Collision Detection for Large Gaming Environments

Young J. Kim (김영준)
http://graphics.ewha.ac.kr
Dept of Computer Sci and Eng
Ewha Womans University

Lecture Information

- Slide credits
  - Ming C. Lin (UNC)
  - Stephane Redon (UNC/INRIA)
  - Naga Govindaraju (UNC)
- Target audience
  - Intermediate level
- This lecture note will be available at http://graphics.ewha.ac.kr

Lecture Goal

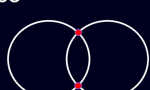
- Learn the basics of discrete and continuous collision detection algorithms as well as their differences
- Discrete collision detection
  - OBB trees (CPU-based)
  - CULLIDE (GPU-based)
- Continuous collision detection
  - Polygonal model
  - Articulated model
    - Simplified
    - Generic

Motivation

- In a cool game like HλLF-LIFE²:
  - Real time dynamics simulation
  - Path planning
  - Collision avoidance of artificial characters
  - Interactive navigation
- What do we need to correctly implement these features?

Problem Statement

- Collision detection: check whether two objects overlap in space



- If a collision occurs
  - Report collision witness features
  - Report the first time of collision
  - Report minimum translational distance to separate the objects (penetration depth)

Applications

- Ubiquitous by nature
  - Robot motion planning
  - Dynamic simulation
  - Haptic rendering
  - Virtual prototyping
  - Interactive walkthroughs
  - Molecular modeling
  - Rapid prototyping
- Computational bottleneck

Assumptions

- Input data
  - Polygonal mesh
  - No topology information is available; i.e., polygonal soup
- Available information
  - Positions and orientations at discrete time samples

Discrete Collision Detection

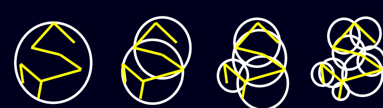
- OBB trees [GLM96]
- CULLIDE [GRLM03]

CD Between Primitives

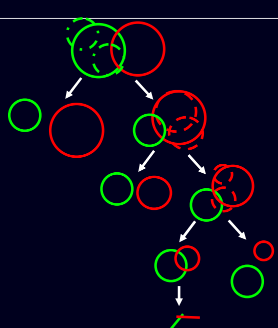
- References
  [Möl97]
  [Held97]
  [Ebe05]
- http://www.realtimerendering.com/int/

Bounding Volume Hierarchies

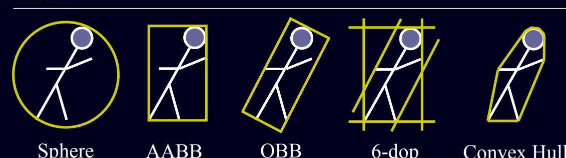
- Model hierarchy:
  - Each node has a simple volume that bounds a set of triangles
  - Children contain volumes that each bound a different portion of the parent's triangles
  - The leaves of the hierarchy usually contain individual triangles
- A binary BVH



BVH-Based Collision Detection



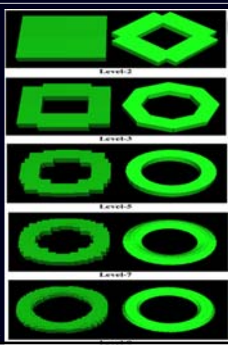
Trade-Off In Choosing BVs



Increasing complexity & tightness of fit

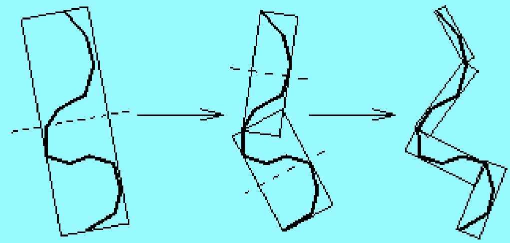
Decreasing cost of (overlap tests + BV update)

Example: AABB VS OBB



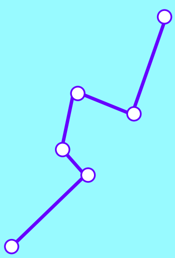
Approximation of a Torus

Building an OBBTree



Recursive top-down construction: partition and refit

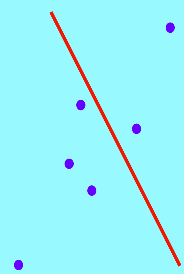
Building an OBB Tree



Given some polygons, consider their vertices...

