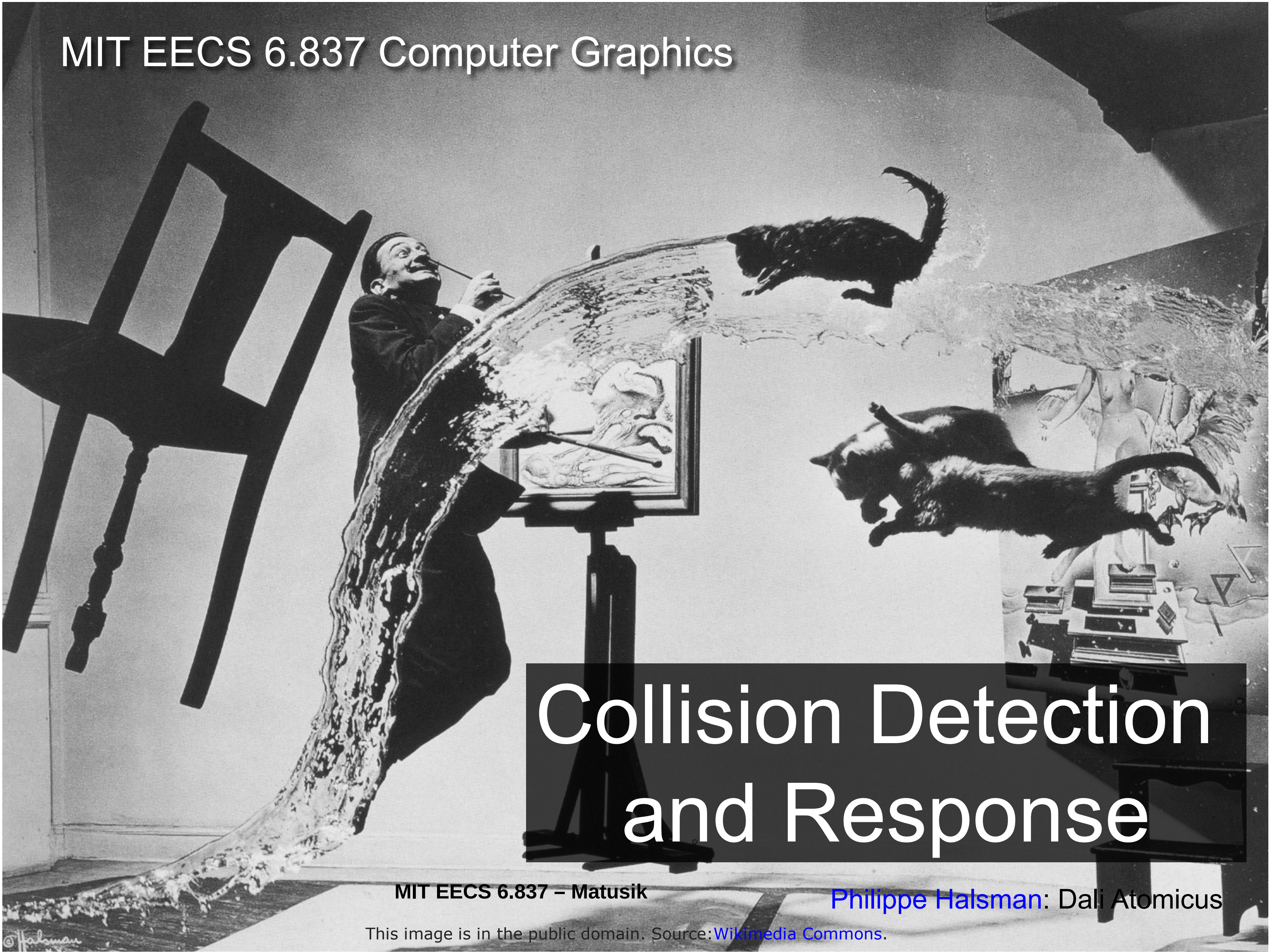


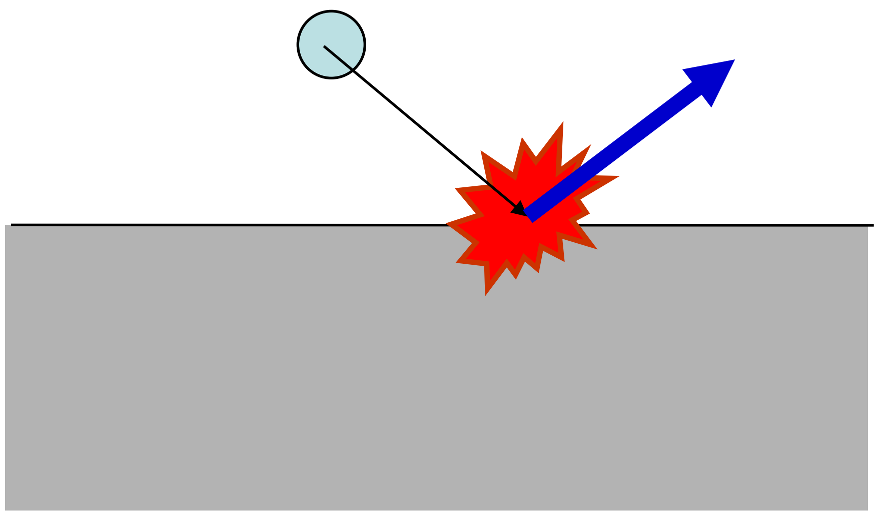
MIT EECS 6.837 Computer Graphics

Collision Detection and Response

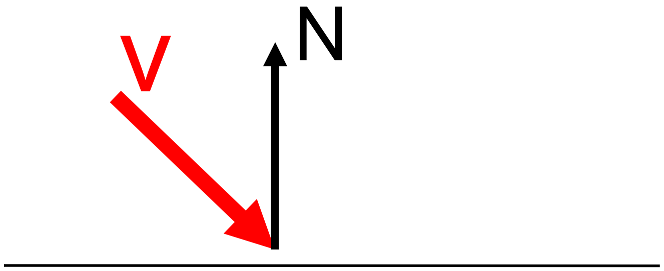
Philippe Halsman: Dali Atomicus

Collisions

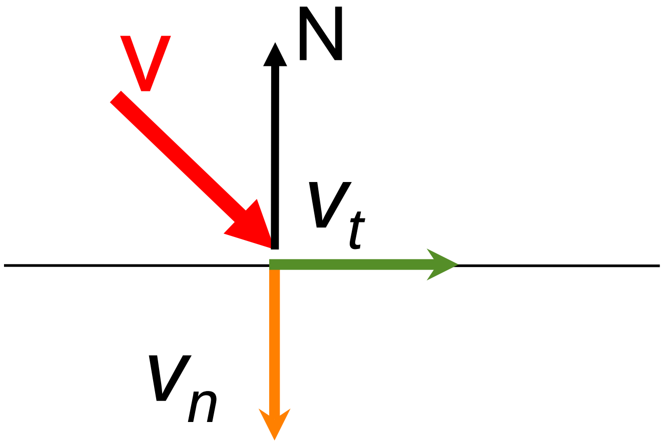
- Detection
- Response
- Overshooting problem (when we enter the solid)



