Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

# 09: BOUNDING VOLUME HIERARCHIES

08/02/2016

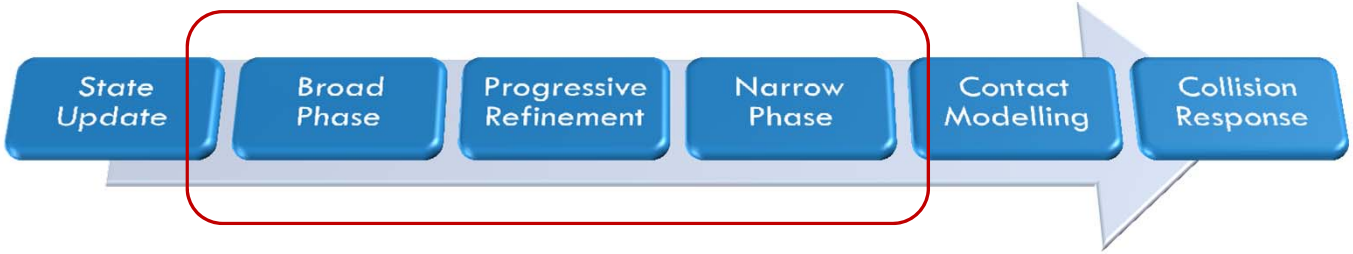
# RECAP: TWO-PHASE COLLISION DETECTION

```
detect (t_curr, {O_0, ..., O_N-1})
    for t ← t_prev to t_curr in steps of Δt_d
        for each object O_i ∈ {O_0, ..., O_N-1}
            move O_i to its position at time t
        for each object O_i ∈ {O_0, ..., O_N-1}
            for each object O_j ∈ {O_i+1, ..., O_N-1}
            if (O_i penetrates O_j)
                collision occurs at simulation time t
    t_prev ← t_curr
```

1. All-pairs weakness
2. Pair Processing weakness
3. Fixed timestep weakness

Naïve collision detection problems [Hubbard 1993]

Collision detection is often broken down further into broadphase (culling pairs) and narrow phase (e.g. primitive triangle intersection tests). In some cases an additional stage is employed such as progressive refinement



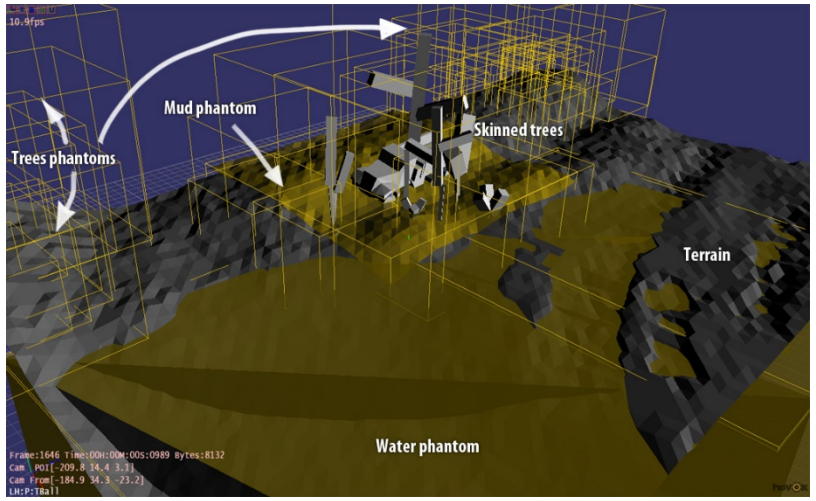

[Hubbard93] Interactive Collision Detection. P. Hubbard, in Proceedings of IEEE Symposium on Research Frontiers in Virtual Reality, 1993

# EXAMPLES

**Quote from havok source code comment**

- Instead of checking whether the moving object is colliding with each of the triangles in the landscape in turn, which would be extremely time-consuming, the bounding box of the moving object is **checked against the bounding volume tree** - first, whether it is intersecting with the top-level bounding volume, then with any of its child bounding volumes, and so on until the check reaches the leaf nodes. A list of any potentially colliding triangles is then passed to the narrowphase collision detection. You can think of the bounding volume tree as a filter to the narrowphase collision detection system.

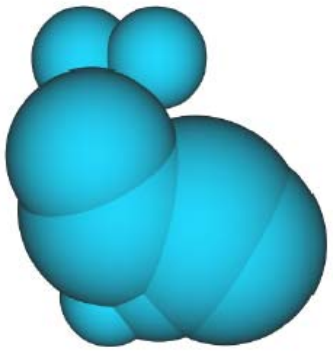
Source: hkpBvTreeShape.h © havok



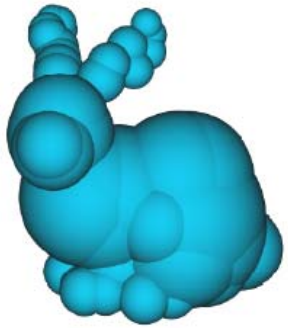

AABB-hierarchy example

Image From [Zagebelnyy13] Pavel Zagrebelny – **Rendering and simulation in offroad driving game** – Gamasutra 06/13/13.

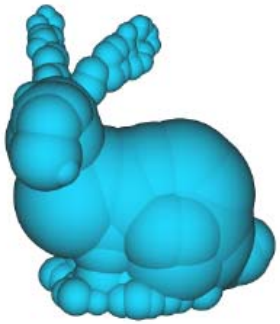
# BOUNDING VOLUME HIERARCHIES



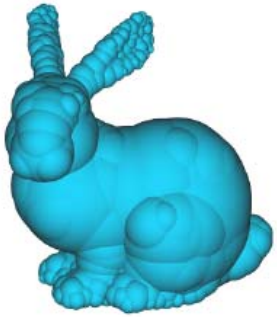
(a) 10 spheres



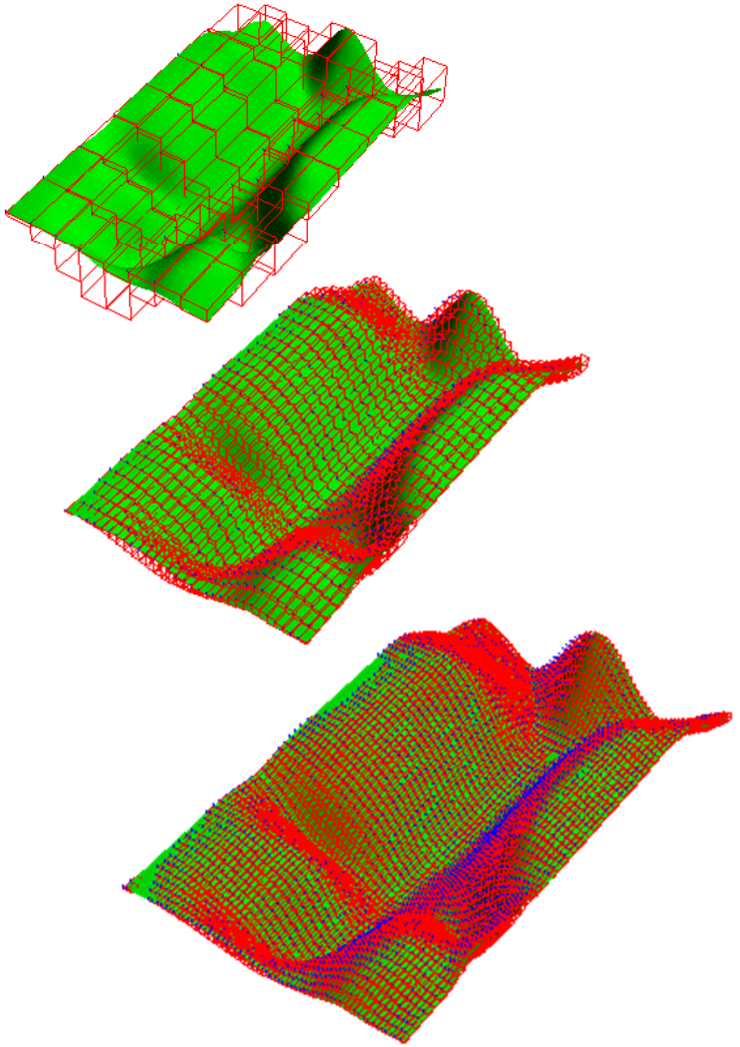
(b) 50 spheres



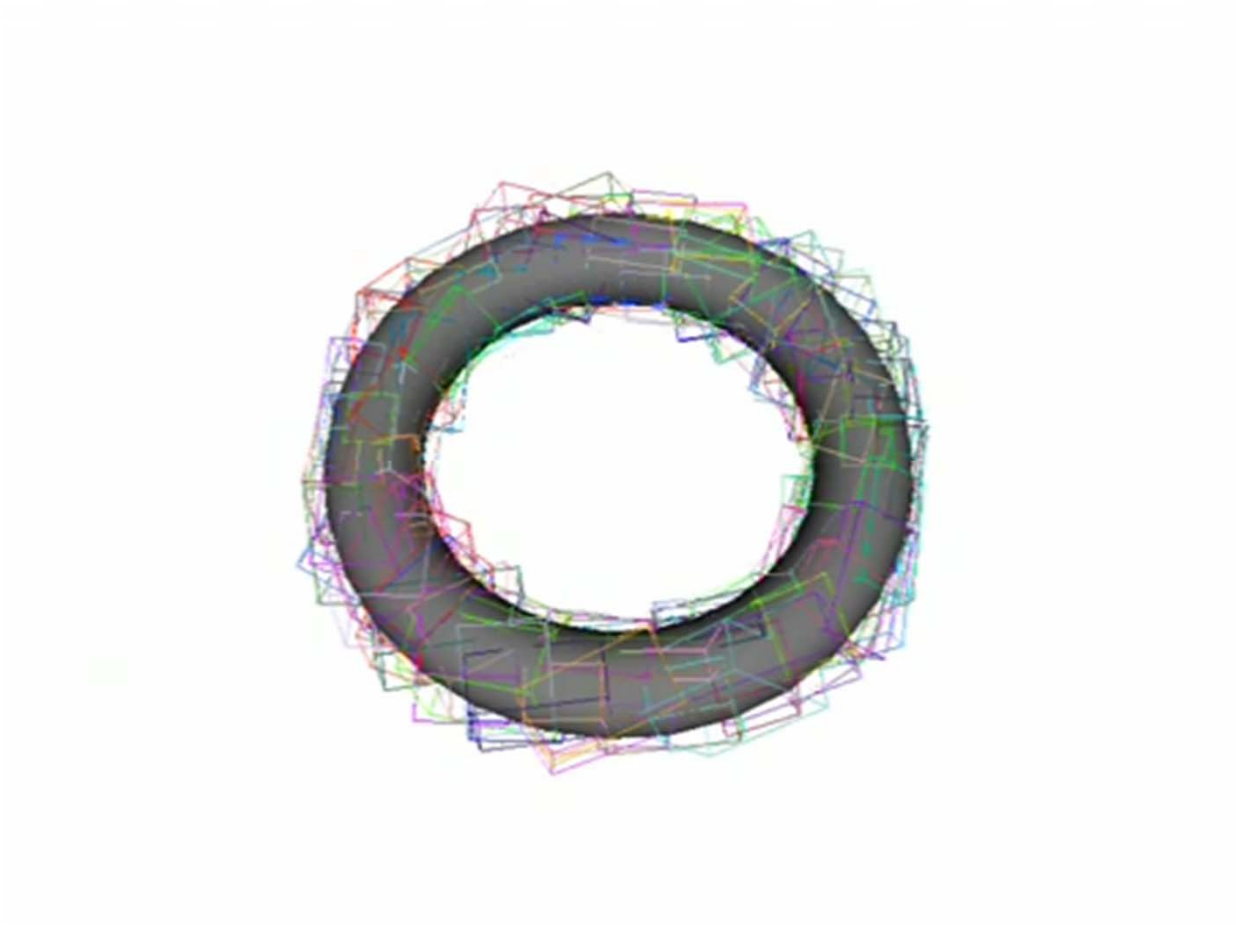
(c) 100 spheres



(d) 500 spheres



# BOUNDING VOLUME HIERARCHIES



© Denison Linus

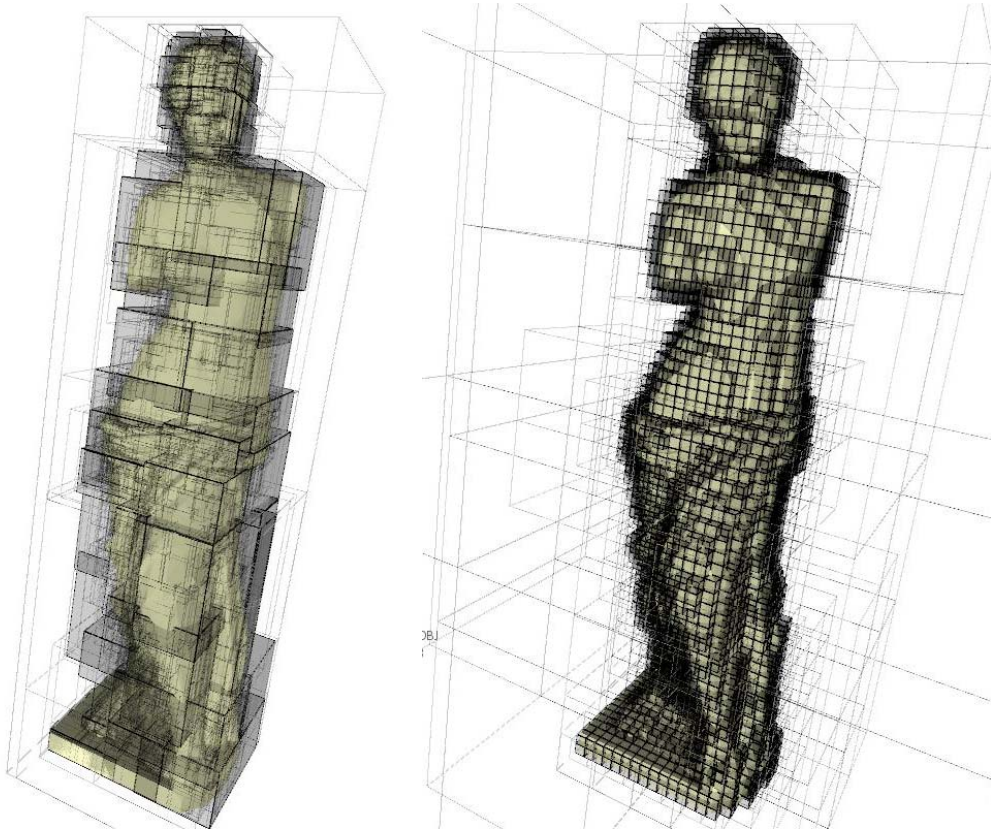
# BOUNDING VOLUME HIERARCHIES

© Machiel Van Hooren

# BVH OVERVIEW

**Overview of Main issues:**

- BVH representation
- BVH generation / fitting
- BVH collision check + update and traversal

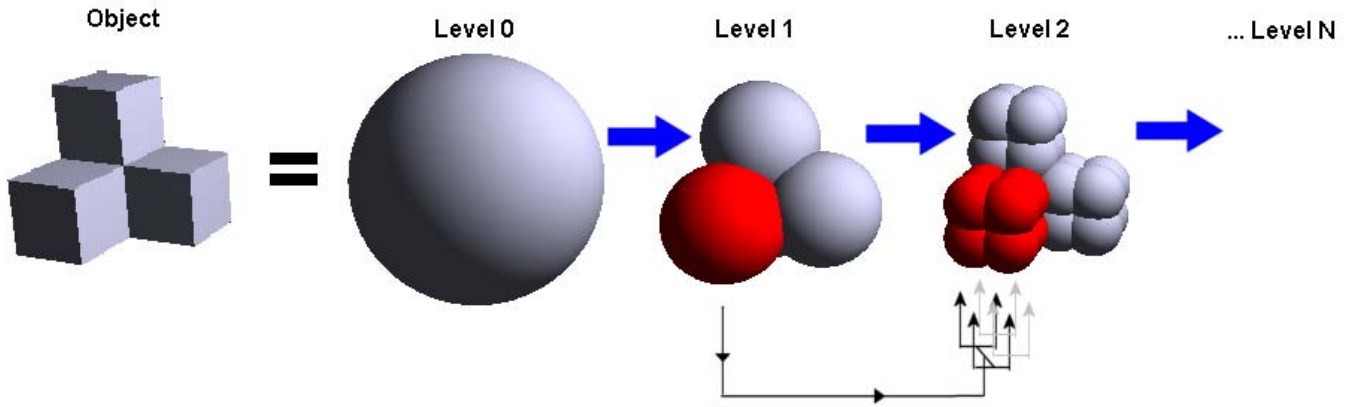


Images © Thomas Diewald

# BOUNDING VOLUME HIERARCHIES

**Model Hierarchy:**

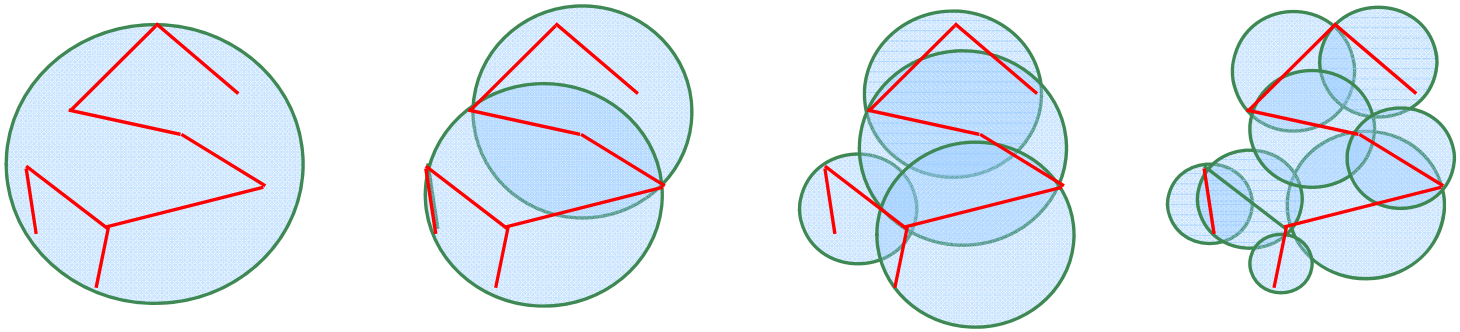
- each node is a simple volume that bounds a set of triangles
- BVH should be conservative over-approximations (for robustness)
- children contain volumes that each bound a different subset of the parent's triangles
  - N.B. Children of each node must collectively cover all triangles covered by the parent



# BVHS AND POLYGONS

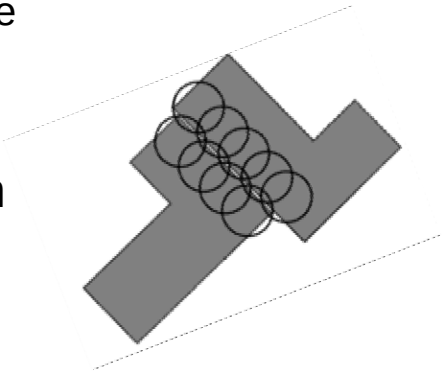
**BVH principally designed for culling primitive level comparisons (pair-processing weakness)**

- Most approaches have leaf node of BVH bounding individual primitives (usually triangles)

