# Collision Detection for Complex Environments

# What is Collision Detection

## Finding coincident geometry

- e.g. what polygons intersect in this environment

## Detection by object

- Don’t care about collisions among an object’s polygons
- Choice of “what is an object” somewhat arbitrary

# Applications

- **Motion planning**
  - **Compute collision-free paths for automated tasks**
- **Virtual prototyping / simulation**
  - **Test functionality of mechanical assemblies**
- **Computer animation**
  - **Physically-based animations**
- **Virtual environments**
  - **Detect interactions of user with environment**
  - **Determine interactions among virtual objects**

# Triangle-Triangle Intersection

- Intersect if edge of one triangle intersects other triangle
  - 6 such tests produces complete tri-tri test
  - Faster methods possible
    - see Moller, "A Fast Triangle-Triangle Intersection Test", *Journal of Graphics Tools*, 2(2), 1997.
- Edge-triangle intersection
  - Find intersection of line and plane
  - Check if intersection is within edge
  - Check if intersection is within triangle

# Testing Complex Models

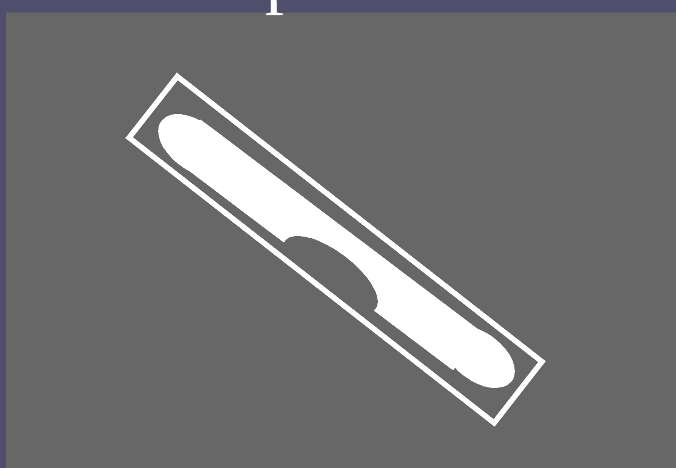
## Bounding volume hierarchies

- Spheres, axis-aligned bounding boxes, oriented bounding boxes
  - simple volume have faster tests, but may require more tests than complex volumes (even asymptotically more)
- Construct hierarchy for each object
- Prune out unnecessary collision tests
  - if two bounding volumes do not collide, their children do not collide

