Navigation Challenges in Massively Destructible Worlds

Paul A. Kruszewski

CTO, Engenuity Technologies, Inc.
4700 de la Savane, Suite 300
Montreal, Quebec H4P 1T7 Canada
paul.kruszewski@engenuitytech.com

Abstract

The creation of life-like believable characters is emerging as the central focus of next-generation game development and is viewed as critical to obtaining true mass-market appeal. Visually-realistic movement in complex physics driven worlds is an increasingly critical component in creating believable characters. The demo is based on our extension to the Unreal 3 engine (UE3) and runs on advanced PCs, the Xbox 360 and PlayStation 3 (PS3) and shows advance navigation behaviors such as dynamic obstacle avoidance, local dynamic path finding and advanced obstacle traversal such as vaulting. A typical first-person shooter (FPS) game level (an underground subway system) is constructed out of physics-simulated objects; the benches and garbage cans can be displaced by gun fire and the pillars and railings can be destroyed by grenades. This allows the player to quickly and significantly disrupt the world making it extremely difficult for the non-player characters (NPCs) to navigate towards the player. NPCs can be given specific movement abilities so that some NPCs can avoid obstacles by circumventing them while more advanced NPCs can vault over them. Upon discovery of blocked passageways such as staircases, NPCs can take alternative routes.

Introduction

Advances in next-generation gaming hardware such as the Xbox 360 and PlayStation 3 and physics engines such as Havok and Ageia PhysX allow the creation and simulation of large and complex physics based games. It remains an open problem to efficiently control the navigation of NPCs so that the result is visually convincing. We present a system that has been developed conjunction with leading game development companies such as Midway Games to solve this problem from a practical game production perspective.

Figure 1. Massive Destructability: State of the Art Physics Driven Game Play (Stranglehold; Courtesy Midway Games 2007).

Goals and Functionalities of a Physics-Aware Navigation System

Physics systems are now capable of correctly simulating the movement of physical objects during a game. Corresponding artificial intelligence (AI) systems need to simulate the movement of NPCs in these environments. Just as rocks must move in a visually realistic way, NPC must navigate in a visually realistic way: e.g., when people are confronted with physical objects, they do not simply want to them but rather they plan how to move around them. Moreover, just as different physical objects move in different ways (e.g., a rock and a piece of paper fall to the ground in completely different ways), different characters have different navigation abilities (e.g., poorly instructed grunt soldiers may stumble around blow up pieces of wall, whereas elite "Indiana Jones" soldier can gracefully vault over the same rubble). An AI navigation system needs to allow game designers to create the entire spectrum of skilled characters that can operate in any kind of environment.

Demonstration of the system

We use as our test bed a section of a subway system called "Underground". Underground is 100m by 70 m and contains 1000 movable physics objects such as benches and garbage cans and numerous destructible physics objects such as rails whose destruction result in to approximately 4000 shards. All art assets including the character models and animations were generated in Maya and exported to our modified version of Epic's Unreal 3 game engine. The final version of the demo will have approximately 32 NPC running in a few physics environment. The system demonstrates three kinds of major dynamic navigation: obstacle avoidance; local path refinement and obstacle traversal. The following screen shots illustrate the type of cluttered environments the NPCs must navigate.