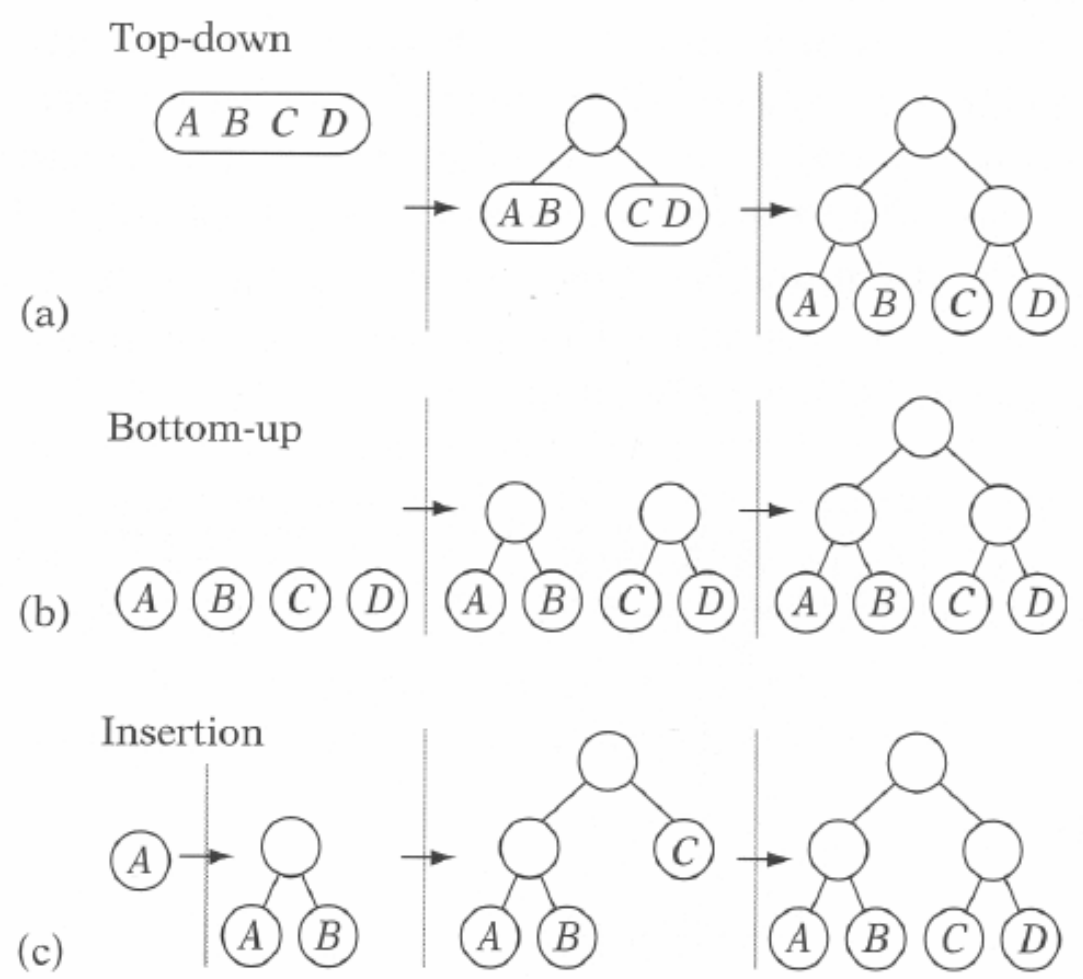


Above Example illustrates a binary sphere tree

- In some approaches, there may be multiple triangles in the smallest BVH node OR there may be more than one BV per triangle

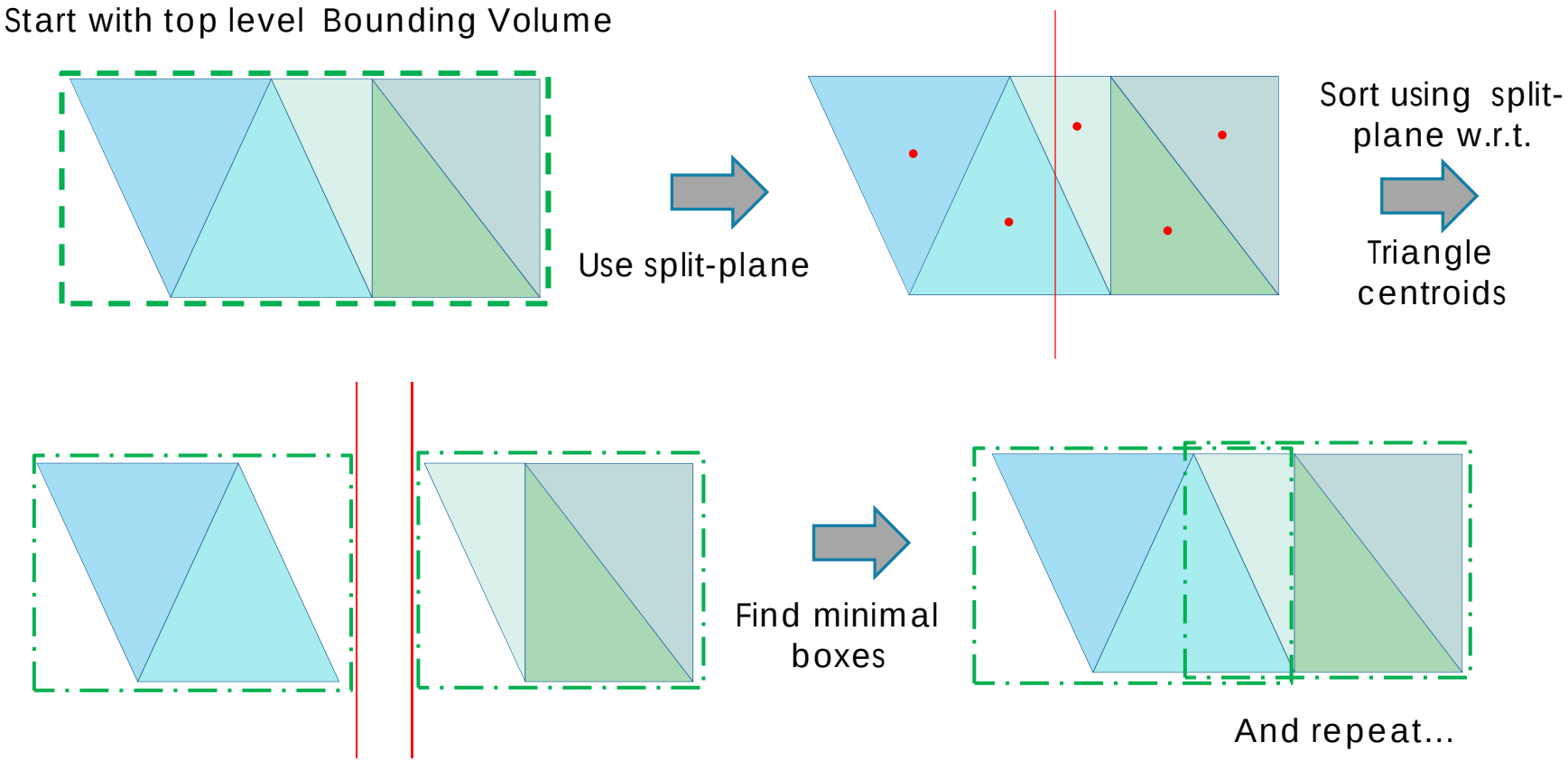


# BVH CONSTRUCTION



# TOP-DOWN CONSTRUCTION: SPLIT

**Recursively split and bound geometric primitives**



# BOTTOM-UP CONSTRUCTION: MERGE

- Wrap all primitives in leaf nodes
- Each level up (chosen based on various heuristics) needs to bound all primitives bound by children

- Parent bounds can be made to enclose child bounds for fitting speed

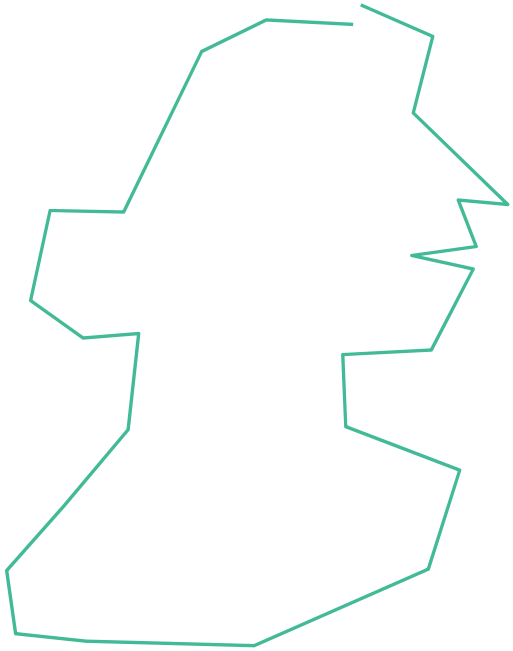


Image from [Quinlan'94] Efficient distance computation between non-convex objects. Quinlan, S. In Proceedings IEEE Robotics and Automation, 1994.

# BOTTOM-UP CONSTRUCTION: MERGE

- **Wrap all primitives in leaf nodes**
- Each level up (chosen based on various heuristics) needs to bound all primitives bound by children
- Parent bounds can be made to enclose child bounds for fitting speed

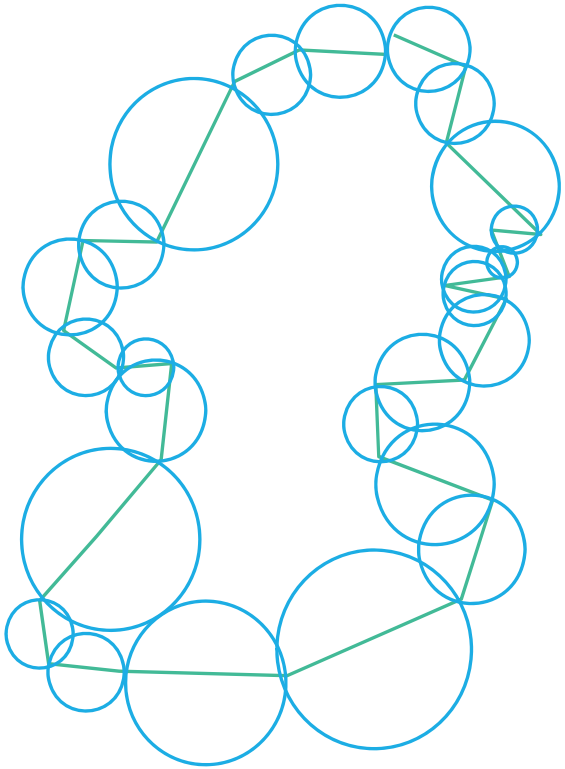


Image from [Quinlan'94] Efficient distance computation between non-convex objects. Quinlan, S. In Proceedings IEEE Robotics and Automation, 1994.

# BOTTOM-UP CONSTRUCTION: MERGE

- Wrap all primitives in leaf nodes
- **Each level up (chosen based on various heuristics) needs to bound all primitives bound by children**
- Parent bounds can be made to enclose child bounds for fitting speed

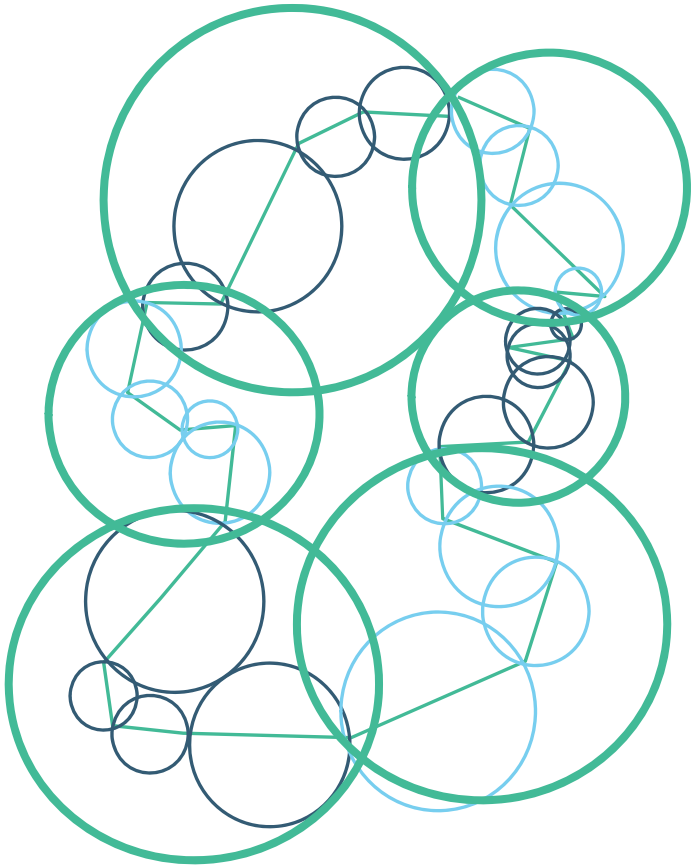


Image from [Quinlan'94] Efficient distance computation between non-convex objects. Quinlan, S. In Proceedings IEEE Robotics and Automation, 1994.

# BOTTOM-UP CONSTRUCTION: MERGE

- Wrap all primitives in leaf nodes
- **Each level up (chosen based on various heuristics) needs to bound all primitives bound by children**
- Parent bounds can be made to enclose child bounds for fitting speed

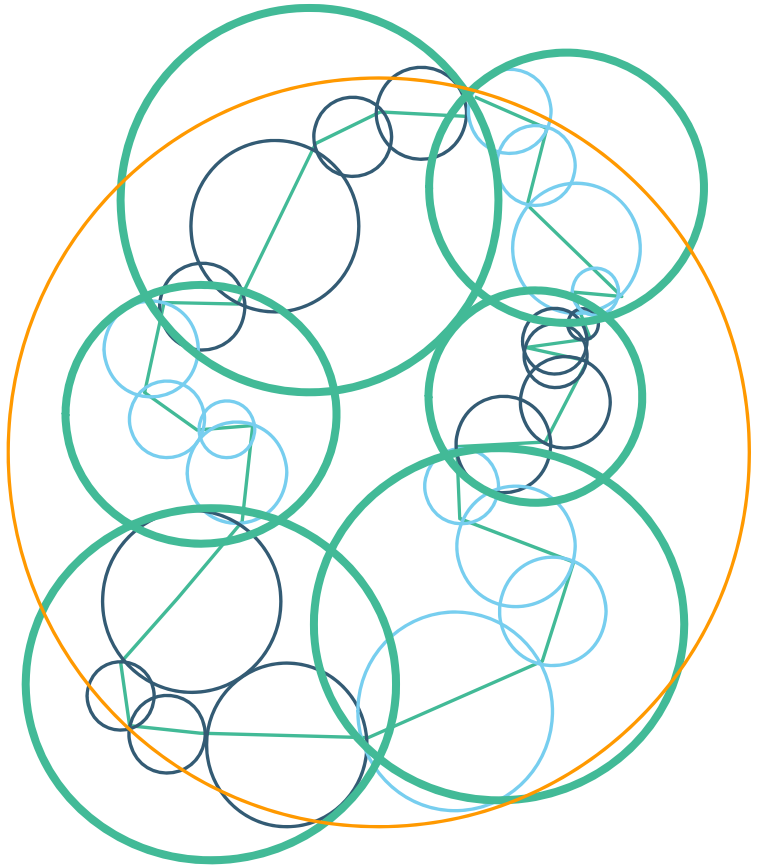


Image from [Quinlan'94] Efficient distance computation between non-convex objects. Quinlan, S. In Proceedings IEEE Robotics and Automation, 1994.

# BVH COLLISION DETECTION

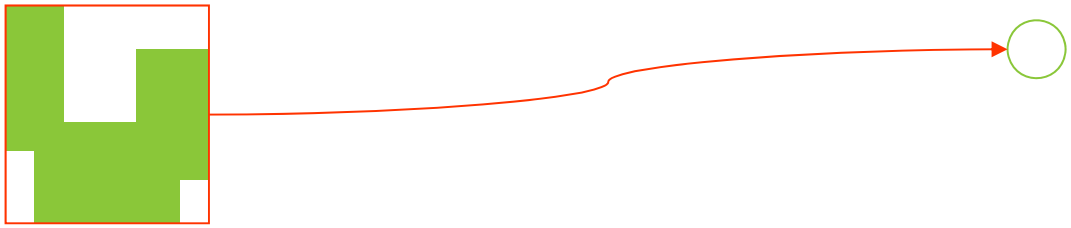


FIGURE © M. LIN, UNC CHAPEL HILL

# BVH COLLISION DETECTION

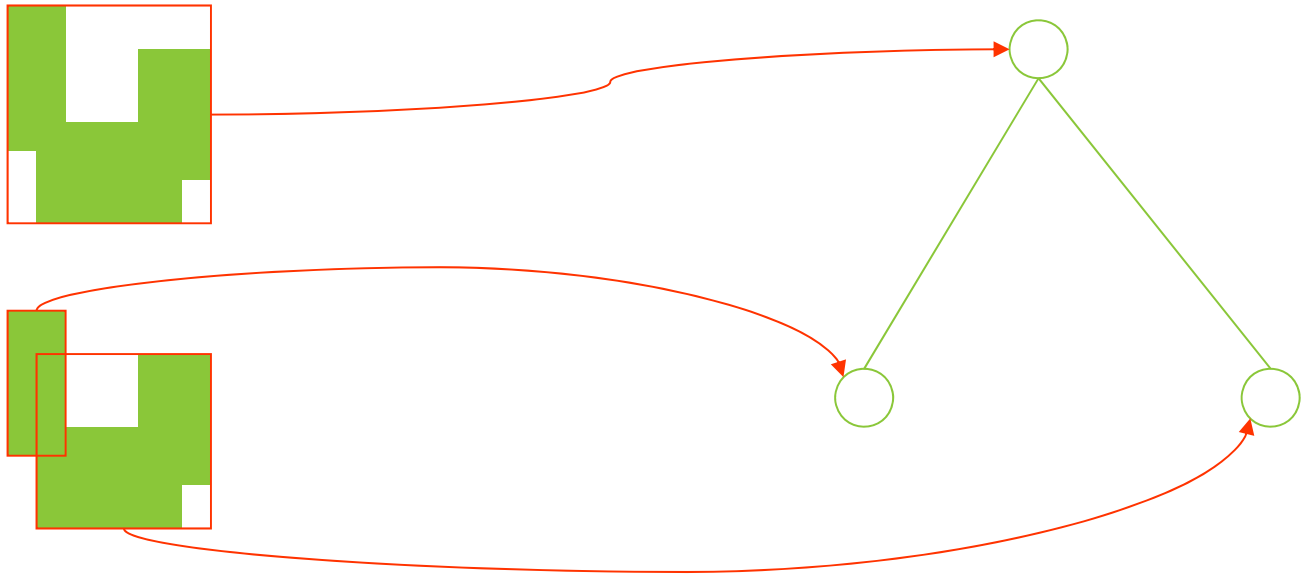


FIGURE © M. LIN, UNC CHAPEL HILL

# BVH COLLISION DETECTION

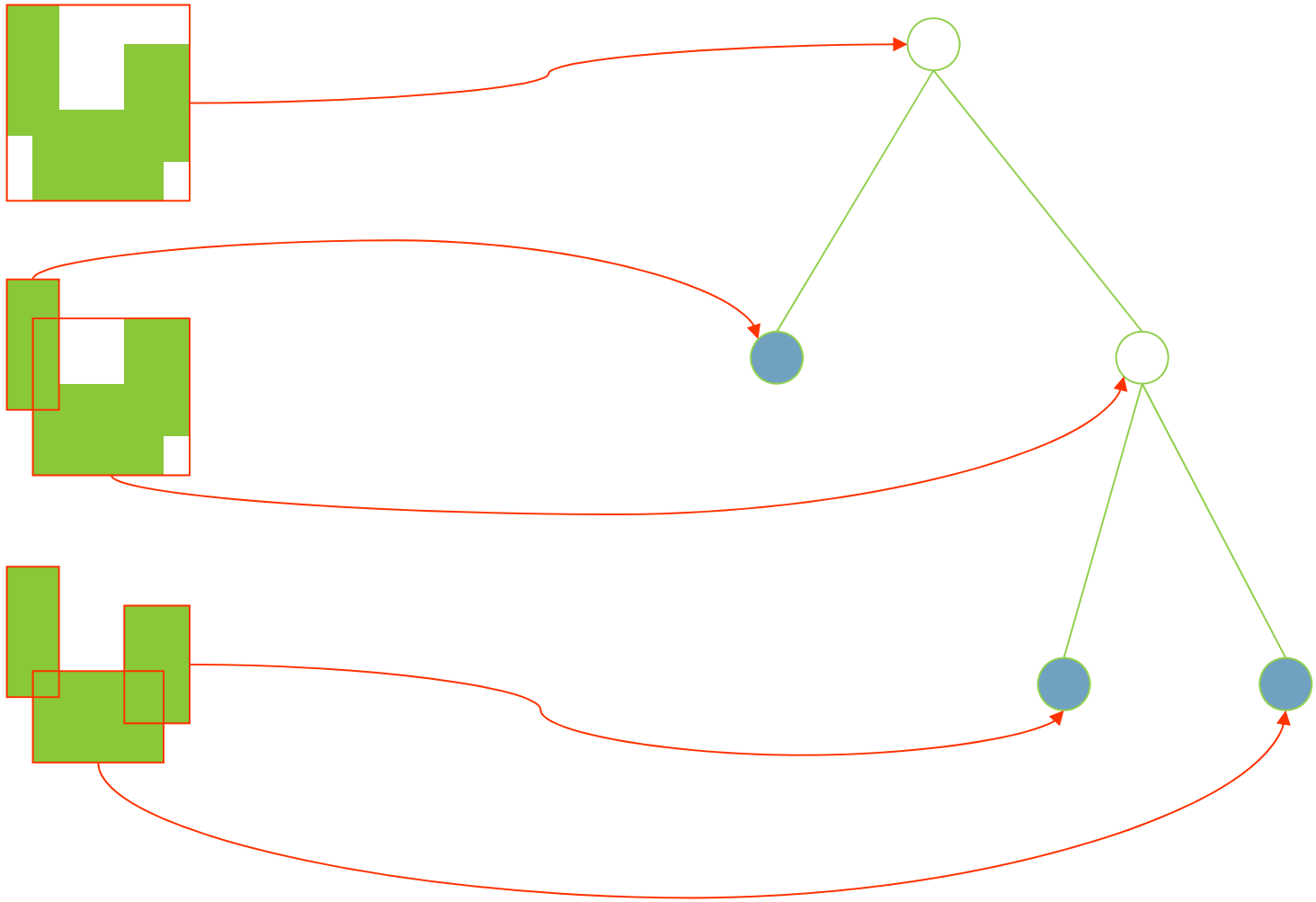
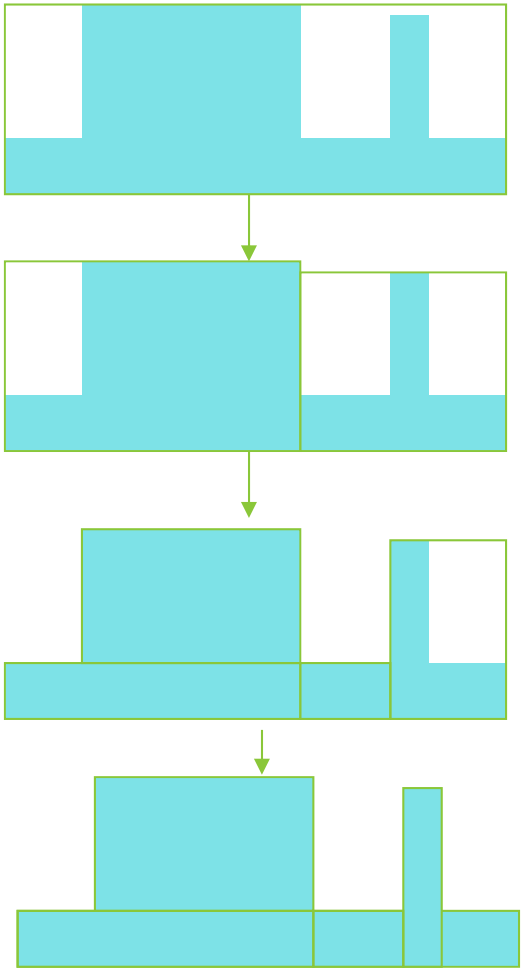


