPH'03

M. C. Lin

Putting it all together...

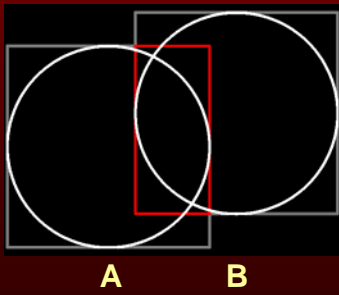
Collision response between 2 deformable objects



SIGGRAPH'03

M. C. Lin

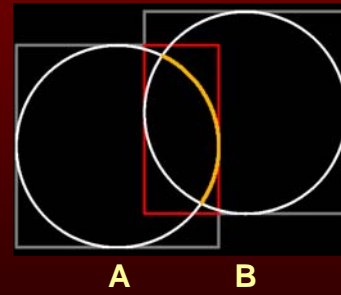
Localize Potential Intersections



SIGGRAPH'03

M. C. Lin

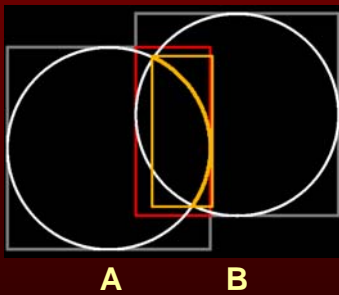
Find Penetrating Points



SIGGRAPH'03

M. C. Lin

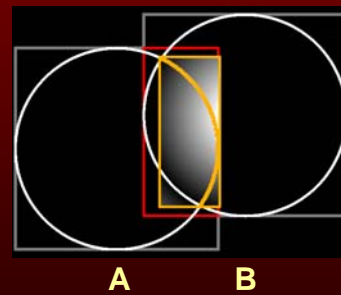
Tighten Localized Region



SIGGRAPH'03

M. C. Lin

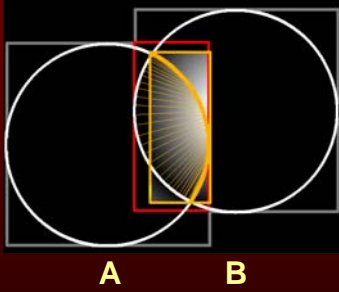
Compute Distance Field



SIGGRAPH'03

M. C. Lin

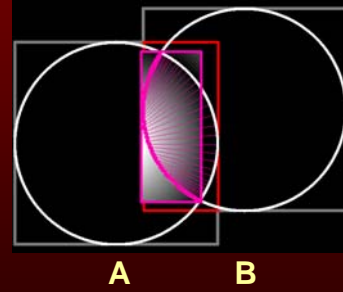
Compute Penetration Depth and Direction



SIGGRAPH'03

M. C. Lin

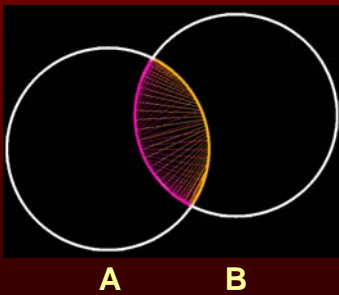
Repeat Process for Other Object



SIGGRAPH'03

M. C. Lin

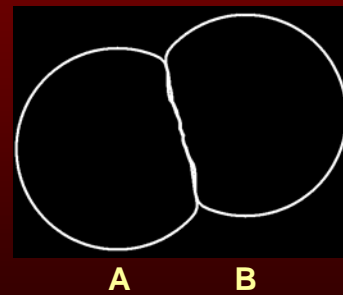
Use Proximity Info for Collision Response



SIGGRAPH'03

M. C. Lin

Separation Forces from Penetration Depth



SIGGRAPH'03

M. C. Lin

Demonstrations

- Simulation test cases:
 - Rigid and deformable objects
 - Different contact scenarios
- Collision response in 2 simulation strategies:
 - Non-penetration constraint, backtracking
 - Unconstrained, penalty-based
- In each demo:
 - No precomputation
 - Interactive frame rates
 - Bounding-box intersection localization
 - Pentium III 800, nVidia GeForce256

SIGGRAPH'03

M. C. Lin

MAP

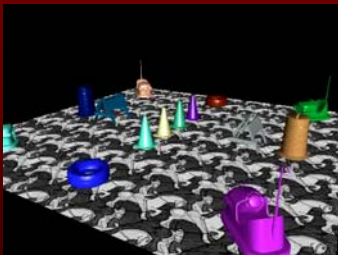


- Non-convex, rigid objects
- Frequent simultaneous close contacts
- Unconstrained, penalty-based

SIGGRAPH'03

M. C. Lin

BUMPER CARS

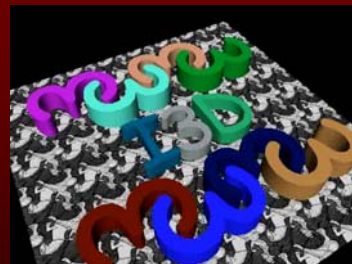


- Convex, rigid objects
- Less frequent contact
- Non-penetration constraint, backtracking

SIGGRAPH'03

M. C. Lin

LINKS

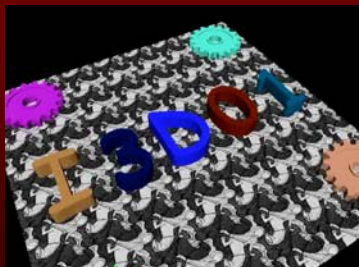


- Non-convex, rigid objects
- Frequent interlocking contacts
- Unconstrained, penalty-based

SIGGRAPH'03

M. C. Lin

GEARS

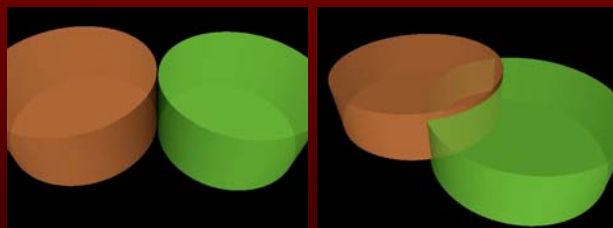


- Non-convex, rigid objects
- Less frequent interlocking contacts
- Unconstrained, penalty-based

SIGGRAPH'03

M. C. Lin

WAVY



- Non-convex, deformable objects
 - Continuous contact
 - Unconstrained, penalty-based
- Resolving extreme penetration

SIGGRAPH'03

M. C. Lin

Performance

Average Total Per-frame Proximity Query Times

| Demo | Objects | Lines | GeForce2 | InfiniteReality2 | ATI Rage Pro LT |
|-------|---------|-------|----------|------------------|-----------------|
| Map | 6 | 719 | 0.281ms | 0.901ms | 0.434ms |
| Gears | 13 | 391 | 0.015 | 0.026 | 0.064 |
| Links | 15 | 440 | 0.020 | 0.052 | 0.038 |
| Cars | 18 | 266 | 0.007 | 0.026 | 0.015 |
| Wavy | 2 | 200 | 1.030 | 2.360 | 2.990 |

Performance timings for dynamics simulations. The number of objects, number of line segments, and the average total time in milliseconds to run proximity queries on all objects in the scene per frame is reported. Timing data was gathered from three machines: a Pentium-III 933MHz desktop with a 64Mb GeForce2, a SGI 300MHz R12000 with InfiniteReality2 graphics, and a Pentium-III 750Mhz laptop with ATI Rage Pro LT graphics.

