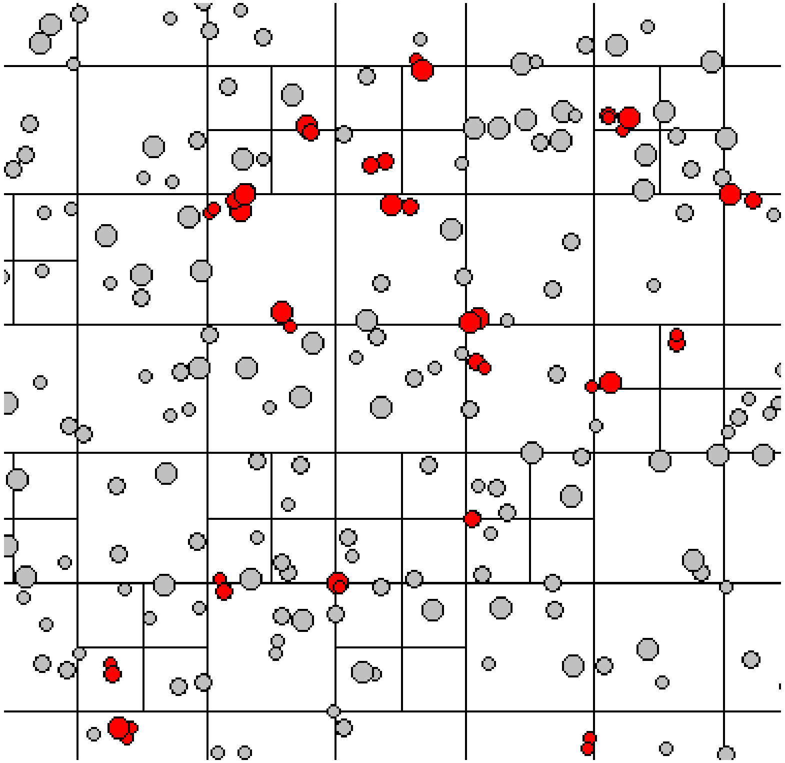


Quadtrees and Applications

Alexandre Chapiro and Tássio Knop

Introduction

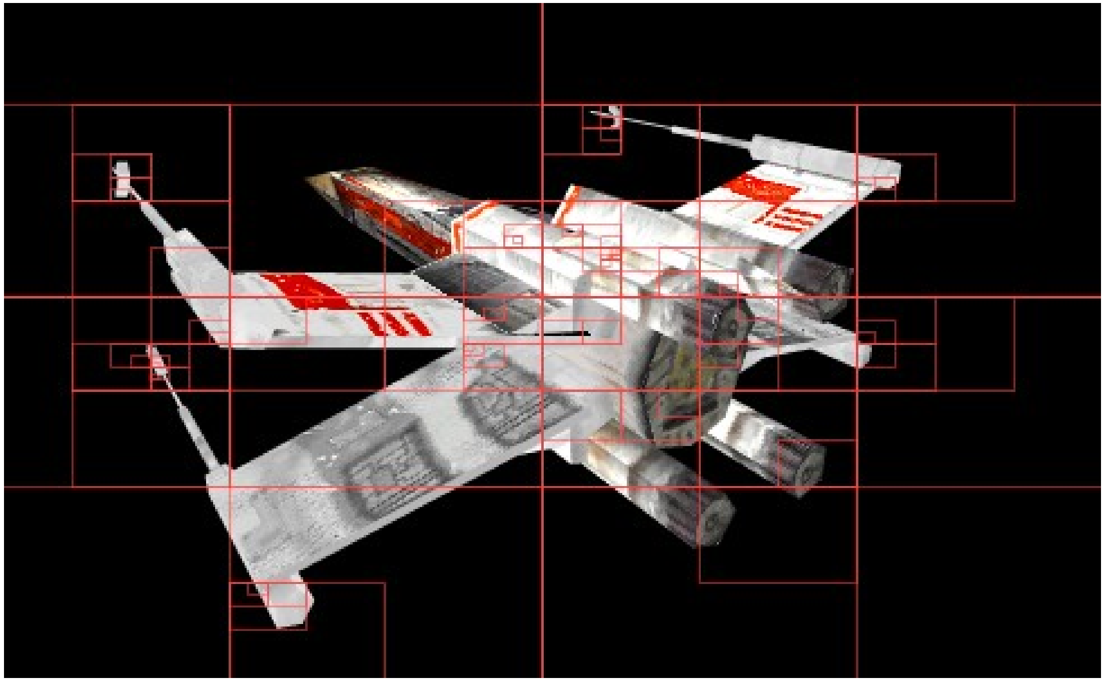
- “Quadtrees” are an important method in Computational Geometry.