FIGURE © M. LIN, UNC CHAPEL HILL

# BVH COLLISION DETECTION



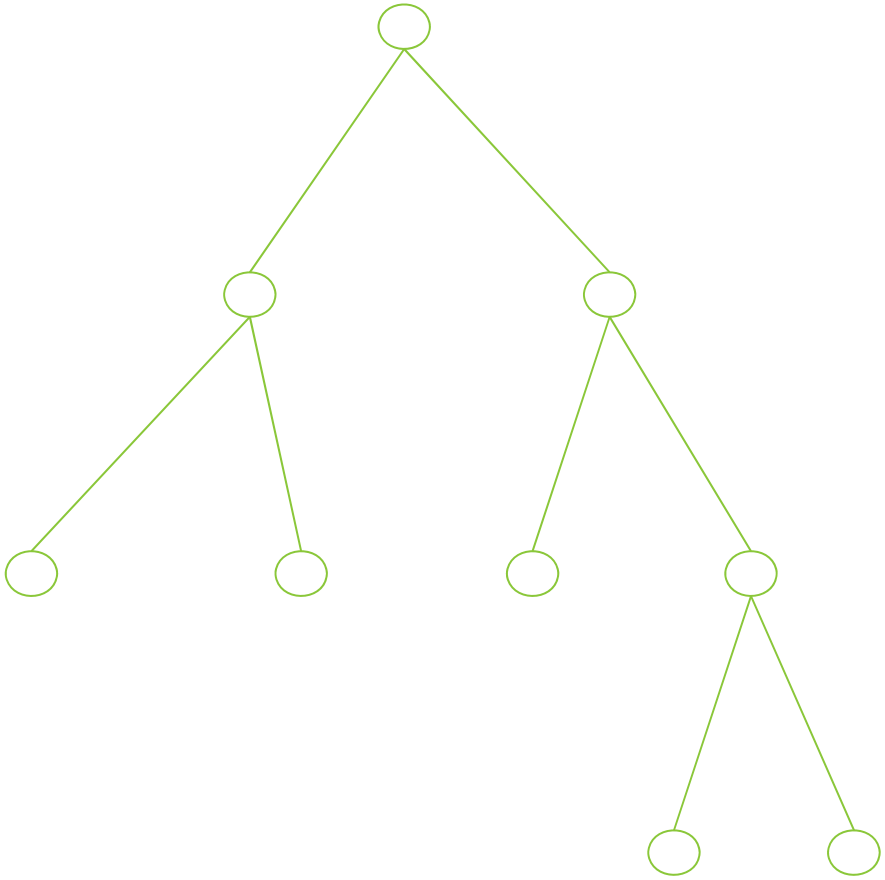
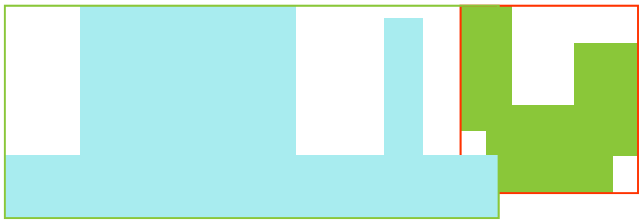


FIGURE © M. LIN, UNC CHAPEL HILL

# BVH COLLISION DETECTION



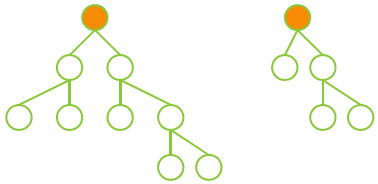
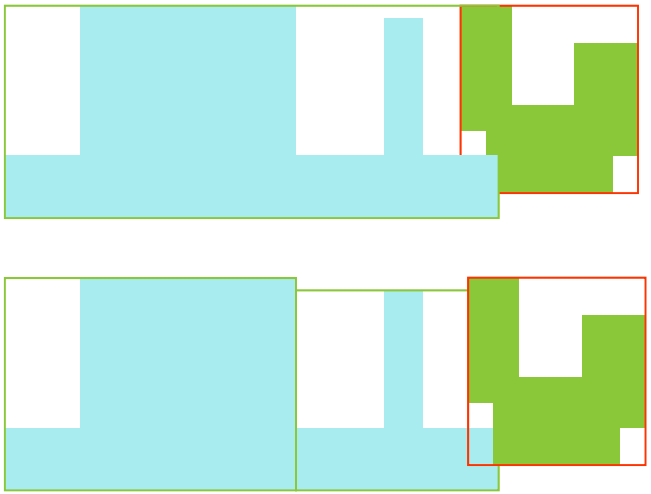


FIGURE © M. LIN, UNC CHAPEL HILL

# BVH COLLISION DETECTION



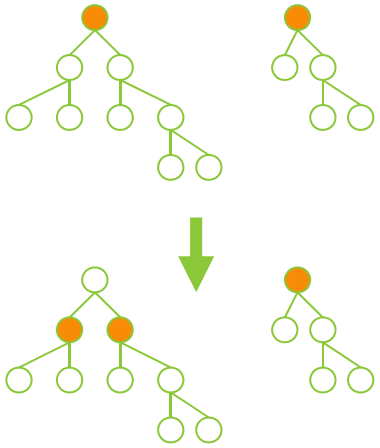
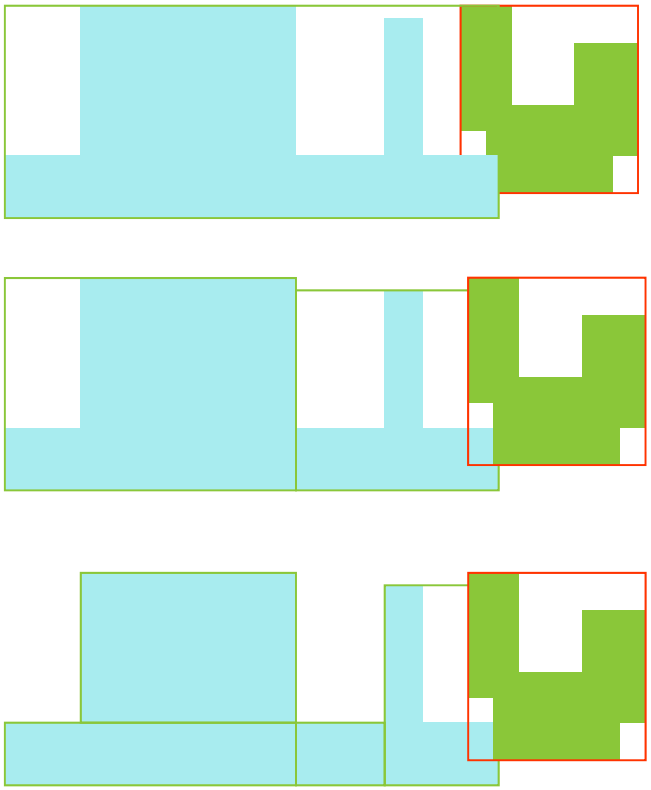


FIGURE © M. LIN, UNC CHAPEL HILL

# BVH COLLISION DETECTION



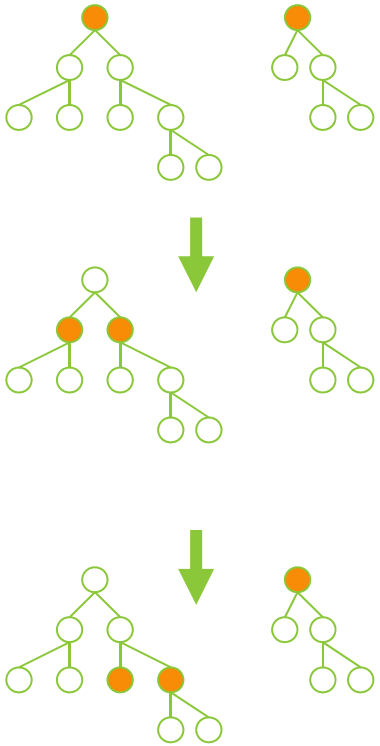
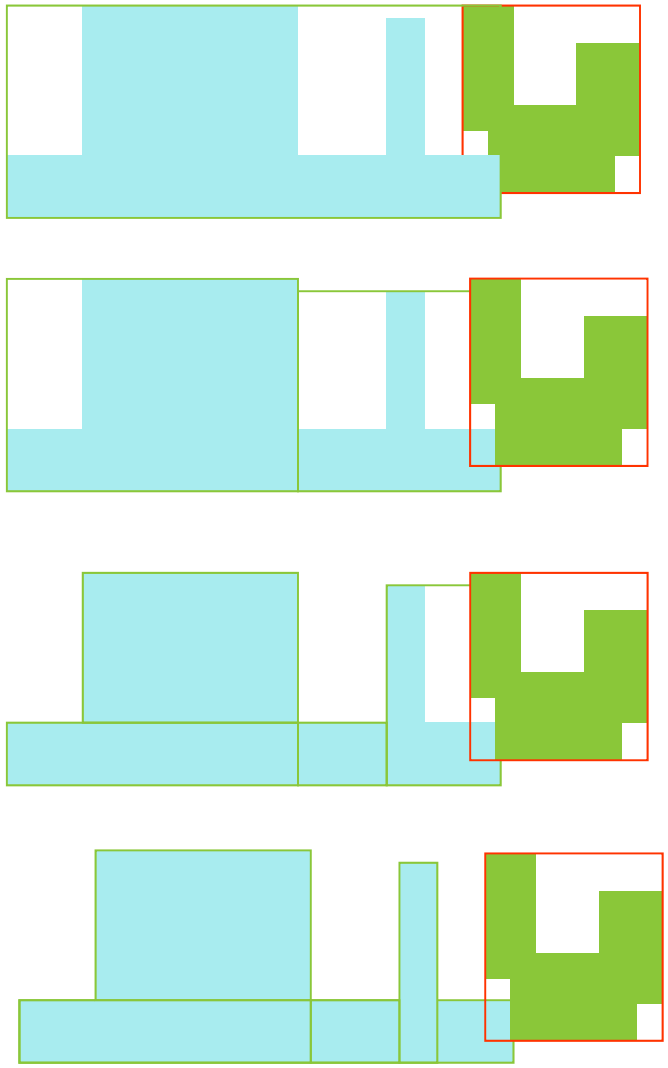


FIGURE © M. LIN, UNC CHAPEL HILL

# BVH COLLISION DETECTION



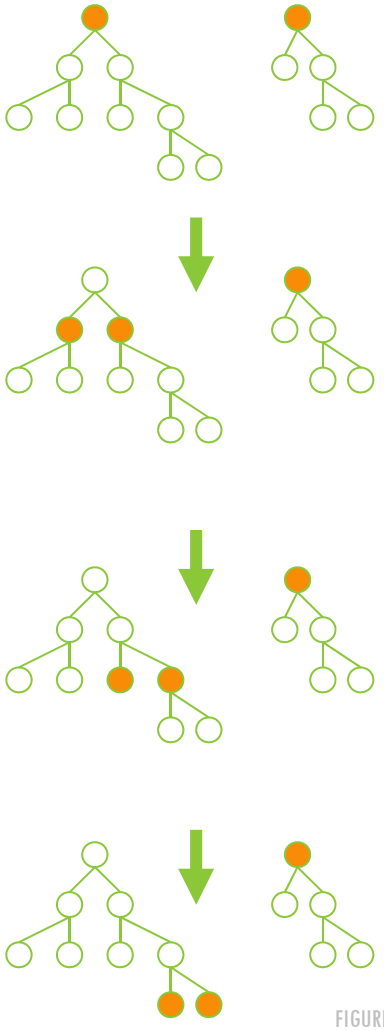
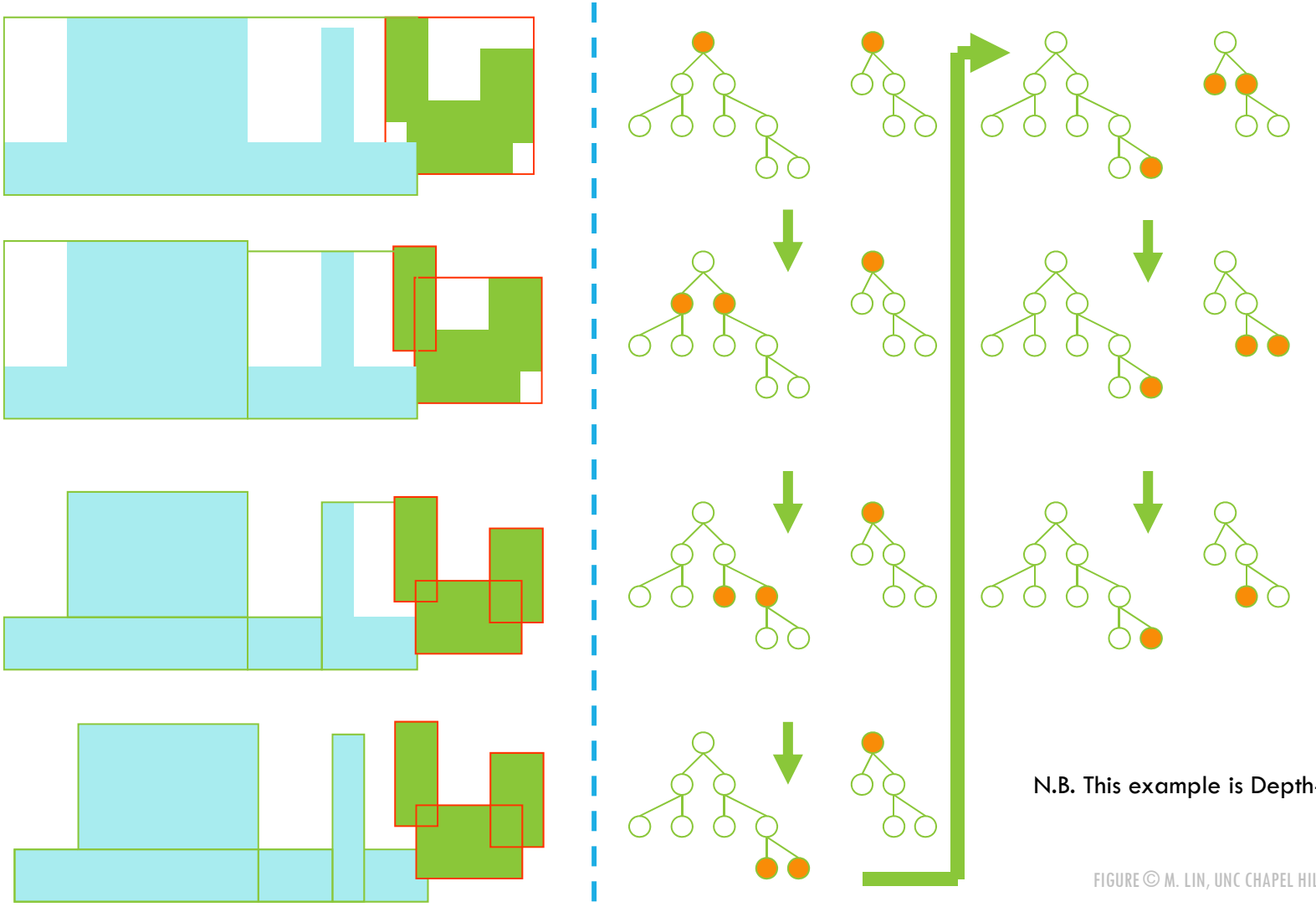


FIGURE © M. LIN, UNC CHAPEL HILL

# BVH COLLISION DETECTION

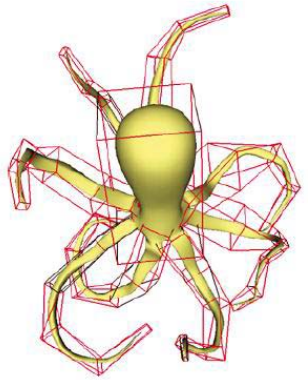


N.B. This example is Depth-first

FIGURE © M. LIN, UNC CHAPEL HILL

# PAIRWISE COLLISION DETECTION USING BVH

```
001: Check for collision between two parent nodes (starting from the roots of
     two given trees)

002: IF there is no interference between two parents,

003:    THEN stop and report “no collision”

004:    ELSE All children of one parent node are checked against all  children
        of the other node

005:      IF there is a collision between the children

006:         THEN IF at leave nodes

007:                   THEN report “collision”

008:                   ELSE go to Step 4

009:              ELSE stop and report “no collision”
```



N.B. This example is Breadth-first

# MULTI-BODY BVH COLLISION DETECTION

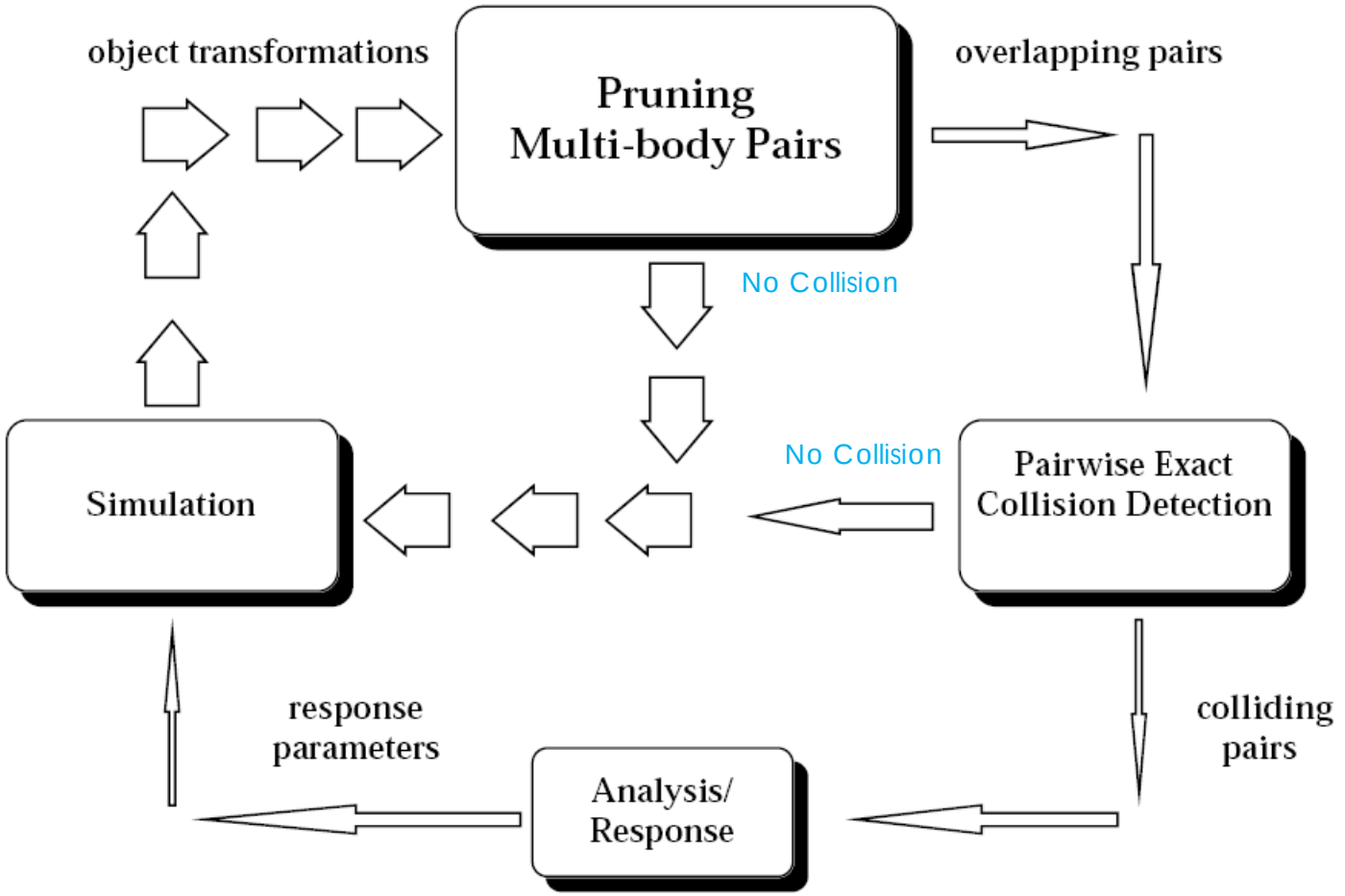



Image from [Cohen95] *I-COLLIDE: An interactive and exact collision detection system for large-scale environments.* J. Cohen, M. Lin , D. Manocha , M. Ponamgi. In Proc. ACM Interactive 3D Graphics 1995