Collision Response for Particles



Collision Response for Particles



$v = v_n + v_t$

normal component
tangential component

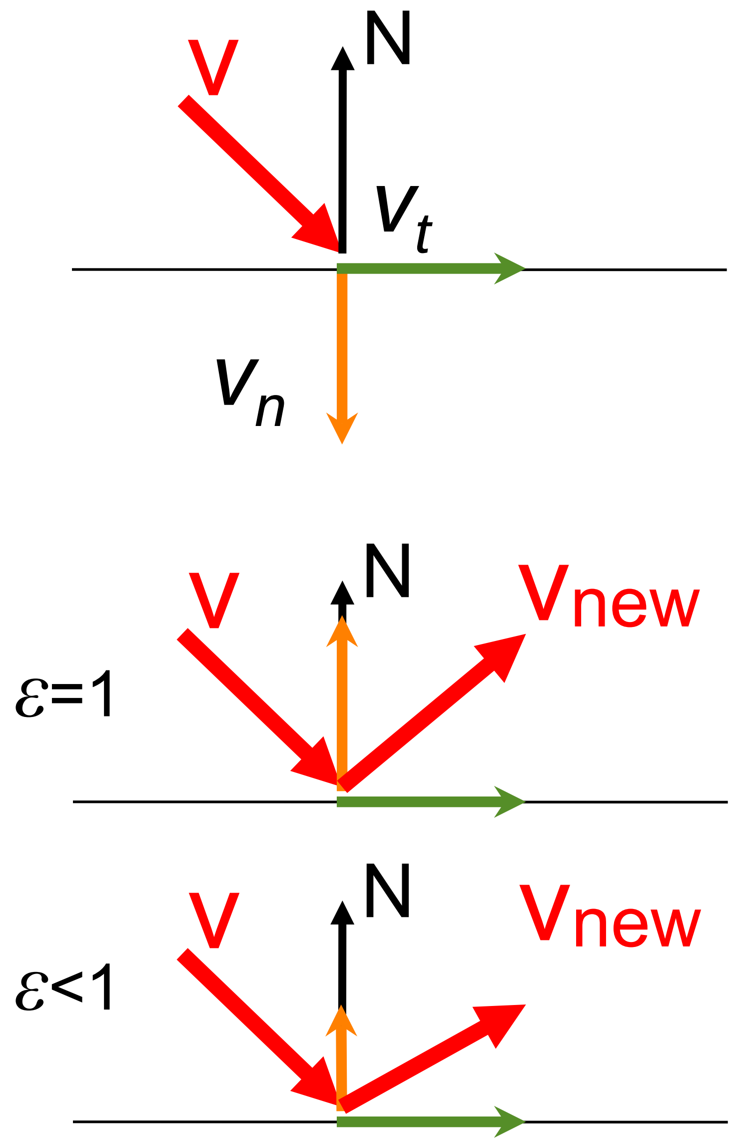
Collision Response for Particles

- Tangential velocity v_t *often* unchanged
- Normal velocity v_n reflects:

$$v = v_t + v_n$$

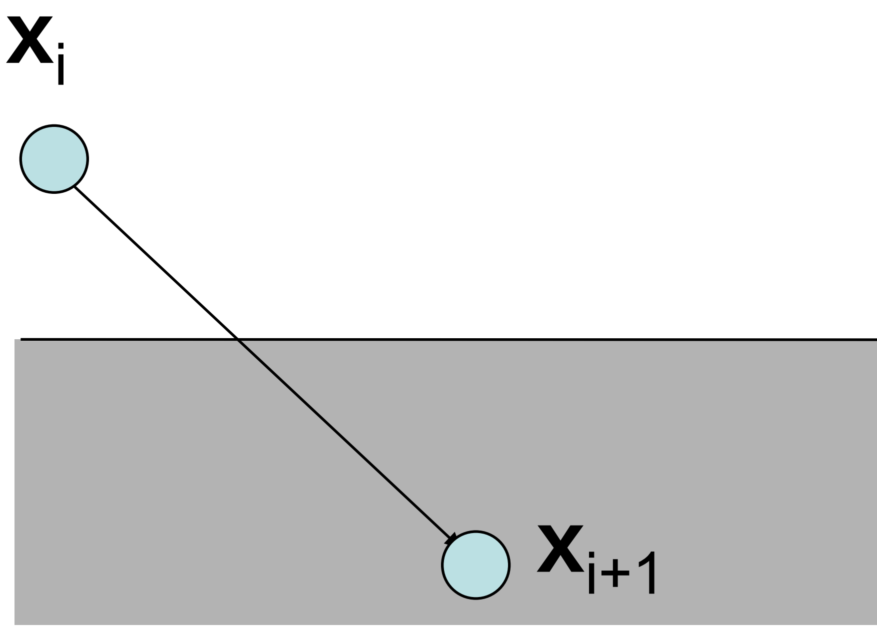
$$v \leftarrow v_t - \varepsilon v_n$$

- Coefficient of restitution ε
- When $\varepsilon = 1$, mirror reflection



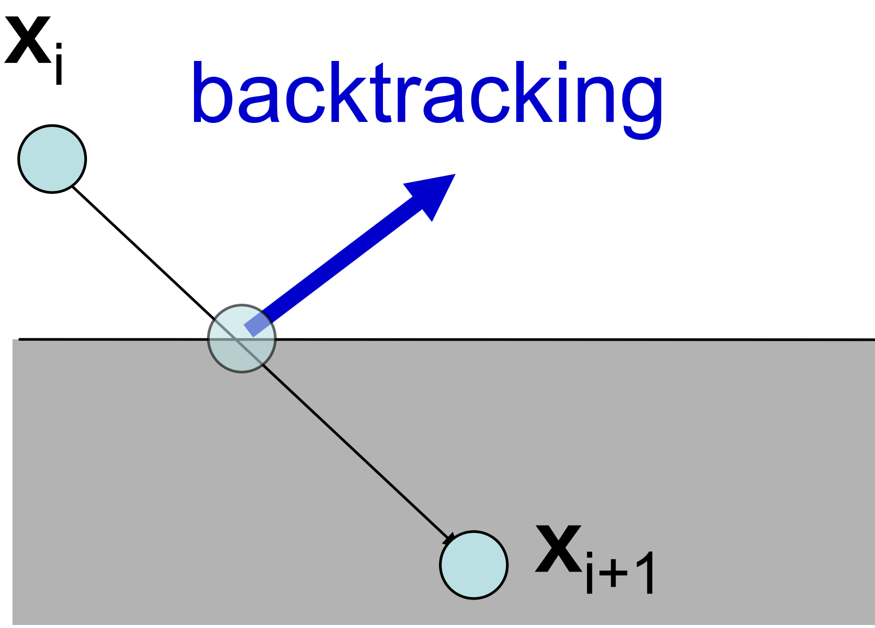
Collisions – Overshooting

- Usually, we detect collision when it is too late: we are already inside



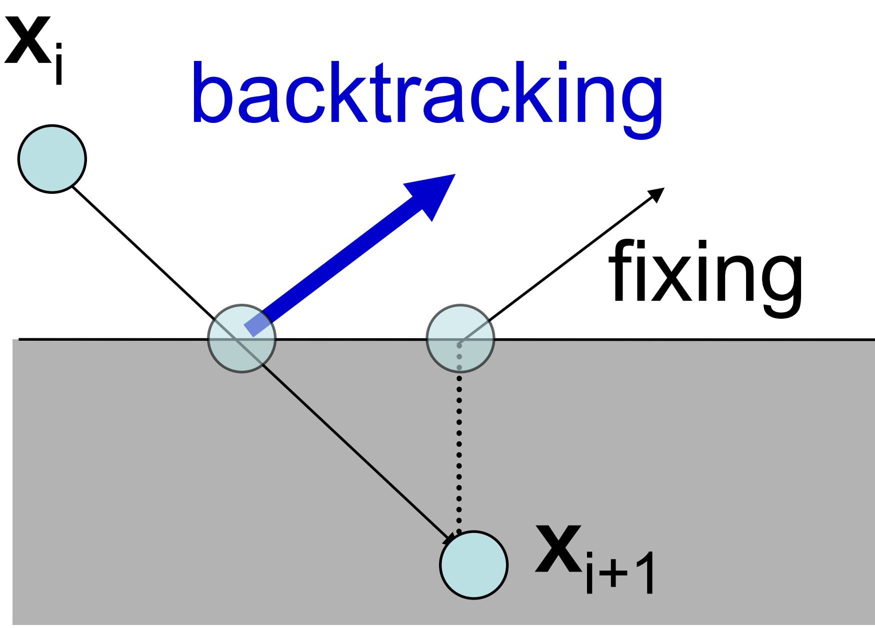
Collisions – Overshooting

- Usually, we detect collision when it is too late: we are already inside
- Solution: Back up
    - Compute intersection point
    - Ray-object intersection!
    - Compute response there
    - Advance for remaining fractional time step



