Resources of techniques use for collision detection in 2D? [closed]

Asked 16 years, 3 months ago Modified 3 years, 10 months ago

Viewed 9k times

11

Closed. This question is seeking recommendations for software libraries, tutorials, tools, books, or other off-site resources. It does not meet Stack Overflow guidelines. It is not currently accepting answers.

We don't allow questions seeking recommendations for software libraries, tutorials, tools, books, or other off-site resources. You can edit the question so it can be answered with facts and citations.

Closed 3 years ago.

Improve this question

What are in your opinion the best resources (books or web pages) describing algorithms or techniques to use for collision detection in a 2D environment?

I'm just eager to learn different techniques to make more sophisticated and efficient games.

algorithm collision-detection

Share

Improve this question

Follow

edited Aug 19, 2017 at 14:44

m87

4,523 ● 3 ● 18 ● 31

asked Aug 27, 2008 at 20:21

borrego

719 ● 8 ● 7

5 Answers

Sorted by: Highest score (default)

14

Collision detection is often a two phase process. Some sort of "broad phase" algorithm for determinining if two objects even have a chance of overlapping (to try to avoid n^2 compares) followed by a "narrow phase" collision detection algorithm, which is based on the geometry requirements of your application.

Sweep and Prune is a well established efficient broad phase algorithm (with a handful of variants that may or may not suit your application) for objects undergoing relatively physical movement (things that move crazy fast or have vastly different sizes and bounding regions might make this unsuitable). The Bullet library has a 3d implementation for reference.

Narrow phase collision can often be as simple as "CircleIntersectCircle." Again the Bullet libraries have good reference implementations. In 3d land when more precise detection is required for arbitrary objects, GJK is among the current cream of the crop - nothing in my knowledge would prevent it from being adapted to 2d (but it might end up slower than just brute forcing all your edges ;)

Finally, after you have collision detection, you are often in need of some sort of collision response. Box 2d is a good starting point for a physical response solution.

Share Improve this answer Follow

answered Sep 18, 2008 at 0:32

Jeff
1,053 ● 1 ● 8 ● 14

2

Metanet Software has published some relevant tutorials. Metanet develops N (Flash-based, for Windows, Mac, Linux) and N+ (for the X360, DS, and PSP).

Share Improve this answer Follow

answered Aug 31, 2008 at 0:52

Nikhil
5,771 ● 1 ● 33 ● 30

2

Personally, I love the work of Paul Bourke.

Also, Paul Nettle used to write on the topic. He has a full 3D collision detection library, but you may be more interested in the ideas behind such libraries (which are very applicable to 2D). For that, see General Collision Detection for Games Using Ellipsoids.

Share Improve this answer

edited Feb 12, 2021 at 19:51

Follow

ggorlen
56.3k ● 7 ● 109 ● 147

answered Aug 27, 2008 at 20:24

Frank Krueger
70.9k ● 48 ● 164 ● 211

The work of Paul Bourke seems offline. Latest version available on the wayback machine is dated 22nd of July 2012. – Domenico De Felice Jan 9, 2014 at 16:39

1

The book 'Real-Time Collision Detection' by Christer Ericson (ISBN: 1-55860-732-3) is a recent (2005) and widely praised book which should give you some good answers.

It starts with a basic primer of some of the maths you will need to know, and then goes into various types of bounding volumes (spheres, axis-aligned bounding boxes, oriented bounding boxes) commonly used in collision detection.

Next up for discussion are numerous algorithms for detecting collisions between various combinations of primitives, such as lines, triangles, spheres, polygons, planes, bounding volumes etc.

Also of importance is the coverage of some of the major methods of spatial division and organisation of your objects (volume hierarchies, BSP trees, Octrees, etc.). This essentially speeds up collision detection, as it allows you to subdivide your objects so you can avoid unnecessary comparisons between objects (e.g. I know from my data structures that object A is too far away to hit object B, so I won't even do a distance check).

It also includes some coverage of how to actually check for collisions between moving objects (intervals, etc) but be aware that even though this is a fairly hefty book and covers the material well, it is for collision *detection*, not *resolution* or *response*. So it will help you determine whether two objects have collided, but not really what to do about it, i.e. how to resolve it. The intersection tests will usually give you the data you need to make such decisions, but in terms of the general problem of writing a *solver*, which uses collision detection routines to detect collisions and then decide what to do about them, this book does not cover that in depth.

Share Improve this answer Follow

answered Sep 2, 2008 at 17:53

Slacker
470 ● 1 ● 4 ● 10

0

If your objects are represented as points in 2D space you can use line intersection to determine if two objects have collided. You can use similar logic to check if an object is inside another object (and thus they have collided even any of their lines are not currently intersecting). The math to do this is quite simple, and should be covered by any textbook on basic geometry. Detecting if an object has passed completely through an object might be a bit more tricky though.

Share Improve this answer

Follow

edited Apr 5, 2013 at 10:45

Quonux

2,991 ● 1 ● 25 ● 32

answered Aug 27, 2008 at 20:29

Anders Sandvig

21k ● 16 ● 61 ● 74