# BVH UPDATE

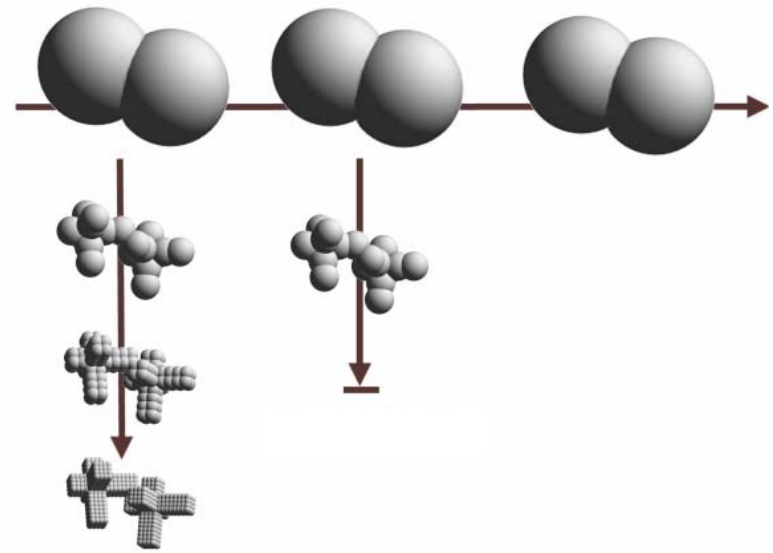
**Note that we cull not only collision checks but node updates**

**Update node only if parent is colliding**

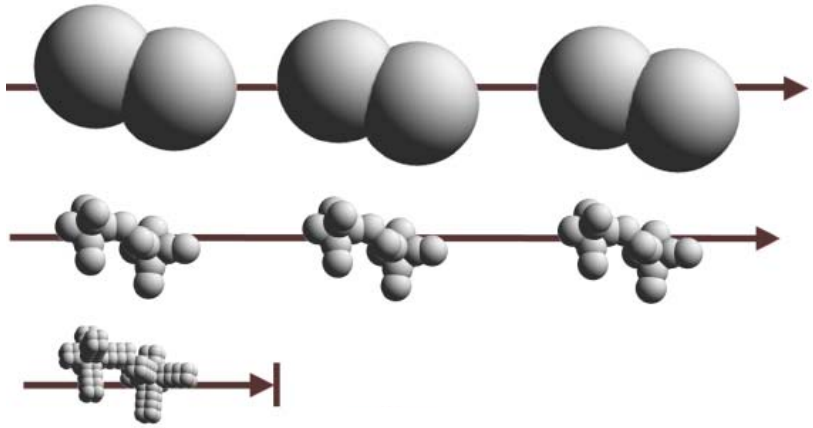
- Broadphase → potential colliding pairs
- For each pair
  - Update position/orientations of immediate children
  - Check collisions of nodes
  - Remove non-colliding nodes
  - Then traverse each colliding node (recursion)

**Two traversal options: Depth first / Breadth first**

# BVH TRAVERSAL



Depth first: pairs fully resolved before any others are processed



Breadth first: resolve everything partially first before moving on

# TIME CRITICAL COLLISION DETECTION [HUBBARD96]

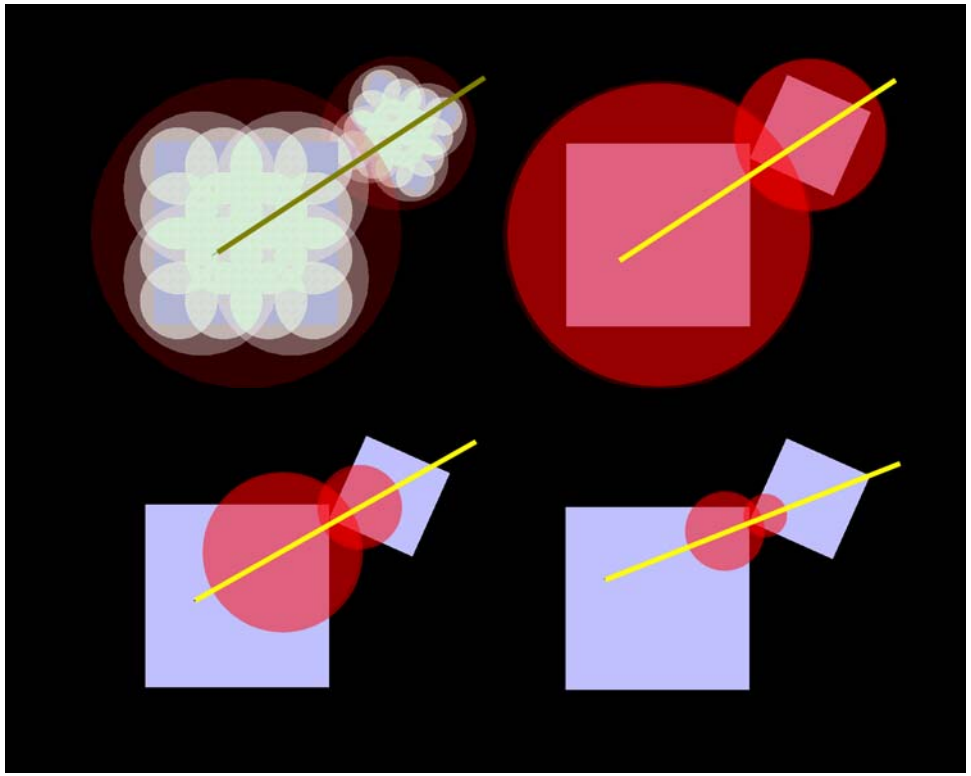
**If we run out of time stop traversal of BVH (Interruptible)**

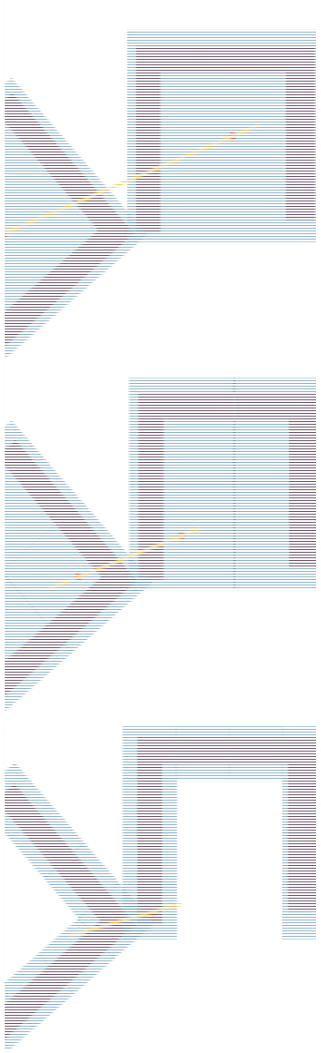
- Any potential colliders will be treated as colliding
- Typically breadth first traversal preferred
- Not widely used with collision response due to stability issues but useful for pure intersection testing in highly complex scenes

**Important question: which potential pairs do we process first**

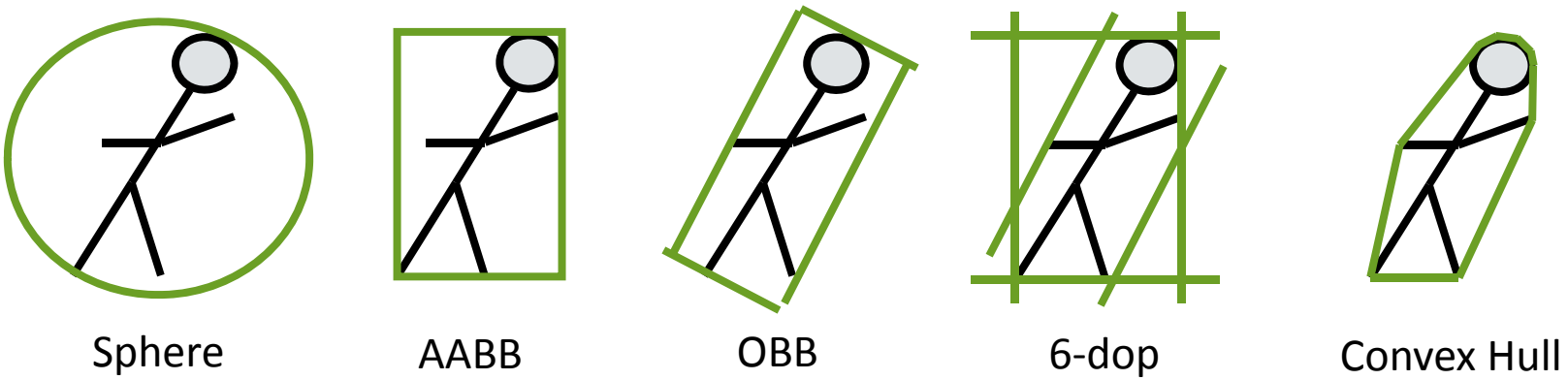
[Hubbard96] P. M. Hubbard. “Approximating polyhedra with spheres for time-critical collision detection”. ACM Trans. Graph., 15(3):179-210, July 1996.

# TIME CRITICAL CONTACT MODEL





# TRADE-OFF IN CHOOSING BVH'S




**increasing complexity & tightness of fit**

**decreasing cost of (overlap tests + BV update)**

FIGURE © M. LIN, UNC CHAPEL HILL

# COMPARING BVH APPROACHES

**Cost Function:** $F = Nu\,C_u + N_v\,C_v + N_p\,Cp$

**where:**

$F$: total cost function for interference detection

$N_u$: no. of bounding volumes updated

$C_u$: cost of updating a bounding volume,

$N_v$: no. of bounding volume pair overlap tests

$C_v$: cost of overlap test between 2 BVs

$N_p$: no. of primitive pairs tested for interference

$C_p$: cost of testing 2 primitives for interference

# MEMORY REQUIREMENTS

**Cost function:** $M = Ni\,Si + NL\,SL$

**where**

$M$ = total memory requirements in bytes

$N_i$ = number of internal nodes

$S_i$ = size of a single internal node in bytes

$N_L$ = number of leaf nodes

$S_L$ = size of a single leaf node in bytes

**Optimised memory strategies:**

- More than one primitive per leaf node
- Leaf node check not required

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

# SOME BVH SCHEMES

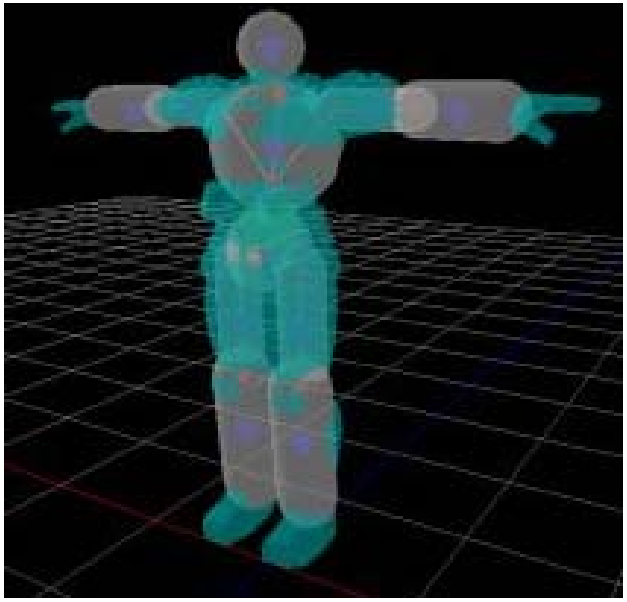
# 1. SPHERE TREE

**Advantages**

- Nodes are rotationally Invariant
- Relatively low memory usage
- Some extension for deformable bodies

**Disadvantages**

- Bad fit for long/thin/flat objects



**Where used, some examples:**

- Reportedly Gran Turismo
- Unreal Tournament 2003 rag-doll: about 15 rigid bodies, spheres+capsules

Ragdoll proxy image © Epic Games

# 1.1 UPDATE

**Spheres are rotationally invariant so just update the positions of the child nodes based on the orientation of the object**

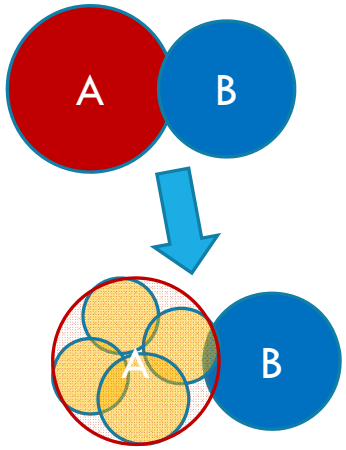
**Assuming c.o.m. is used as reference:**

```
for i=0 to number of children
    S_A.child[i].x(t) = R(t) * S_A.child[i].x(0) + x(t);
```

Where $\boldsymbol{x}(t)$ is position of object, $\boldsymbol{R}(t)$ is orientation matrix of object and $S_A.child[i].\boldsymbol{x}(0)$ is position of BVH node in Body space

# 1.2 COLLISION TESTING

```
bool CollisionCheck ( spheretree S_A, spheretree S_B)
{
  if ( distance(S_A.x(t), S_B.x(t)) < S_B.radius + S_A.radius )
  {
  for i = 0 to num_children_of _S_A, denoted S_A.child[i]
    S_A.child[i].x(t) = R(t) * S_A.child[i].x(0) + x(t);
    CollisionCheck (S_B , S_A.child[i]);
  }
  else return false;
}
```




**In practice depth first recursion can be problematic (esp. For time critical collision detection)**

- Instead we might place $S_B, S_A \rightarrow child[i]$ on a potential colliders list for the next iteration of processing

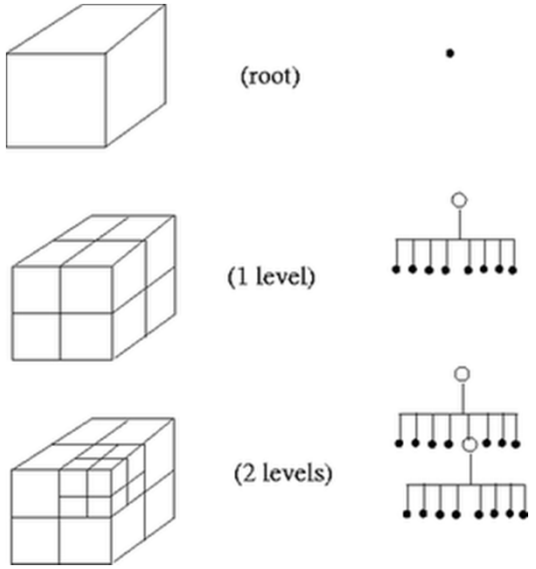
# 1.3 GENERATION: OCTREE-BASED

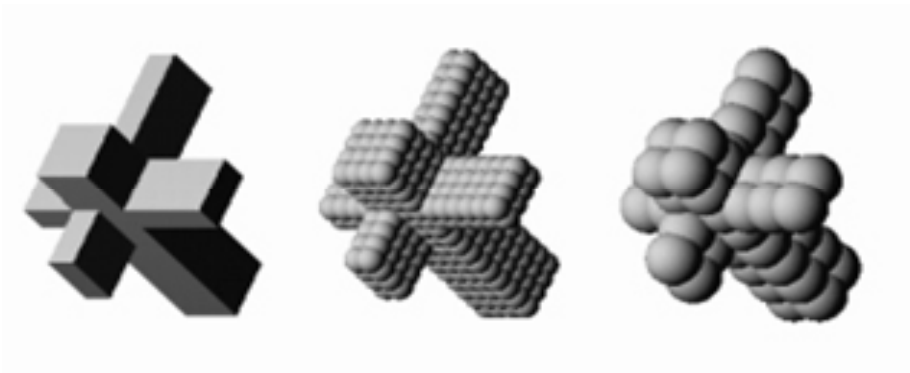
**Simple technique: Decompose object into Voxel Octtree**

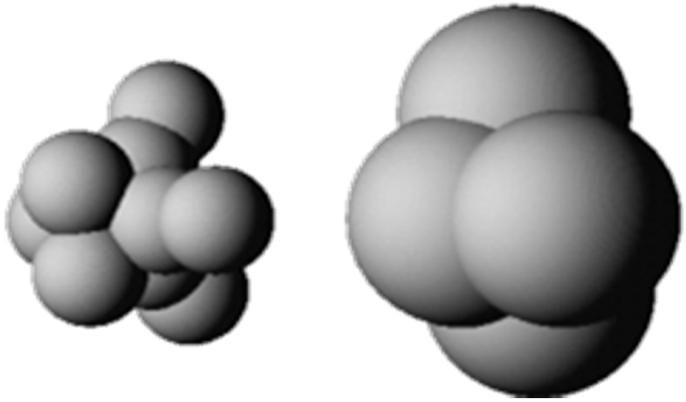
- Relatively common data structure

**Each occupied node of the octree (an object-space ABB) is bounded by a sphere (0.5*diagonal of octree cell)**

**Quick to code BUT not very efficient fit**

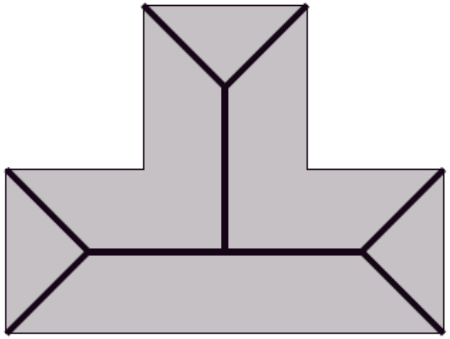


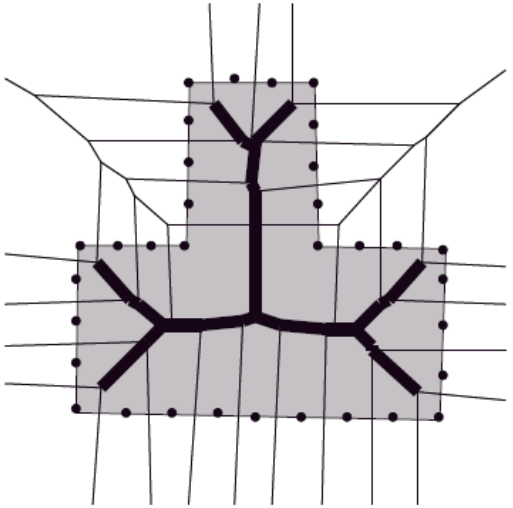


# 1.3 GENERATION: MEDIAL AXIS-BASED

- Non-uniform cell-sizes usually provide much better fit
- Common example is to use a medial-axis (m-rep) base