SIGGRAPH'03

M. C. Lin

Effects of Changes in Error Tolerance

Effects of Error Tolerance on Performance of Wavy

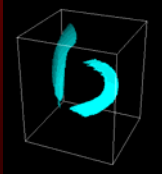
| Error | GeForce2 | InfiniteReality2 | ATI Rage Pro LT |
|-------|----------|------------------|-----------------|
| $d/4$ | 0.710ms | 1.270ms | 5.560ms |
| $d/2$ | 0.315 | 1.000 | 1.850 |
| d | 0.211 | 0.930 | 0.895 |
| $2d$ | 0.176 | 0.879 | 0.631 |
| $4d$ | 0.165 | 0.876 | 0.535 |

The effect on performance when changing the distance error tolerance d . We used proximity queries on the **wavy** demo with no collision response. The error determines the number of pixels used in the image-based operations. Systems with low graphics performance are more directly affected by the choice of d (see ATI Rage Pro LT); however, as the error is increased there is less dependence on graphics performance and the faster laptop CPU overtakes the InfiniteReality2 system.

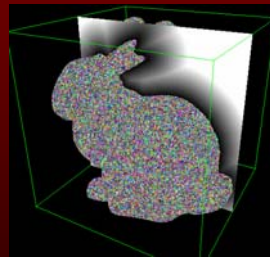
SIGGRAPH'03

M. C. Lin

Recent Results: Extension to 3D



3D intersections



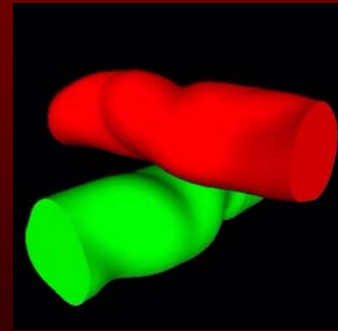
3D distance field

SIGGRAPH'03

M. C. Lin

Real-time queries: Deformable models

- Non-convex, complex 3D deformable objects
- ~20000 polygons each cylinder
- Continuous contact
- Unconstrained, penalty-based
- Intersection region in blue
- Distance field gradients in red and green
- No pre-computation



Resolving extreme penetration

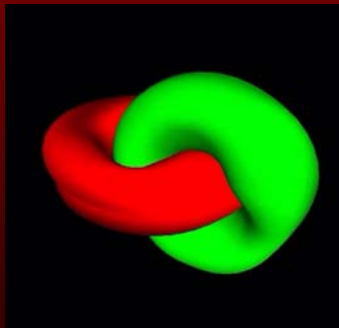
[Hoff, Zafarakis, Lin, Manocha'02]

SIGGRAPH'03

M. C. Lin

Real-time queries: Deformable models

- Non-convex, complex 3D deformable objects
- ~20000 polygons each cylinder
- Continuous contact
- Unconstrained, penalty-based
- Intersection region in blue
- Distance field gradients in red and green
- No pre-computation



Resolving extreme penetration

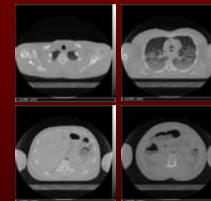
[Hoff, Zafarakis, Lin, Manocha'02]

SIGGRAPH'03

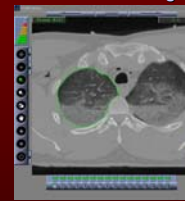
M. C. Lin

NURBS Torso Models [Seeger et al.'00]

Sample Slices from the Visible Human CT Data Set



SURFdriver Surface Reconstruction Program



NCAT Phantom (anterior view)



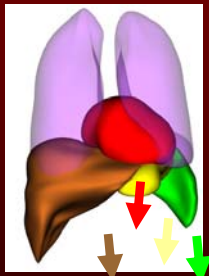
3D NURBS Organ Models



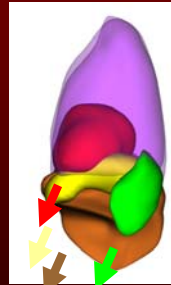
SIGGRAPH'03

M. C. Lin

4D NURBS Respiratory Model



Anterior View



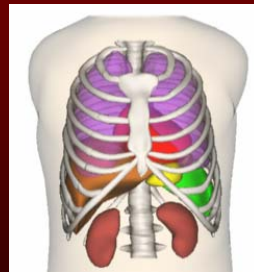
Left Lateral View

Motion Analysis of the Diaphragm, Heart, Liver, Spleen, and Stomach

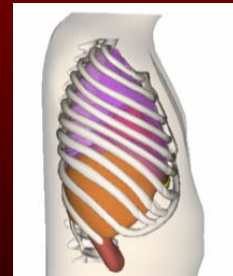
SIGGRAPH'03

M. C. Lin

Proximity Queries between Organs



Anterior View



Right Lateral View

[Seeger et al. '00]

SIGGRAPH'03

M. C. Lin

Modeling the Heart Structures

Heart exterior

Contact
computations
between Inner
chambers



SIGGRAPH'03

M. C. Lin

Proximity Queries using Graphics H/W

- Performs all proximity queries
- Handles general non-convex & deformable primitives
- Requires no pre-computation or complex hierarchical data structures
- Fast and efficient

SIGGRAPH'03

M. C. Lin

Some Observations

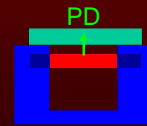
- Appropriate for most of applications where numerical accuracy is not the key
- Running time related to the rendering time
- Error bound provides smooth dial between performance and level of approximation
- Varying the max depth in the hierarchical localization allows balancing load between CPU and graphics

SIGGRAPH'03

M. C. Lin

Definition

- Penetration Depth (PD)
 - Minimum translational distance to separate two intersecting objects



SIGGRAPH'03

M. C. Lin

Motivation

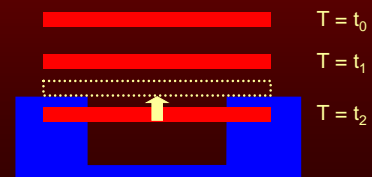
- Contact Handling in Rigid Body Simulation
 - Penalty-based method for contact resolution
 - Time stepping in general simulation framework

SIGGRAPH'03

M. C. Lin

Motivation

- Penalty-based Method

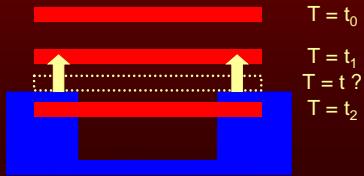


SIGGRAPH'03

M. C. Lin

Motivation

- Time stepping method
 - Estimate the time of collision (TOC)

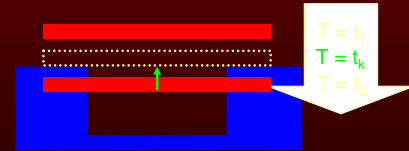


SIGGRAPH'03

M. C. Lin

Motivation

- Time stepping method using PD
 1. Compute the penetration depth
 2. Estimate the TOC by interpolation in time domain



SIGGRAPH'03

M. C. Lin

PD Applications

- Rigid body dynamic simulation
- Robot motion planning for autonomous agents and animated characters
- Haptic rendering
- Tolerance verification for CAD models