Building an OBB Tree

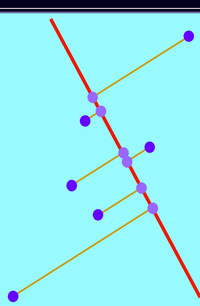
… and an arbitrary line



Building an OBB Tree

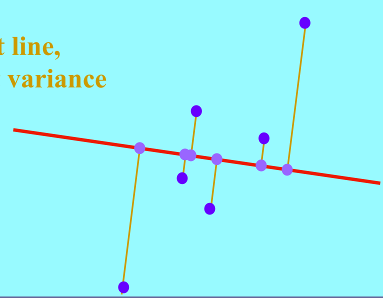
Project onto the line

Consider variance of distribution on the line

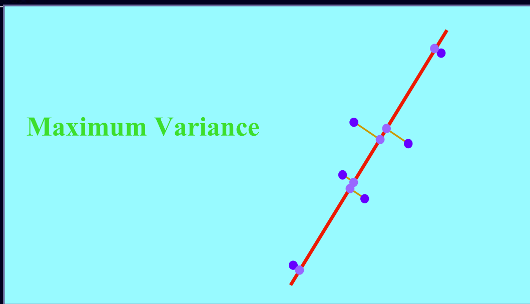


Building an OBB Tree

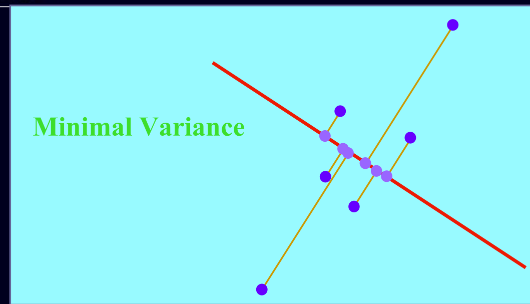
Different line, different variance



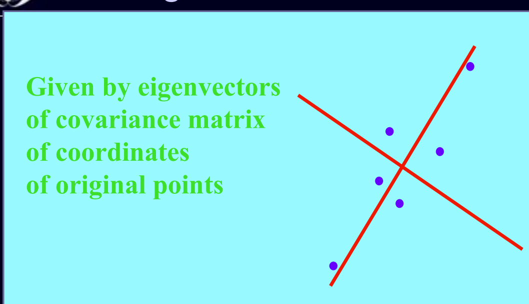
Building an OBB Tree



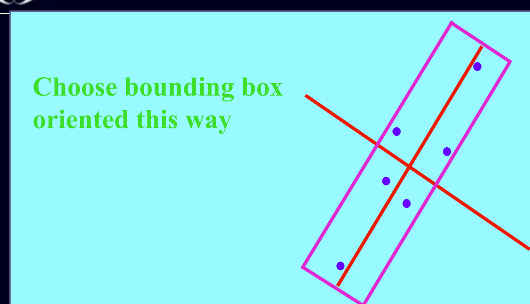
Building an OBB Tree



Building an OBB Tree



Building an OBB Tree



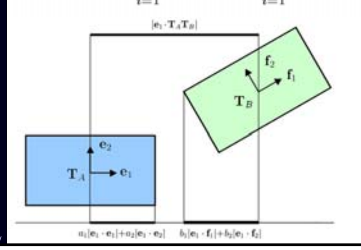
Tree Traversal

Hierarchy of tests

Separating Axis Theorem

- The boxes are disjoint if

$$|\mathbf{a} \cdot \mathbf{T}_A\mathbf{T}_B| > \sum_{i=1}^{3} a_i|\mathbf{a} \cdot \mathbf{e}_i| + \sum_{i=1}^{3} b_i|\mathbf{a} \cdot \mathbf{f}_i|$$