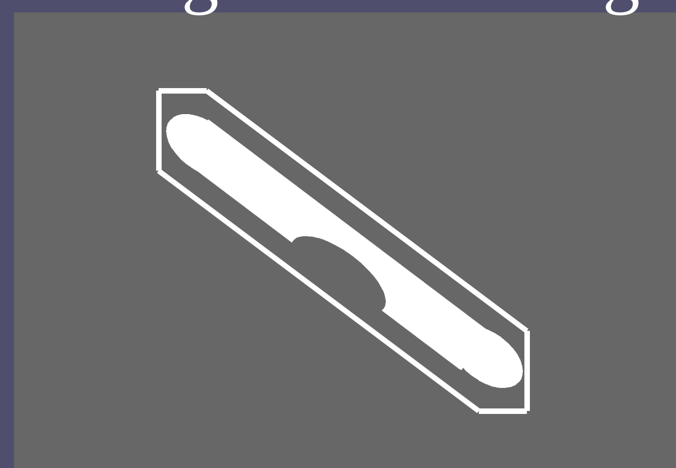
# Bounding Volume Examples

Sphere

Axis-Aligned Bounding Box

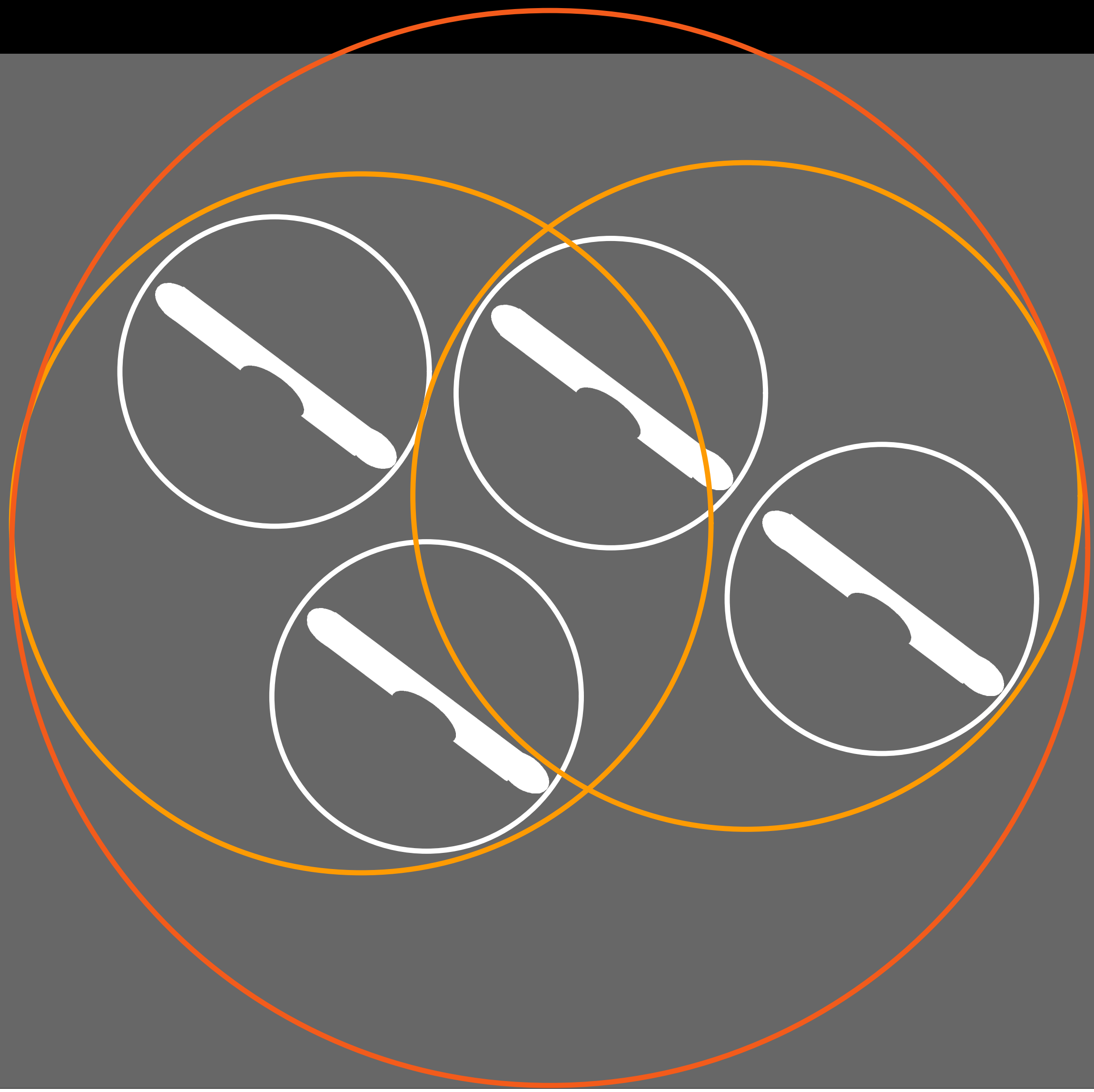


Oriented Bounding Box

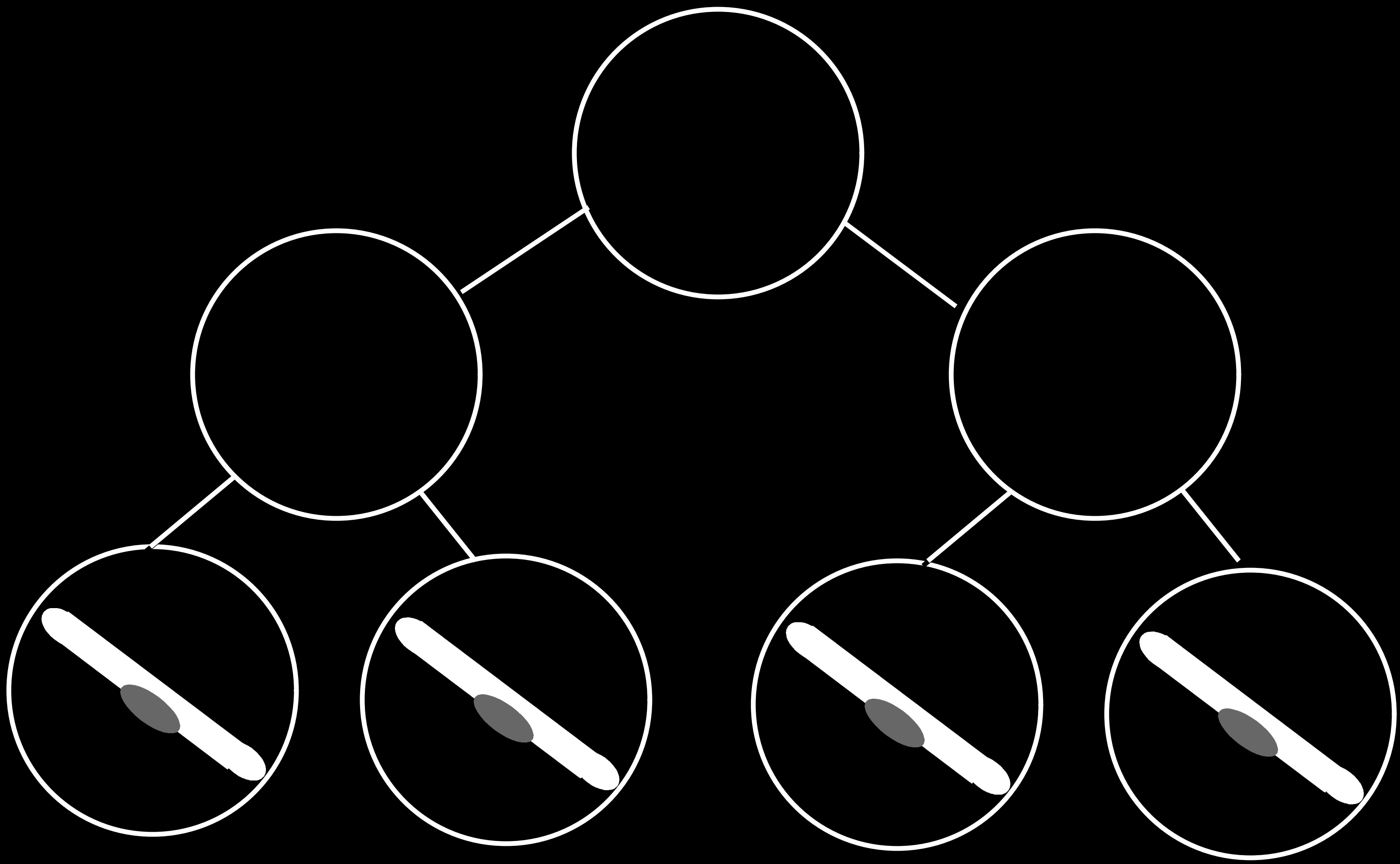


General Slab Intersection

# Bounding Volume Hierarchy Example



# Bounding Volume Hierarchy



# Computing Hierarchies

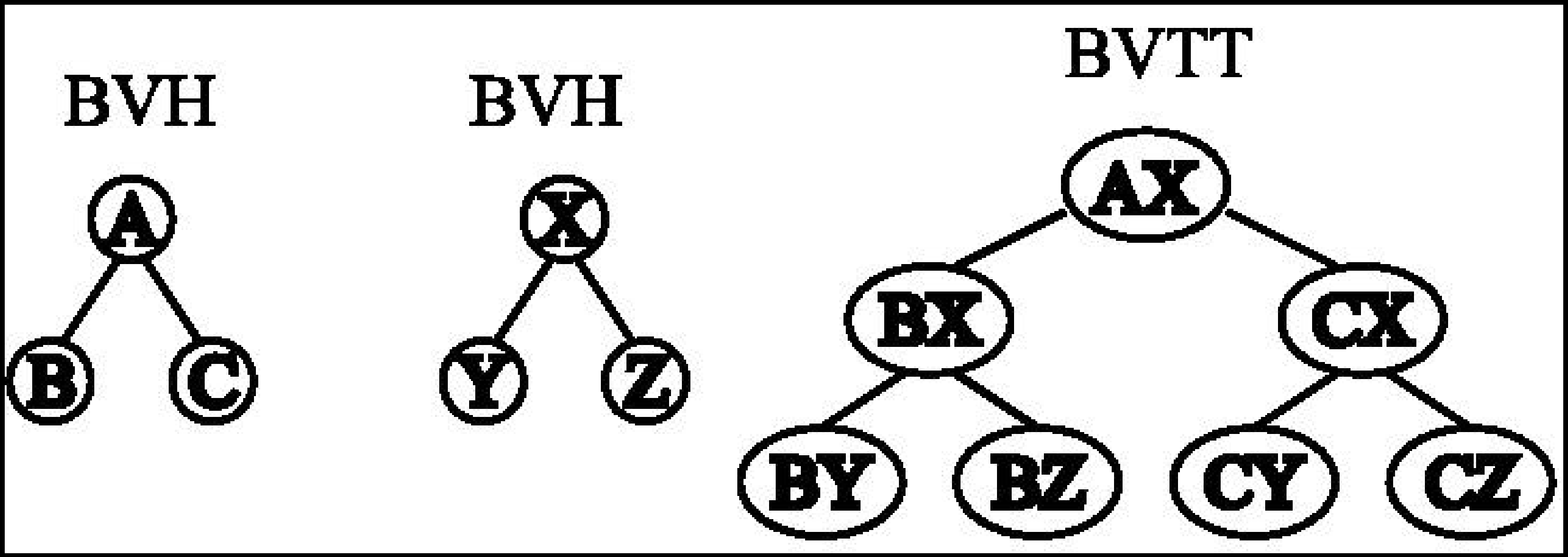
**Top-down**

**Bottom-up**

**Minimize volume/surface area**

**Computing “good” hierarchies is difficult**

# Bounding Volume Test Trees




Indicate order of traversing a pair of bounding volume hierarchies

Not generally represented explicitly

# Cost of Proximity Queries

$$T = N_{bv} \times C_{bv} + N_p \times C_p$$

# Video

**Gottschalk, Lin, Manocha. "OBB-Tree: A Hierarchical Structure for Rapid Interference Detection."** ***Proceedings of SIGGRAPH 96*****.**

- **Top-down method of OBB tree construction**
- **Fast OBB-OBB overlap test with separating axis theorem**
- **OBB trees *aymptotically faster* than sphere or AABB trees for parallel close proximity**

# N-body Collision Detection

**Given n moving objects and m stationary objects, find all pairs of intersecting objects.**

# Goals

## Speed

- want algorithm to be output sensitive, something like O(n+s)

## Accuracy

- accuracy to the precision of the models

## Assumptions

- temporal coherence
- no assumptions about accelerations or velocities of objects, except that sampling rate is high enough to detect collisions

# Worst Case Complexity

**Assuming none of the stationary objects are intersecting each other (or we are not concerned with these intersections), worst case output size:**

$$s = O\,(n^2 + nm)$$

# Space Partitioning Approaches

Partition space into small units.