SIGGRAPH'03

M. C. Lin

Previous Work

- Convex polytopes
 - [Cameron '86 '97], [Dobkin et al. '93], [Agarwal et al. '00], [Bergen '01], [Kim et al. '02]
- Local solutions for deformable models
 - [Susan and Lin '01], [Hoff et al. '01]
- No solutions existed for non-convex models

SIGGRAPH'03

M. C. Lin

Overview

- Preliminaries
 - Minkowski sum-based Framework
- Basic PD Algorithm: A Hybrid Approach
- Acceleration Techniques
 - Object Space Culling
 - Hierarchical Refinement
 - Image Space Culling
- Application to Rigid Body Dynamic Simulation

SIGGRAPH'03

M. C. Lin

Overview

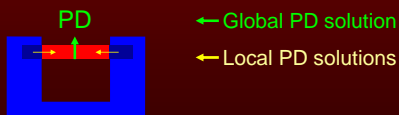
- Preliminaries
 - **Minkowski sum-based Framework**
- Basic PD Algorithm: A Hybrid Approach
- Acceleration Techniques
 - Object Space Culling
 - Hierarchical Refinement
 - Image Space Culling
- Application to Rigid Body Dynamic Simulation

SIGGRAPH'03

M. C. Lin

Preliminaries

- Local solutions might not have any relevance to a global solution

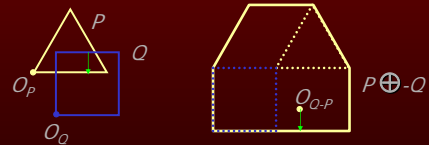


SIGGRAPH'03

M. C. Lin

Preliminaries

- Minkowski sum and PD
 - $P \oplus -Q = \{ p-q \mid p \in P, q \in Q \}$
 - PD := minimum distance between $O_{Q,P}$ and the surface of $P \oplus -Q$

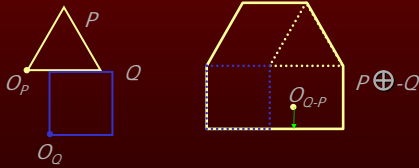


SIGGRAPH'03

M. C. Lin

Preliminaries

- Minkowski sum and PD
 - $P \oplus -Q = \{ p-q \mid p \in P, q \in Q \}$
 - PD := minimum distance between $O_{Q,P}$ and the surface of $P \oplus -Q$



SIGGRAPH'03

M. C. Lin

Preliminaries

- Decomposition property of Minkowski sum
 - If $P = P_1 \cup P_2$, then $P \oplus Q = (P_1 \oplus Q) \cup (P_2 \oplus Q)$
- Computing Minkowski sum
 - Convex: $O(n \log(n))$
 - where n is the number of features
 - Non-Convex: $O(n^6)$ computational complexity
 - In theory, use the convolution or the decomposition property
 - In practice, very hard to implement

SIGGRAPH'03

M. C. Lin

Overview

- Preliminaries
 - Minkowski sum-based Framework
- **Basic PD Algorithm: A Hybrid Approach**
- Acceleration Techniques
 - Object Space Culling
 - Hierarchical Refinement
 - Image Space Culling
- Application to Rigid Body Dynamic Simulation

SIGGRAPH'03

M. C. Lin

PD Algorithm : A Hybrid Approach

- $P \oplus Q = (P_1 \oplus Q) \cup (P_2 \oplus Q)$
 - where $P = P_1 \cup P_2$

SIGGRAPH'03

M. C. Lin

PD Algorithm : A Hybrid Approach

- $P \oplus Q = (P_1 \oplus Q) \cup (P_2 \oplus Q)$
 - where $P = P_1 \cup P_2$
- **Precomputation: Decomposition**

SIGGRAPH'03

M. C. Lin

PD Algorithm : A Hybrid Approach

- $P \oplus Q = (P_1 \oplus Q) \cup (P_2 \oplus Q)$
 - where $P = P_1 \cup P_2$
- Precomputation: Decomposition
- Runtime:
 - **Object Space: Pairwise Minkowski sum computation**

SIGGRAPH'03

M. C. Lin

PD Algorithm : A Hybrid Approach

- $P \oplus Q = (P_1 \oplus Q) \cup (P_2 \oplus Q)$
 - where $P = P_1 \cup P_2$
- Precomputation: Decomposition
- Runtime:
 - Object Space: Pairwise Minkowski sum computation
 - Image Space: Union by graphics hardware

[Kim, Otaduy, Lin & Manocha, SCA'01]

SIGGRAPH'03

M. C. Lin

PD Computation Pipeline

Precomputation

Convex
Decomposition

Run-time PD Query

Pairwise
Minkowski Sum
(Object Space)



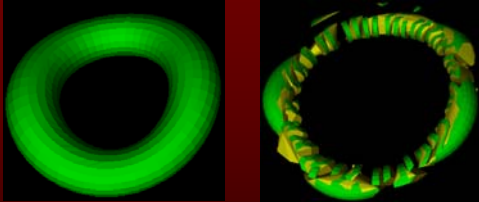
Closest
Point Query
(Image Space)

[Kim, Otaduy, Lin & Manocha, SCA'02]

SIGGRAPH'03

M. C. Lin

Convex Surface Decomposition



- [Ehmann and Lin '01]
- Decompose an object into a collection of convex surface patches
- Compute the convex hull of each surface patch

SIGGRAPH'03

M. C. Lin

Pairwise Minkowski Sum

- Algorithms

- Convex hull property of convex Minkowski sum
 - $P \oplus Q = \text{ConvHull}\{v_i + v_j \mid v_i \in V_P, v_j \in V_Q\}$, where P and Q are convex polytopes
- Topological sweep on Gauss map [Guibas '87]
- Incremental surface expansion [Rossignac '92]

SIGGRAPH'03

M. C. Lin

Closest Point Query

Goal

- Given a collection of convex Minkowski sums, compute the shortest distance from the origin to the surface of their union
- An exact solution is computationally expensive \Rightarrow Approximation using graphics hardware

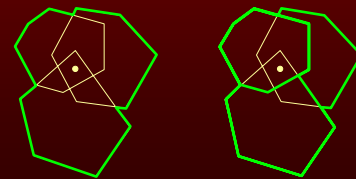
SIGGRAPH'03

M. C. Lin

Closest Point Query

- Main Idea

- Incrementally expand the current front of the boundary

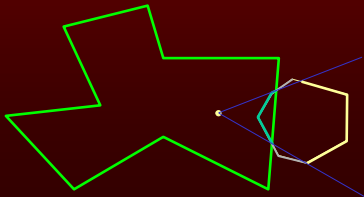


SIGGRAPH'03

M. C. Lin

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

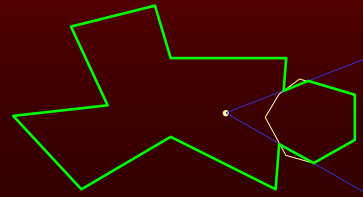


SIGGRAPH'03

M. C. Lin

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



SIGGRAPH'03

M. C. Lin

Overview

- Preliminaries
 - Minkowski sum-based Framework
- Basic PD Algorithm: A Hybrid Approach
- **Acceleration Techniques**
 - Object Space Culling
 - Hierarchical Refinement
 - Image Space Culling
- Application to Rigid Body Dynamic Simulation