- They are used in several applications, such as games, physically based simulations, etc.

- Can be used to generate “smart” meshes.

Introduction

- Quadtrees used to assist backface culling.

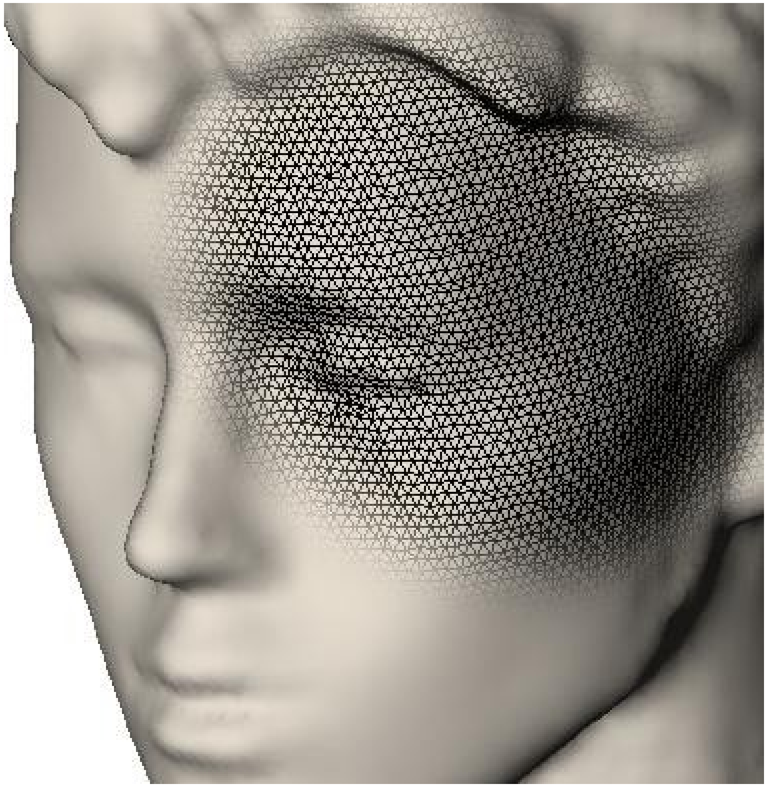


Meshes

- Sometimes, a regular mesh is not good enough.

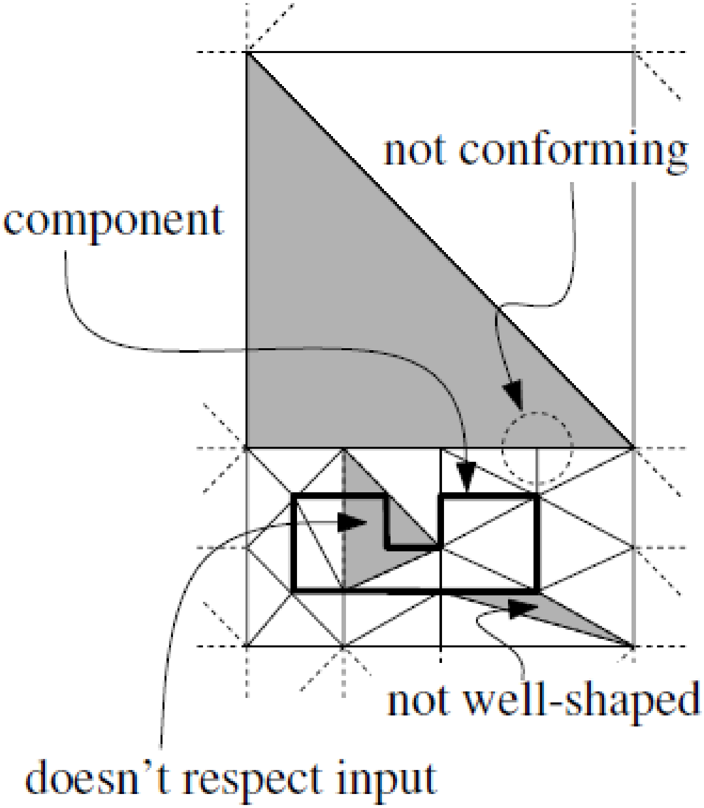
  - A compromise between mesh size and computational speed is needed.