Only test for collisions between objects in each of these units.

Examples

- Uniform Spatial Subdivision
- Octrees
- K-d trees, R-trees, etc.

# Uniform Spacial Sudivision

Partition space into a large number of boxes.

Decide which boxes each object falls into.

Only test for collisions between pairs of objects in each box.

Very simple.

Very memory intensive.

Difficult/impossible to choose proper box size.

# Octree

- Start with all of space in a single cube.
- Recursively subdivide cube into 8 equal sub-cubes until either the cube contains fewer than a set number of objects or the level of recursion reaches some maximum.

# Octree (cont.)

Cell size adapts to arrangment of objects in the environment.

Memory used more efficiently than uniform spacial subdivision,  but may still be wasteful.

Still difficult to choose cutoff sizes.

# Object Sorting

Sort objects in space to determine which object pairs overlap.

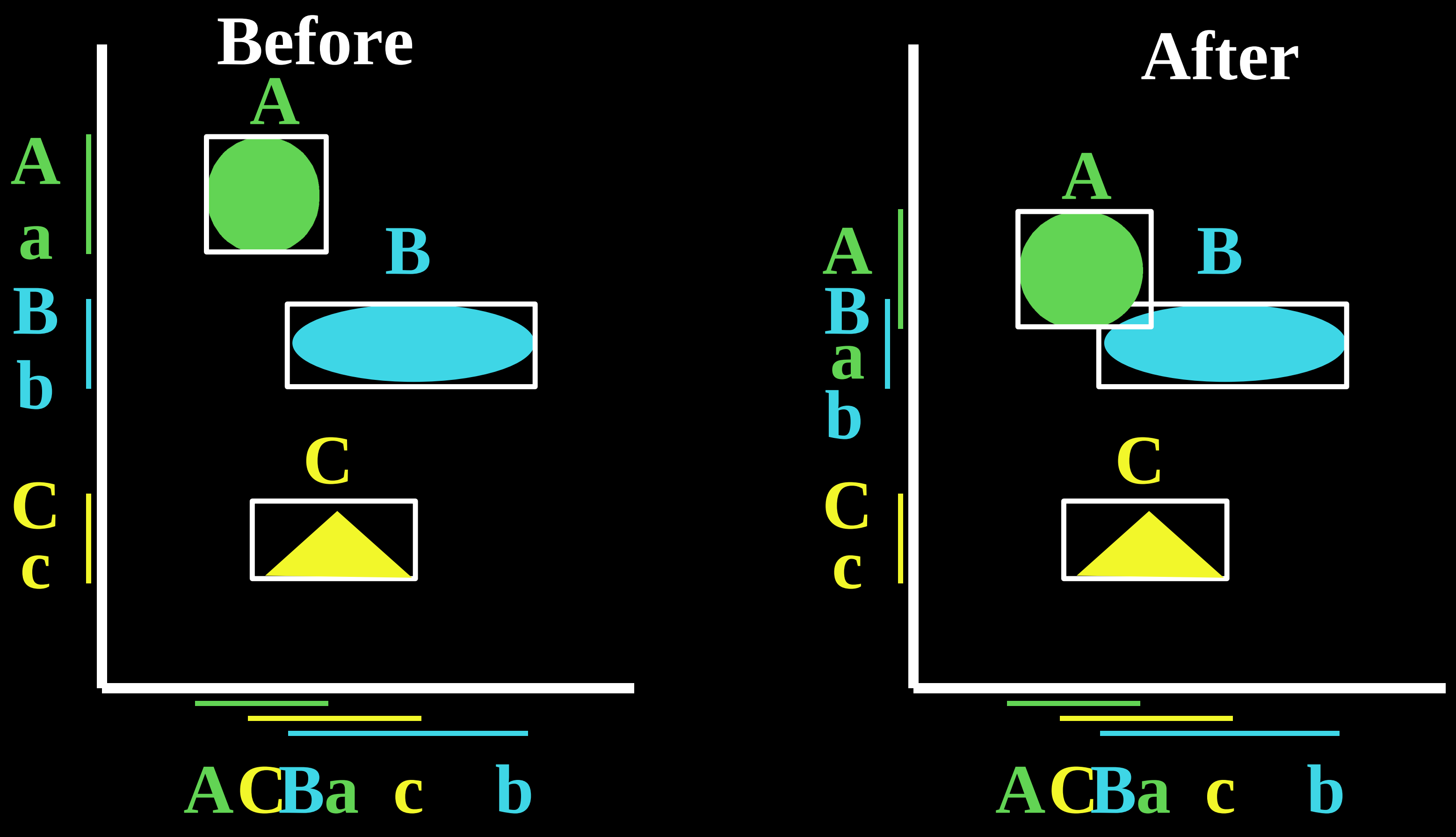
Test only these pairs for collisions.