SIGGRAPH'03

M. C. Lin

Motivation

- PD is shallow in practice
- Convex decomposition has $O(n)$ convex pieces in practice
 - Culling strategy is suitable and very effective

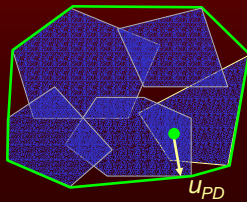
SIGGRAPH'03

M. C. Lin

Object Space Culling

- Basic Idea

- If we know the upper bound on PD, u_{PD} , we do not need to compute the Mink. sum of pairs whose Euclidean dist is more than u_{PD}



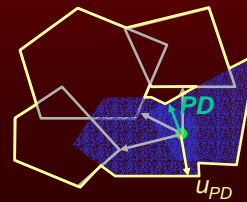
SIGGRAPH'03

M. C. Lin

Object Space Culling

- Basic Idea

- If we know the upper bound on PD, u_{PD} , we do not need to compute the Mink. sum of pairs whose Euclidean dist is more than u_{PD}



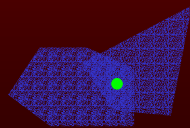
SIGGRAPH'03

M. C. Lin

Object Space Culling

- Basic Idea

- If we know the upper bound on PD, u_{PD} , we do not need to compute the Mink. sum of pairs whose Euclidean dist is more than u_{PD}



SIGGRAPH'03

M. C. Lin

Image Space Culling


- Rendering only once for the Minkowski sums containing the origin
- Refine the upper bound for every view frustum
- View frustum culling

SIGGRAPH'03

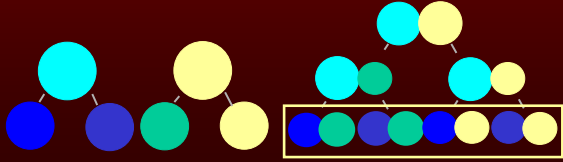
M. C. Lin

Hierarchical Culling

- BVH Construction (Ehmann and Lin '01)




- Hierarchical Culling



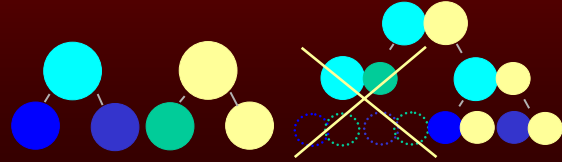
SIGGRAPH'03 M. C. Lin

Hierarchical Culling

- BVH Construction (Ehmann and Lin '01)



- Hierarchical Culling



SIGGRAPH'03 M. C. Lin

Hierarchical Refinement

Precomputation

Convex
Decomposition

→

BVH
Construction

Run-time PD Query

Culling

→

Pairwise
Minkowski Sum

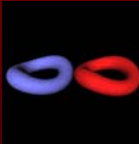
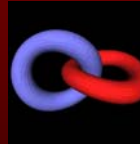
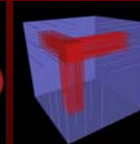

→

Closest
Point Query

Refine the current PD estimate, and
go to the next level of BVH

SIGGRAPH'03 M. C. Lin

PD Benchmarks

| Touching Tori | Interlocked Tori | Interlocked Grates | Touching Alphabets |
|---|--|---|---|
|  |  |  |  |
| 0.3 sec (4 hr) | 3.7 sec (4 hr) | 1.9 sec (177 hr) | 0.4 sec (7 min) |

- With Accel. (Without Accel.)

SIGGRAPH'03 M. C. Lin

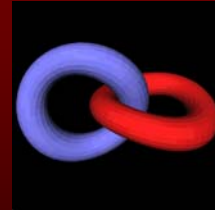
PD Benchmarks

| Models | Tri | Convex Pieces | PD w/o Accel. | PD w/ Accel |
|--------------------|-----------|---------------|---------------|-------------|
| Touching Tori | 2000 | 67 | 4 hr | 0.3 sec |
| Interlocked Tori | 2000 | 67 | 4 hr | 3.7 sec |
| Interlocked Grates | 444, 1134 | 169, 409 | 177 hr | 1.9 sec |
| Touching Alphabets | 144,152 | 42, 43 | 7 min | 0.4 sec |

SIGGRAPH'03

M. C. Lin

Hierarchical Culling Example



| Level | Cull Ratio | Mink Sum | HW Query | PD _{est} |
|-------|------------|-----------|-----------|-------------------|
| 3 | 31.2% | 0.219 sec | 0.220 sec | 0.99 |
| 5 | 96.7% | 0.165 sec | 0.146 sec | 0.53 |
| 7 | 98.3% | 1.014 sec | 1.992 sec | 0.50 |

SIGGRAPH'03

M. C. Lin

Implementation

- SWIFT++ [Ehmann and Lin '01]
- QHULL
- OpenGL for closest point query

SIGGRAPH'03

M. C. Lin

Implementation

```

void DrawUnionOfConvex(ConvexObj *ConvexObjs, int NumConvexObjs)
{
    glClearDepth(0);
    glClearStencil(0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT | GL_STENCIL_BUFFER_BIT);

    glEnable(GL_DEPTH_TEST);
    glEnable(GL_STENCIL_TEST);
    for (int i=0; i<NumConvexObjs; i++)
        for (int j=0; j<NumConvexObjs; j++)
        {
            glDepthMask(0);
            glColorMask(0,0,0,0);
            glDepthFunc(GL_LESS);
            glStencilFunc(GL_ALWAYS,1,1);
            glStencilOp(GL_KEEP,GL_REPLACE,GL_KEEP);
            ConvexObjs[j].DrawFrontFaces();

            glDepthMask(1);
            glColorMask(1,1,1,1);
            glDepthFunc(GL_GREATER);
            glStencilFunc(GL_EQUAL,0,1);
            glStencilOp(GL_ZERO,GL_KEEP,GL_KEEP);
            ConvexObjs[j].DrawBackFaces();
        }
}

```

SIGGRAPH'03

M. C. Lin

Accuracy of PD computation

- Our algorithm computes an upper bound to PD
- Image space computation determines the tightness of the upper bound
 - Pixel resolution
 - Z-buffer precision
- In practice, with 256×256 pixel resolution, the algorithm rapidly converges to the PD

SIGGRAPH'03

M. C. Lin

Overview

- Preliminaries
 - Minkowski sum-based Framework
- Basic PD Algorithm: A Hybrid Approach
- Acceleration Techniques
 - Object Space Culling
 - Hierarchical Refinement
 - Image Space Culling
- **Application to Rigid Body Dynamic Simulation**

SIGGRAPH'03

M. C. Lin

Application to Rigid Body Simulation

- Interpenetration is often unavoidable in numerical simulations
- Need for a consistent and accurate measure of PD
- Penalty-based Method
 - $F = (k \cdot d)n$
 - d : PD, n : PD direction, k : stiffness constant

SIGGRAPH'03

M. C. Lin

Application to Rigid Body Simulation

- Time Stepping Method
 - Estimate the time of collision (TOC)
 - s : separation dist before interpenetration
 - d : PD, n : PD dir after interpenetration
 - v_s : relative velocity of closest features
 - v_d : relative velocity of PD features