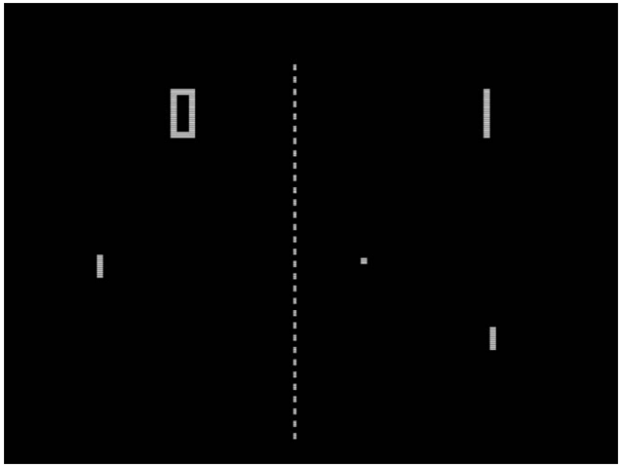
Collisions – Overshooting

- Usually, we detect collision when it is too late: we are already inside
- Solution: Back up
    - Compute intersection point
    - Ray-object intersection!
    - Compute response there
    - Advance for remaining fractional time step
- Other solution: Quick and dirty hack
    - Just project back to object closest point



Questions?

- Pong: ε =?
- http://www.youtube.com/watch?v=sWY0Q_lMFfw
- http://www.xnet.se/javaTest/jPong/jPong.html



Animation removed due to copyright restrictions.

Collision Detection in Big Scenes

- Imagine we have *n* objects. Can we test all pairwise intersections?
  - Quadratic cost $O(n^2)$!

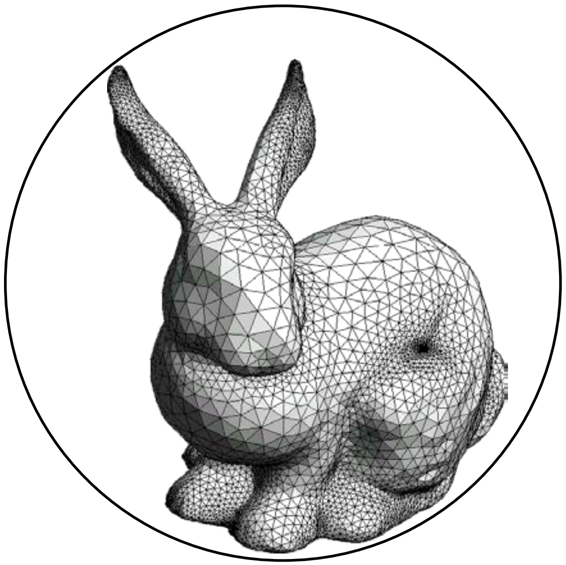
- Simple optimization: separate static objects
  - But still $O(\text{static} \times \text{dynamic} + \text{dynamic}^2)$

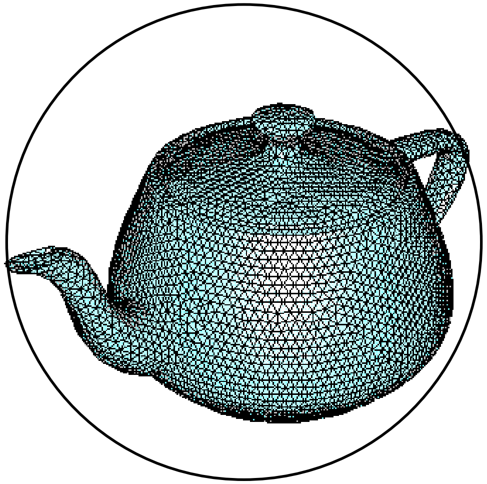
Hierarchical Collision Detection

- Use simpler conservative proxies (e.g. bounding spheres)
- Recursive (hierarchical) test
  - Spend time only for parts of the scene that are close
- Many different versions, we will cover only one

Bounding Spheres

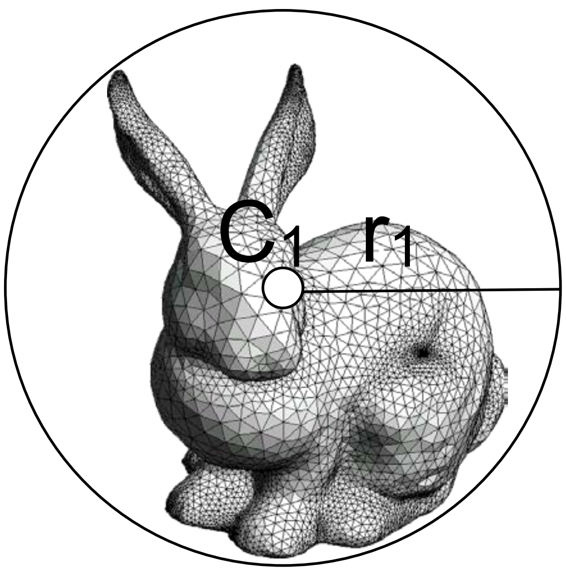
- Place spheres around objects
- If spheres do not intersect, neither do the objects!
- Sphere-sphere collision test is easy.

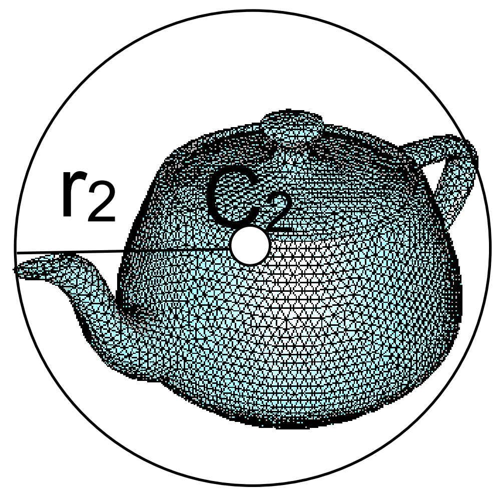




Sphere-Sphere Collision Test

- Two spheres, centers C_1 and C_2, radii r_1 and r_2
- Intersect only if $||C_1C_2||<r_1+r_2$