(methods for determining overlaps and testing collisions are often independent).

# Dimension Reduction

- project bounding boxes of objects onto planes or lines
- “sort” in lower-dimensional space
- determine overlaps in lower-dimensional space
- overlap in all lower-dimensional spaces indicates overlap in the higher dimensional space

# Dimension Reduction (2D)




**Box A moves to overlap box B**

# Bounding Boxes

## Conservatively-sized cubes

- Shape invariant for object rotations

## Arbitrary AABBs for tighter fit

- Update incrementally as objects rotate

# Fast Bounding Box Updates

## Convex objects

- Use local walk to update extremal vertices

## Non-convex objects

- Precompute convex hull or other tight, convex bounding volume
- Apply incremental walk to bounding volume

# List Sorting

- Use temporal coherence
- Typically sort in O(n) time
- Each swap toggles a 1D overlap status

**Advantage - simple and efficient**

**Disadvantage - many swaps/overlaps in 1D**

# Range Queries in 2D and 3D

## Interval/Segment Trees

- O(nlogn) to construct
- Each query O(logn)
- Inserting/Deleting a range O(logn)

## Advantage - fewer overlaps in higher dimensions

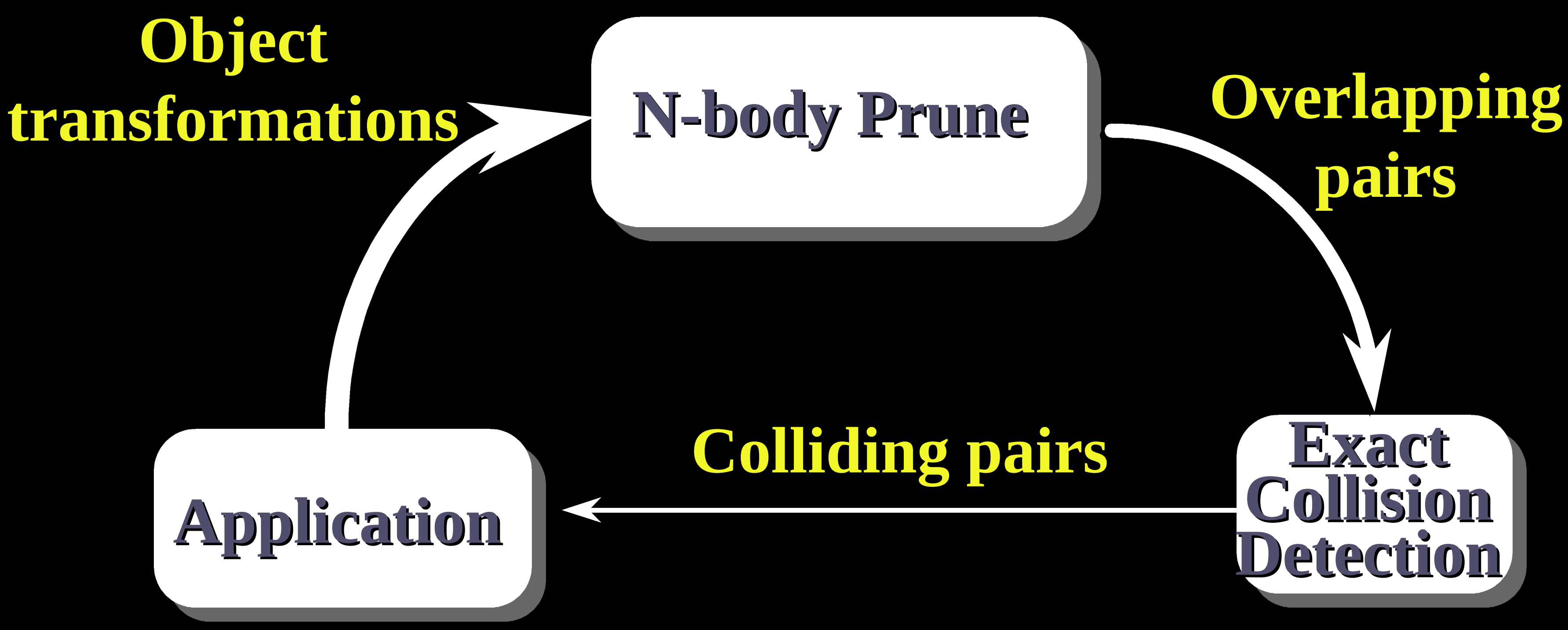
## Disadvantage - more complex structures, with larger overhead (constant factors matter)