SIGGRAPH'03

M. C. Lin

Application to Rigid Body Simulation

– $x(t)$: 1D distance function between closest features and PD features projected to PD direction

– $x(0) = s, x(T) = d$

– $dx/dt(0) = \mathbf{v}_s \cdot \mathbf{n}, dx/dt(T) = \mathbf{v}_d \cdot \mathbf{n}$

– Compute the roots of $x(t) = 0$

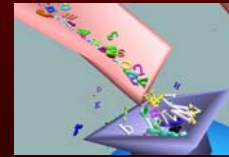
SIGGRAPH'03

M. C. Lin

Rigid Body Simulation Demo



Example 1
Average complexity: 250 triangles
60 Convex Pieces / Object



Example 2
Average complexity: 250 triangles
200 letters and alphabets

SIGGRAPH'03

M. C. Lin

Rigid Body Simulation Demo



SIGGRAPH'03

M. C. Lin

Summary

- First practical PD algorithm using a hybrid approach
- Acceleration techniques:
 - Object space culling
 - Image space culling
 - Hierarchical refinement
- Application to rigid body simulation

SIGGRAPH'03

M. C. Lin

Proximity Queries

- **Object-Space Methods**
 - Considerable pre-processing for hierarchy construction
 - Difficult to achieve real-time performance on complex deformable and/or breaking models
- **Image-Space Techniques**
 - Primitive rasterization ↔ sorting in screen-space
 - Applicable to interference tests

SIGGRAPH'03

M. C. Lin

Problems with Image-Based Methods

- Closed models
- Frame buffer readbacks – slow
 - NVIDIA GeForce 4, Dell Precision Workstation with 1Kx1K depth buffer taking 50ms

SIGGRAPH'03

M. C. Lin

Goals

- Interactive Performance
- Complex objects
 - Large number of objects
 - High primitive count
 - Non-convex objects
 - Open and closed objects
- Non-Rigid Motion
 - Deformable bodies
 - Changing topology

SIGGRAPH'03

M. C. Lin

Overview

- **Potentially Colliding Set (PCS) computation**
- **Exact collision tests on the final PCS**



GPU based PCS
computation

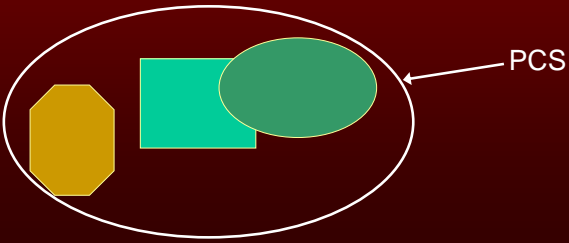
Using
CPU

SIGGRAPH'03

M. C. Lin

Potentially Colliding Set (PCS)

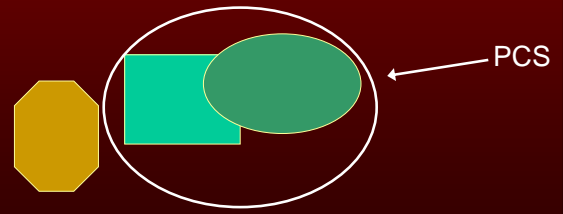
- A PCS consists of the set of objects that can potentially collide with each other



SIGGRAPH'03

M. C. Lin

Potentially Colliding Set (PCS)

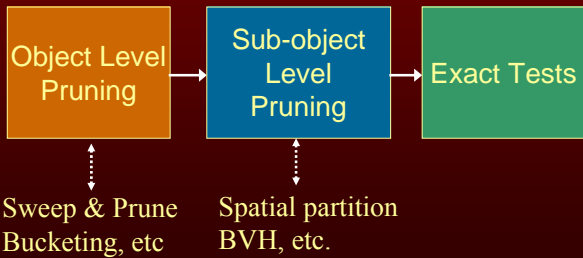


SIGGRAPH'03

M. C. Lin

Comparison

- Object Level Pruning ↔ Broad Phase
- Sub-object Level Pruning ↔ Narrow Phase



SIGGRAPH'03

M. C. Lin

Visibility Computations

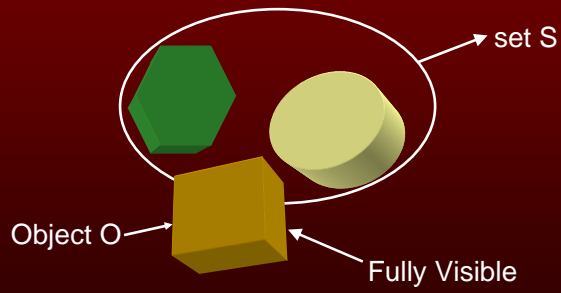
Lemma 1: *An object O does not collide with a set of objects S if O is fully visible with respect to S*

→ Utilize visibility for PCS computation

SIGGRAPH'03

M. C. Lin

Collision Detection using Visibility Computations



SIGGRAPH'03

M. C. Lin

PCS Pruning

Lemma 2: Given n objects O_1, O_2, \dots, O_n , an object O_i does not belong to PCS if it does not collide with $O_1, \dots, O_{i-1}, O_{i+1}, \dots, O_n$

→ Prune objects that do not collide

SIGGRAPH'03

M. C. Lin

PCS Pruning

$O_1 O_2 \dots O_{i-1} O_i O_{i+1} \dots O_{n-1} O_n$

SIGGRAPH'03

M. C. Lin

PCS Pruning

$O_1 O_2 \dots O_{i-1} O_i$

SIGGRAPH'03

M. C. Lin

PCS Pruning

$O_i O_{i+1} \dots O_{n-1} O_n$

SIGGRAPH'03

M. C. Lin

PCS Computation

- Each object tested against all objects but itself
- Naive algorithm is $O(n^2)$
- Linear time algorithm
 - Uses two pass rendering approach
 - Conservative solution

