CS425: Game Programming 1
Lecture 6: Collision Detection

9/18/2013

Lecturer: Jyh-Ming Lien

1 Introduction

1.1 Collision Detection

1. Intersection between two line segments
2. Intersection between a point and a polygon
3. Intersection between two polygons
4. Intersection between n polygons

For many collision detection methods and applications, the point of intersection of two line segments are used. Therefore, we start this lecture by looking into the computation of the intersection between two line segments and then intersection between a point and a polygon. We will then discuss intersections between two polygons and intersections among n polygons.

2 Two line segments

Let the two segments have endpoints a and b and c and d, and let L_{ab} and L_{cd} be the lines containing the two segments. A common method is to solve slop-intercept equation for L_{ab} and L_{cd} simultaneously. However, definition involves slop has many special cases, e.g. a vertical line has slop to the infinity. Instead, we will use a parametric representation of the two segments.

parameterization Let $A = b - a$ and $C = d - c$; these vectors point along the segments. Any point on the line L_{ab} can be represented as the vector sum:

$$p(s) = a + sA$$

parameter s The variable s is called the parameter of this equation. Consider the values obtained for $s = 0$, $s = 1$, and $s = 1/2$:

$$p(0) = a\ ,$$

$$p(1) = a + A = a + b - a = b\ \text{, and,}$$

$$p(1/2) = (a + b)/2$$

.

Another line Similarly we can represent L_{cd} as $= q(t) = c + tC$. The line segment between c and d is corresponding to values between $t = 0$ and $t = 1$.

Intersection The intersection between the parameterized line is then:

$$\begin{aligned}
D &= a_0(d_c - c_1) + b_0(c_1 - d_1) + d_0(b_1 - a_1) + c_0(a_1 - b_1) \\
s &= [a_0(d_1 - c_1) + c_0(a_1 - d_1) + d_0(c_1 - a_1)]/D \\
t &= [a_0(c_1 - b_1) + c_0(a_1 - c_1) + d_0(b_1 - a_1)]/D
\end{aligned}$$

Here a_0 and a_1 are the x and y coordinates of point a. Same rule applies to points b_i, c_i, d_i, where $i = \{1, 2\}$.

Question: What does it mean when $D = 0$?

Implementation bool SEGSEGINT(double a[2], double b[2], double c[2], double d[2], double x[2])

Problems:

Implementation Let us try again.
SEGSEGINT(double a[2], double b[2], double c[2], double d[2], double x[2])

3 A point and a polygon

Given a 2d point q and a polygon P, check if q is inside P or not. Let P be represented as a list of n vertices, i.e., $P = \{v_1, v_2, \cdots, v_n\}$.

P is convex This can be done in $O(n)$ time by

P is NOT convex This can be done in several ways.

Implementation Let us implement the case that P is not convex.
bool INTERSECT(double q[2], polygon P)

Problems:

Implementation Let us try again.
bool INTERSECT(double q[2], polygon P)

4 Two polygons

Given two polygons $P = \{v_1, v_2, \cdots, v_n\}$ and $Q = \{u_1, u_2, \cdots, u_m\}$.

If both P and Q are circles

If both P and Q are axis-aligned boxes

If both P and Q are oriented boxes An oriented box is a box whose edges are not parallel to the x or y axes.

If both P and Q are convex The intersection can be found in $O(n+m)$ time by

If both P and Q are NOT convex Intersection can be found in several ways. Let use try brute force first.

Implementation bool PolyPolyInt(polygon P, polygon Q)

Bounding Volume (BV) Bound P and Q using circles, boxes, or oriented boxes.

Implementation with BV bool PolyPolyInt(polygon P, polygon Q)

Bounding Volume Hierarchy (BVH) Bound P and Q using a tree of circles, boxes, or oriented boxes.

Implementation with BVH In this case we will have to build BVH first.
BVHtree PolyPolyInt(polygon P)

bool PolyPolyInt(polygon P, polygon Q)

5 Multibody collision detection: n polygons

Given n polygons $\{P_0, P_1, P_2, \cdots, P_n\}$, report pairs of polygons $\{P_i, P_j\}$ such that $P_i \cup P_j \neq \emptyset$.

Brute Force Make $O(n^2)$ poly-poly intersection calls.

Plan sweep This is one of the most efficient way but very difficult to implement.

Interval tree Slower than plan sweep but simpler to implement. The main idea is to project the polygons to x and y axes and the check which intervals overlap.

1D Interval tree Given a list of n intervals $\mathcal{I} = \{I_1, I_2, \cdots, I_n\}$, a node of this 1D interval tree is constructed as the following.