Separating Axis Theorem

Two polytopes A and B are disjoint iff there exists a separating axis which is:

perpendicular to a face from either
or
perpendicular to an edge from each

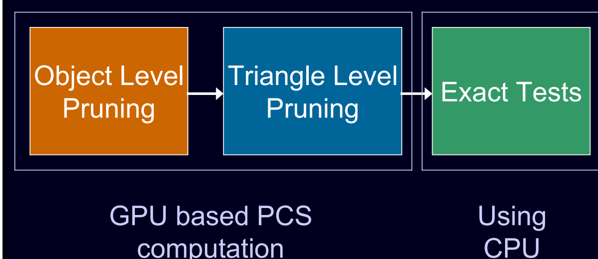
Implications of Theorem

Given two generic polytopes, each with E edges and F faces, number of candidate axes to test is:
$2F + E^2$

OBBs have only E = 3 distinct edge directions, and only F = 3 distinct face normals. OBBs need at most 15 axis tests.

Because edge directions and normals each form orthogonal frames, the axis tests are rather simple.

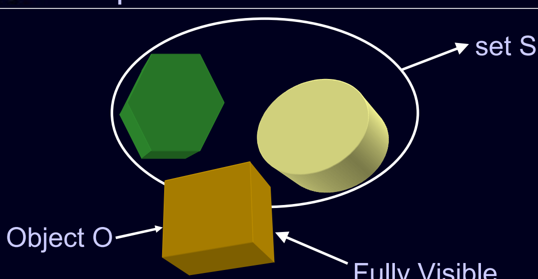
CULLIDE Algorithm



Visibility Computations

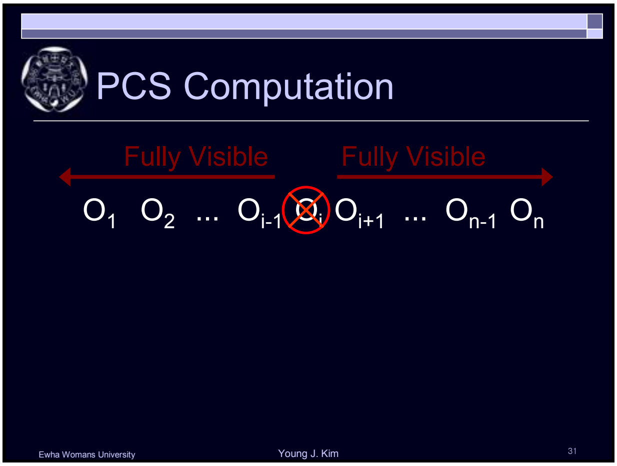
Lemma 1: *An object O does not collide with a set of objects S if O is fully visible with respect to S*

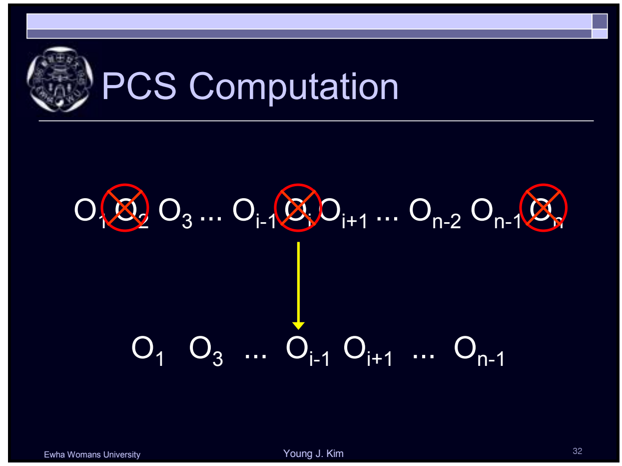
Collision Culling Using Visibility Computations