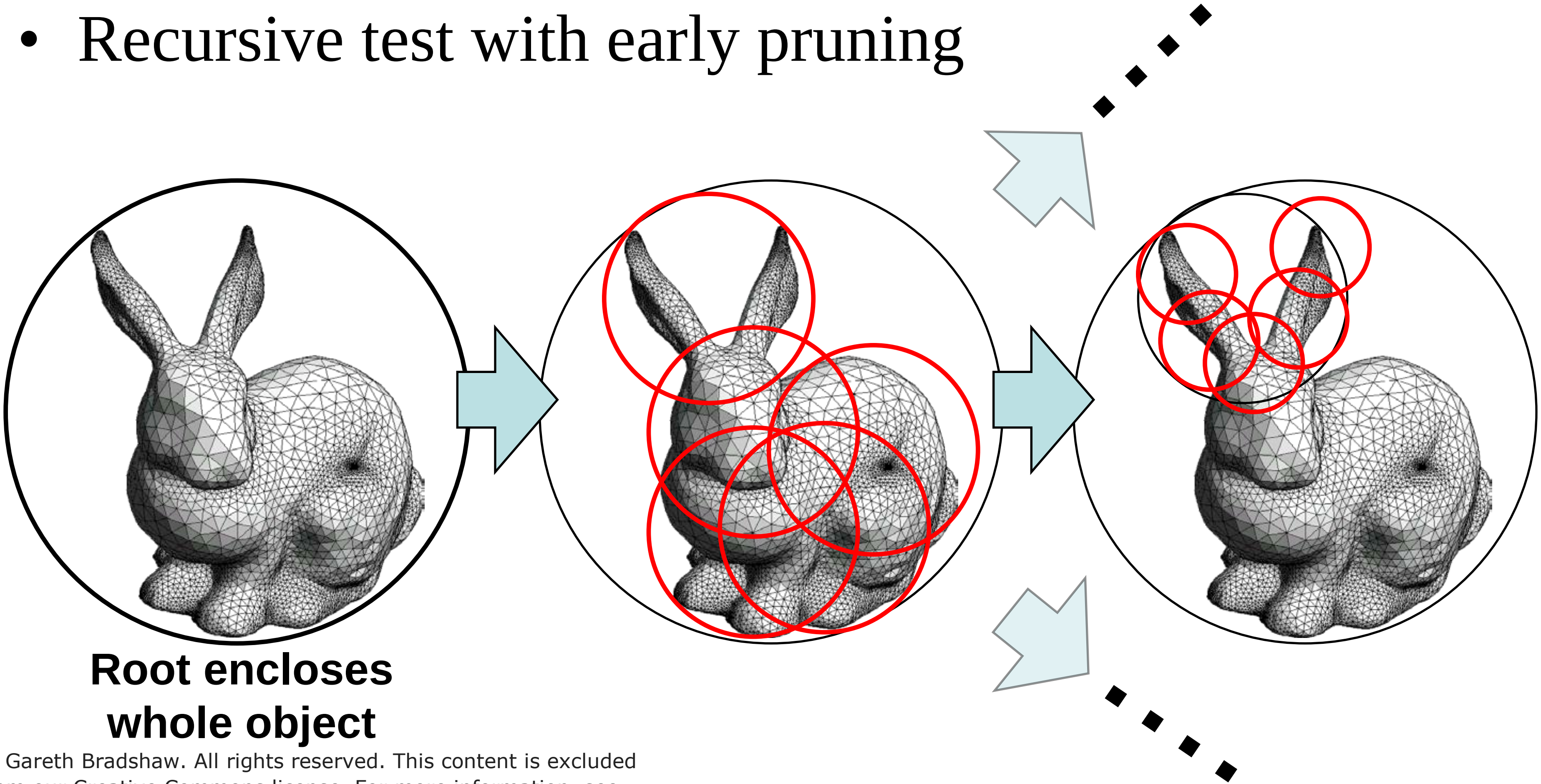




Courtesy of Patrick Laug. Used with permission.

Hierarchical Collision Test

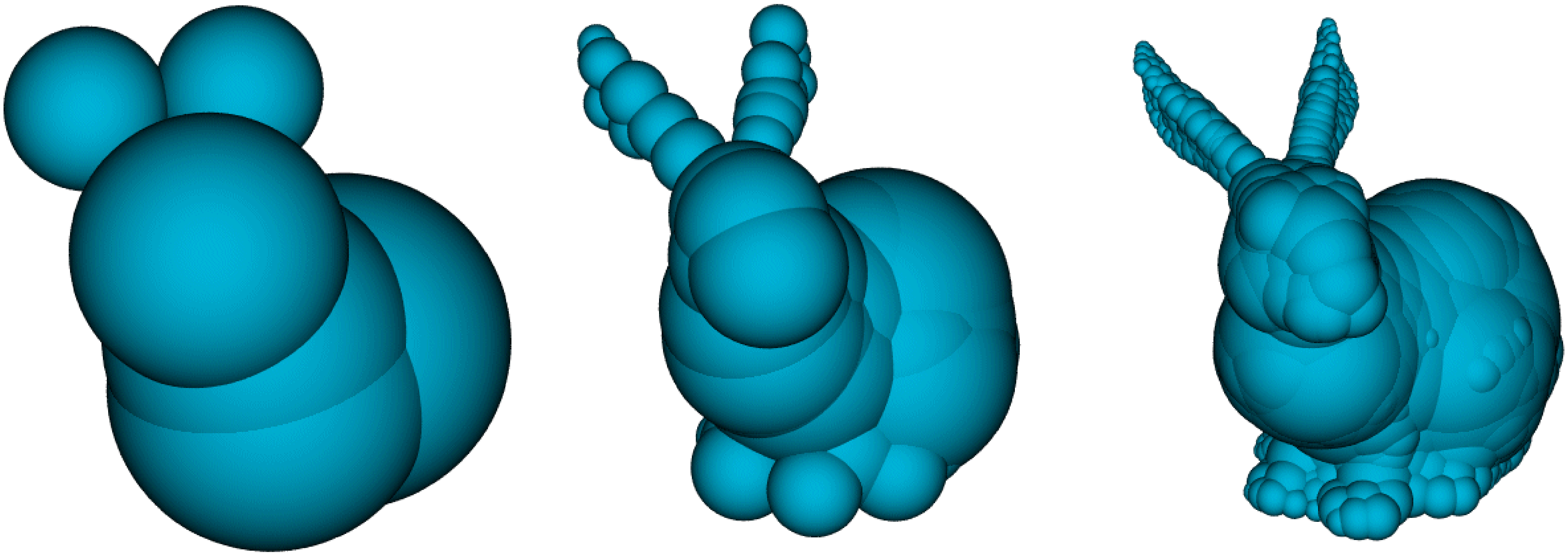
- Hierarchy of bounding spheres
  - Organized in a tree
- Recursive test with early pruning