# I-Collide

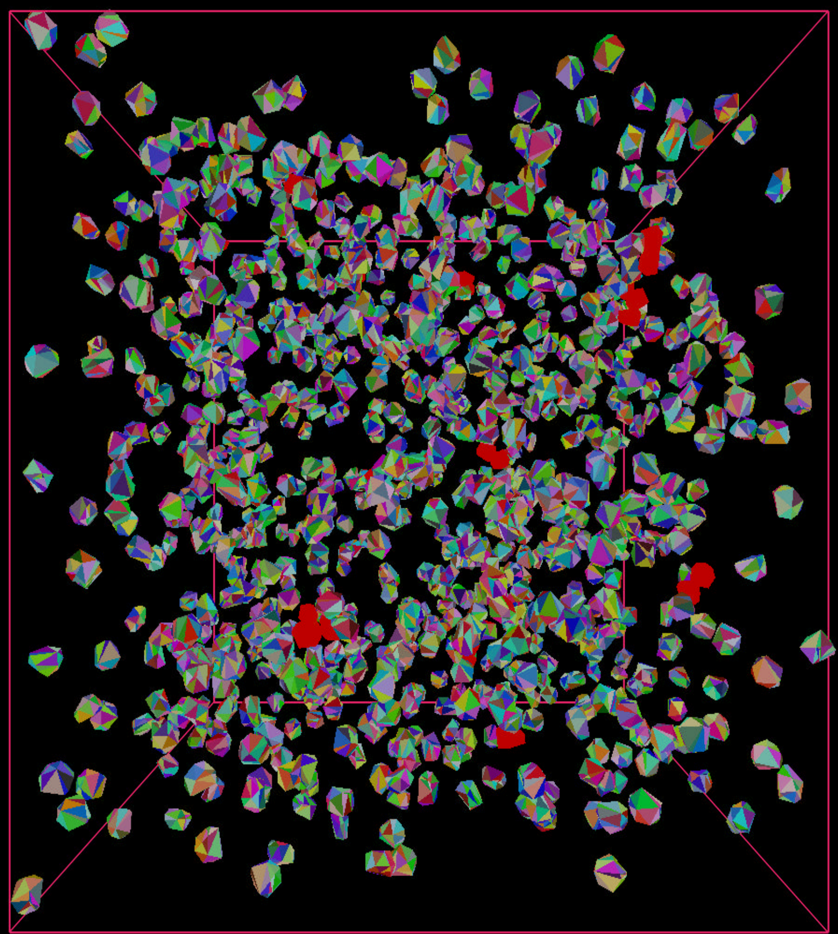
## Collision detection for convex polyhedra

