Collision detection

Collision detection is used to find out which objects to collide against other objects. Normally it is broken down into several phases:

Broad Phase

Broad phase detection is typically a computationally low cost operation that quickly answers the question, “Which objects have a strong possibility of colliding?” Approaches include Sweep and Prune, and Spatial Partitioning.

SAT

zdroj

The Separating Axis Theorem, SAT for short, is a method to determine if two convex shapes are intersecting. The algorithm can also be used to find the minimum penetration vector which is useful for physics simulation and a number of other applications. SAT is a fast generic algorithm that can remove the need to have collision detection code for each shape type pair thereby reducing code and maintenance.

“If two convex objects are not penetrating, there exists an axis for which the projection of the objects will not overlap.”

No Intersection

First lets discuss how SAT determines two shapes are not intersecting. In figure 5 we know that the two shapes are not intersecting. A line is drawn between them to illustrate this.

[http://www.dyn4j.org/wp-content/uploads/2010/01/sat-ex-1.png]

If we choose the perpendicular line to the line separating the two shapes in figure 5, and project the shapes onto that line we can see that there is no overlap in their projections. A line where the projections (shadows) of the shapes do not overlap is called a separation axis. In figure 6 the dark grey line is a separation axis and the respective colored lines are the projections of the shapes onto the separation axis. Notice in figure 6 the projections are not overlapping, therefore according to SAT the shapes are not intersecting.

[http://www.dyn4j.org/wp-content/uploads/2010/01/sat-ex-2.png]

SAT may test many axes for overlap, however, the first axis where the projections are not overlapping, the algorithm can immediately exit determining that the shapes are not intersecting. Because of this early exit, SAT is ideal for applications that have many objects but few collisions (games, simulations, etc).

Spatial Partitioning

zdroj

Spatial Partitioning is the act of dividing up a continuous space into several discrete areas based on a few simple rules (the rules change with each technique for spatial partioning).

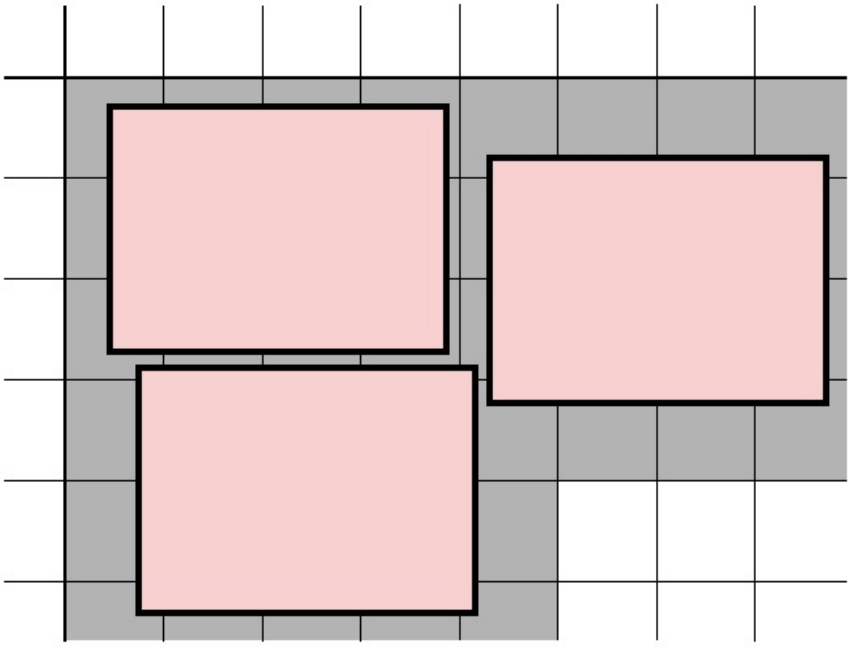
The rules of our gridding system are as follows:

- A cell is defined as a square having discrete boundaries (e.g. it describes an exact “piece” of space). 6
- If an entity’s axis-aligned bounding box overlaps with a cell, the entity will be inserted into that cell.

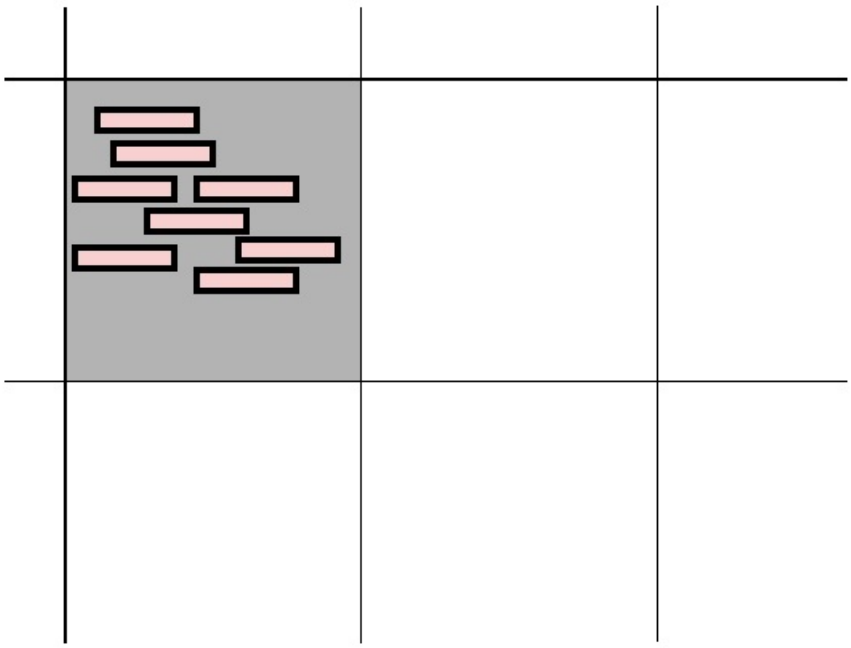
- An entity can be inserted into multiple cells.
- The grid will be discarded and rebuilt after every world update.
- Looking for coliding pairs requires iterating over every occupied cell of the grid.
- We must track which pairs have already been tested against each other.