- Quadtrees can be used to generate non-uniform meshes.



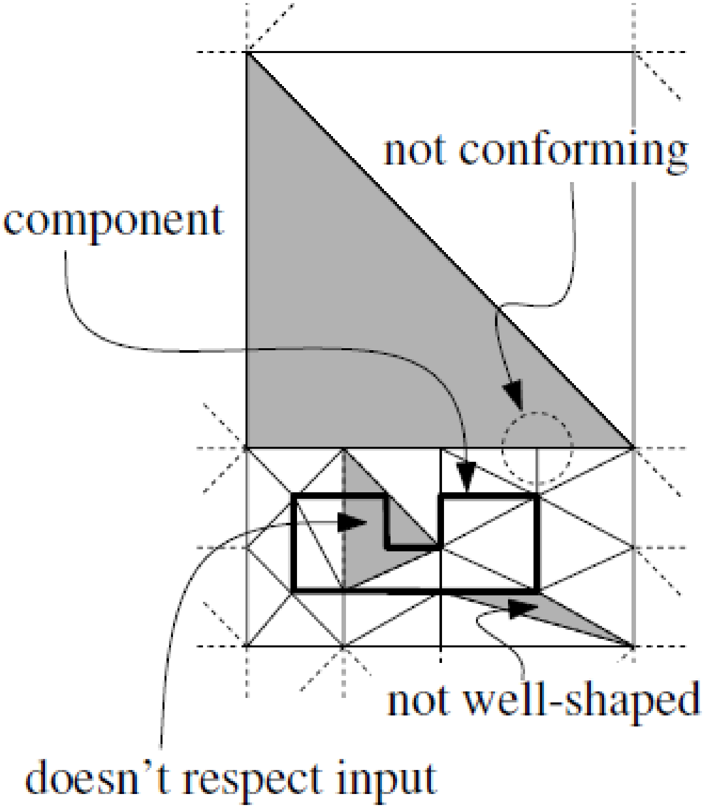
Meshes

- Some proprieties are important for a “good” mesh:
- Conformity: a triangle is not allowed to have a vertex of another triangle in the interior of one of its edges.
- Respect the input: the edges of the components must be contained in the union of the edges of the mesh triangles



Meshes

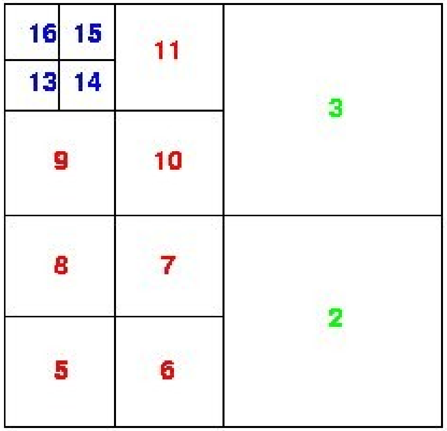
- Some proprieties are important for a “good” mesh:
- The mesh triangles must be well-shaped: the angles of any mesh triangle should not be too large nor too small. In particular, we require them to be in the range from 45◦ to 90◦ .
- The mesh must be non-uniform: it should be fine near the edges of the components and coarse far away from the edges.

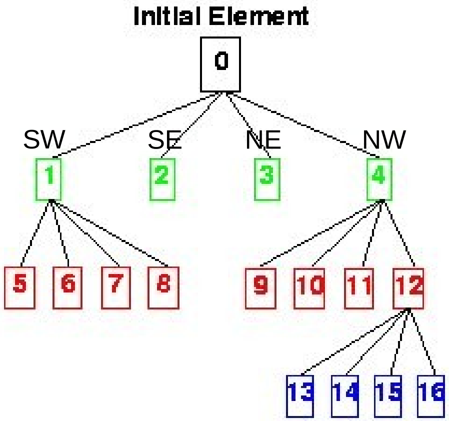


Quadtrees

- A Quadtree is a kind of tree where every non-leaf node has four children.

