April 25, 2006

Collision Detection and the use of The Screw

Different Collision Detection Scenarios

1. Collision detection in a static image
2. Collision detection in animation

- Collision detection in animation is not a trivial problem
- The objects of polygons in the scene can move in arbitrary paths
- Even if the objects are moving with constant velocity, still a difficult problem

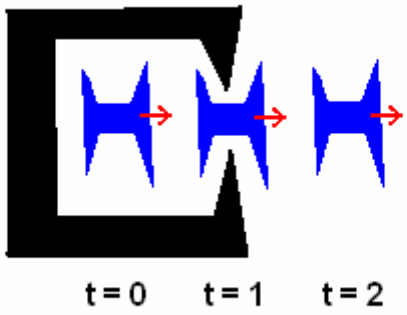

Figure 1. Object moving with constant speed

As seen in Figure 1, the blue object is moving towards the right at a constant speed. We want to know if it collides with the black object. If we were to take snapshots every dt amount of time, something like the above can happen. At all three snapshots, there is no collision. However, in reality, if the object is moving with a smooth transition, then there will be collision from t = 0 to t = 2. One cannot just decrease dt, because however small it is, something like the scenario in Figure 1 can always happen.

Two blobs collide

Suppose there are two blobs moving in animation, then at some point, the two blobs collide. What can we do to prevent this? Or is there an alternative?

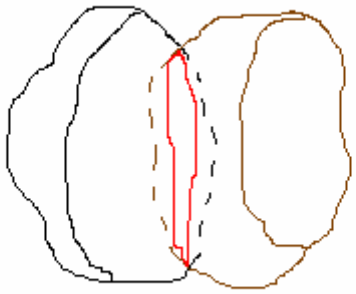

Figure 2. Two blobs collide

In Figure 2, there is a left blob in black, and a right blob in right. The red region is the surface formed between the two objects. This surface is NOT necessarily flat.

Rollback

One method to prevent this collision is to “rollback” in the animation sequence. If we are able to move backwards in time back several time frames, then we will be able to prevent the collision. However, this is not always possible.

Push/Pull apart

Another tweak would be to somehow “push” the two blobs apart slightly. This can be effective if the penetration is not very deep, so we can just offset by a small amount.

1. Need to find the direction in which to pull the blobs apart.
2. Can do this by using the normals around the edges of the red surface.
3. If the penetration is not too deep, we may not need to add all the normals and average them.
4. Instead, use a weighted normal that is weighted by the triangles.

Frame of reference

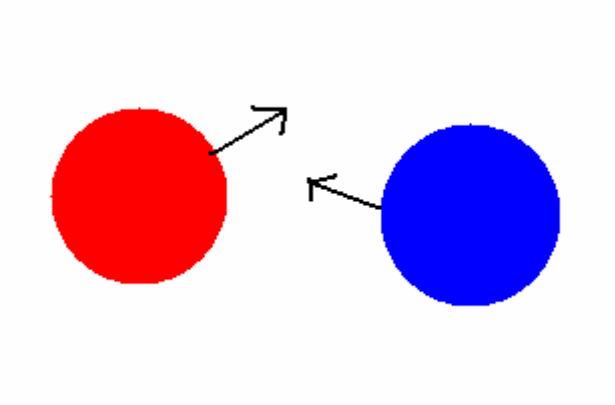

Figure 3. two moving objects

Figure 3 shows a red ball moving in some direction and a blue ball moving in some direction. To simplify the collision detection problem, we can change the frame of references.

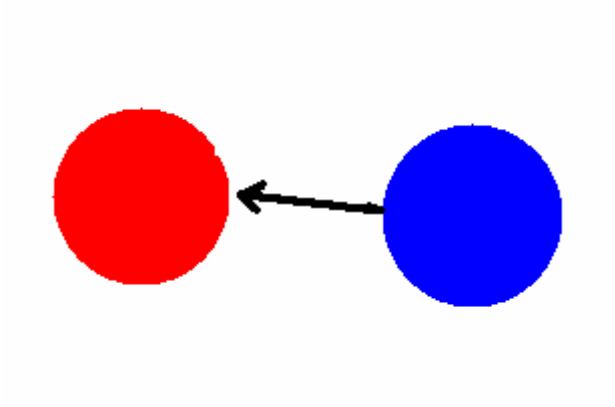

Figure 4. From Red's frame of reference

If we were to look from the red ball's frame of reference, the red ball is standing still, but the blue ball is moving towards the red. To find this vector, simply subtract Red's vector from Blue's in the original.

To further simplify, instead of having a ball/ball detection, we can add one ball's radius to the other and change the first ball into a point. The result will look like something in Figure 5. Both the left and right configurations results in the same collision detection.