SIGGRAPH'03

M. C. Lin

PCS Computation: First Pass

Render →

$O_1 O_2 \dots O_{i-1} O_i O_{i+1} \dots O_{n-1} O_n$

SIGGRAPH'03

M. C. Lin

PCS Computation: First Pass

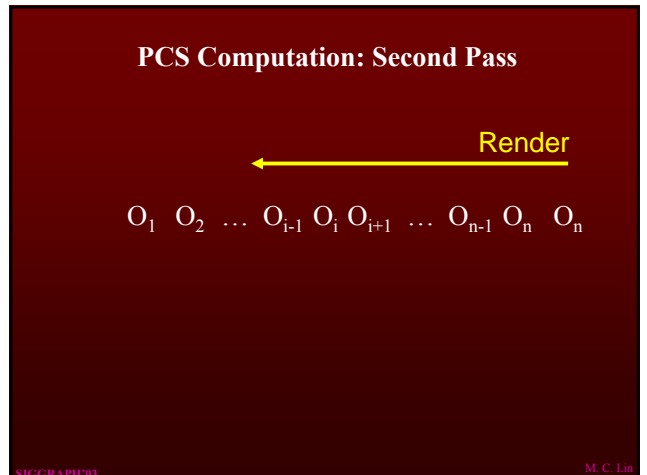
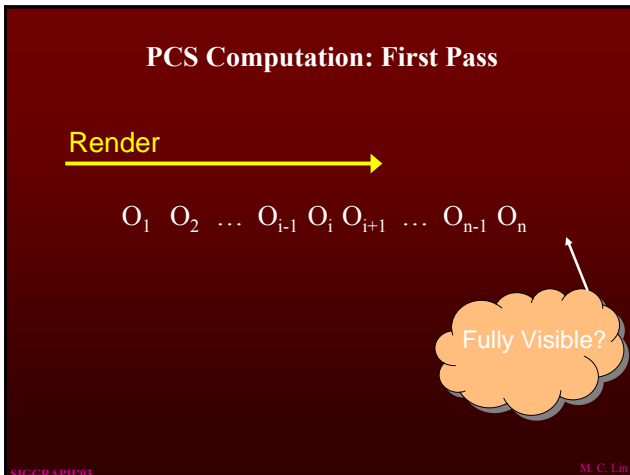
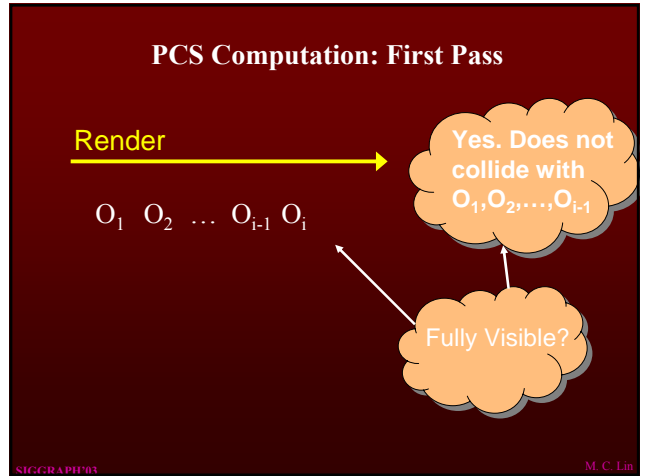
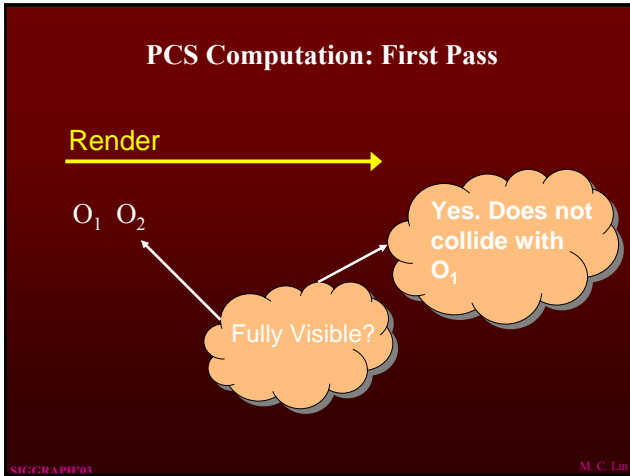
Render →

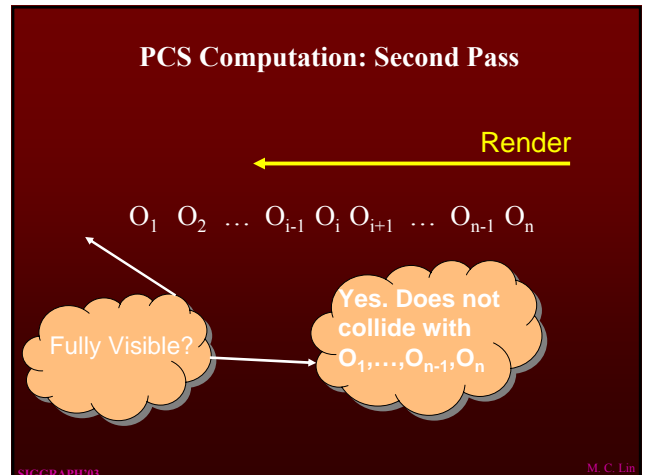
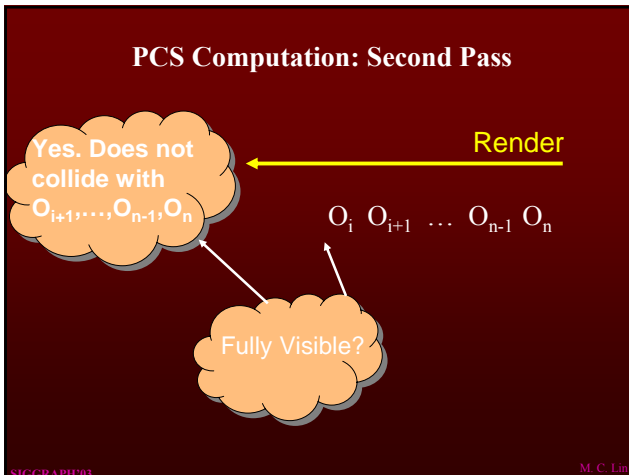
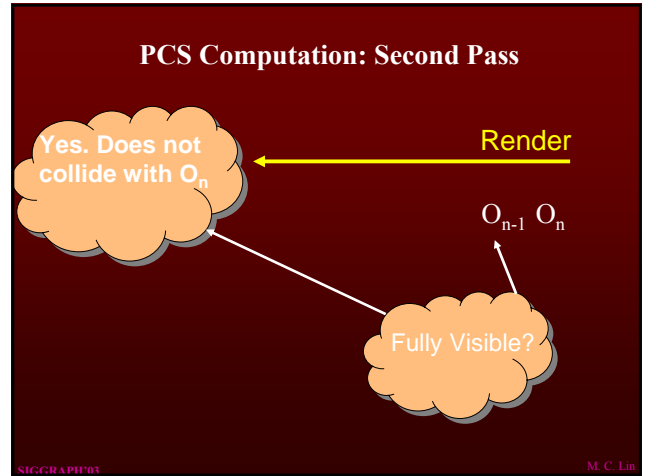
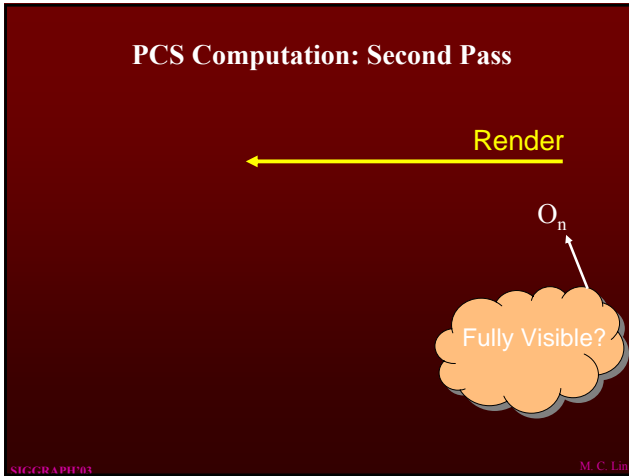
O_1

Fully Visible?

