**Time-Critical Collision Detection Using an Average-Case Approach**

**Motivation & Collision Detection**

Fundamental operation:
- Virtual prototyping
- Haptic rendering
- Interaction in VR
- Rigid body simulation

**Controlling the Errors**

Factors that contribute to imprecision in a model and simulation:
- Numerical error in computation
- Approximations in the abstraction
- Inaccuracy in the data
- Inaccuracy in the computation of collision (point of collision, normals, \( \hat{O} \))
- Barzel et al., 1996: physically plausible simulation.

**Related Work**

- Approximating Polyhedra with Spheres for Time-Critical Collision Detection [Hubbard, 1996]
- Graceful Degradation of Collision Handling in Physically Based Animation [Dingliana, O'Sullivan, 2000]

**Main Idea**

- Our approach is applicable to most BV hierarchies.
- We estimate the probability of intersection for a set of polygons.
- The estimation is also done for inner nodes.
- We do not need any polygons during runtime, but only parameters describing the polygon distribution.

1. The application can control the runtime by specifying the desired quality (accuracy).
2. The probabilities can guide the BV traversal into those parts that allow for faster convergence.

**Our Average-Case Approach**

Set of plausible paths for a cannonball.

Our goal: Decrease the accuracy of collision detection in a controlled way to speed up the simulation.

We concentrate on collision detection between rigid bodies.
Basic Traversal Algorithm

```c
priority queue q;

traverse(A, B) {
    while (not empty) {
        if (A & B) {
            // A & B overlap
            return "no collision";
        }
        if (A & !B) {
            // A contains B
            break;
        }
        if (!A & B) {
            // B contains A
            break;
        }
        // A and B do not overlap
        // ... traverse... 
    }
}
```

Estimating Probabilities

- Conceptually, partition $A \cap B$ into a grid.
- Probability of collision in cell is high, if it contains "enough" polygons.
- Estimate probability of collision by:
  - $Pr[c(A \cap B) \geq x]$: probability that at least $x$ collision cells exist in $A \cap B$.
  - $LB(A \cap B)$: lower bound for the probability that a collision really takes place in a collision cell. (for detail $\rightarrow$ paper)

\[
\text{estimateProbability}(A, B) = \max_{x \in \text{cells}} \left\{ Pr[c(A \cap B) \geq x] \cdot (1 - (1 - LB(A \cap B))^x) \right\}
\]

Necessary Parameters

What parameters are necessary to compute $Pr[c(A \cap B) \geq x]$?

- $s = \#$ cells contained in $A \cap B$
- $s_x = \#$ possible collision cells from $A$ contained in $A \cap B$
- $s_y = \#$ possible collision cells from $B$ contained in $A \cap B$

![Average-case approach: probability of being a possible collision cell is evenly distributed among all cells](image)

Probability Computations

Given $s$, $s_x$, $s_y$, compute $Pr[c(A \cap B) \geq x]$ by balls into bins model.

![What is the probability that at least $x$ bins get a red and a blue ball?](image)

Average-Distribution Trees (ADB Trees)

Partitioning of $A \cap B$ is too expensive during runtime!

- ADB tree:
  - Conceptually, partition each BV of the hierarchy into a fixed number of cells.
  - Store $\#$ possible collision cells with nodes
- Use precomputed possible collision cells to determine $s_x$ and $s_y$
- BVs $A$ and $B$ are equally sized.
- BVs $A$ and $B$ are differently sized.
- Please look at our paper

Lookup Tables

- $\text{estimateProbability}(A, B) = \max_{x \in \text{cells}} \left\{ Pr[c(A \cap B) \geq x] \cdot (1 - (1 - LB(A \cap B))^x) \right\}$
- The equation assumes the maximum at $a$ small $x \rightarrow$ bound $x$
- Store probabilities in lookup table.
- Number of possible collision cells is bounded by preprocessing
Results

Preprocessing

- ADB-tree based on AABBs.
- 8x8 cells per node.

Benchmark Scenario

- Objects are scaled uniformly to a cube of size 2^x.
- Average collision detection time for a complete revolution (5000 steps).
- Pentium-IV, 2GHz, 1 GB main memory.

Video

Time-Critical Collision Detection Using an Average-Case Approach
Jan Klein, Gabriel Zachmann

Time and Quality vs Complexity

- Runtime increases if \( p_{min} \) increases.
- Error rate decreases if \( p_{min} \) increases.

Time vs Quality

- Performance improves as \( p_{min} \) increases.
Conclusion & Future Work

Conclusion
- Average-case approach.
- It uses probability estimations to balance speed and quality.
- Results show speedup of about a factor 3 to 6 with only about 4 % errors.

Future Work
- Non-polygonal geometry
- Broad phase of collision detection
- Deformable objects
- Other BV hierarchies (DOP tree, restricted boxtree)

Thank you!

Heinz Nixdorf Institute
University of Paderborn
Institute of Computer Science
Germany
E-Mail: janklein@uni-paderborn.de