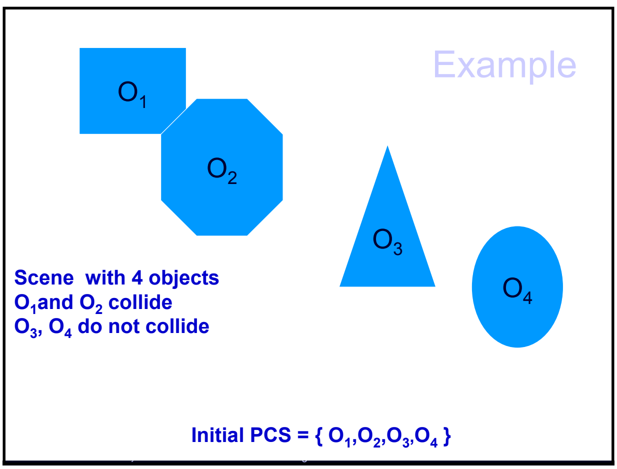


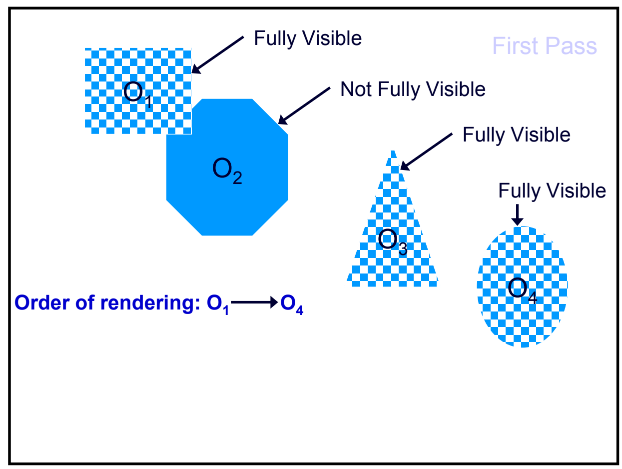
Potentially Colliding Set

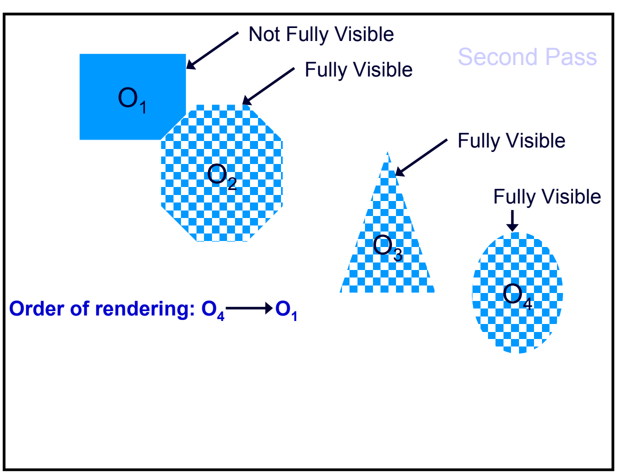
Lemma 2: *Given n objects $O_1, O_2, \ldots, O_n$, an object O_i does not belong to PCS if it does not collide with $O_1, \ldots, O_{i-1}, O_{i+1}, \ldots, O_n$*

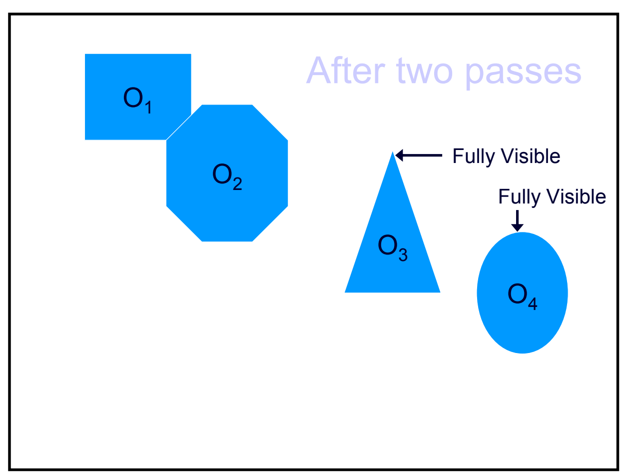

PCS Computation
Fully Visible
Fully Visible
O_1 O_2 … O_{i-1} O_i O_{i+1} … O_{n-1} O_n
Ewha Womans University
Young J. Kim
31