- Can be used to represent a spacial separation:



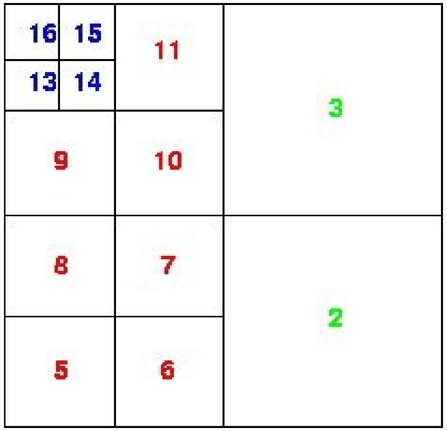


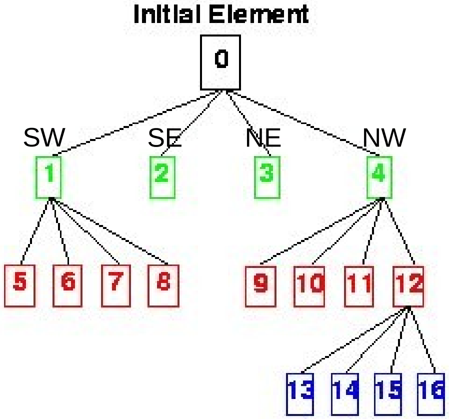
Quadtrees

- The Quadtree is built recursively.

  - Begins splitting square into four parts.

  - Recursively builds another Quadtree in each square that contains more than 'n' elements.





Complexity

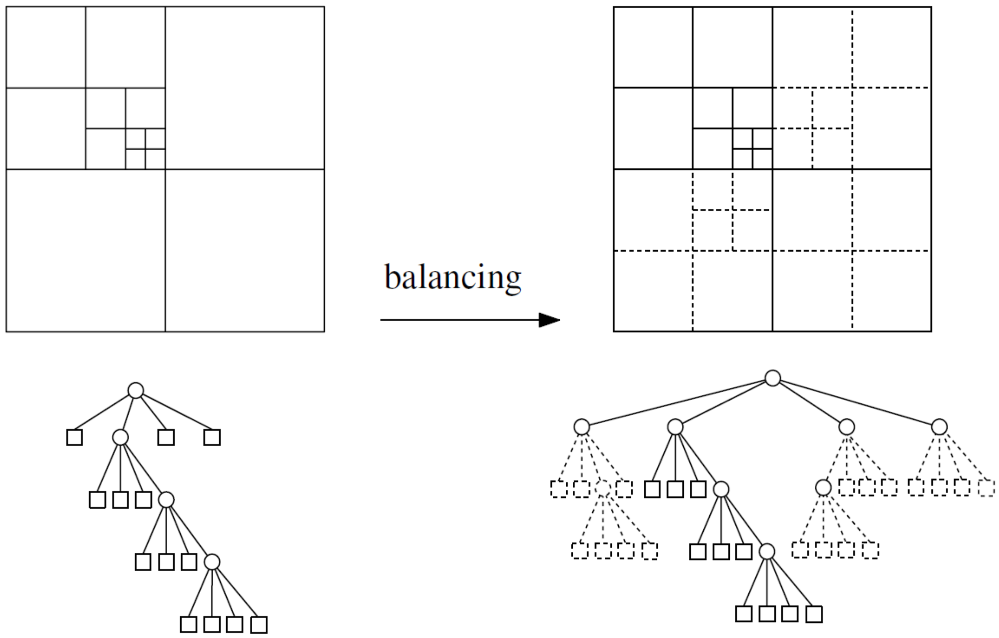
- Theorem 1: A Quadtree of depth 'd' storing a set of 'n' points has O((d + 1)n) nodes and can be constructed in O((d + 1)n) time.

- Theorem 2: Let T be a Quadtree of depth 'd'. The neighbour of a given node 'ν' in T in a given direction can be found in O(d + 1) time.

Balanced Quadtree

- A Quadtree can be very unbalanced, which makes one big square have many small adjacent squares.

Variant: Balanced Quadtree.

