CS 314: More Collision Detection

Robert Bridson

November 13, 2008

This is unfinished text: just a point-form summary of the lecture.

- We can extend the raytracing model to testing moving points versus rigidly-moving objects, by transforming the point into the object space (in the object's coordinate system it's stationary—it's just the object coordinate system itself is moving with respect to the world)
- Another example is particularly useful for characters in a game: model them as simple axis-aligned cylinders. If the objects in the world are, say, axis-aligned boxes, then collision detection can be separated between axes—you can break it down into a 1D problem along y and a 2D problem in the xz plane.
- To check if the cylinder overlaps with a box, first check if their intervals along the y-axis overlap. If so, then check if the cylinder's cross-section in xz (a circle) overlaps with the box's cross-section (a rectangle).
- A nice way to do the circle-rectangle test is to find the closest point in the rectangle to the circle's centre. This can be done by simply clamping the coordinates of the circle's centre to the bounds of the box. Then check if the distance between the circle's centre and the closest point in the rectangle is less than the circle's radius—this is equivalent to checking for overlap.
- Finding closest points is often very useful for simplifying collision checks, and will be useful in responding to collisions too.
- One particularly nice thing about smooth geometry like circles and spheres, for collisions, is that the normal is well-defined everywhere: at the corner of a box, there are multiple possible normals which can pose a problem later on when we try to respond to the collision...
- Also note that for checking overlap with a big collection of objects, a BVH can be used just as for raytracing: test overlap with the root box of the tree first, then traverse to children as necessary.