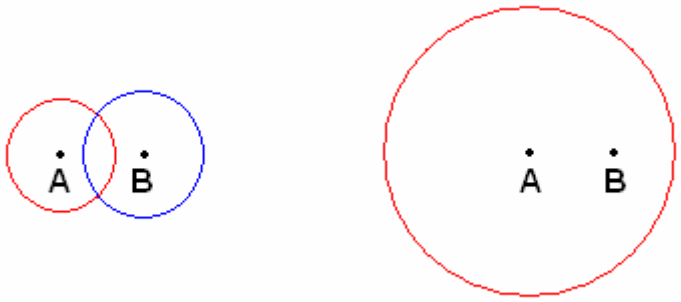

Figure 5. Two of the same collisions

Collision types

Given two meshes, where are essentially 3 different scenarios that can happen when they collide.

1. Point on Object 1 collides with a Face on Object 2
2. Face on Object 1 collides with Point on Object 2
3. Edge of Object 1 collides with Edge on Object 2

Note: The edge – edge detection of #3 can also be generalized to point – edge, edge – point, or point – point.

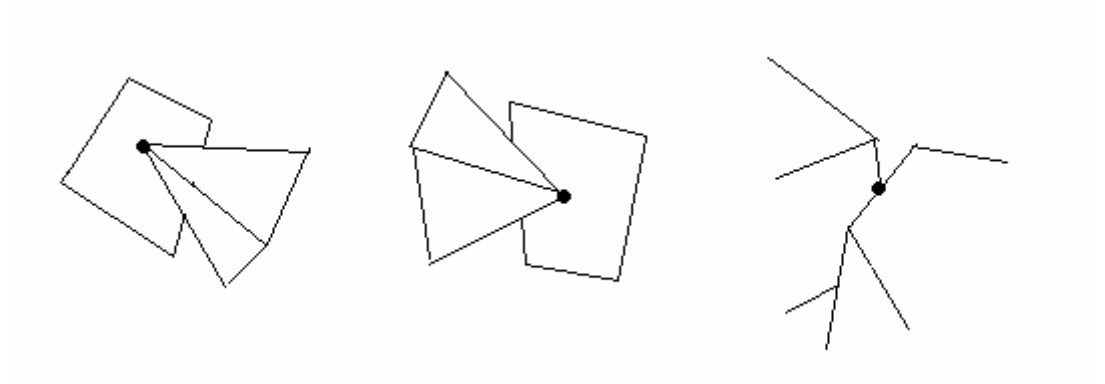

Figure 6. Types of collisions

However, types 1 and 2 are essentially the same if we just change the frame of reference.

To test the point – face collision, first check if the point is on the plane of the face. Do a dot product of the point on the plane. If the result is 0, then the point is on the plane, otherwise, it is not. Next, check if the point is within the polygonal face. A simple way is to shoot several rays outwards from the point. If each ray passes an edge an odd number of times, then he point is inside the polygon.

To test the edge – edge collision, one can form a tetrahedron using the two edges. If we were able to calculate the volume of this tetrahedron, then we know the two edges are coplanar when the volume equals zero. Once we test they are coplanar, next test if they intersect each other.

Screw motion

There is an initial position of the object.
Pick a final destination of the object allowing rotation in any direction
There will always be a path from initial position of object to final position where we are allowed to rotate around the path's axis. This can be visualized as a "screw" motion. Note that this is not unique, because we can always rotate in the opposite direction.

This helps with the problem of solving objects in the scene moving with arbitrary paths. Now we can show the motion of the object, and be possible to test if collide with another object. All objects in scene can then move with a screw motion.

This also works for 2D space.

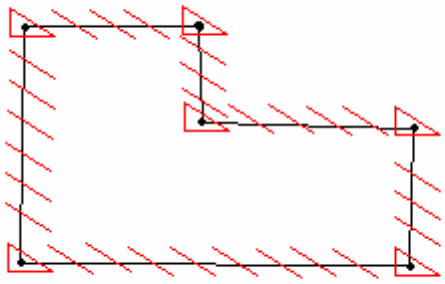

Figure 7. Screw motion where there is no rotation between control points

Figure 7 shows the screw motion of the triangle between the vertices of the black polygon. There is no rotation of the triangle between each control point, so the path of the translation is a straight line between consecutive control points.

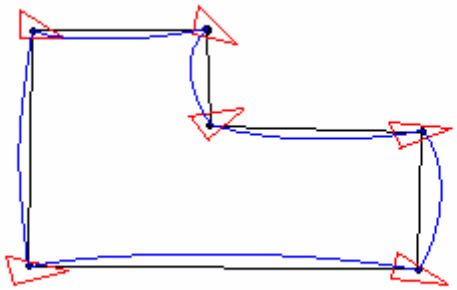

Figure 8. Screw motion where there is rotation between control points

Figure 8 shows a random rotation of the triangle between control points. The path of the screw is no longer straight. The blue line is the path the triangle will take from one point to the next. Along this line, the triangle will gradually rotate into the configuration of the next point.