Real Time Collision Detection in 3D Games

Richard McAleer and Bradford G. Nickerson
March 22, 2006

Abstract

Collision detection in modern games involves a systematic refining of object space to reduce the number of needed comparisons. Methods for performing such refinement were studied, including culling (collision frustum), bounding volumes (Axis-Aligned Bounding Boxes – AABB's and Oriented Boundng Boxes - OBB's) and finally a direct collision testing method (General Separating Axis Test). The hypothesis is that as more levels of abstraction are employed, the time necessary for performing collision detection at each time step of a game simulation will decrease, resulting in improved frame rates. Using a simple vehicle simulation system, the vehicles path was captured during a run of the simulation and later 'played back' using the different collision methods and varying levels of scene complexity (increased number of objects) while capturing frame rate data and run time of current collision detection algorithm at each time step and averaging the collected data. The experimental data shows that the collision frustum culling algorithm is ineffective when combined with the AABB and OBB, giving worse performance when combined with AABB's in all cases and varying results with OBB's and no significantly better performance. Collision frustum culling does, however, give significantly improved results when combined with the general separating axis test (see Table 1). The results were at first surprising, but make sense as the AABB, OBB and collision frustum are all constant time algorithms (for pairs of objects in AABB/OBB case and when comparing a single object to the frustum in the case of the collision frustum). They also have a comparable number of steps to perform a collision test (Maximum of 3 for AABB, 4 for collision frustum and 15 for OBB). The general separating axis test, however, is $O(e_1*e_2)$, where e_1 and e_2 are the number of edges in each object being compared. It therefore makes sense that if we have objects with many edges being compared we can have much worse running time if we must test against all objects in the scene and using a constant time algorithm to eliminate unnecessary comparisons our computation time will improve.

Table 1. Precomputed path test results for collision detection.

Frame Rate (Frames/second)					
Number of Objects	General Separating Axis Test	AABB	OBB	View Frustum	
				AABB	OBB
10	30.96	30.70	29.90	29.67	29.92
100	25.53	26.89	26.90	25.99	26.14
1000	11.29	15.19	15.20	14.89	14.88
Actual Time for Algorithm (in milliseconds)					
10	0.203776	0.00857287	0.010705	0.0179493	0.0202375
100	1.98974	0.0280095	0.0446972	0.0447027	0.046627
1000	19.4600	0.151315	0.320517	0.269872	0.278576