Root encloses whole object

Examples of Hierarchy

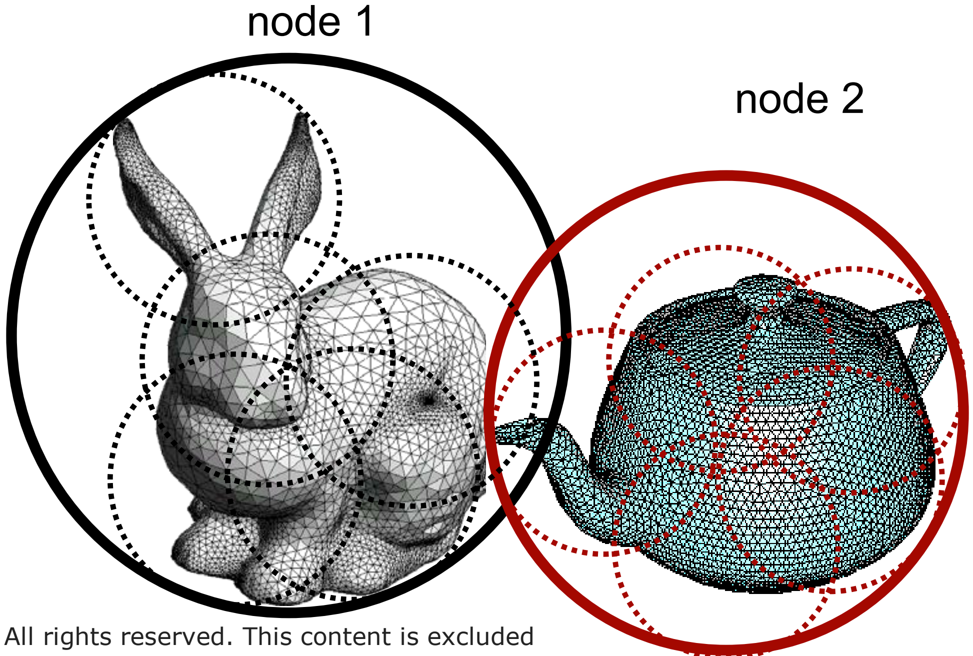
- http://isg.cs.tcd.ie/spheretree/



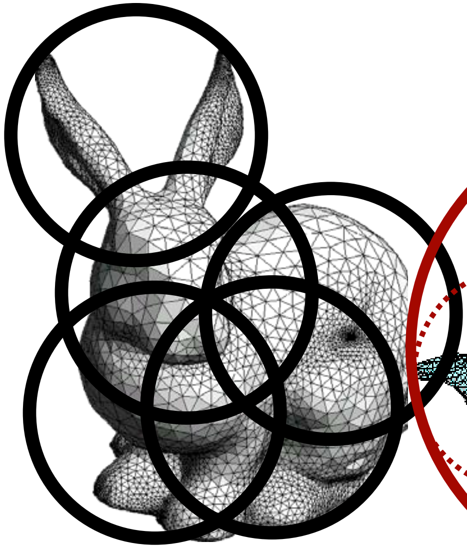
Pseudocode (simplistic version)

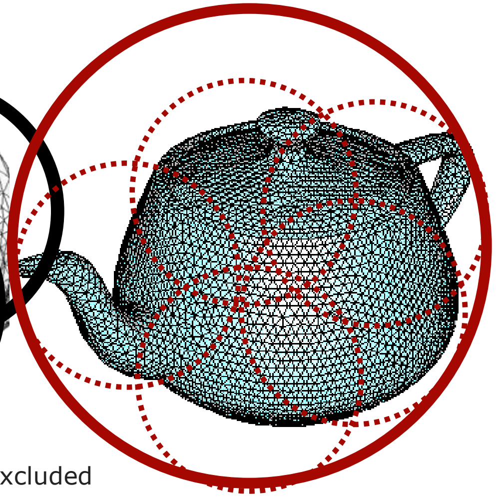
```
boolean intersect(node1, node2)
   // no overlap? ==> no intersection!
   if (!overlap(node1->sphere, node2->sphere)
      return false

   // recurse down the larger of the two nodes
   if (node1->radius()>node2->radius())
      for each child c of node1
         if intersect(c, node2) return true
   else
      for each child c f node2
        if intersect(c, node1) return true

   // no intersection in the subtrees? ==> no intersection!
   return false
```

```
boolean intersect(node1, node2)
   if (!overlap(node1->sphere, node2->sphere)
      return false
   if (node1->radius()>node2->radius())
      for each child c of node1
         if intersect(c, node2) return true
   else
      for each child c f node2
        if intersect(c, node1) return true
   return false
```



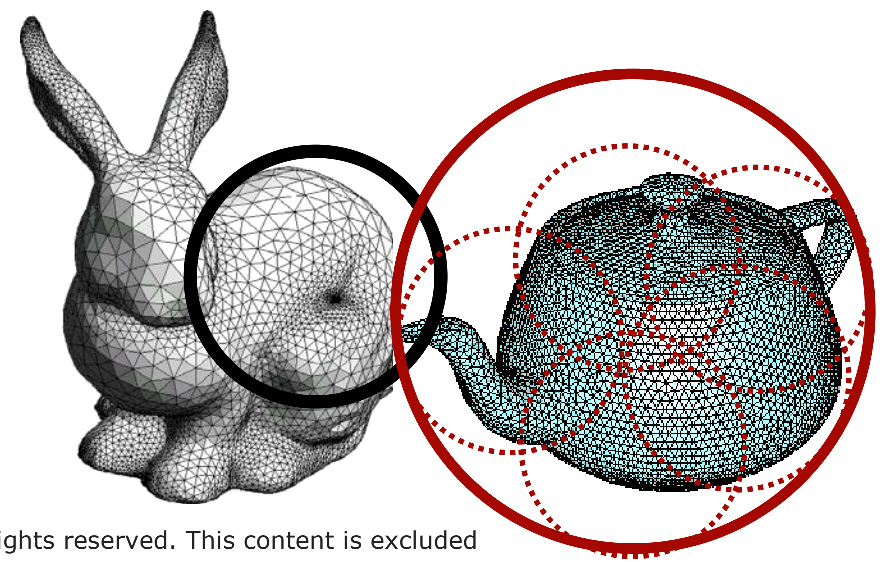
```
boolean intersect(node1, node2)
   if (!overlap(node1->sphere, node2->sphere)
      return false
   if (node1->radius()>node2->radius())
      for each child c of node1
         if intersect(c, node2) return true
   else
      for each child c f node2
        if intersect(c, node1) return true
   return false
```



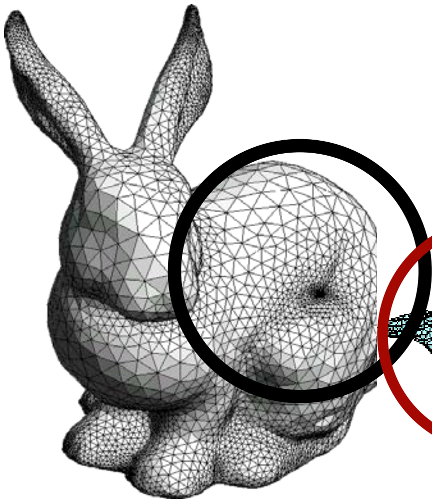


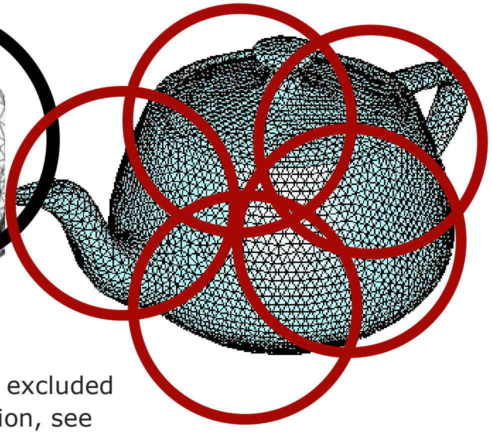
Courtesy of Patrick Laug. Used with permission.

```
boolean intersect(node1, node2)
   if (!overlap(node1->sphere, node2->sphere)
      return false
   if (node1->radius()>node2->radius())
      for each child c of node1
         if intersect(c, node2) return true
   else
      for each child c f node2
        if intersect(c, node1) return true
   return false
```

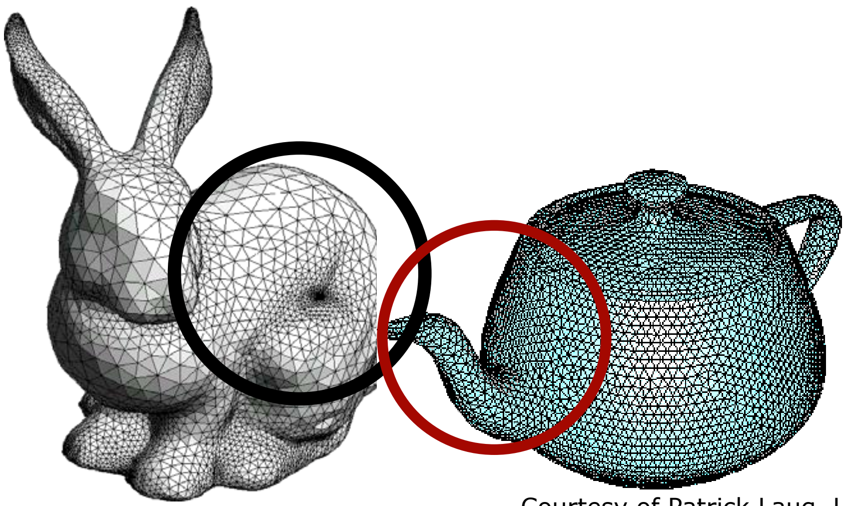


```
boolean intersect(node1, node2)
   if (!overlap(node1->sphere, node2->sphere)
      return false
   if (node1->radius()>node2->radius())
      for each child c of node1
         if intersect(c, node2) return true
   else
      for each child c f node2
        if intersect(c, node1) return true
   return false
```