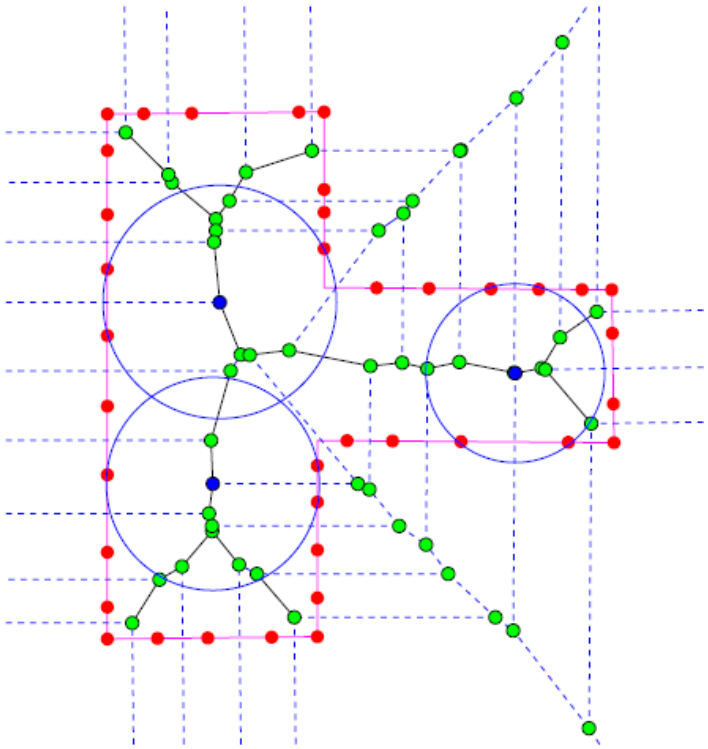


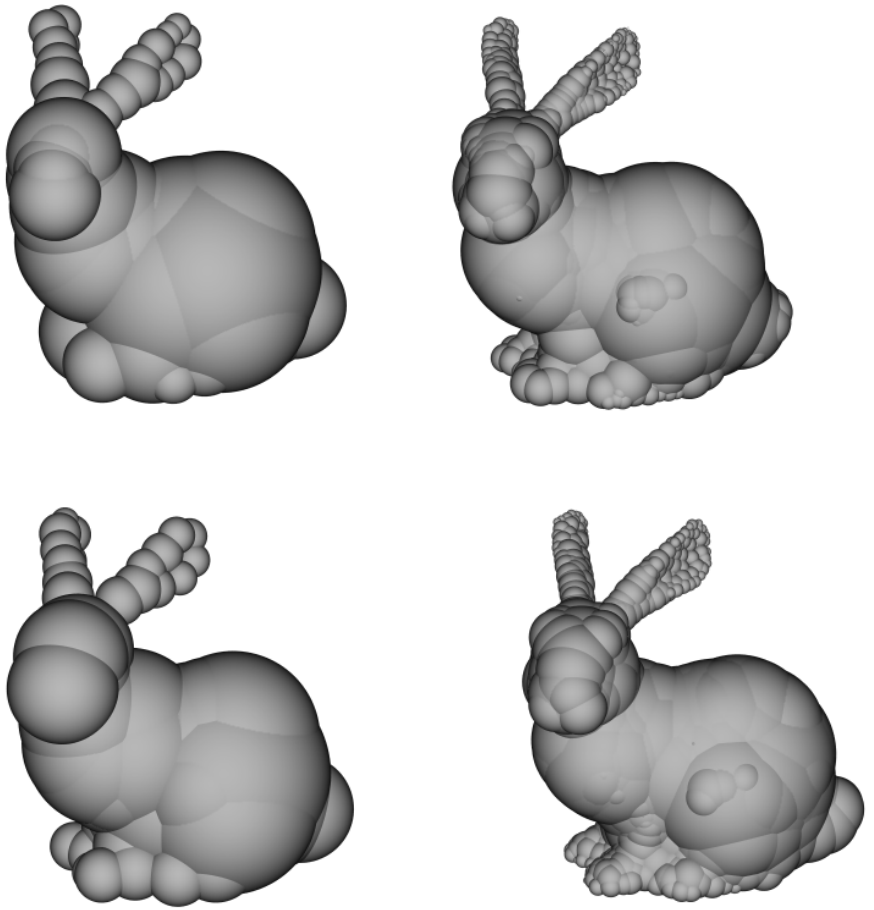
Medial axis of T- shape



Voronoi diagram used to approximate medial axis

Documentation + source available http://isg.cs.tcd.ie/spheretree/



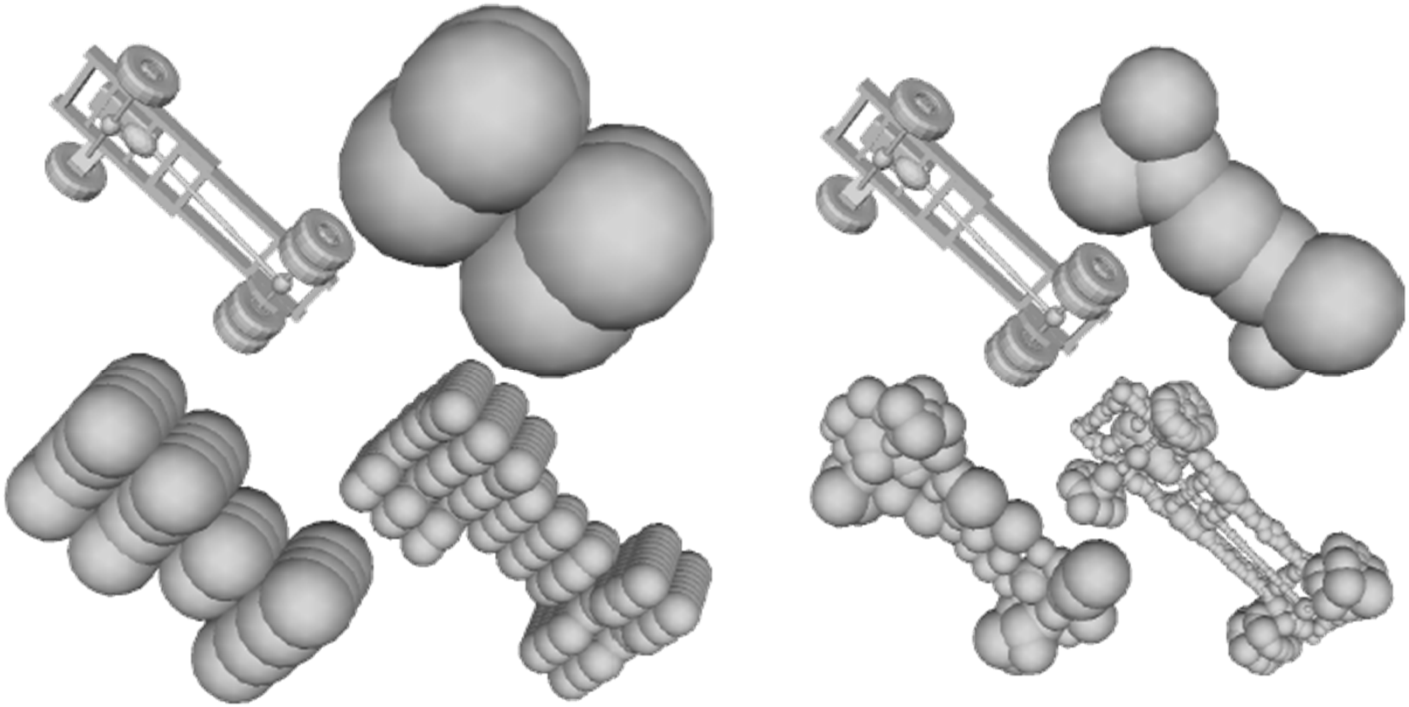


Spheres are placed on the Voronoi vertices that are inside the object.

http://isg.cs.tcd.ie/spheretree/

# COMPARISON

Medial-axis based sphere tree (right) tends to be about twice as efficient at bounding objects, compared to octree (left)



[Hubbard96] "Approximating Polyhedra with Spheres for Time-Critical Collision Detection" ACM Transactions on Graphics, 15(3). 1996
[Bradshaw] Sphere tree generation (code, binaries and papers): http://isg.cs.tcd.ie/spheretree

# 2. AABB TREES

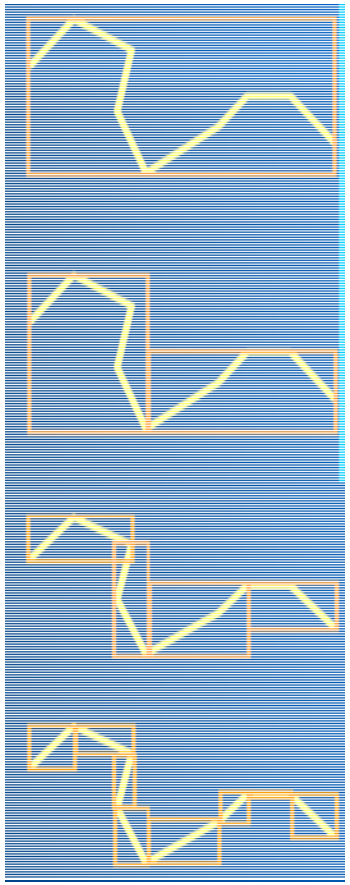
**Advantages**

- Applies almost equally well to deformable objects
- Easy to compute structure
- Easy intersection tests
- Often tighter fit than sphere tree
- Low update – if object doesn't rotate

**Disadvantages**

- Tree geometry varies with object rotation
- If object rotates, must recompute AABB at each frame

**Generally very useful at toplevel – usefulness is debatable at finer levels (depends on application)**



# 3. OBB TREES

**Advantages**

- Better fit than AABB and spheres
- Rotationally invariant in body space for rigid body

**Disadvantages**

- More expensive to update: must re-orient and reposition
- More expensive intersection test: **Two orders of magnitude** slower than checking two spheres for overlap
- More difficult to generate tree

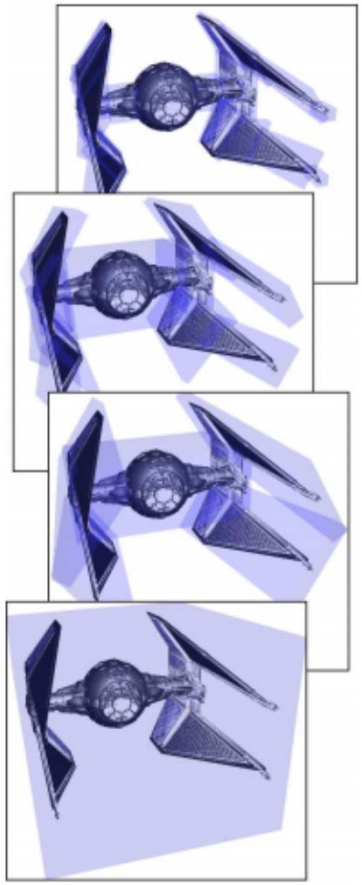
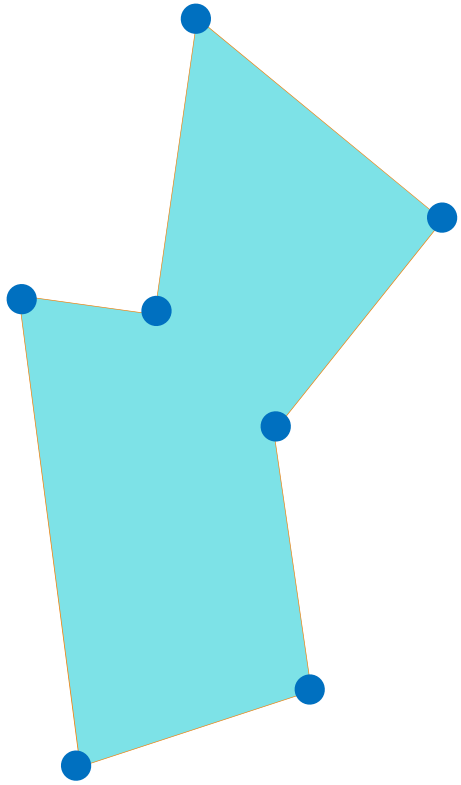


Image © K. A. Anderson and C. Bay

# OBB CONSTRUCTION

**For a set of polygons**

- Consider their vertices

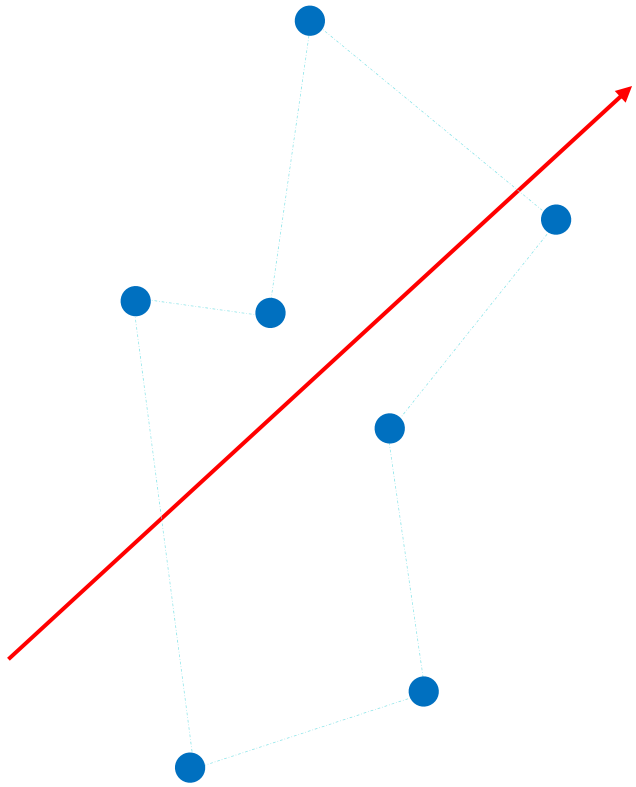


# OBB CONSTRUCTION

**For a set of polygons**

- Consider their vertices

**And an arbitrary line**



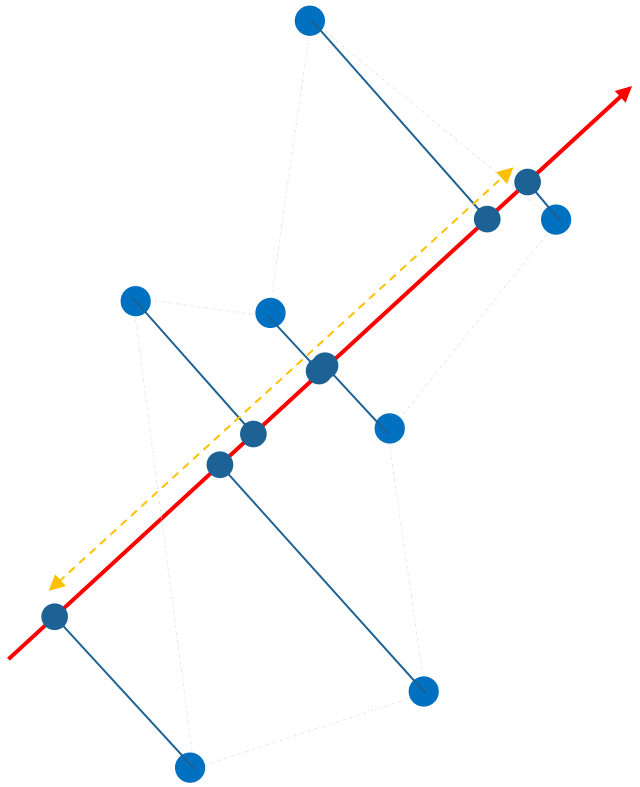
# OBB CONSTRUCTION

**For a set of polygons**

- Consider their vertices

**And an arbitrary line**

- Project the vertices onto the line
- Consider their variance (spread)



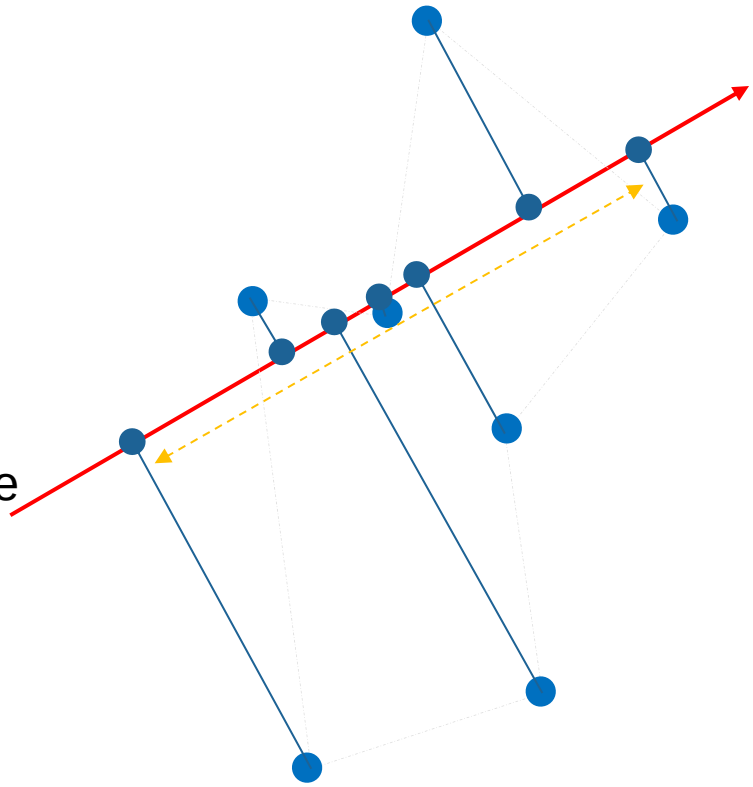
# OBB CONSTRUCTION

**For a set of polygons**

- Consider their vertices

**And an arbitrary line**

- Project the vertices onto the line
- Consider their variance (spread)
- Different line → different variance



# OBB CONSTRUCTION

**For a set of polygons**

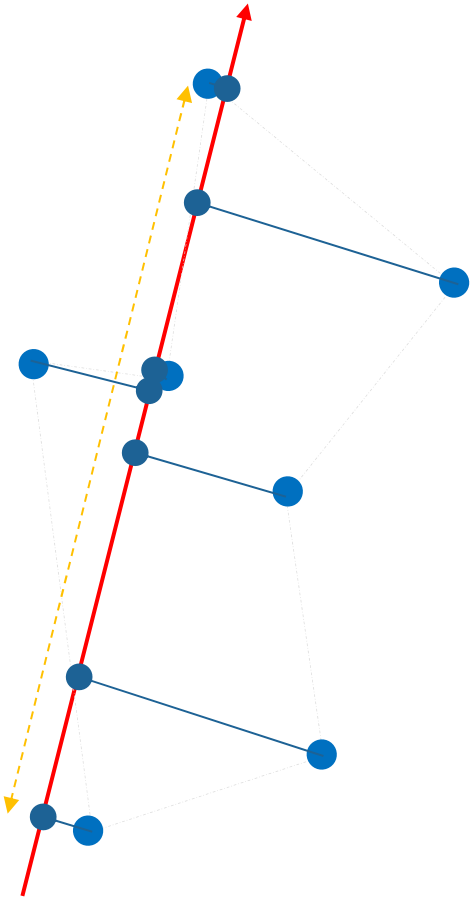
- Consider their vertices

**And an arbitrary line**

- Project the vertices onto the line
- Consider their variance (spread)
- Different line → different variance

**Given**

- the maximum variance



# OBB CONSTRUCTION

**For a set of polygons**

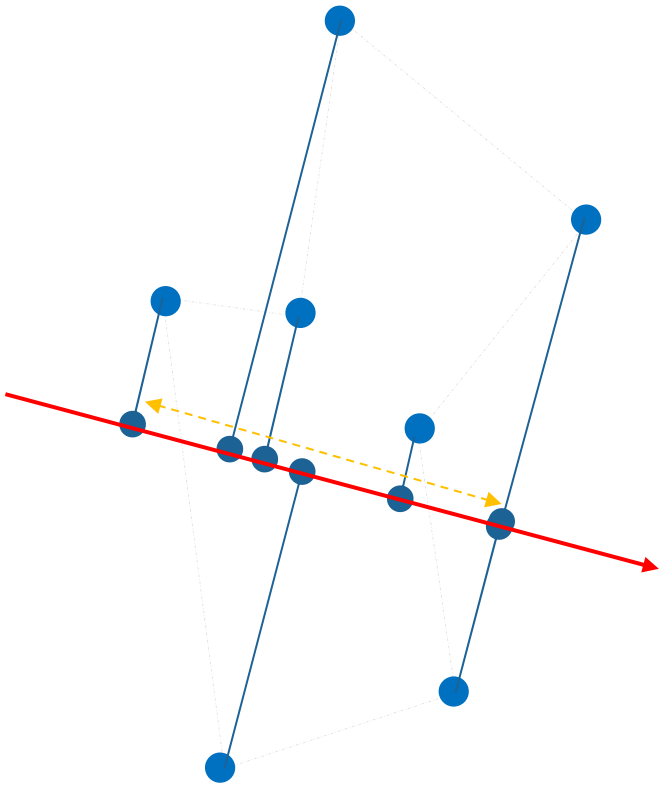
- Consider their vertices

**And an arbitrary line**

- Project the vertices onto the line
- Consider their variance (spread)
- Different line → different variance

