How does 3D collision / object detection work?

Asked 14 years, 11 months ago Modified 10 years, 7 months ago

Viewed 46k times

58

I'v always wondered this. In a game like GTA where there are 10s of thousands of objects, how does the game know as soon as you're on a health pack?

There can't possibly be an event listener for each object? Iterating isn't good either? I'm just wondering how it's actually done.

algorithm data-structures 3d collision-detection

Share Follow

edited Dec 25, 2009 at 23:49

Peter Mortensen

31.6k ● 22 ● 109 ● 133

asked Dec 25, 2009 at 5:32

jmasterx

54k ● 99 ● 326 ● 572

6 Answers Sorted by: Highest score (default)

65

There's no one answer to this but large worlds are often space-partitioned by using something along the lines of a quadtree or kd-tree which brings search times for finding nearest neighbors below linear time (fractional power, or at worst O(N^(2/3)) for a 3D game). These methods are often referred to as *BSP* for binary space partitioning.

With regards to collision detection, each object also generally has a *bounding volume* mesh (set of polygons forming a convex hull) associated with it. These highly simplified meshes (sometimes just a cube) aren't drawn but are used in the detection of collisions. The most rudimentary method is to create a plane that is perpendicular to the line connecting the midpoints of each object with the plane intersecting the line at the line's midpoint. If an object's bounding volume has points on both sides of this plane, it is a collision (you only need to test one of the two bounding volumes against the plane). Another method is the enhanced GJK distance algorithm. If you want a tutorial to dive through, check out NeHe Productions' OpenGL lesson #30.

Incidently, bounding volumes can also be used for other optimizations such as what are called *occlusion queries*. This is a process of determining which objects are behind other objects (occluders) and therefore do not need to be processed / rendered. Bounding volumes can also be used for *frustum culling* which is the process of determining which objects are outside of the perspective

viewing volume (too near, too far, or beyond your field-of-view angle) and therefore do not need to be rendered.

As Kylotan noted, using a bounding volume can generate false positives when detecting occlusion and simply does not work at all for some types of objects such as toroids (e.g. looking through the hole in a donut). Having objects like these occlude correctly is a whole other thread on *portal-culling*.

Share Follow

edited Dec 26, 2009 at 10:42

answered Dec 25, 2009 at 6:51

charstar

1,167 ● 8 ● 13

---

8 +1. Back in the '80s I did 3D modeling for architects, and we had to use every optimization trick under the sun to make it work in anything like reasonable time on that era's PCs. These two approaches were very high on the list.
– Joe Mabel Dec 25, 2009 at 7:07

Great answer! I have just started to implement some game-stuff on my own (see here: stackoverflow.com/questions/1916259/…) and bounding volumes were one of the ideas that I had come up with for pre-filtering my *very* expensive collision-detection. I had not even occurred to me that I could re-use it for occlusion detection. This is a **huge** help! – RBarryYoung Dec 25, 2009 at 7:10

4 The problem with using coarse bounding volumes for occlusion is that sometimes you will get false positives (eg.

thinking that an object is obscured by a human's bounding cylinder when in fact you should be able to see it through his legs and over his shoulders). Really the authoritative occlusion volume should be smaller than the object in question – Kylotan Dec 25, 2009 at 14:46

+1 on Kylotan's comment. Absolutely true. I can try to make that more clear. – charstar Dec 26, 2009 at 8:48

1 The coarse bounding volume just gives you a first approximation: then you actually have to *do* the comparison. But it means that you can cheaply eliminate a lot of stuff that isn't even near each other from ever being examined closely (and expensively). – Joe Mabel Dec 27, 2009 at 8:27

9

Quadtrees and Octrees, another quadtree, are popular ways, using space partitioning, to accomplish this. The later example shows a 97% reduction in processing over a pair-by-pair brute-force search for collisions.

Share Follow

edited Dec 31, 2012 at 20:53

answered Dec 25, 2009 at 6:26

joshperry

42.2k ● 16 ● 92 ● 104

4

A common technique in game physics engines is the sweep-and-prune method. This is explained in David Baraff's SIGGRAPH notes (see Motion with Constraints chapter). Havok definitely uses this, I think it's an option in Bullet, but I'm not sure about PhysX.

The idea is that you can look at the overlaps of AABBs (axis-aligned bounding boxes) on each axis; if the projection of two objects' AABBs overlap on all three axes, then the AABBs must overlap. You can check each axis relatively quickly by sorting the start and end points of the AABBs; there's a lot of temporal coherence between frames since usually most objects aren't moving very fast, so the sorting doesn't change much.

Once sweep-and-prune detects an overlap between AABBs, you can do the more detailed check for the objects, e.g. sphere vs. box. If the detailed check reveals a collision, you can then resolve the collision by applying forces, and/or trigger a game event or play a sound effect.

Share Follow

answered Dec 25, 2009 at 16:50

celion

4,004 ● 28 ● 20

2

Correct. Normally there is not an event listener for each object. Often there is a non-binary tree structure in memory that mimics your games map. Imagine a metro/underground map. This memory strucutre is a collection of things in the game. You the player, monsters and items that you can pickup or items that might blowup and do you harm. So as the player moves around the game the player object pointer is moved in the game/map memory structure.

see How should I have my game entities knowledgeable of the things around them?

Share Follow

edited May 23, 2017 at 12:02

Community Bot

1 ● 1

answered Dec 25, 2009 at 5:43

kingchris

1,757 ● 22 ● 29

1

I would like to recommend the solid book of Christer Ericson on real time collision detection. It presents the basics of collision detection while providing references on the contemporary research efforts.

Real-Time Collision Detection (The Morgan Kaufmann Series in Interactive 3-D Technology)

Share Follow

answered Dec 21, 2010 at 21:11

wojakzek

287 ● 1 ● 3 ● 6

1

There are a lot of optimizations can be used. Firstly - any object (say with index i for example) is bounded by cube, with center coordinates `CXi`, `CYi`, and size `Si` Secondly - collision detection works with estimations:

a) Find all pairs cubes i,j with condition: `Abs(CXi-CXj)<(Si+Sj) AND Abs(CYi-CYj)<(Si+Sj)`

b) Now we work only with pairs got in a). We calculate distances between them more accurately, something like `Sqrt(Sqr(CXi-CXj)+Sqr(CYi-CYj))`, objects now represented as sets of few numbers of simple figures - cubes, spheres, cones - and we using geometry formulas to check these figures intersections.

c) Objects from b) with detected intersections are processed as collisions with physics calculating etc.

Share Follow

edited Apr 29, 2014 at 23:42

user3396151

answered Dec 25, 2009 at 6:18

user224564

1,323 ● 1 ● 10 ● 14