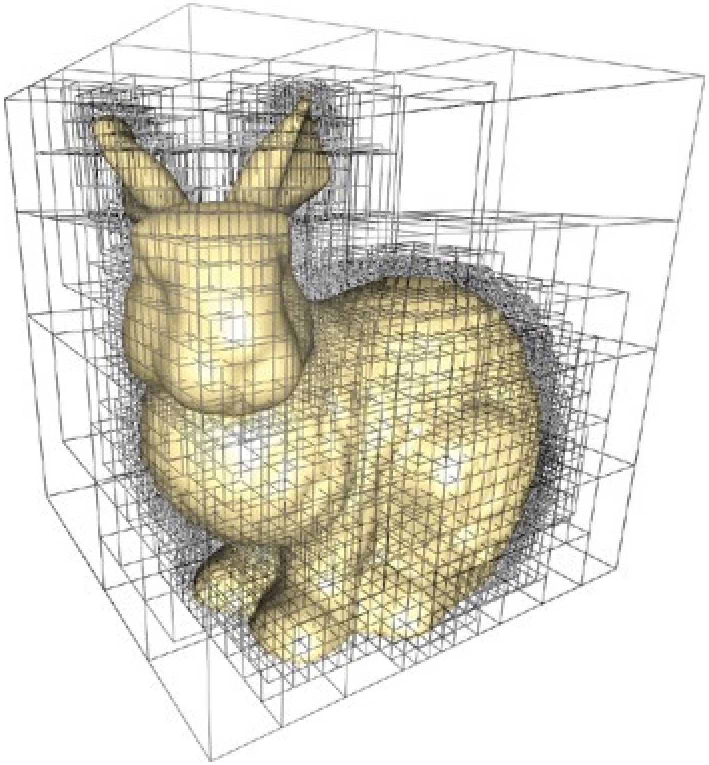


Balanced Quadtree

- Theorem 3: Let T be a Quadtree with 'm' nodes. Then the balanced version of T has O(m) nodes and it can be created in O((d + 1)m) time.

Extensions

- Quadtrees can easily be generalized to greater dimensions. (3D → Octrees)



Application – Collision Detection

- In physical simulations, video games and computational geometry, collision detection involves algorithms for checking for collision, i.e. intersection, of two given solids.

Application – Collision Detection

- Collision detection is widely used in games. Without it, for instance, characters would go through walls. Pool simulations clearly require a good collision detector.

Player 2 breaks.

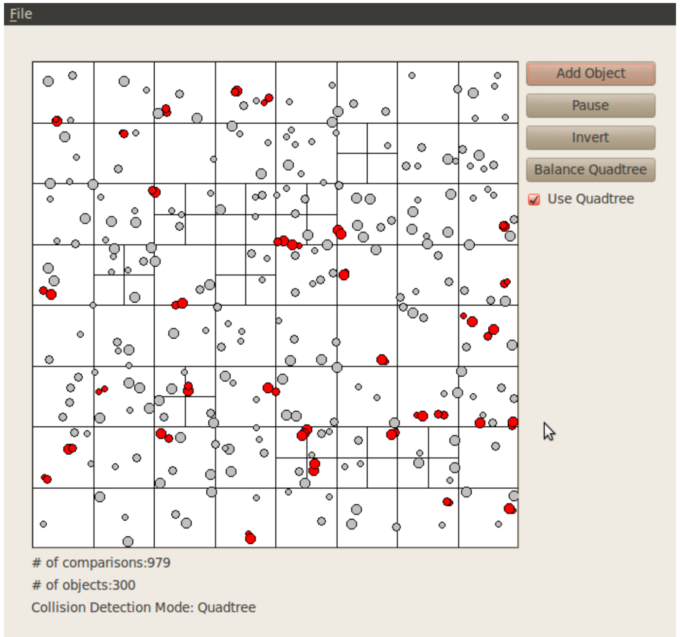
Quadtrees in Collision Detection

- A Brute Force algorithm for Collision Detection would take O(n^2) time. (Too slow!)

- Creating a Quadtree allows us to analyse only adjacent squares, reducing the number of comparisons greatly.

Our Application

- A Quadtree is created. Each square has no more than 10 elements.
- Quadtree balance was implemented.