- Tracks pair-wise closest distances
- Reports distance and collisions
- Performs exact collision tests
- Sub-linear in terms of object complexity
- Output sensitive in terms of number of objects
- Fast in practice

# System Architecture



# Multi-body simulation



1000 moving polyhedra

Runs at over 15 frames per second on Onyx RE R4400

# I_COLLIDE Results

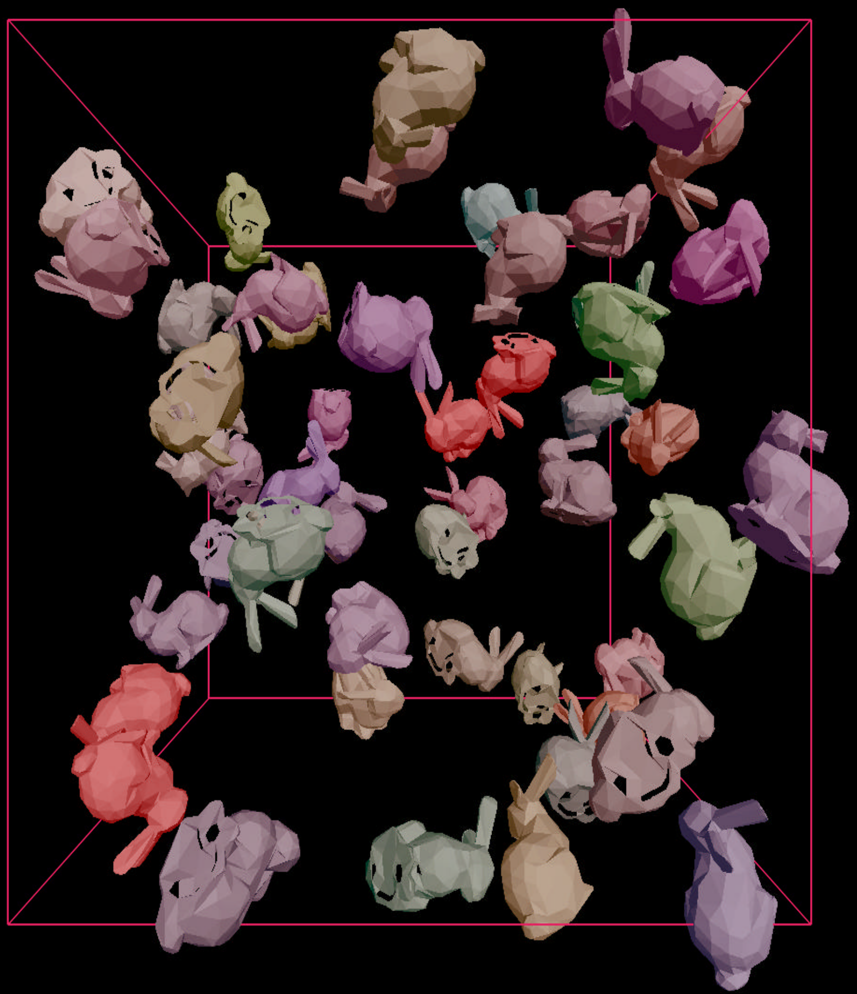
- Fairly linear with respect to number of objects and object/volume density
- N-body simulation with 500 objects runs at about 30 frames/sec (without graphics).

# V-Collide

Collision detection for large polygonal environments

- Combines I-Collide pruning algorithm with RAPID intersection test
- Provides efficient collision detection for n arbitrary polygonal models
- Improves space efficiency over I-Collide implementation

# Multi-bunny Simulation



# Video

Cohen, Lin, Manocha, and Ponamgi. **"I-COLLIDE: An Interactive and Exact Collision Detection System for Large-Scale Environments."** *Proceedings of 1995 Symposium on Interactive 3D Graphics.*