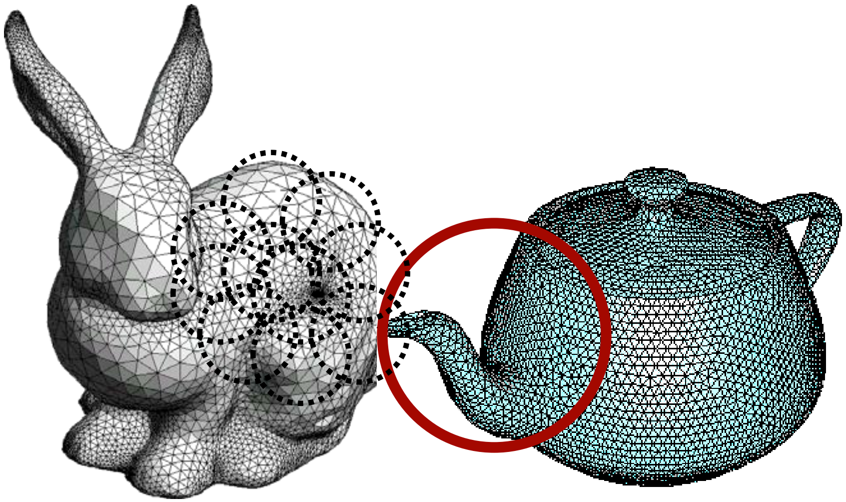


```
boolean intersect(node1, node2)
   if (!overlap(node1->sphere, node2->sphere)
      return false
   if (node1->radius()>node2->radius())
      for each child c of node1
         if intersect(c, node2) return true
   else
      for each child c f node2
        if intersect(c, node1) return true
   return false
```



Courtesy of Patrick Laug. Used with permission.

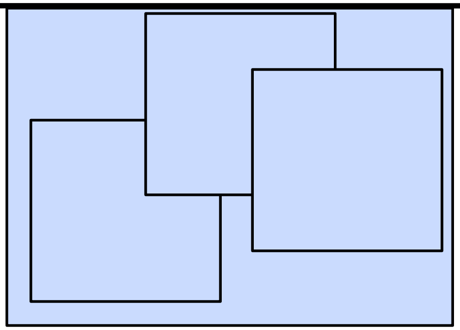
```
boolean intersect(node1, node2)
   if (!overlap(node1->sphere, node2->sphere)
      return false
   if (node1->radius()>node2->radius())
      for each child c of node1
         if intersect(c, node2) return true
   else
      for each child c f node2
        if intersect(c, node1) return true
   return false
```



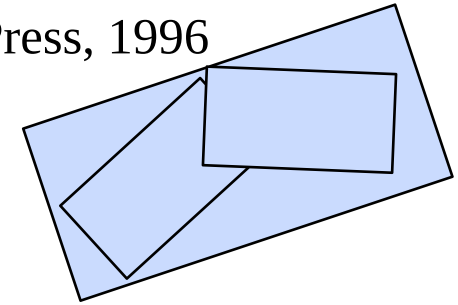
Pseudocode (with leaf case)

```
boolean intersect(node1, node2)
   if (!overlap(node1->sphere, node2->sphere)
      return false

   // if there is nowhere to go, test everything
   if (node1->isLeaf() && node2->isLeaf())
      perform full test between all primitives within nodes

   // otherwise go down the tree in the non-leaf path
   if ( !node2->isLeaf() && !node1->isLeaf() )
      // pick the larger node to subdivide, then recurse
   else
      // recurse down the node that is not a leaf

   return false
```

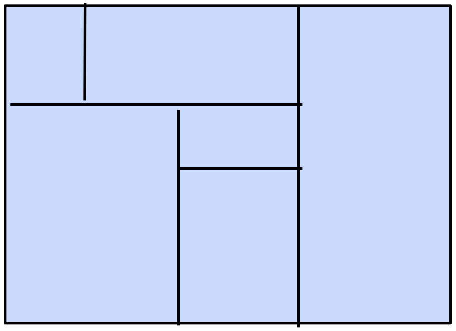
Other Options



- Axis Aligned Bounding Boxes
  - "R-Trees"
- Oriented bounding boxes
  - S. Gottschalk, M. Lin, and D. Manocha. "OBBTree: A hierarchical Structure for rapid interference detection," Proc. Siggraph 96. ACM Press, 1996



- Binary space partitioning trees; kd-trees



Questions?

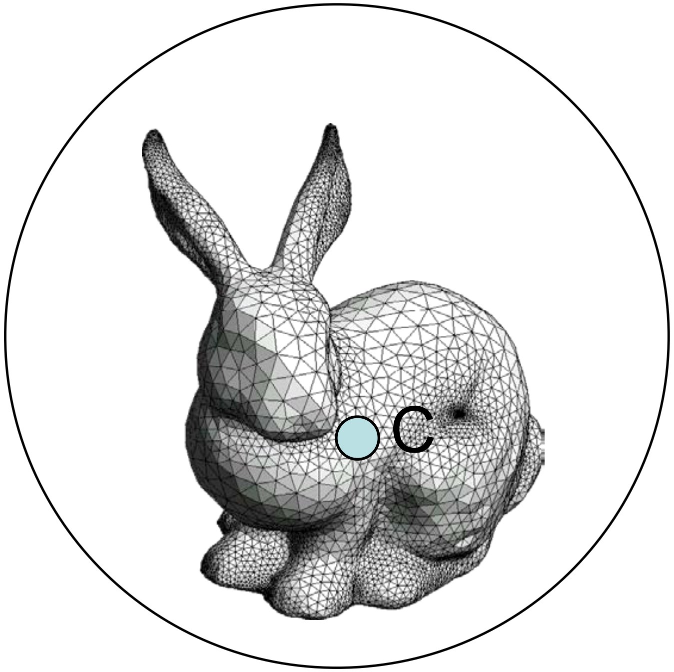
- http://www.youtube.com/watch?v=b_cGXtc-nMg
- http://www.youtube.com/watch?v=nFd9BIcpHX4&feature=related
- http://www.youtube.com/watch?v=2SXixK7yCGU

Hierarchy Construction

- Top down
  - Divide and conquer
- Bottom up
  - Cluster nearby objects
- Incremental
  - Add objects one by one, binary-tree style.