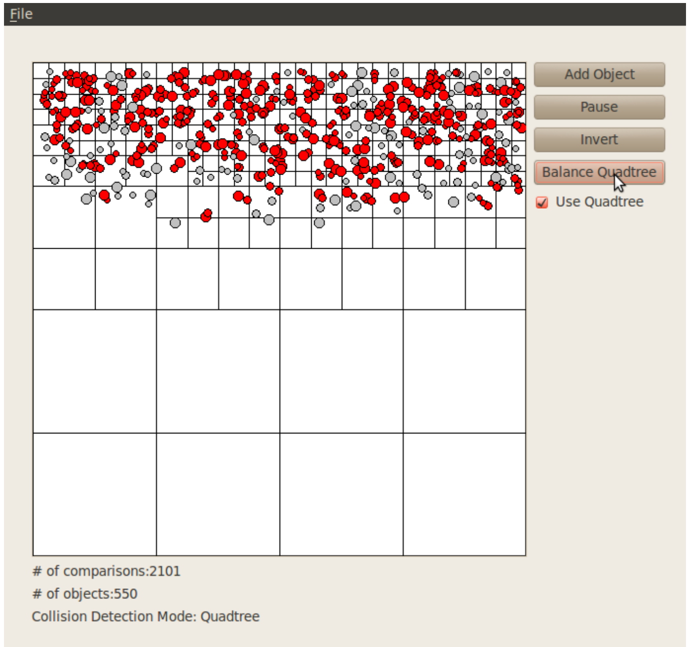


☑ Use Quadtree

# of comparisons:979

# of objects:300

Collision Detection Mode: Quadtree



☑ Use Quadtree

# of comparisons:2101

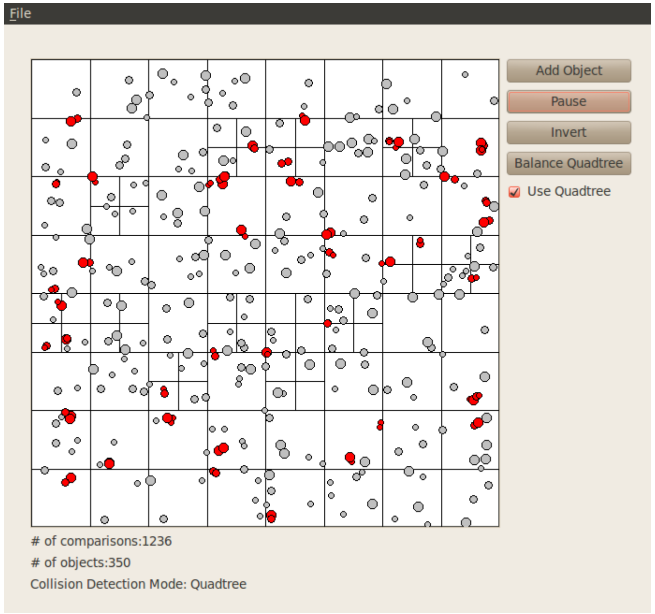
# of objects:550

Collision Detection Mode: Quadtree

Our Application

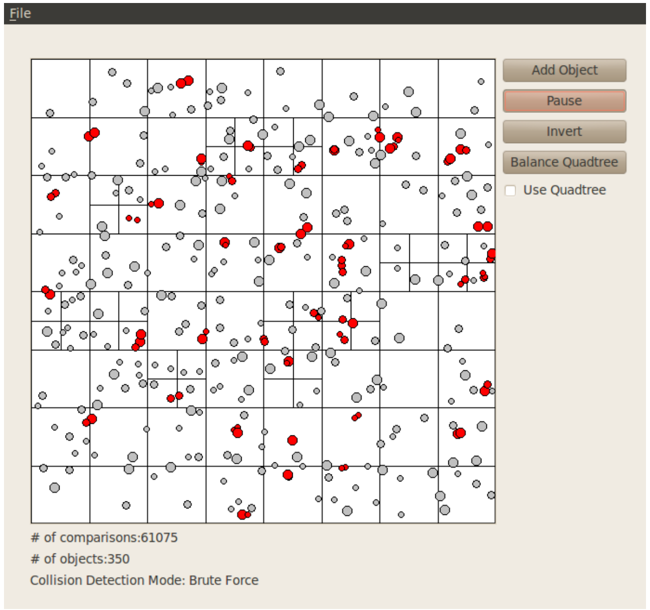
- Noticeable Improvement over Brute Force.

  → About x50.



☑ Use Quadtree

# of comparisons:1236
# of objects:350
Collision Detection Mode: Quadtree



□ Use Quadtree

# of comparisons:61075
# of objects:350
Collision Detection Mode: Brute Force

References

→ M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, Springer-Verlag, 1997.

→ http://en.wikipedia.org/wiki/Quadtree

→ http://en.wikipedia.org/wiki/Collision_detection