**Given**

- the maximum variance
- and the minimum variance



# OBB CONSTRUCTION

**For a set of polygons**

- Consider their vertices

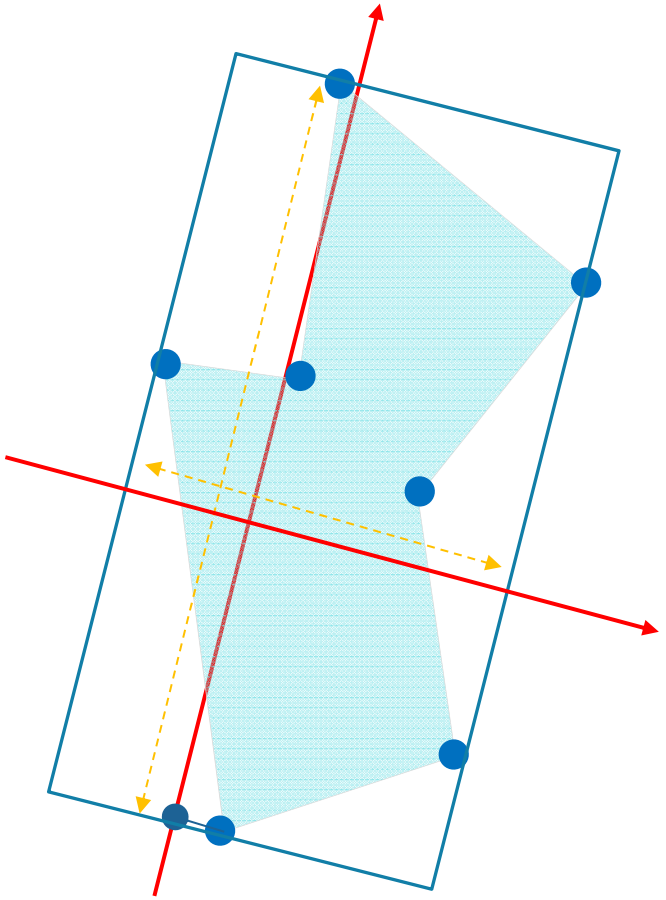
**And an arbitrary line**

- Project the vertices onto the line
- Consider their variance (spread)
- Different line → different variance

**Given**

- the maximum variance
- and the minimum variance

**Find orientation of the best fitting box**

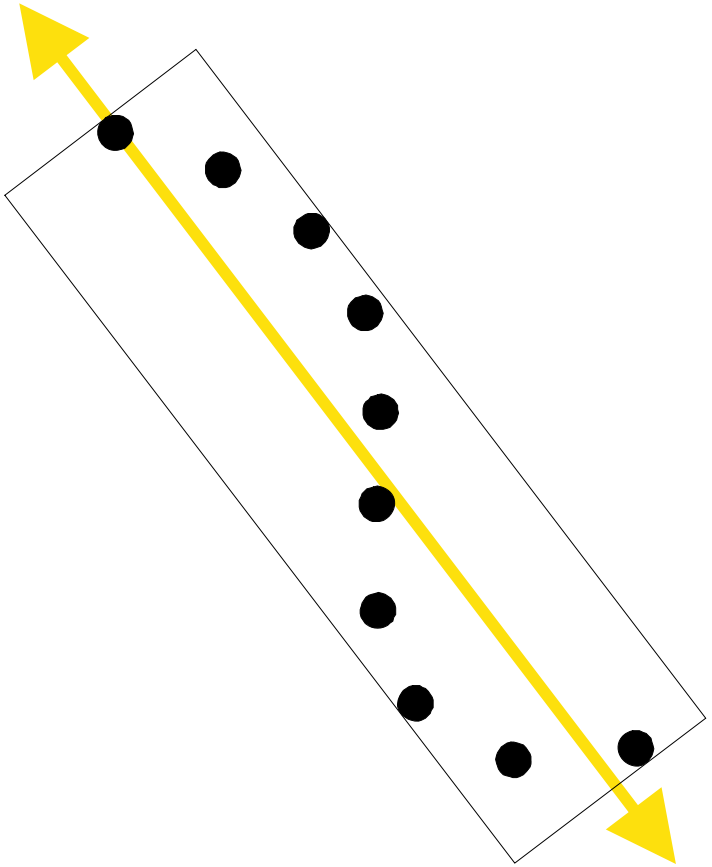


based on eigenvectors of the covariance matrix of the vertices

# BUILDING AN OBB TREE: FITTING

**Covariance matrix of point coordinates describes statistical spread of cloud.**

**The OBB is aligned with directions of greatest and least spread**



# FITTING OBBS

**Let the vertices of the i'th triangle be the points $ai$, $bi$, and $ci$, then the mean $\mu$ and covariance matrix $C$ can be expressed in vector notation as:**

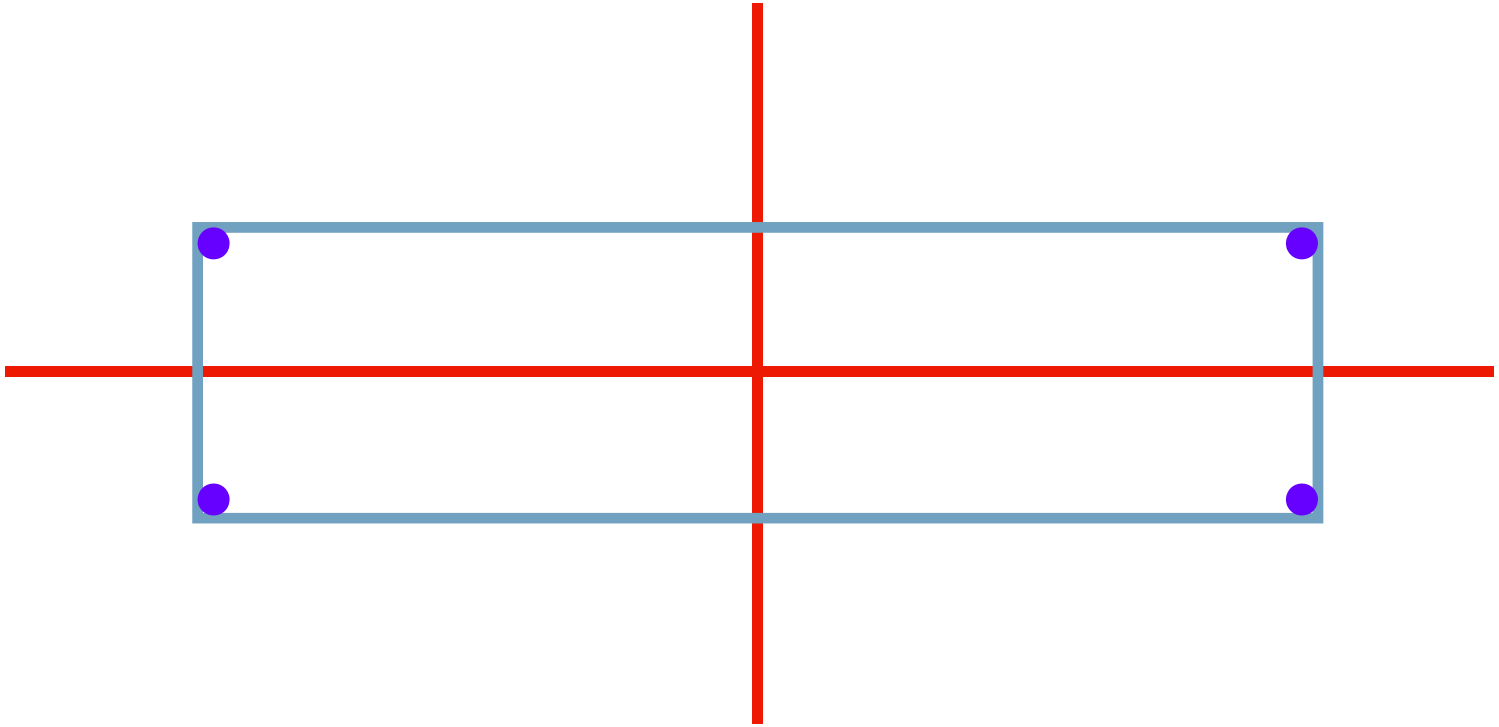
- $\boldsymbol{\mu} = \frac{1}{3n}\sum_{i=0}^{n}\left(\boldsymbol{a}^i + \boldsymbol{b}^i + \boldsymbol{c}^i\right)$
- $c_{jk} = \frac{1}{3n}\sum_{i=0}^{n}\left(\bar{\boldsymbol{a}}_j^i\bar{\boldsymbol{a}}_k^i + \bar{\boldsymbol{b}}_j^i\bar{\boldsymbol{b}}_k^i + \bar{\boldsymbol{c}}_j^i\bar{\boldsymbol{c}}_k^i\right), \quad 1 \le j,k \le 3$

**where n is the number of triangles, and**

- $\bar{\boldsymbol{a}}^i = \boldsymbol{a}^i - \boldsymbol{\mu},\ \bar{\boldsymbol{b}}^i = \boldsymbol{b}^i - \boldsymbol{\mu},\ \bar{\boldsymbol{c}}^i = \boldsymbol{c}^i - \boldsymbol{\mu}$