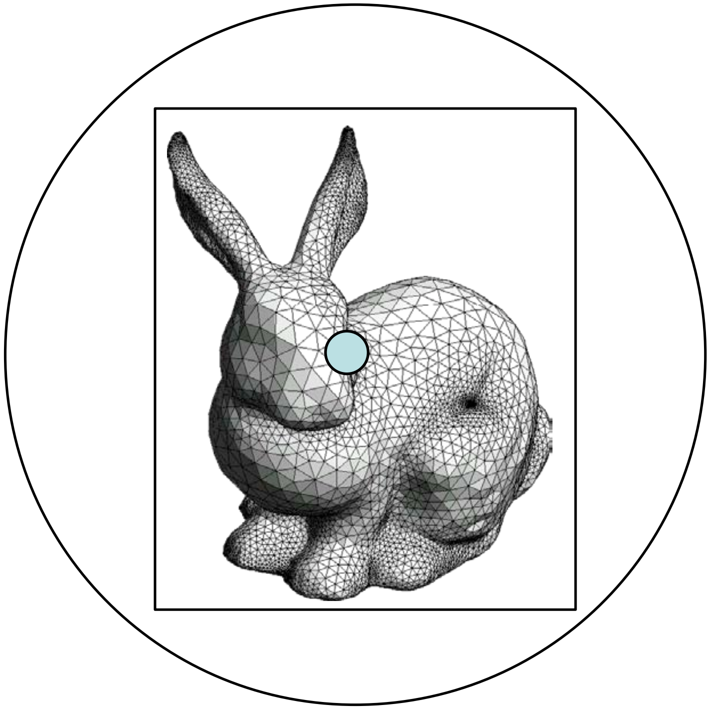
Bounding Sphere of a Set of Points

- Trivial given center C
  - radius = $\max_i \|C-P_i\|$



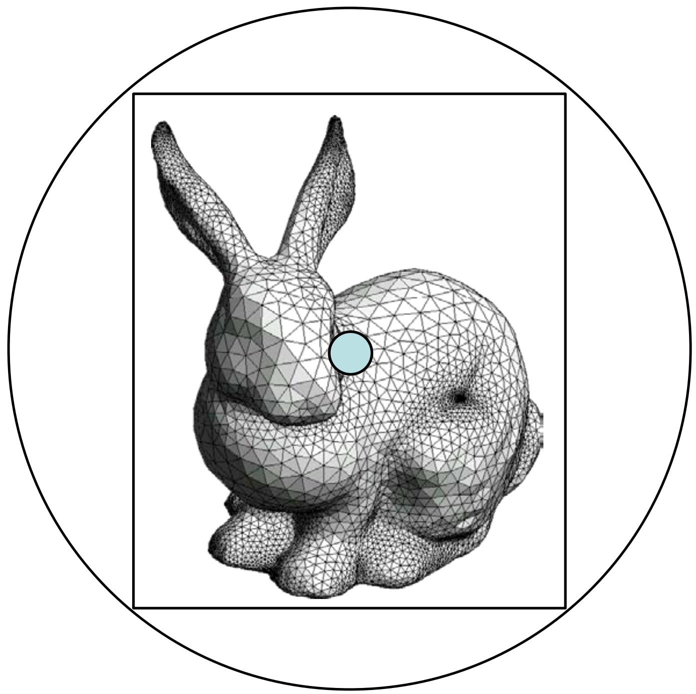
Bounding Sphere of a Set of Points

- Using axis-aligned bounding box
  - *center*= $((x_{min}+x_{max})/2, (y_{min}+y_{max})/2, (z_{min}, z_{max})/2)$
  - Better than the average of the vertices because does not suffer from non-uniform tessellation



Bounding Sphere of a Set of Points

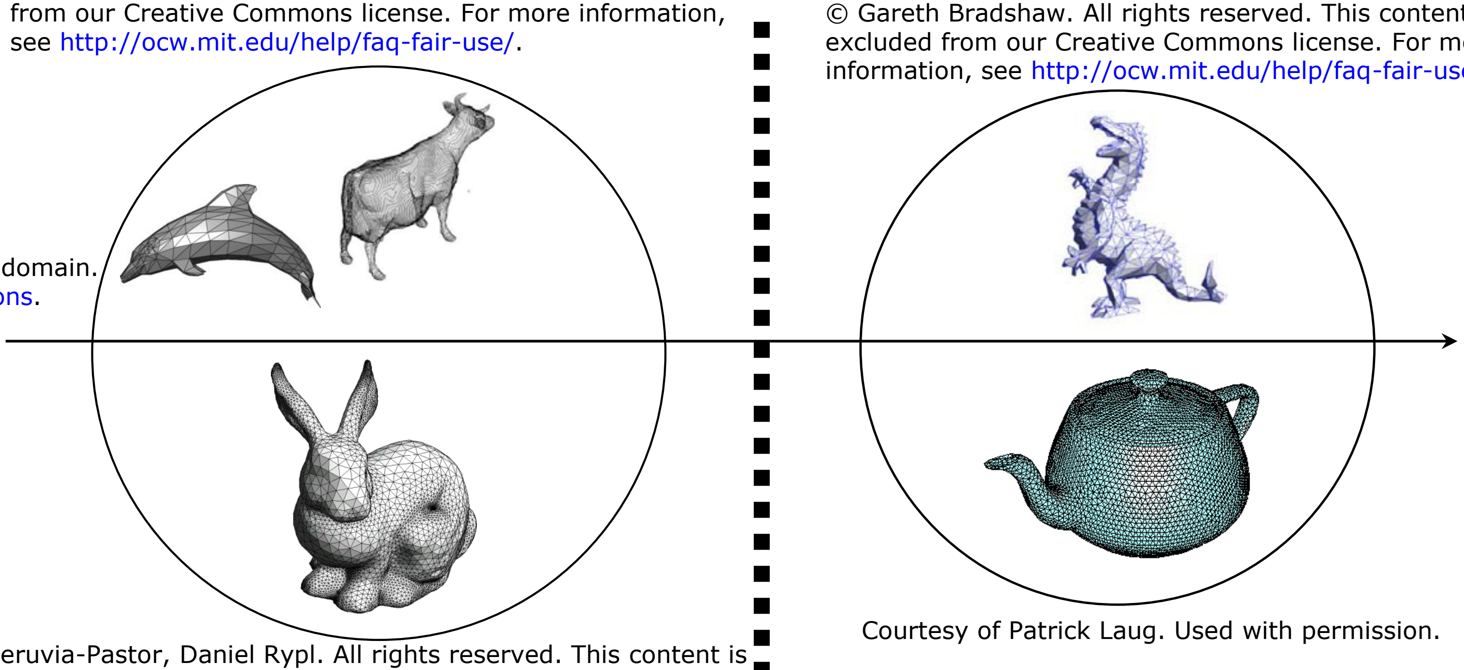
- Using axis-aligned bounding box
  - *center=* $((x_{min}+x_{max})/2,\ (y_{min}+y_{max})/2,\ (z_{min},\ z_{max})/2)$
  - Better than the average of the vertices because does not suffer from non-uniform tessellation



Questions?

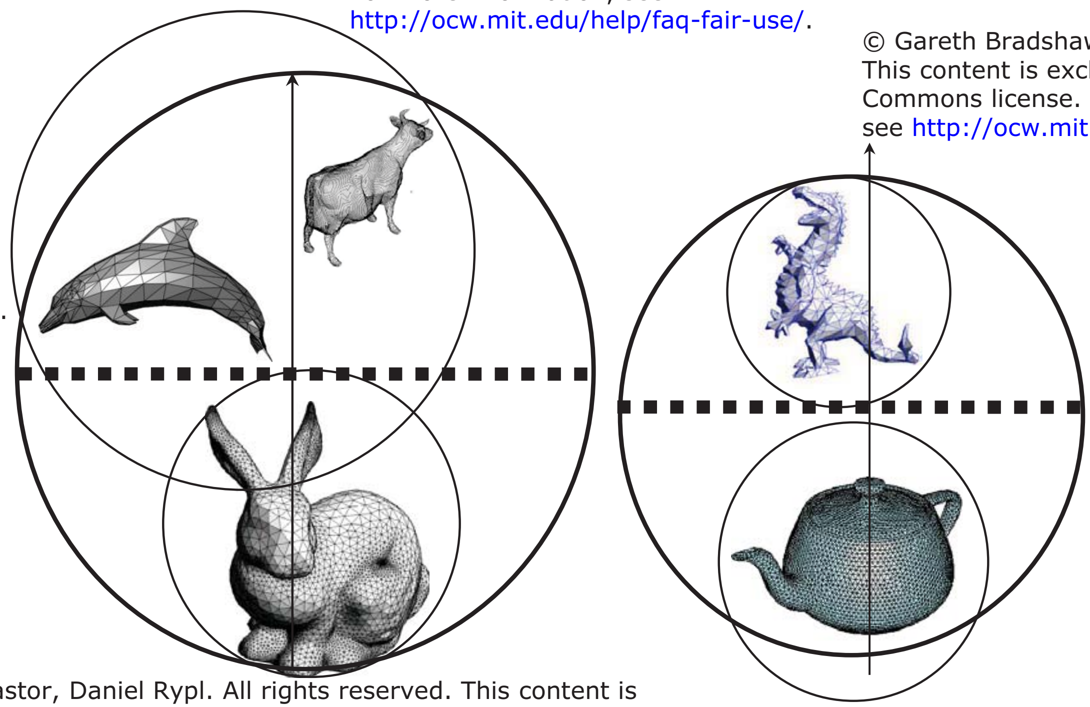
Top-Down Construction

- Take longest scene dimension
- Cut in two in the middle
  - assign each object or triangle to one side
  - build sphere around it