PCS Computation
O_1 O_2 O_3 … O_{i-1} O_i O_{i+1} … O_{n-2} O_{n-1} O_n
O_1 O_3 … O_{i-1} O_{i+1} … O_{n-1}
Ewha Womans University
Young J. Kim
32

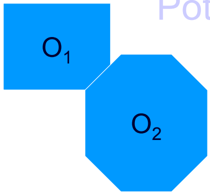

Example
O_1
O_2
O_3
O_4
Scene with 4 objects
O_1 and O_2 collide
O_3, O_4 do not collide
Initial PCS = { O_1,O_2,O_3,O_4 }


First Pass
Fully Visible
Not Fully Visible
Fully Visible
Fully Visible
O_1
O_2
O_3
O_4
Order of rendering: $O_1 \rightarrow O_4$


Second Pass
Not Fully Visible
Fully Visible
Fully Visible
Fully Visible
O_1
O_2
O_3
O_4
Order of rendering: $O_4 \rightarrow O_1$


After two passes
O_1
O_2
Fully Visible
Fully Visible
O_3
O_4

Potential Colliding Set



PCS $=\{O_1, O_2\}$

Visibility Queries

- We require a query
  - Tests if a primitive is fully visible or not
- Modern GPU hardware supports occlusion queries
  - Test if a primitive is visible or not
  - Examples - HP_Occlusion_test, NV_occlusion_query

Demo

Continuous Collision Detection

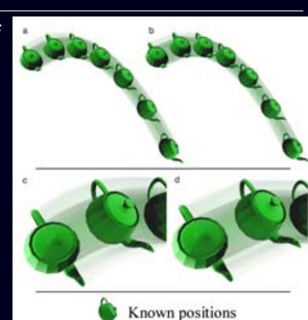
- General Polygonal Models [RKC00]
- Articulated Models [RKLM04a, RKLM04b]

Discrete VS Continuous CD

- Discrete collision detection
  - Detect interpenetrations only at successive positions
- Continuous collision detection

Continuous Collision Detection

- The actual motion of the object is not known
- Use an arbitrary in-between motion



CCD Between Primitives

- Only two (non-degenerate) types
  - Edge/Edge case
  - Vertex/Face case
- How to solve
  - Algebraically [Can86, RKC00]
  - Interval-based root finding [RKC02]

Continuous Overlap Test

- Continuous OBB test [RKC00]

$$|\mathbf{a} \bullet \mathbf{T}_A \mathbf{T}_B| > \sum_{i=1}^{3} a_i |\mathbf{a} \bullet \mathbf{e}_i| + \sum_{i=1}^{3} b_i |\mathbf{a} \bullet \mathbf{f}_i|$$

  - e_1, e_2, e_3, f_1, f_2, f_3: OBB axes
  - a_1, a_2, a_3, b_1, b_2, b_3: half-size OBB axes
  - T_A, T_B: OBB centers
  - a: separating axis
- Apply the interval arithmetic to the LHS and RHS

Interval Arithmetic

$$I = [a,b] = \{x \in \mathbb{R},\ a \leqslant x \leqslant b\}$$

$$\begin{aligned} I_n &= [a_1,b_1] \times \ldots \times [a_n,b_n] \\ &= \{\mathbf{x} = (x_1,\ldots,x_n) \in \mathbb{R}^n,\ a_i \leqslant x_i \leqslant b_i \quad \forall i,\ \ 1 \leqslant i \leqslant n\} \end{aligned}$$

$$\begin{aligned} [a,b] + [c,d] &= [a+c, b+d] \\ [a,b] - [c,d] &= [a-d, b-c] \\ [a,b] \times [c,d] &= [min(ac,ad,bc,bd), max(ac,ad,bc,bd)] \\ 1/[a,b] &= [1/b, 1/a] \quad \text{if} \quad a > 0 \text{ or } b < 0 \\ [a,b]/[c,d] &= [a,b] \times (1/[c,d]) \quad \text{if} \quad c > 0 \text{ or } d < 0 \\ [a,b] &\leqslant [c,d] \quad \text{if} \quad b \leqslant c \end{aligned}$$