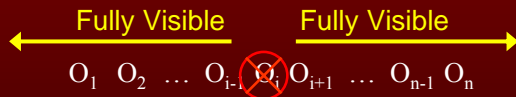
SIGGRAPH'03

M. C. Lin





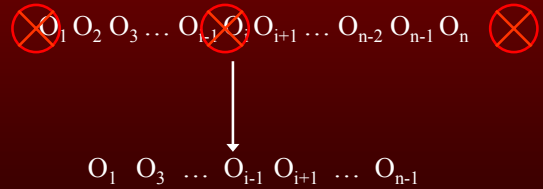
PCS Computation



SIGGRAPH'03

M. C. Lin

PCS Computation



SIGGRAPH'03

M. C. Lin

Sub-Object Level: Overlap Localization

- Each object is composed of sub-objects
- We are given n objects O_1, \dots, O_n
- Compute sub-objects of each object O_i that overlap with sub-objects of other objects

SIGGRAPH'03

M. C. Lin

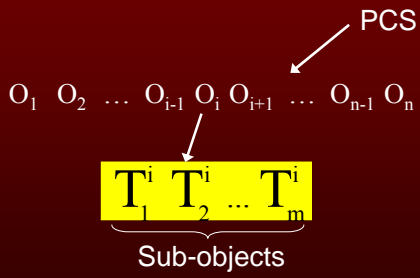
Sub-Object Level: Overlap Localization

- Our solution
 - Test if each sub-object of O_i overlaps with sub-objects of O_1, \dots, O_{i-1}
 - Test if each sub-object of O_i overlaps with sub-objects of O_{i+1}, \dots, O_n
- Linear time algorithm
- Extend the two pass approach

SIGGRAPH'03

M. C. Lin

Sub-Object Level: Overlap Localization



SIGGRAPH'03

M. C. Lin

Sub-Object Overlap Localization: First Pass

Render sub-objects

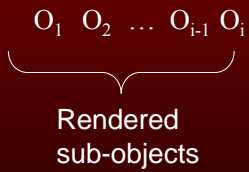


SIGGRAPH'03

M. C. Lin

Sub-Object Overlap Localization: First Pass

Render sub-objects

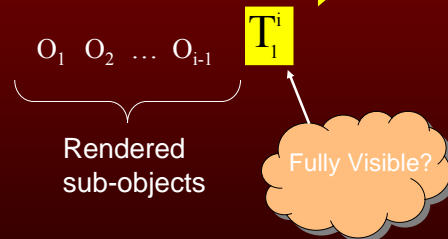


SIGGRAPH'03

M. C. Lin

Sub-Object Overlap Localization: First Pass

Render sub-objects



SIGGRAPH'03

M. C. Lin

Sub-Object Overlap Localization: First Pass

Render sub-objects



$O_1 O_2 \dots O_{i-1}$

T_2^i

Rendered sub-objects

Fully Visible?

SIGGRAPH'03

M. C. Lin

Sub-Object Overlap Localization: First Pass

Render sub-objects



$O_1 O_2 \dots O_{i-1}$

T_m^i

Rendered sub-objects

Fully Visible?

Yes. Does not collide with sub-objects of O_1, O_2, \dots, O_{i-1}

SIGGRAPH'03

M. C. Lin

Sub-Object Overlap Localization: First Pass

Render sub-objects



$O_1 O_2 \dots O_{i-1}$

T_m^i

Rendered sub-objects

Fully Visible?

Avoids self-collisions!

SIGGRAPH'03

M. C. Lin

Sub-Object Overlap Localization: First Pass

Render sub-objects



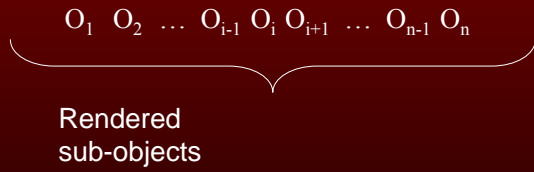
$O_1 O_2 \dots O_{i-1} O_i$

Rendered sub-objects

SIGGRAPH'03

M. C. Lin

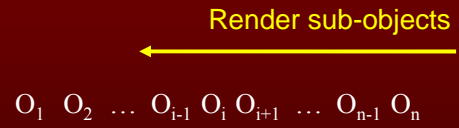
Sub-Object Overlap Localization: First Pass



SIGGRAPH'03

M. C. Lin

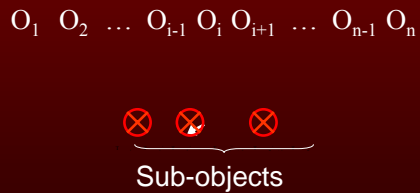
Sub-Object Overlap Localization: Second Pass



SIGGRAPH'03

M. C. Lin

Sub-Object Level Overlap Localization



SIGGRAPH'03

M. C. Lin

Visibility Queries

- We require a query
 - Tests if a primitive is fully visible or not
- Current hardware supports occlusion queries
 - Test if a primitive is visible or not
- Our solution
 - Change the sign of depth function

SIGGRAPH'03

M. C. Lin

Visibility Queries

| | | Depth function | |
|---------------|------|----------------|------|
| | | GEQUAL | LESS |
| All fragments | Pass | Pass | Fail |
| | Fail | Fail | Pass |

Occlusion query Query not supported

Examples - HP_Occlusion_test, NV_occlusion_query

SIGGRAPH'03

M. C. Lin

Bandwidth Analysis

- Read back only integer identifiers
 - Independent of screen resolution

SIGGRAPH'03

M. C. Lin

Optimizations

- First use AABBs as object bounding volume
- Use orthographic views for pruning
- Prune using original objects

SIGGRAPH'03

M. C. Lin

Implementation

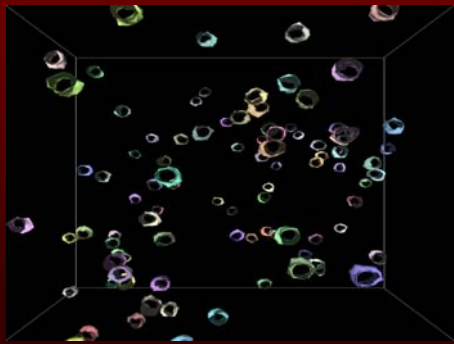
- Dell Precision workstation
- Dell M50 Laptop

[Govindaraju, Redon, Lin, Manocha, GH'03]

SIGGRAPH'03

M. C. Lin

Test Models: Environment 1

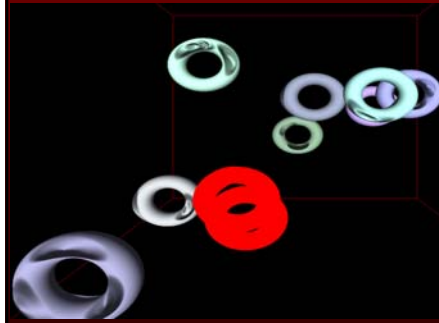


- 100 deforming cylinders
- Total – 20K triangles

SIGGRAPH'03

M. C. Lin

Test Models: Environment 2

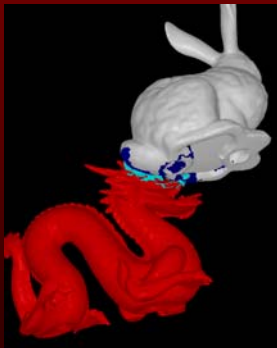


- Deforming torii
- Each torus – 20K triangles

SIGGRAPH'03

M. C. Lin

Test Model: Environment 3



- 250K Dragon
- 35K Bunny

SIGGRAPH'03

M. C. Lin

Test Model: Environment 4



- Breaking dragon – 250K triangles
- Bunny – 35K triangles

[Govindaraju, Redon, Lin, Manocha, GH'03]

