Quadtree vs Red-Black tree for a game in C++?

Asked 16 years ago Modified 8 years, 5 months ago Viewed 11k times

8

I have been looking for a quadtree/quadtree node implementation on the net for ages. There is some basic stuff but nothing that I would be able to really use it a game.

My purpose is to store objects in a game for processing things such as collision detection. I am not 100% certain that a quadtree is the best data structure to use, but from what I have read it is. I have already coded a Red-Black tree, but I don't really know if the performance would be good enough for my game (which will be an adventure 3rd person game like Ankh).

How would I write a basic but complete quadtree class (or octree) in C++? How would you use the quad tree for collisions?

c++ quadtree

Share
Improve this question
Follow

edited Dec 16, 2008 at 14:45

asked Dec 16, 2008 at 14:27

Brock Woolf
47.3k ● 51 ● 124 ● 145

This is a bit vague. I think the tree implementation needs to know what kind of data you want to store, since that affects the details on how insert works (splitting, etc). – unwind Dec 16, 2008 at 14:29

6 Answers

Sorted by: Highest score (default)

18

Quadtrees are used when you only need to store things that are effectively on a plane. Like units in a classic RTS where they are all on the ground or just a little bit above it. Essentially each node has links to 4 children that divide the node's space up into evenly distributed quarters.

Octrees do the same but in all three dimensions rather than just two, and thus they have 8 child nodes and partition the space up into eights. They should be used when the game entities are distributed more evenly among all three dimensions.

If you are looking for a binary tree - like a red-black tree - then you want to use a data structure called a binary space partitioning tree (BSP tree) or a version of it called the KD Tree. These partition space into halves using a plane, in the KD tree the planes are orthogonal (on the XZ, XY, ZY axes) so sometimes it works better in a 3D scene. BSP trees divide the scene up using planes in any orientation, but they can be quite useful, and they were used as far back as Doom.

Now because you've partitioned the game space you now don't have to test every game entity against every other game entity to see if they collide, which is an O(n^2) algorithm at best. Instead you query the data structure to return the game entities within a sub-region of the game space, and only perform collision detection for those nodes against each other.

This means that collision detection for all game entities should be n O(nlogn) operation (at worst).

A couple of extra things to watch out for:

- Make sure you test game entities from adjacent nodes, not just the ones in the current node, since they could still collide.
- Rebalance the data structure after the entities have moved since you may have empty nodes in the data structure now, or ones that contain too many entities for good performance (also the degenerate case of all entities being in the same node).

Share Improve this answer

Follow

edited Jul 20, 2016 at 19:17

Smi
14.3k ● 9 ● 60 ● 65

answered Dec 16, 2008 at 16:25

Dominik Grabiec
10.6k ● 5 ● 41 ● 46

10

A red-black tree is not a spatial index; it can only sort on a single ordinal key. A quadtree is (for two dimensions) a spatial index that allows fast lookup and elimination of points. An Octree does the same thing for three dimensions.

Share Improve this answer Follow

answered Dec 16, 2008 at 16:10

ConcernedOfTunbridge Wells
66.5k ● 15 ● 148 ● 198

5

The reason to use a quadtree is because you can then split on x- and y-coordinates, an octree on x, y and z, making collision detection trivial.

Quadtree: if an element is not in the topleft, it wont collide with one in topright, bottomleft or bottomright.

It is a very basic class, so I don't understand what you are missing in implementations you found.

I would not write such a class, I'd just borrow it from a project with a suitable license.

Share Improve this answer Follow

edited Dec 16, 2008 at 16:07

answered Dec 16, 2008 at 16:02

Stephan Eggermont
15.9k ● 1 ● 40 ● 65

exactly. the point of a quadtree is not fast traversal of the tree, but avoiding the traversal in the first place. – Jimmy Dec 16, 2008 at 16:03

3

I warmly suggest you to use a rendering engine, Ogre3D for instance. As far as I know it supports Octrees for scene management. But you can extend the Octree-based class as you wish. I used to code the stuff I needed by myself, but for complex projects, it's just not the right way.

Share Improve this answer Follow

edited Dec 16, 2008 at 17:23

answered Dec 16, 2008 at 17:11

tunnuz
23.9k ● 33 ● 95 ● 135

0

Trees in general are problematic for this in that any item inserted can lie on a boundary, and all the methods of dealing with that situation are fairly unsatisfactory.

You'll most likely want to sort your objects into moveable and static, and check anything that moved on a given frame against the static objects.

BSP Trees are the accepted solution for static geometry (boundary cases handled by splitting the object into two pieces), for dynamic try something like Sort and Sweep (also known as Sweep and Prune).

Share Improve this answer Follow

answered Dec 16, 2008 at 17:09

MontyGomery

0

Right now STANN is the best open source implementation.

http://sites.google.com/a/compgeom.com/stann/

Share Improve this answer Follow

answered Jul 5, 2010 at 20:14

Chad Brewbaker

2,569 ● 2 ● 19 ● 26