Choosing cell size:

Cell size plays a large role in how efficient the grid can be. In the first figure, the effect of a very small cell size is seen when entities are large. Each entity overlaps many cells, and because the spatial grid iterates over cells, and not entities, there is actually more work for it to do than a brute force entity-to-entity comparison.



In the second figure, a very large cell size is paired with small entities. In this case, only one cell will need to be visited, but each entity will need to be tested against every other entity, which is, again, the same as a brute force entity-to-entity comparison.



Mid Phase

This is where you normally do a simple collision check such as a bounding box or bounding sphere check because they would be faster than your narrow phase. Most spatial index structures are going to return some false positives, and it's best to catch them early instead of calling the more expensive narrow phase collision routines.

Bounding Volume Hierarchy

zdroj

A bounding volume hierarchy (BVH) is a tree structure on a set of geometric objects. All geometric objects are wrapped in bounding volumes that form the leaf nodes of the tree. These nodes are then grouped as small sets and enclosed within larger bounding volumes. These, in turn, are also grouped and enclosed within other larger bounding volumes in a recursive fashion, eventually resulting in a tree structure with a single bounding volume at the top of the tree. Bounding volume hierarchies are used to support several operations on sets of geometric objects efficiently.

Voronoi Region

zdroj

A Voronoi region is associated with each feature of an object: each vertex, edge and face.
Each Voronoi region is the set of points that are closest to that feature.
To find the closest feature to a given point, find the Voronoi region that contains it
– Easy to exploit coherence (remember the
closest feature from the last frame)

Convex Shape Decomposition

zdroj1
zdroj2
zdroj3

One common strategy for dealing with large, complex models is to partition them into pieces that are easier to handle. Many problems can be solved more efficiently when the input is convex. While decomposition into convex components results in pieces that are easy to process, such decompositions can be costly to construct and often result in representations with an unmanageable number of components.

In theory, there is a method, that can decompose objects of arbitrary dimension, although for very high dimension objects, the complexity is high. For 2D and 3D objects, such decomposition is fast.

The decomposition is achieved by minimizing the total cost of decomposition under some concavity constraints, which guarantees to be globally optimal in many situations.

Narrow Phase

The narrow phase is the most specific and most expensive collision detection check. It is the fine grained, “What part of object A colided with object B?” step. It is typically computationally intense, and thus cannot be performed on every pair of objects in the time between game updates (e.g. the next drawn frame). Examples of narrow phase techniques are the Hyperplane Separation Theorem (also known as the Separating Axis Theorem) 1, Pixel Perfect Collision Detection 2, and Swept Tests.

Minkowski Difference

zdroj

The Minkowski Difference consists of the convex hull of every point on the boundary of one shape subtracted from every point on the boundary of the other shape. Its easier to imagine this as shrinking one of the shapes down until it becomes a point, and expanding the other shape by the shape of the first. Configuration space is the space that the Minkowski difference resides within - the set of vector differences of two objects.

It turns out that when the Minkowski Difference (written MD from now on) of two shapes contains the origin, the two shapes are intersecting. Furthermore, the distance from the origin to the MD is actually the distance between the two shapes, and the vector between the two gives you the Minimum Translational Distance which is the globally shortest distance you can move either shape from penetration until they both just touch.

This is how you build MD for two OBBs. Each edge in the MD is made up of a vertex from one shape subtracted from an edge from the other. In order to know which vertices we subtract from which edges, we need to define the concept of a supporting vertex.

A supporting vertex is simply a vertex which is "most in the direction of" a given direction vector. Mathematically, this can be found as the vertex which has the greatest dot product with a given direction vector.

Once we have this we can simply loop through all the edges of an object, finding the supporting vertex in the other object for each edge by using reversed edge normal. The edge normal is reversed because we want to find the vertex "most opposite" this edge. As we do so, we project the origin onto each edge, and keep track of the distance - we're looking for the least negative distance, which will be the Minimum Translational Distance.

Once we've gone through edges of the first object, we do the same thing with the other object, this time using supporting vertices from the first one, again tracking the least negative distance. If we're only concerned about penetration we can terminate our search as soon as we find a positive distance - since this represents a separating axis.