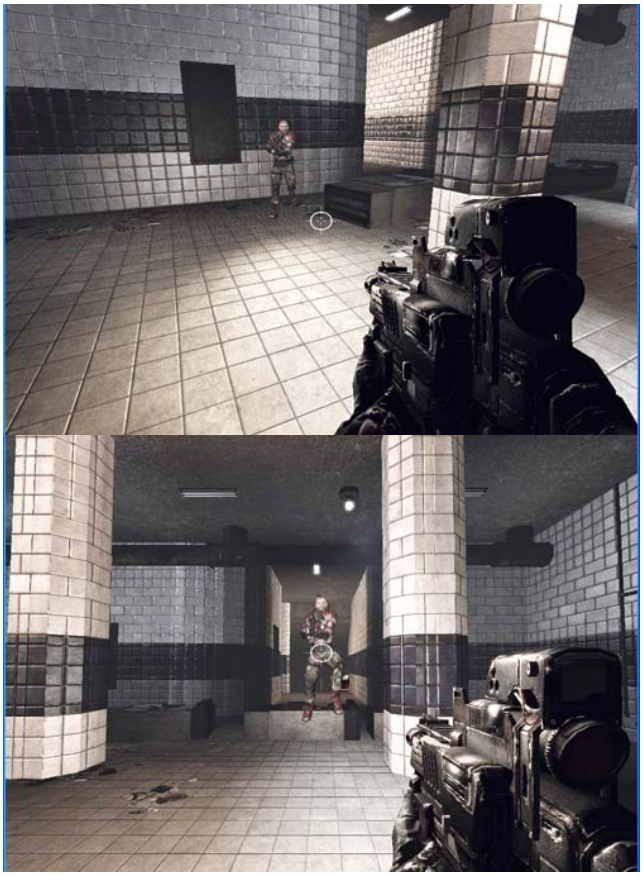

Figure 2, 3. NPCs can either go around or jump over the physics-driven objects such as barrels, benches, and shattered walls.

Obstacle Avoidance

Our obstacle avoidance system was originally influenced by Craig Reynold's seminal work steering behaviours. Since then it has been highly optimized and made significantly more robust.

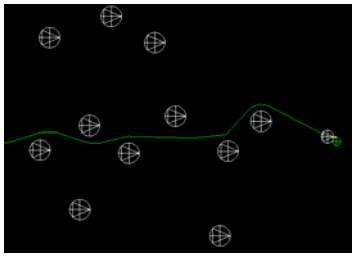

Figure 4. Resulting motion trail generated by obstacle avoidance.

Dynamic Local Path Refinement

Steering behaviours are relatively efficient as they only look forward for their avoidance, however this implies that NPCs can be caught in local minimum, especially when the obstacles are not circular in shape. In order to overcome these limitations, we can developed system called dynamic path refinement (DPR). DPR calculates polygon hull around objects that are blocking the NPC along its path.

DPR then does a local search on the free space defined by the navigation graph that results from the polygon hulls.

Figure 5. Initial path before DPR.

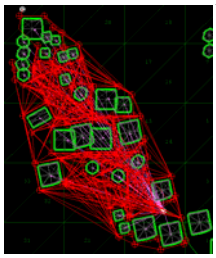

Figure 6. Interim local search graph based on polygon hulls.

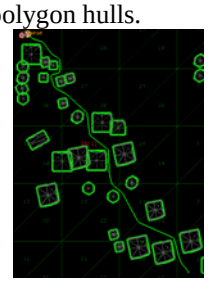

Figure 7. Final motion trail.

Obstacle Kicking and Traversal

Of course, circumventing obstacles is not only way to deal with obstacles that are blocking a NPC, alternative strategies can include kicking (or shooting) obstacles out of the way or jumping over them. Our system has a notion of frustration. This allows the system to know when its progress is being impeded. A call back mechanism informs the higher level AI system of what is blocking its progress, given this information the higher level brain can decide to seek out an alternative route or merely kick it out of the way (e.g., a small box). Jumping over an obstacle can be another strategy. For this to work with the game animation system, typically the NPC needs to line itself up a specific spot and location on the vaultable object so that the animation system can play a clean animation clip.

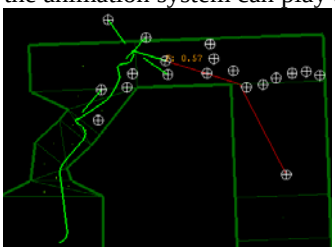

Figure 8. Kicking obstacles out of the way.

Figure 9. Jumping over obstacles.

References

Reynolds, C. 1987. Flocks, herds, and schools: A distributed behavior model. In Proceedings of ACMSIGRAPH 87, Annual Conference Series, ACM SIGGRAPH.

Acknowledgements

The AI.implant system is the result of many years of hard teamwork. The author would like to thank the entire AI.implant development team at ETI for its ongoing excellent work.