Demo – Engine Removal

CCD for Articulated Models

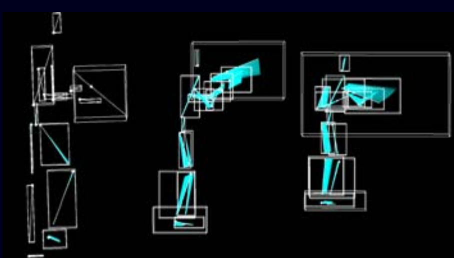
- Avatars in Virtual Environments [RKLM04a]
  - Articulated body

Pipeline

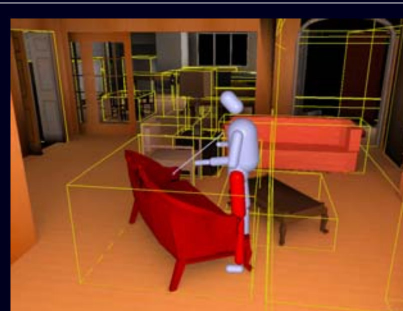
Motion Interpolation → BVH Construction → BVH Culling → SV Generation → Collision Checking and TOC estimation

Motion Interpolation

BVH Construction



BVH Culling



Swept Volume Generation

Offset surfaces

$$\boldsymbol{x}(t,s) = \boldsymbol{b}(t) + s\boldsymbol{\delta}(t)$$

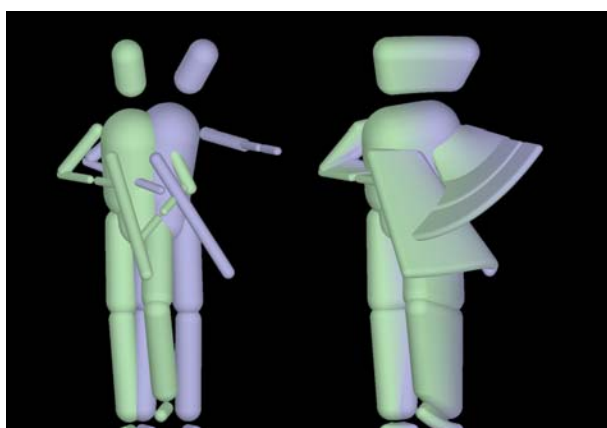
$$\boldsymbol{x}_d(t,s) = \boldsymbol{x}(t,s) \pm d\, \boldsymbol{n}(t,s)$$

Pipe surfaces

$$\boldsymbol{K}(t,\theta) = \boldsymbol{C}(t) + R(\cos\theta \boldsymbol{b_1}(t) + \sin\theta \boldsymbol{b_2}(t))$$

$$\boldsymbol{b_1}(t) = \frac{\boldsymbol{C}'(t) \times \boldsymbol{C}''(t)}{\|\boldsymbol{C}'(t) \times \boldsymbol{C}''(t)\|}$$

$$\boldsymbol{b_2}(t) = \frac{\boldsymbol{C}'(t) \times \boldsymbol{b_1}(t)}{\|\boldsymbol{C}'(t) \times \boldsymbol{b_1}(t)\|}$$