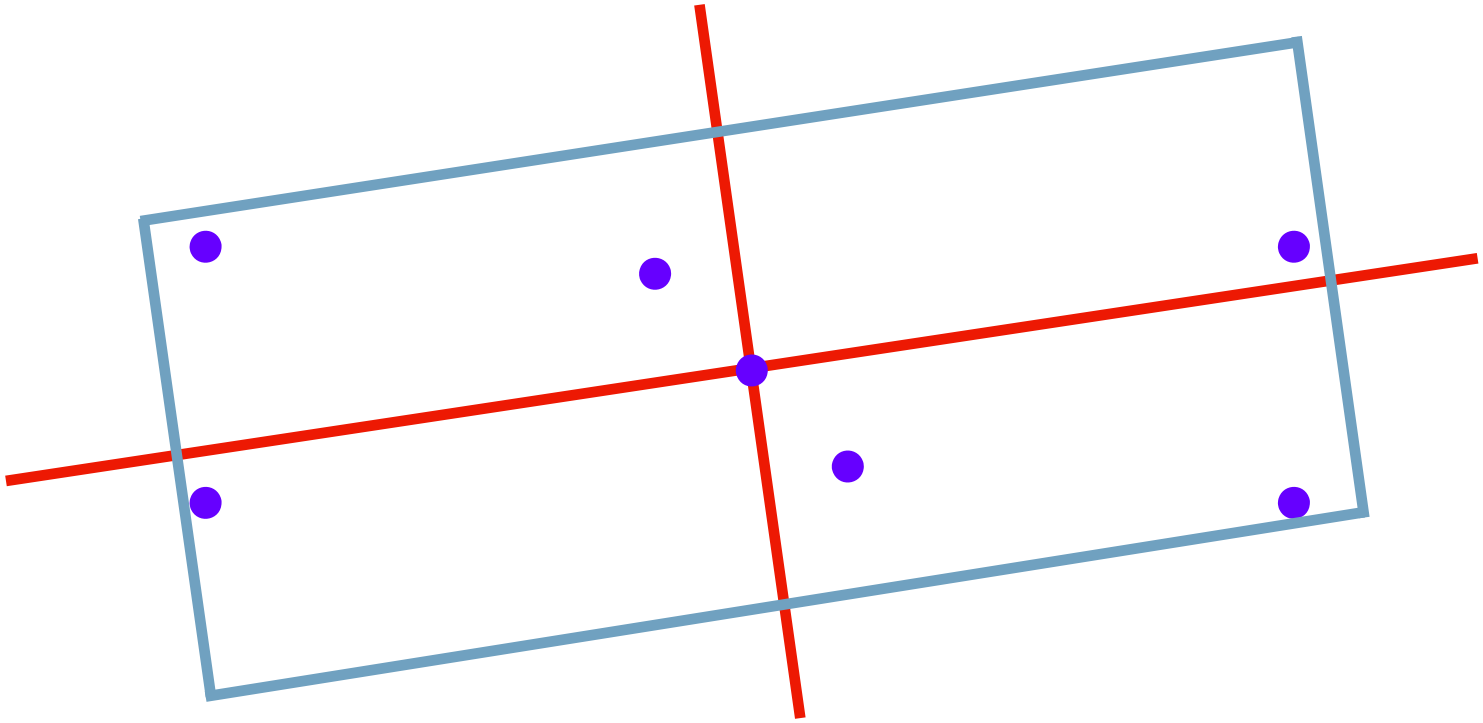
# BUILDING AN OBB TREE

**Good box**



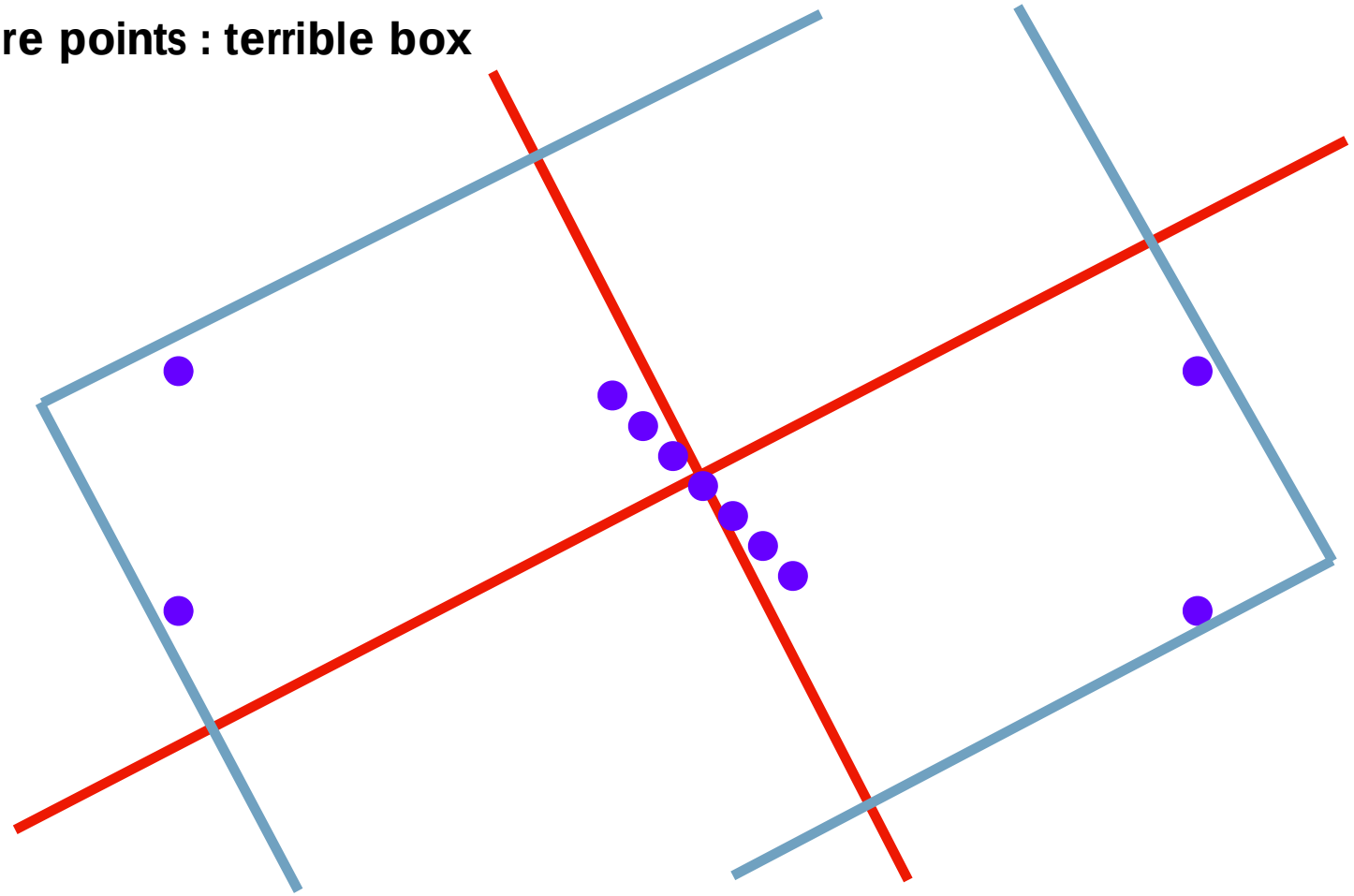
# BUILDING AN OBB TREE

**Add points : worse box**



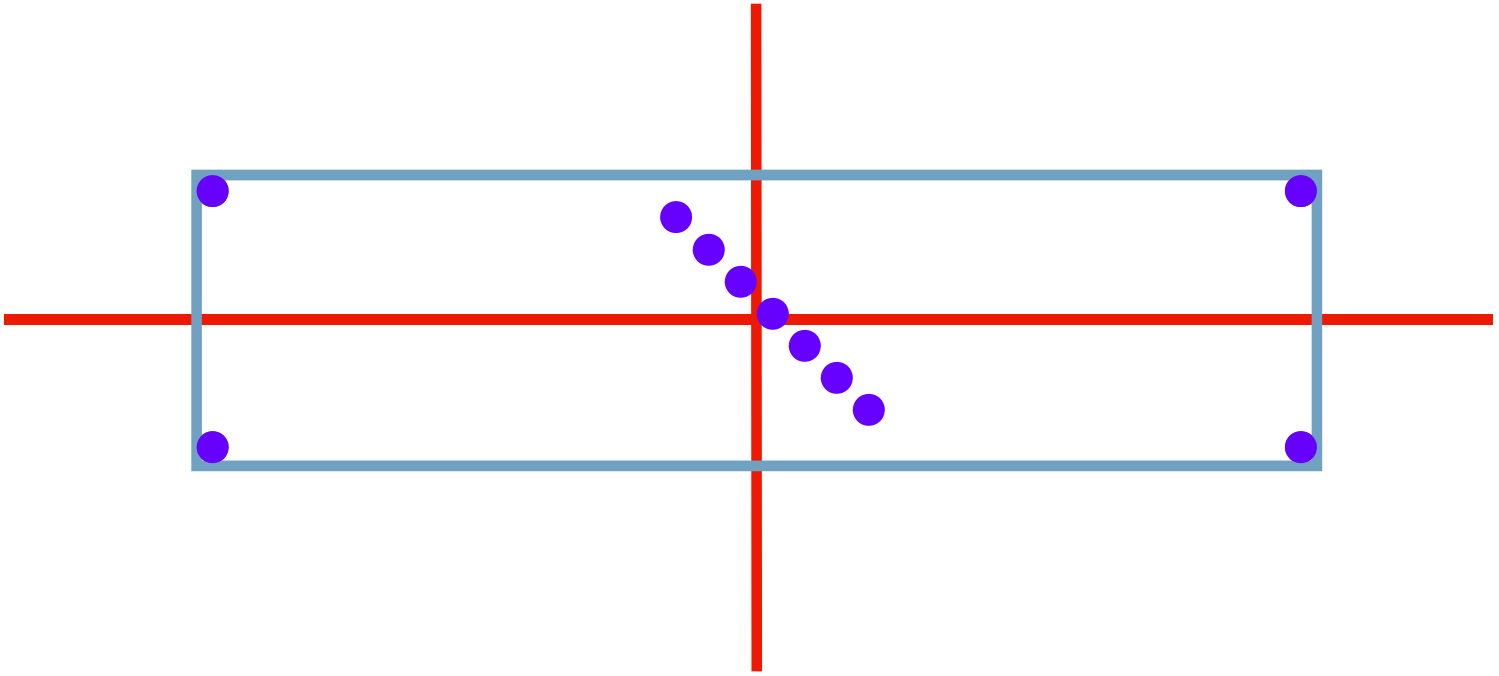
# BUILDING AN OBB TREE

**More points : terrible box**



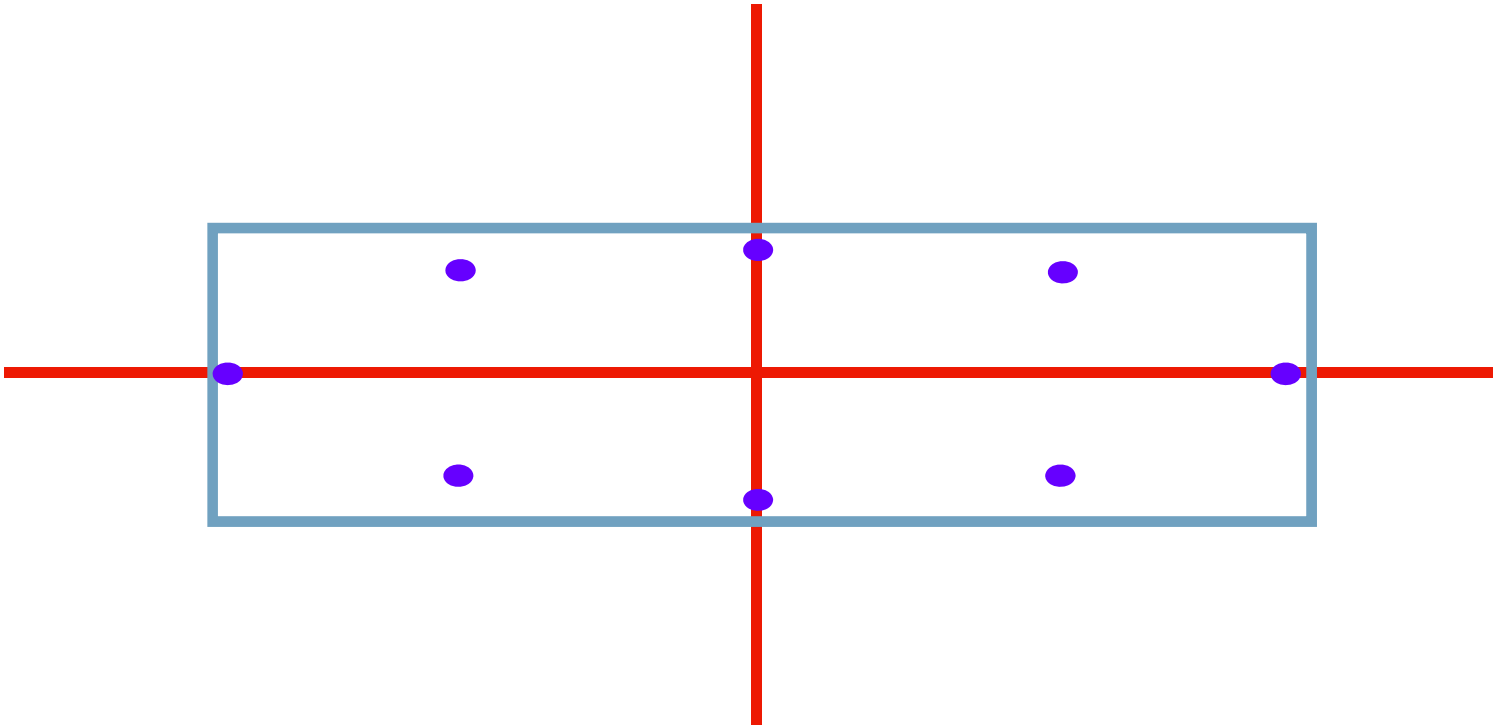
# BUILDING AN OBB TREE

**Compute with extremal points only**



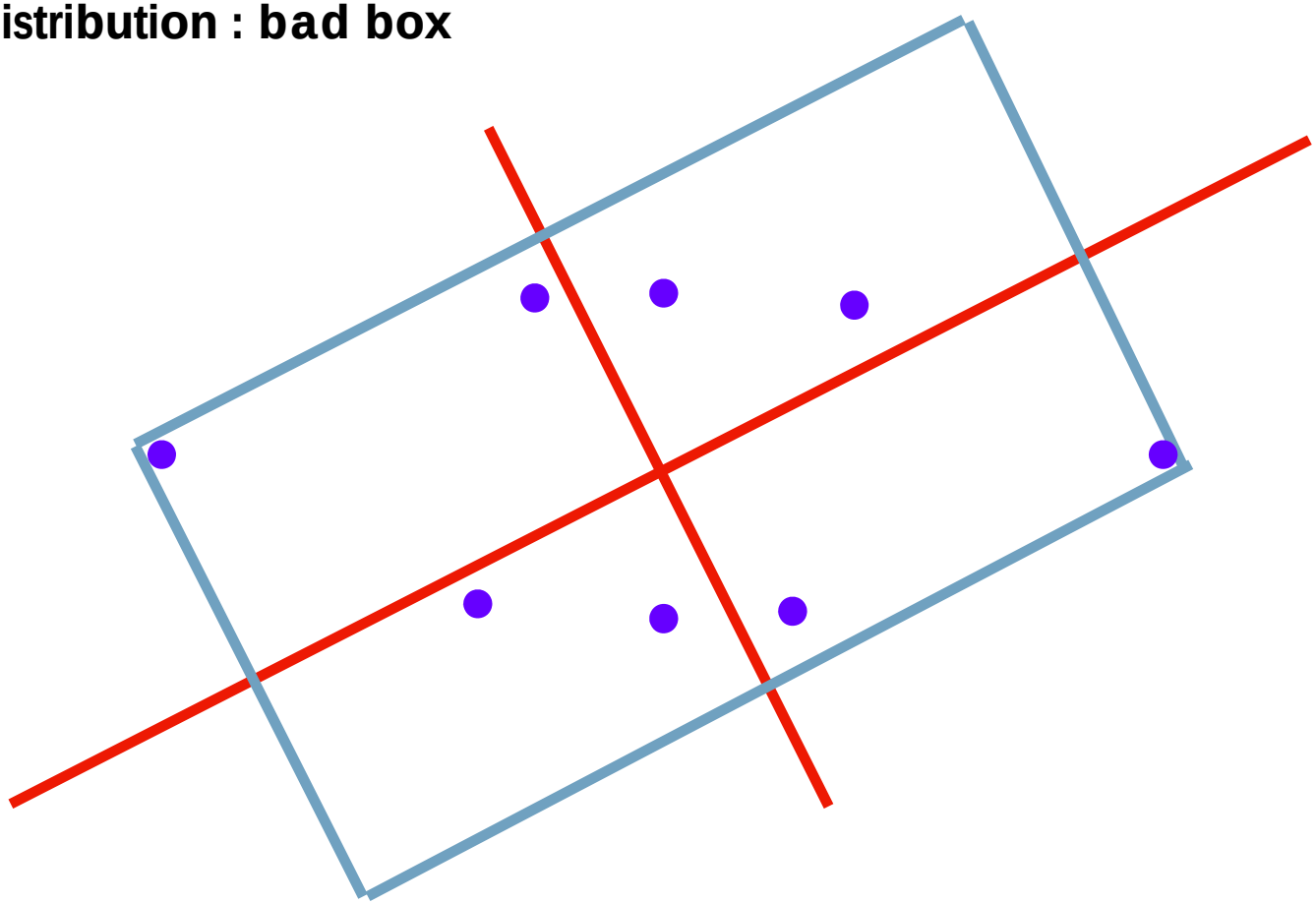
# BUILDING AN OBB TREE

**“Even” distribution : good box**



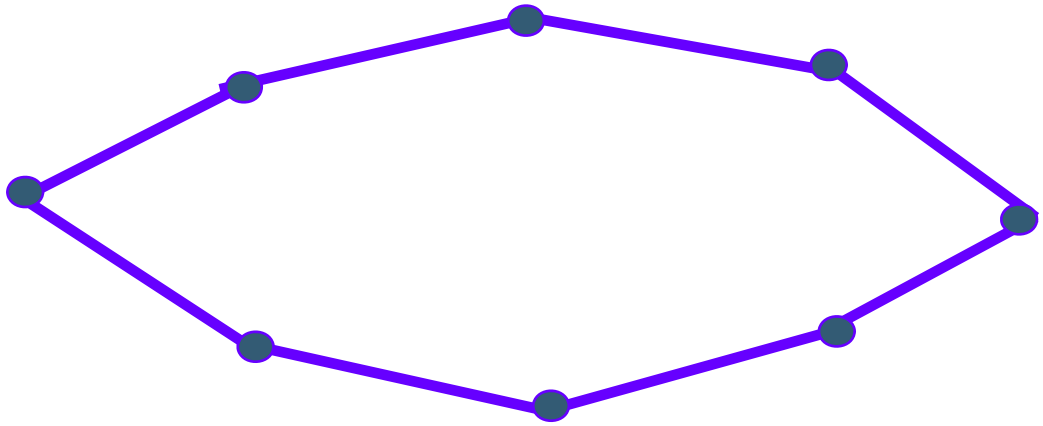
# BUILDING AN OBB TREE

**“Uneven” distribution : bad box**



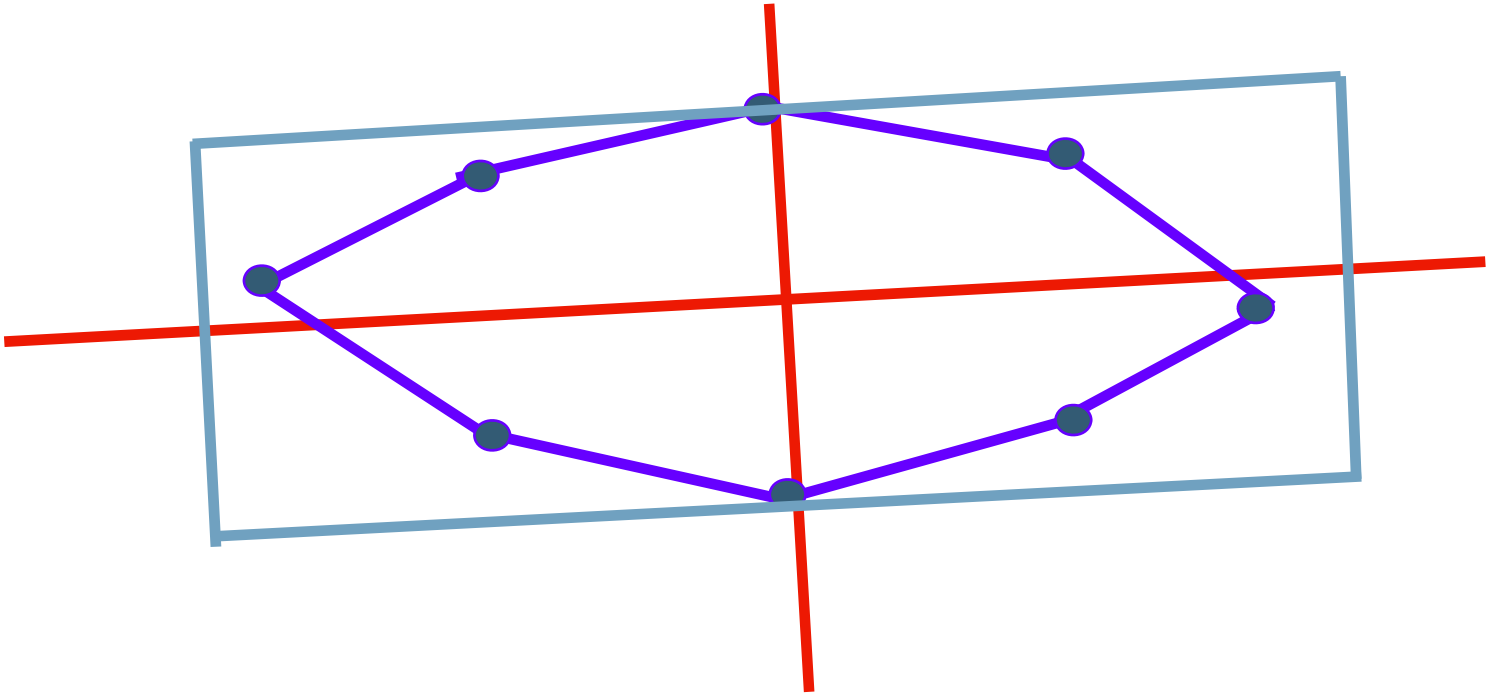
# BUILDING AN OBB TREE

**Fix : compute facets of convex hull**

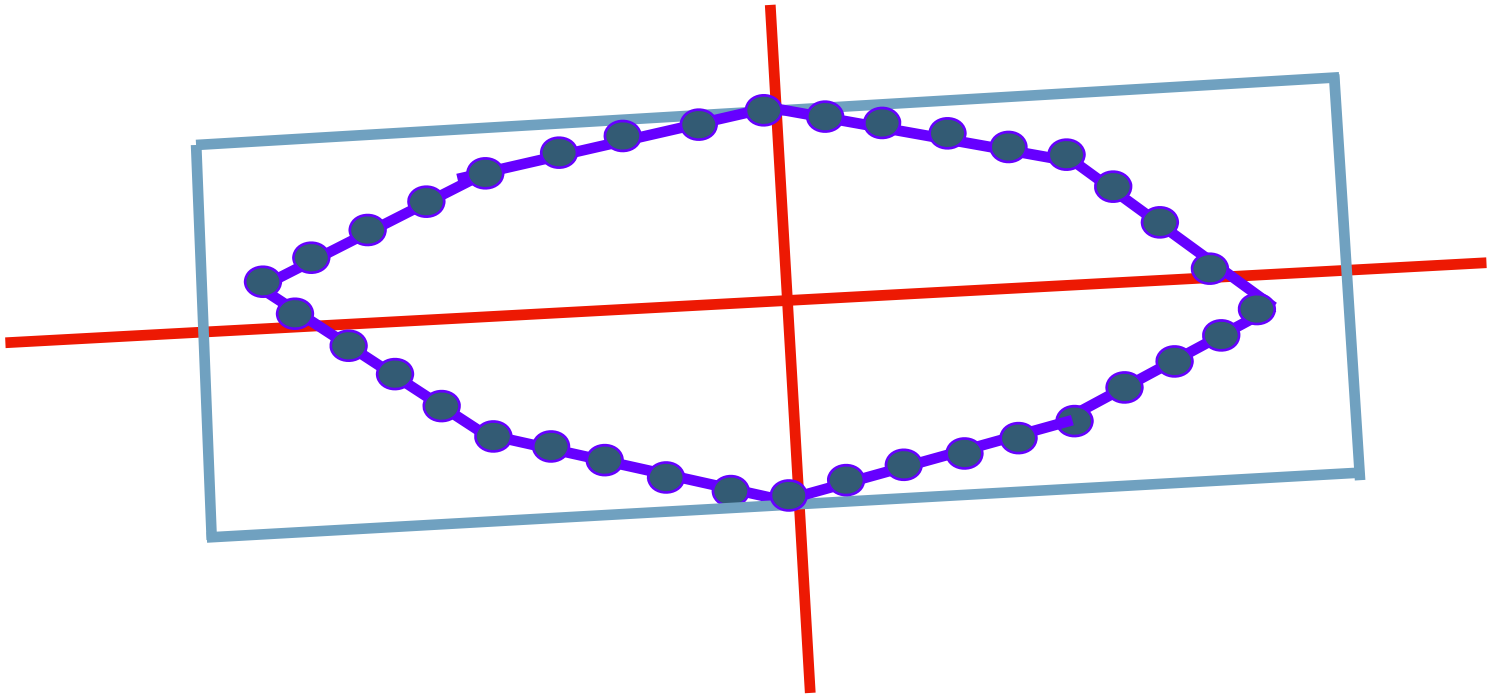


# BUILDING AN OBB TREE

**Better : integrate over facets**



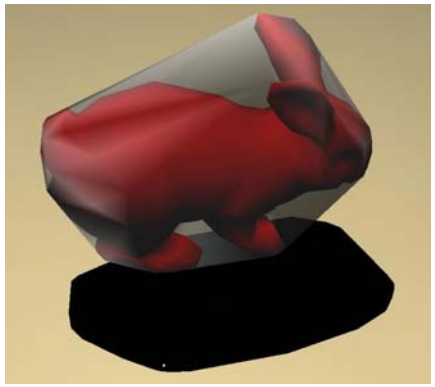
# BUILDING AN OBB TREE

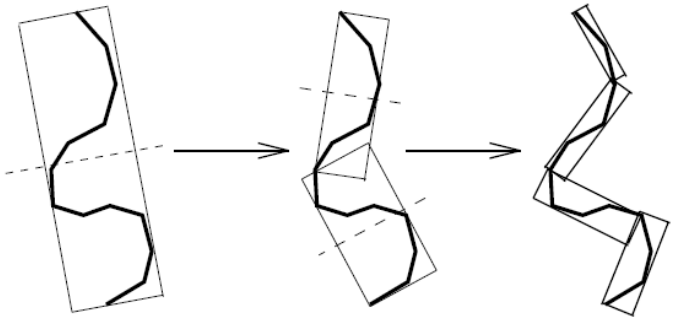


# OBB GENERATION

**For each primitive group, by Principal component analysis (covariance based)**

- Find convex hull* of object(e.g. http://www.qhull.org/)
- Essentially a representation of the enclosing ellipsoid
- Find best BB to fit this



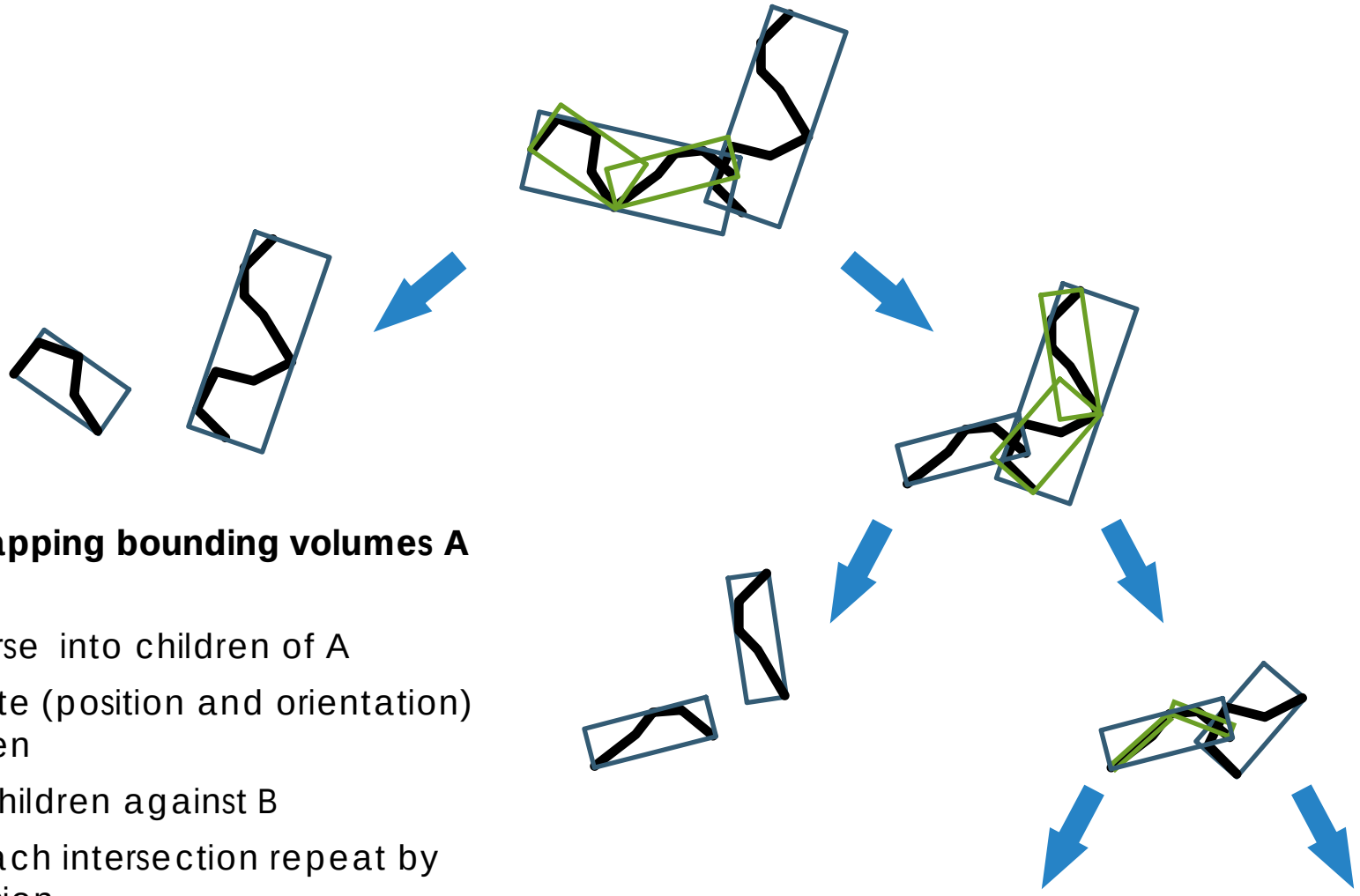


**Building the tree: typically top-down(split operation)**

**Cost:**

- O(n log n) fitting time for single BV
- O(n $\log^2$ n) fitting time for entire tree

# TREE TRAVERSAL



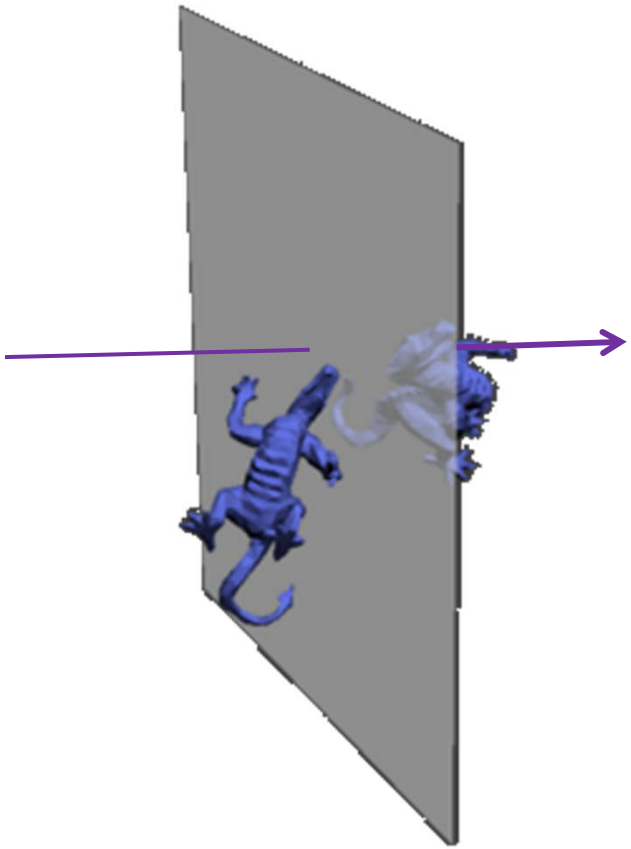
**If overlapping bounding volumes A and B:**

- Traverse into children of A
- Update (position and orientation) chidren
- Test children against B
- For each intersection repeat by recursion

# OBB INTERSECTION TEST

**Separating axis theorem:** Two convex polytopes are disjoint iff there exists a separating axis orthogonal to a face of either polytope or orthogonal to an edge from each polytope.