SIGGRAPH'03

M. C. Lin

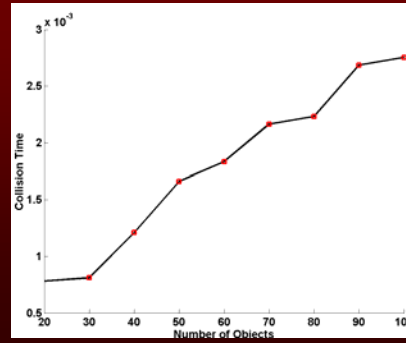
Live Demo of Environment 4

- Dell M50 laptop, 2.4GHz Pentium IV-M CPU, NVIDIA Quadro4 700GoGL GPU, 1GB memory running Windows XP

SIGGRAPH'03

M. C. Lin

Performance

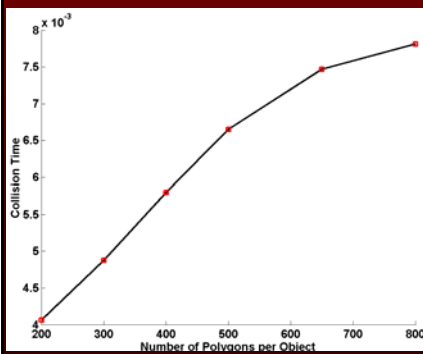


Number of objects v/s collision time

SIGGRAPH'03

M. C. Lin

Performance

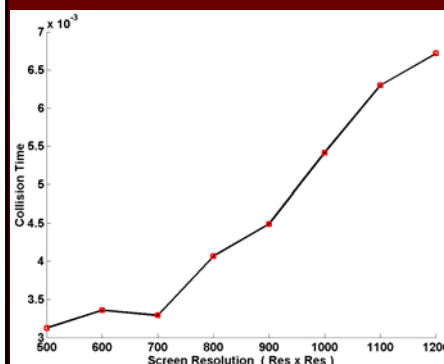


Number of polygons v/s collision time

SIGGRAPH'03

M. C. Lin

Performance



Screen resolution v/s collision time

SIGGRAPH'03

M. C. Lin

Advantages

- No coherence
- No assumptions on motion of objects
- Works on generic models
- A fast pruning algorithm
- No frame-buffer readbacks
- Applicable to deformable & breaking models

SIGGRAPH'03

M. C. Lin

Limitations

- No distance or penetration depth information
- Resolution limited accuracy
- No self-collision detection

SIGGRAPH'03

M. C. Lin

Summary

- First collision detection using GPU
 - Applicable to polygon soups, open boundary, etc.
 - No assumptions on motion
 - Allowing change of topology and dynamic geometry
 - Unified approach for object & sub-object level culling
- Linear time PCS computation algorithm
- No frame buffer readbacks

SIGGRAPH'03

M. C. Lin

Organization

- Motion Planning
- Proximity Queries
- **Physically-based Simulation**

SIGGRAPH'03

M. C. Lin

Visual Simulation of Ice Growth



SIGGRAPH'03

M. C. Lin

Motivation

- Special Effects
- Computer Animation



A scene from X-Men 2

SIGGRAPH'03

M. C. Lin

Motivation

- “We built a particle system that produced the effect of fingers of ice ... which was tricky, since the particles didn't do that automatically”
- Arnon Manor, Cinesite

SIGGRAPH'03

M. C. Lin

Our approach

- A method for modeling and simulating visually complex structure of ice – both geometrically and optically
 - Physical Simulation using Phase Field Methods
 - Simulation Acceleration using GPUs & Banded Opt.
 - Controlling Complexity
 - Post-Processing for Rendering

[Kim and Lin, SCA'03]

SIGGRAPH'03

M. C. Lin

Phase Field Methods

- Temperature PDE

$$\frac{\partial T}{\partial t} = a^2 \nabla^2 T + K \frac{\partial p}{\partial t}$$

- Phase PDE

$$\tau \frac{\partial p}{\partial t} = \nabla \cdot (\varepsilon^2 \nabla p) - \frac{\partial}{\partial x} \left(\varepsilon \frac{\partial \varepsilon}{\partial \theta} \frac{\partial p}{\partial y} \right) + \frac{\partial}{\partial y} \left(\varepsilon \frac{\partial \varepsilon}{\partial \theta} \frac{\partial p}{\partial x} \right) + p(1-p) \left(p - \frac{1}{2} + m \right)$$

SIGGRAPH'03

M. C. Lin

Ice Morphology



dendritic

plate

isotropic

SIGGRAPH'03

M. C. Lin

GPU Acceleration

- Fields map easily to GPU
- Framework of [Harris et al. 2002]
 - Fields sent as textures
 - Fragment program runs PDE per texel
 - ~100 lines of Cg!

SIGGRAPH'03

M. C. Lin

Performance

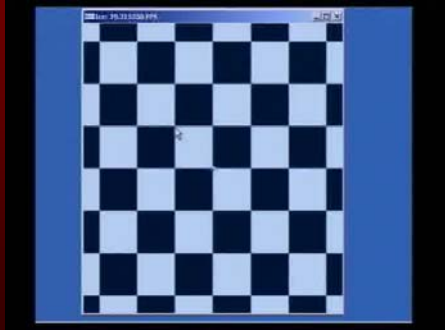
- Only unbanded currently possible
- No early exit yet on GPU

| Grid Size | CPU(Hz) | GPU(Hz) | Speedup |
|-----------|---------|---------|---------|
| 64x64 | 250 | 624 | 2.50x |
| 128x128 | 25 | 236 | 9.44x |
| 256x256 | 8 | 67.47 | 8.43x |
| 512x512 | 3.5 | 17.67 | 5.05x |
| 1024x1024 | 1.08 | 3.77 | 3.49x |

SIGGRAPH'03

M. C. Lin

Real-Time Capture



[Kim and Lin, SCA'03]

SIGGRAPH'03

M. C. Lin

Future Work

- Extension to motion planning of deformable robots and other types of constraints
- PD computation for rotational movement using GPU
- Continuous Collision Detection using GPUs
- Physically-based modeling of paint media, phase transition (melting, solidification, etc)

SIGGRAPH'03

M. C. Lin

Collaborators

Bill Baxter
Tim Culver
Mark Foskey
Maxim Garber
Naga Govindaraju
Kenny Hoff
John Keyser
Theodore Kim
Young Kim
Dinesh Manocha
Miguel Otaduy
Charles Pisula
Stephan Redon
Andrew Zaferakis

SIGGRAPH'03

M. C. Lin

Acknowledgements

Stephen Ehmann
Stefan Gottschalk
Sarah Hoff
David Hsu
Eric Larsen
Jean-Claude Latombe
Jean-Paul Laumond

SIGGRAPH'03

M. C. Lin

Acknowledgements

Army Research Office

DOE ASCI Program

Intel Corporation

National Institute of Health

National Science Foundation

Office of Naval Research

University of North Carolina at Chapel Hill

SIGGRAPH'03

M. C. Lin

The End

SIGGRAPH'03

M. C. Lin