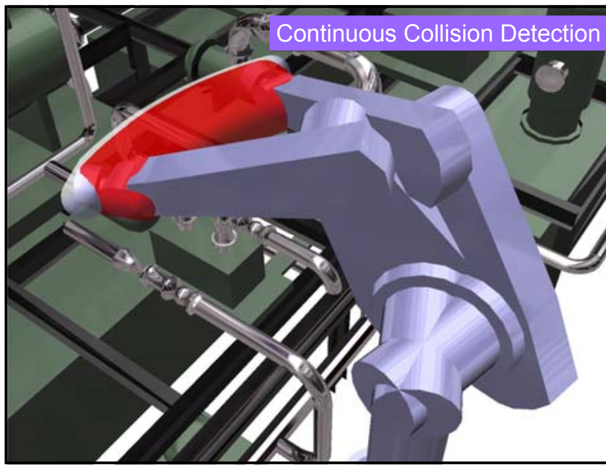
Dynamically generated swept surfaces

TOC Estimation

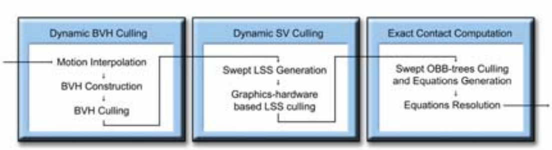
Demo

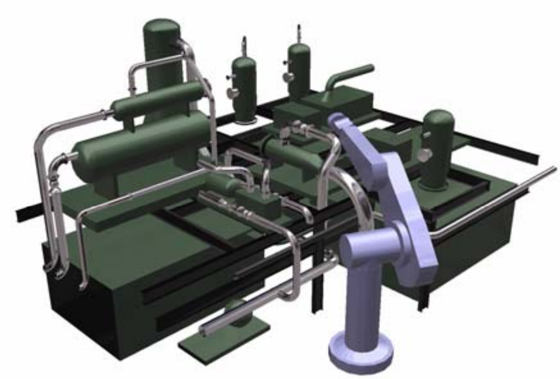
CCD for Generic Articulated Models

- Multiple culling steps [RKLM04b]
- Near interactive rates



Pipeline



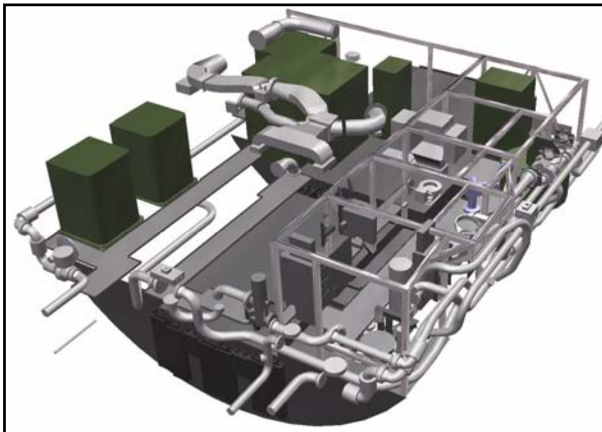


“Pipes” environment - 38,000 triangles

Implementation and Results

Angle θ_i^{max}	Dynamic BVH Culling		Dynamic SV Culling		Exact Contact Computation		Total Time for CCD	
	COL	NO-COL	COL	NO-COL	COL	NO-COL	COL	NO-COL
1	0.0014	0.0012	11.9552	2.9801	11.8	3.3123	23.7566	6.2936
5	0.0017	0.0013	16.1848	4.3327	17.9427	3.6844	32.3713	8.0184
15	0.0018	0.0015	22.1523	3.9728	30.5652	5.4548	52.7193	9.4291
30	0.0018	0.0013	18.6973	4.4477	85.4134	18.7761	104.1125	23.2251

Average timings for the "Pipes" environment (in milliseconds)