**Separating Axis:** an axis on which the projections of two polytopes don't overlap.



# SEPARATING AXIS TEST (IN ONE SLIDE)

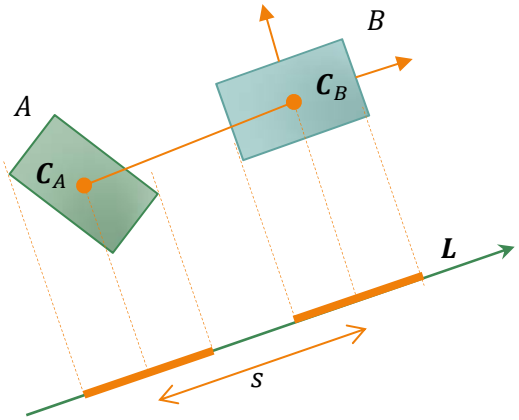
Determine Candidate Axes → Find Midpoint Separation → Find Half-length of projected intervals

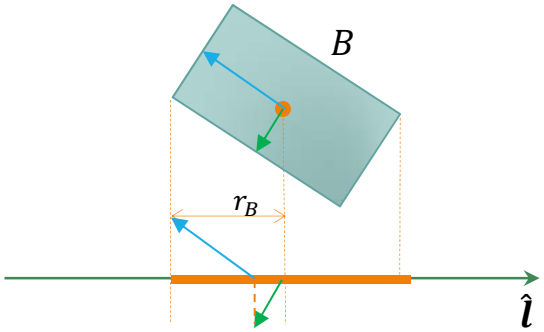
Face normal directions

$$\boldsymbol{u_0^A} \quad \boldsymbol{u_1^A} \quad \boldsymbol{u_2^A}$$
$$\boldsymbol{u_0^B} \quad \boldsymbol{u_1^B} \quad \boldsymbol{u_2^B}$$

Cross product of edge directions (for boxes these are orthogonal to face normals)

$$\boldsymbol{u_0^A \times u_0^B} \quad \boldsymbol{u_0^A \times u_1^B} \quad \boldsymbol{u_0^A \times u_2^B}$$
$$\boldsymbol{u_1^A \times u_0^B} \quad \boldsymbol{u_1^A \times u_1^B} \quad \boldsymbol{u_1^A \times u_2^B}$$
$$\boldsymbol{u_2^A \times u_0^B} \quad \boldsymbol{u_2^A \times u_1^B} \quad \boldsymbol{u_2^A \times u_2^B}$$




$$s = \left|(\boldsymbol{C_A} - \boldsymbol{C_B}) \cdot \hat{\boldsymbol{l}}\right|$$

$$r_B = e_1^B \left|\boldsymbol{u_1^B} \cdot \hat{\boldsymbol{l}}\right| + e_2^B \left|\boldsymbol{u_2^B} \cdot \hat{\boldsymbol{l}}\right| + e_3^B \left|\boldsymbol{u_3^B} \cdot \hat{\boldsymbol{l}}\right|$$

# COMPARISON

Cost Function: $F = N_u \, x \, C_u \; + \; N_v \, x \, C_v + N_p \, x \, C_p$

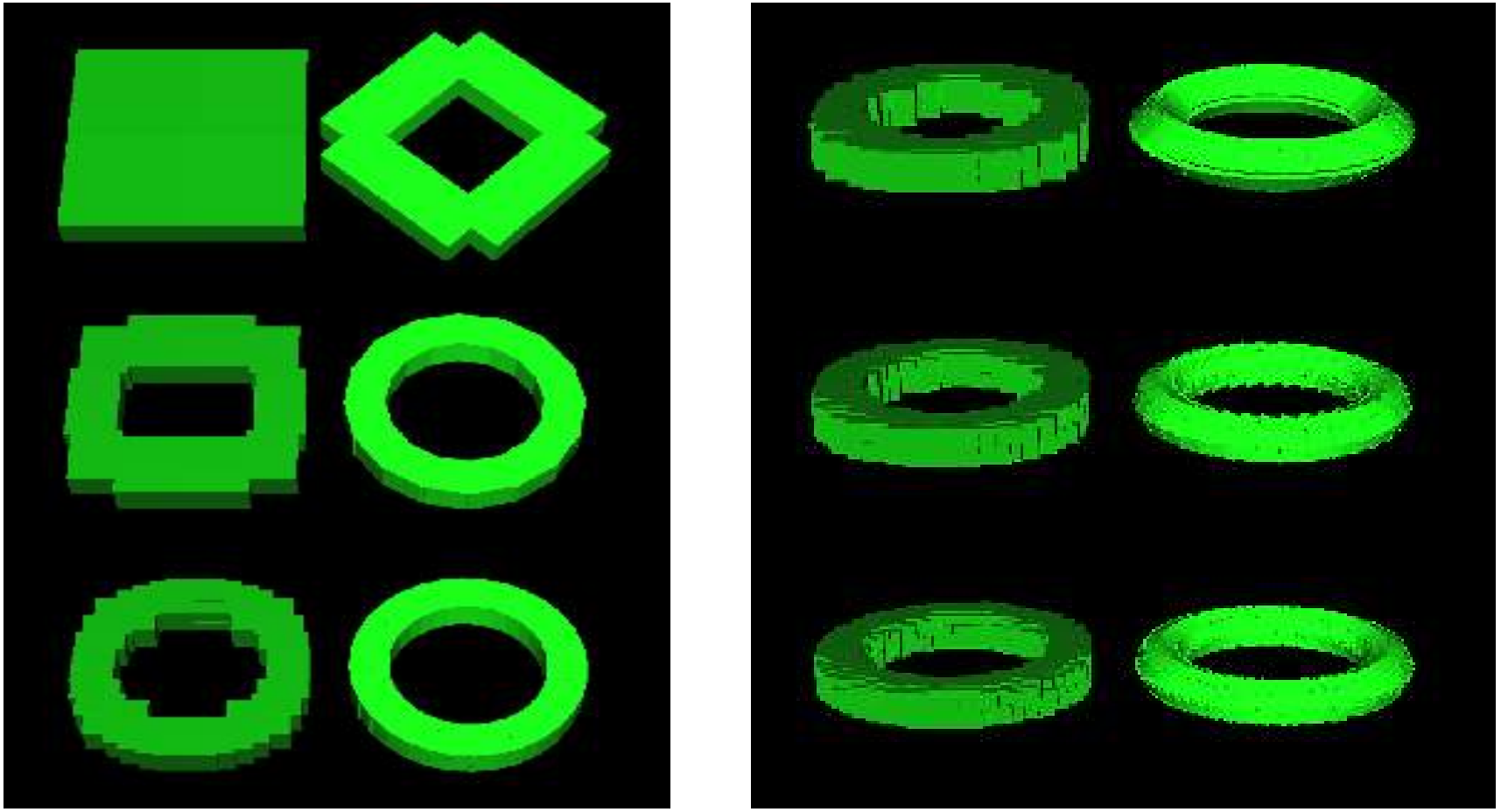
| | |
|---|---|
| $F$: | total cost function for interference detection |
| $N_u$: | no. of bounding volumes updated |
| $C_u$: | cost of updating a bounding volume, |
| $N_v$: | no. of bounding volume pair overlap tests |
| $C_v$: | cost of overlap test between 2 BVs |
| $N_p$: | no. of primitive pairs tested for interference |
| $C_p$: | cost of testing 2 primitives for interference |

**Volume-node intersection $C_v$ is one-order of magnitude slower than that for sphere trees or AABBs.**

**Primary advantage for OBB: Low $N_v$ and $N_p$**

- In general, OBBs can bound geometry more tightly than AABB Trees and sphere trees.

# OBB VS AABB COMPARISON

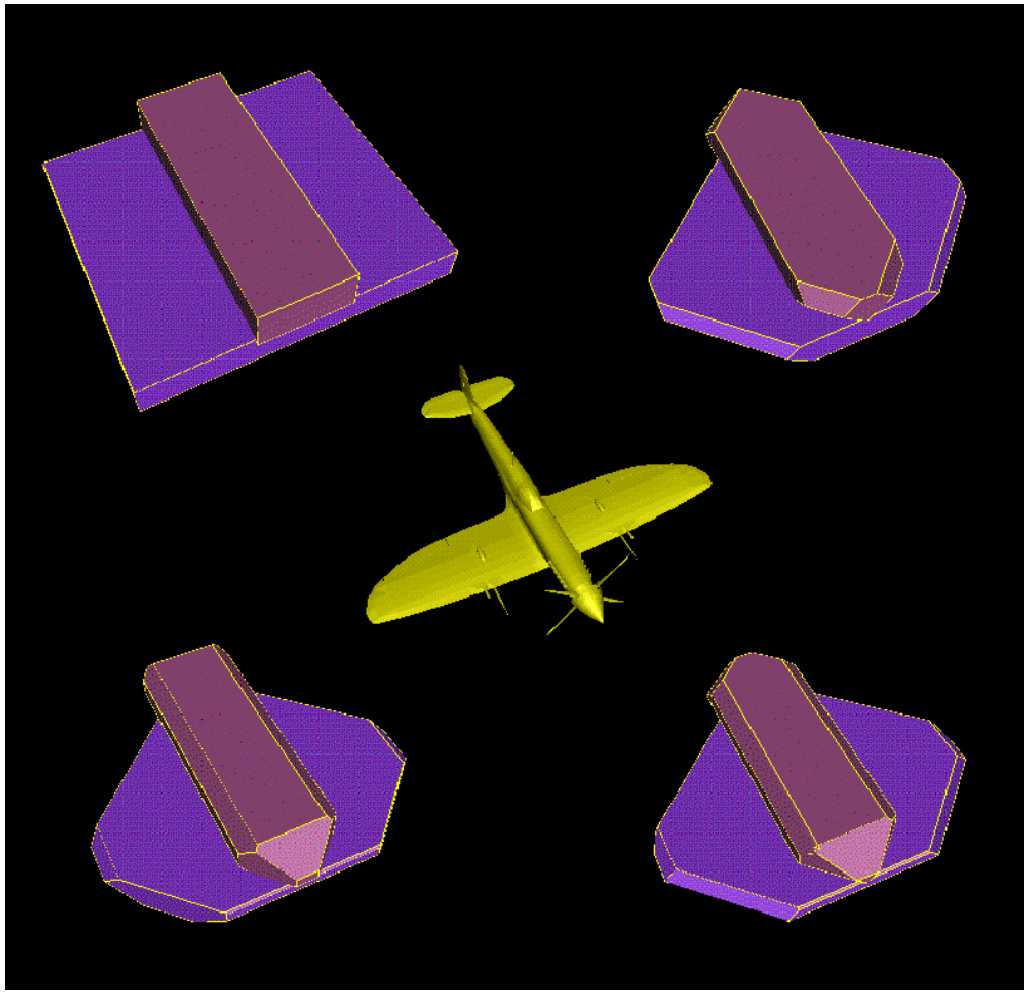


OBB converges faster to the shape of the torus

# K-DOP TREE: LEVEL 1

6-DOP

14-DOP



18-DOP

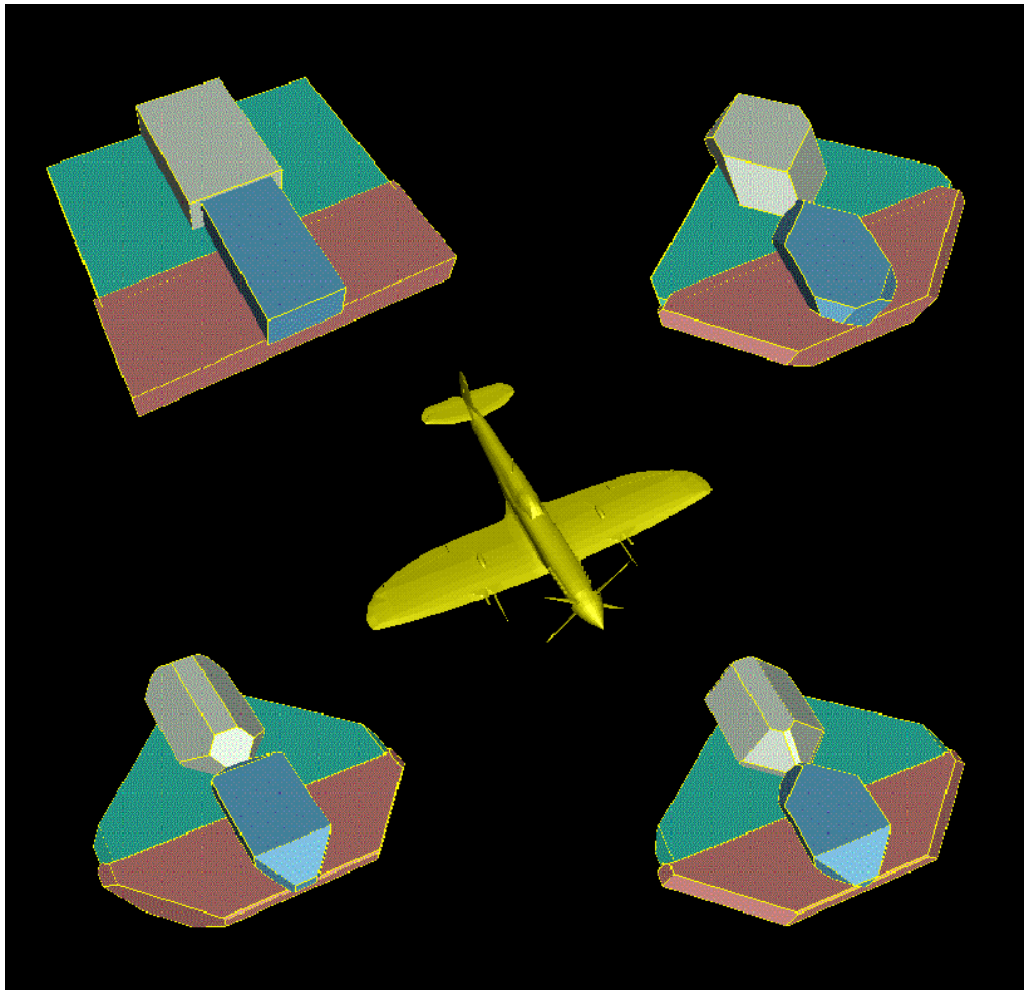
26-DOP

Images © Martin Held

# K-DOP TREE : LEVEL 2

6-DOP

14-DOP



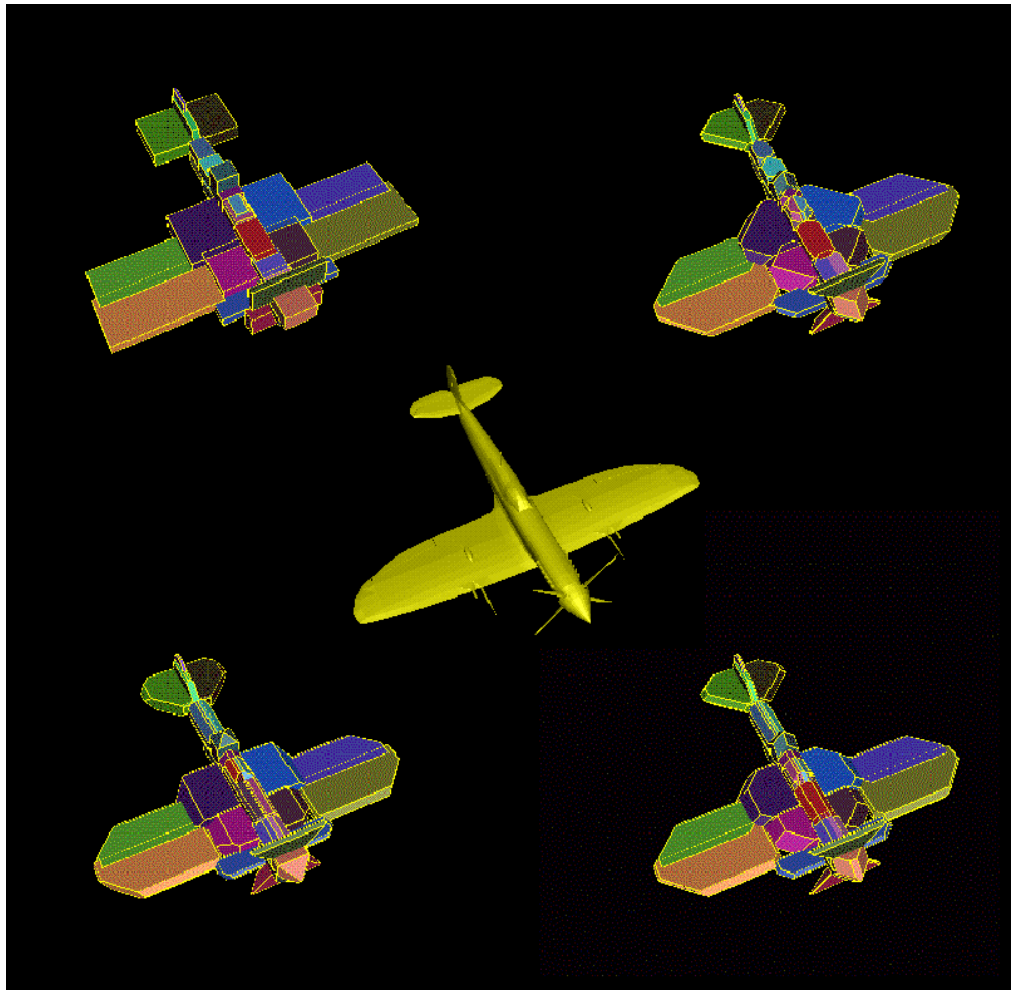
18-DOP

26-DOP

Images © Martin Held

# K-DOP TREE: LEVEL 3

6-DOP



14-DOP

18-DOP

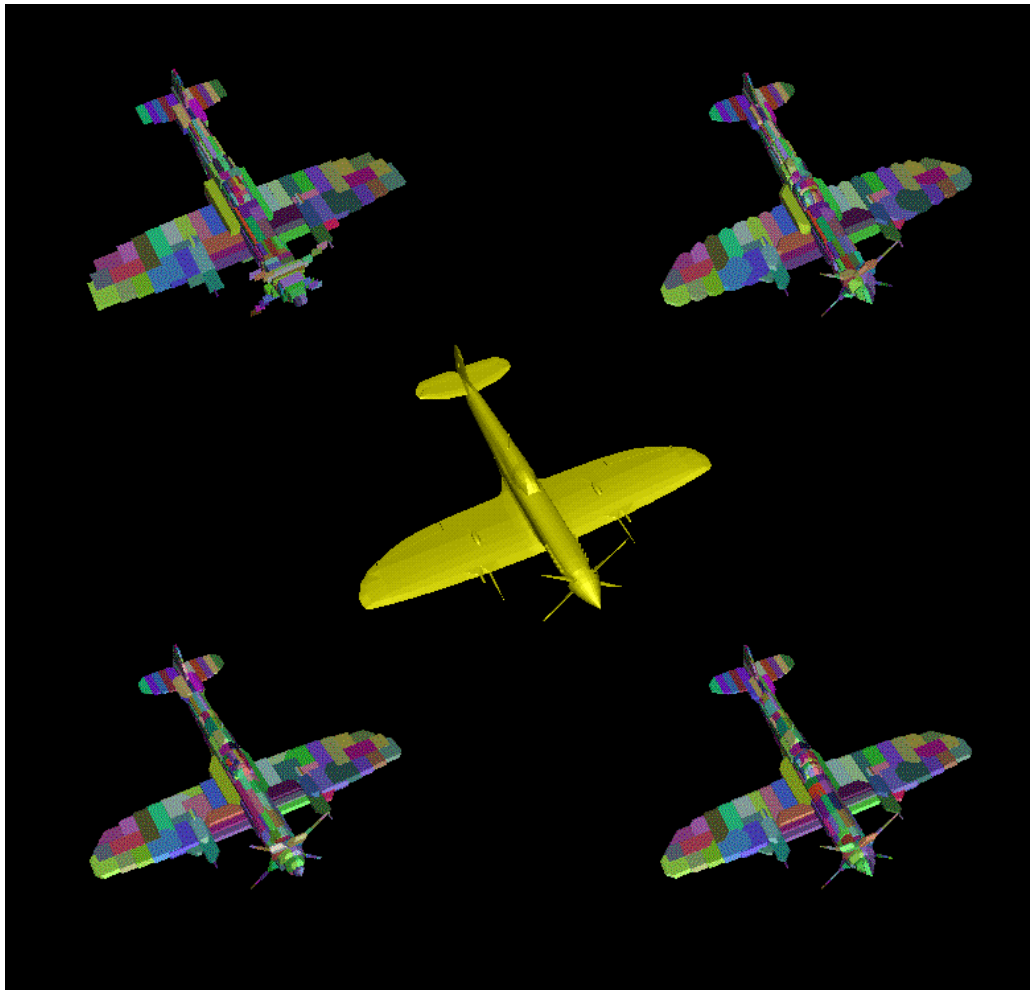
26-DOP

Images © Martin Held

# K-DOP TREE : LEVEL 4

6-DOP

14-DOP



18-DOP

26-DOP

Images © Martin Held