Top-Down Construction - Recurse

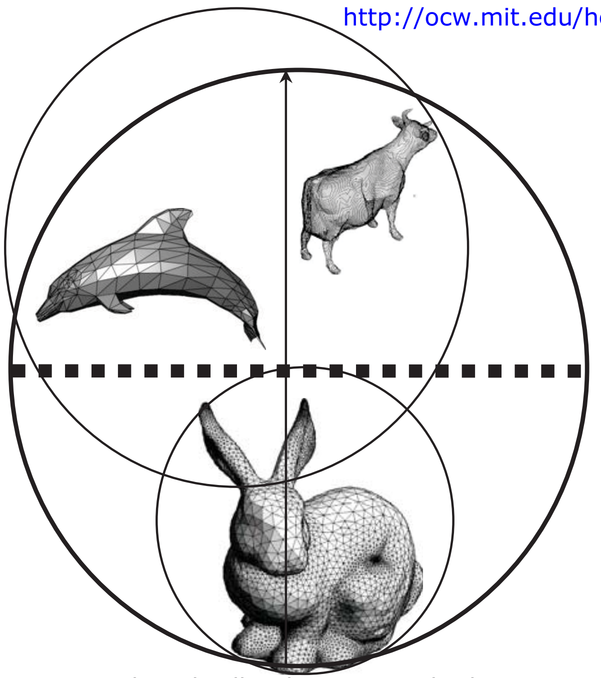
- Take longest scene dimension
- Cut in two in the middle
  - assign each object or triangle to one side
  - build sphere/box around it

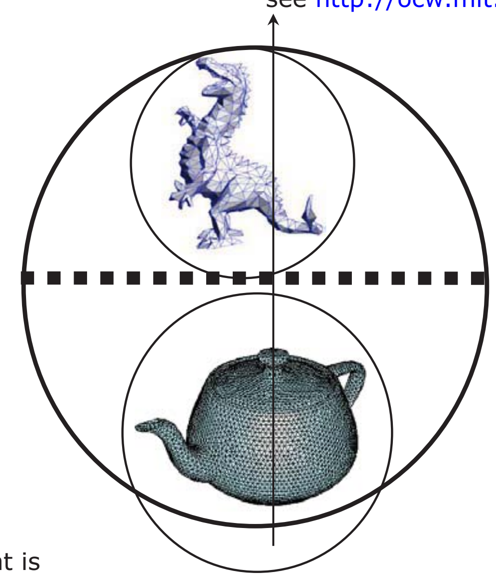


Top-Down Construction - Recurse

- Take longest scene dimension
- Cut in two in the middle
  - assign each object or triangle to one side
  - build sphere/box around it

Questions?



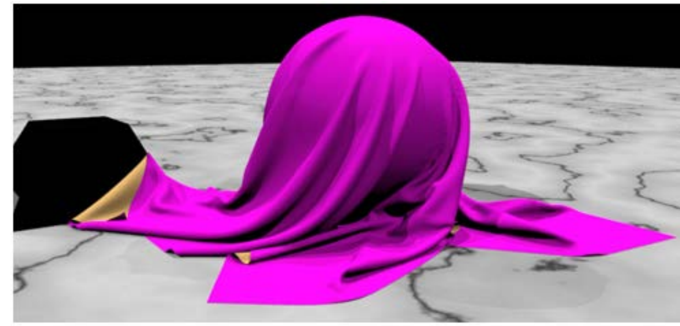


Reference

Image of the cover of the book, "Real Time Collision Detection," by Christer Ericson has been removed due to copyright restrictions.

The Cloth Collision Problem

- A cloth has many points of contact
- Stays in contact
- Requires
  - Efficient collision detection
  - Efficient numerical treatment (stability)



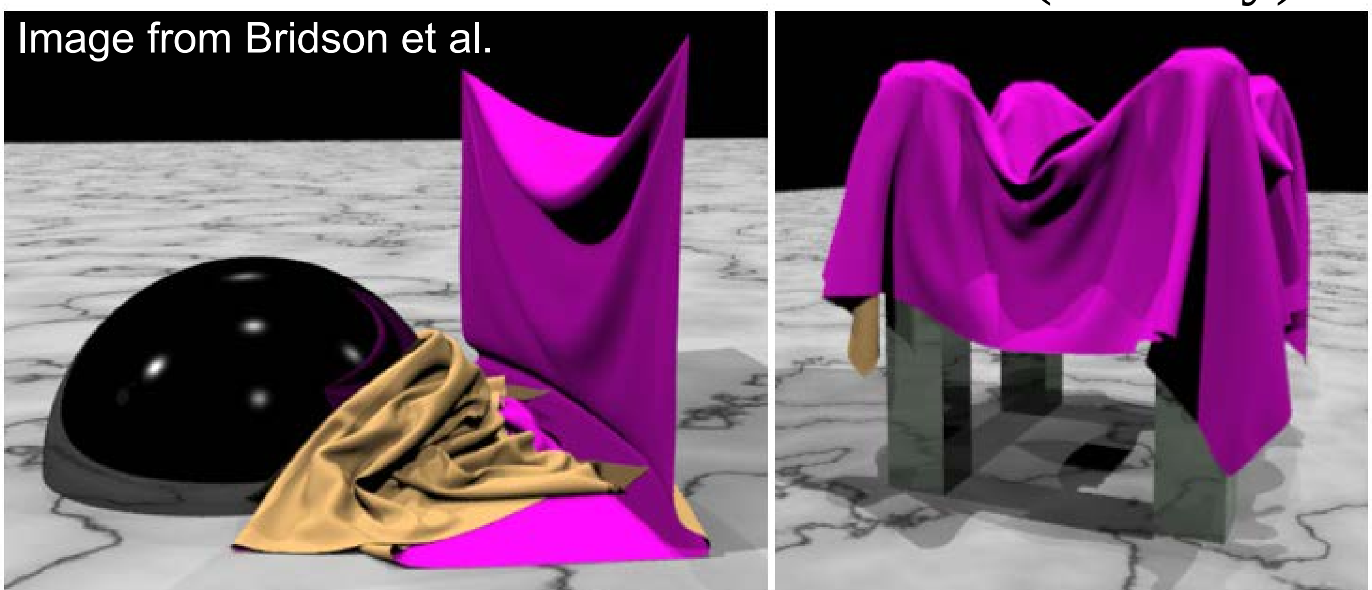


Image from Bridson et al.

Robust Treatment of Simultaneous Collisions

David Harmon, Etienne Vouga, Rasmus Tamstorf, Eitan Grinspun

6.837 Computer Graphics
Fall 2012