"Auxiliary Machine Room" environment - 180,000 triangles

Implementation and Results

Angle θ_i^{max}	Stages	Dynamic BVH Culling		Dynamic SV Culling		Exact Contact Computation		Total Time for CCD	
		COL	NO-COL	COL	NO-COL	COL	NO-COL	COL	NO-COL
1°	1+2+3	0.33	0.33	41.58	18.54	7.01	2.06	48.92	19.04
	1+3	0.34	0.33	-	-	7.33	1.79	7.67	1.24
	2+3	-	-	47.60	41.68	23.29	15.00	70.89	43.53
	3	-	-	-	-	82.24	75.05	82.24	75.05
30°	1+2+3	0.33	0.33	30.83	20.18	116.42	46.15	147.58	48.31
	1+3	0.33	0.33	-	-	121.93	43.43	122.26	115.40
	2+3	-	-	56.14	44.74	147.00	63.72	203.14	98.90
	3	-	-	-	-	190.79	98.41	190.79	98.41
60°	1+2+3	0.34	0.33	43.13	19.05	480.45	91.74	523.92	40.91
	1+3	0.35	0.33	-	-	577.70	73.06	578.05	36.79
	2+3	-	-	62.23	44.70	519.39	107.86	581.62	70.08
	3	-	-	-	-	649.48	107.29	649.48	107.29

Average timings for the "Auxiliary machine room" environment (in milliseconds)

Conclusions

- The collision detection problem is essential for mimicking the physical reality
- A rule of thumb
  - Find a collision detection library best suited for your target models
    - Polygonal soup, polytope, deformable
  - Ask yourself what you want from the library
    - Collision determination, witness features, first time of collision, penetration depth, self-collision

Hot Research Topics

- Self-collision detection
- Deformable bodies
  - Discrete CD
  - Continuous CD
- Big data
  - Polygonal mesh
  - Point data set
- Multi-resolution CD
  - Contact dependent
- GPU-based algorithms
- Curved surfaces

Web Sites

- RAPID (OBB-trees)
  http://www.cs.unc.edu/~geom/OBB/OBBT.htm
- CULLIDE
  http://gamma.cs.unc.edu/CULLIDE/
- AVATAR
  http://gamma.cs.unc.edu/Avatar/
- ARTICULATE
  http://gamma.cs.unc.edu/Articulate/

References

[Cam90] S. Cameron. Collision detection by four-dimensional intersection testing. *IEEE Transactions on Robotics and Automation*, 6:291–302, 1990.

[Can86] J. Canny. Collision detection for moving polyhedra. *IEEE Trans. Pattern and Mach. Intell.*, 8:200–209, 1986.

[Ebe05] D. Eberly, Magic software, http://www.magic-software.com/Intersection.html

[GLM96] OBBTree: A Hierarchical Structure for Rapid Interference Detection, *Proc. of SIGGRAPH* 1996

[GRLM03] N. Govindaraju, S. Redon, M. Lin, and D. Manocha. CULLIDE: Interactive collision detection between complex models in large environments using graphics hardware. *Proc. of ACM SIGGRAPH/Eurographics Workshop on Graphics Hardware*, pages 25–32, 2003.

[Held97] Martin Held. ERIT: A collection of efficient and reliable intersection tests. Journal of Graphics Tools, 2(4):25-44, 1997

[Mol97] Tomas Möller. A fast triangle-triangle intersection test. Journal of Graphics Tools, 2(2):25-30, 1997

[RKC00] S. Redon, A. Kheddar, and S. Coquillart. An algebraic solution to the problem of collision detection for rigid polyhedral objects. *Proc. of IEEE Conference on Robotics and Automation*, 2000.

[RKC02] S. Redon, A. Kheddar, and S. Coquillart. Fast continuous collision detection between rigid bodies. *Proc. of Eurographics (Computer Graphics Forum)*, 2002.

[RKLM04a] S. Redon, Y. Kim, M. Lin, and D. Manocha. Interactive and continuous collision detection for avatars in virtual environments. *IEEE Virtual Reality*, 2004.

[RKLM04b] S. Redon, Y. Kim, M. Lin, and D. Manocha. Fast continuous collision detection for articulated models. *ACM Solid Modeling and Applications*, 2004.

[SSL02] F. Schwarzer, M. Saha, and J.-C. Latombe. Exact collision checking of robot paths. In *Workshop on Algorithmic Foundations of Robotics (WAFR)*, Dec. 2002.

[Xav97] P. Xavier. Fast swept-volume distance for robust collision detection. In *Proceedings of International Conference on Robotics and Automation*, 1997.

Thank You!

For more info, visit http://graphics.ewha.ac.kr or email me to kimy@ewha.ac.kr.