Intelligent Sneaking Agents for Games

CIS 499 SENIOR PROJECT DESIGN DOCUMENT

Brittany Fields
Advisor: Dr. Norman Badler
University of Pennsylvania

PROJECT ABSTRACT

The goal of this project is to develop AI for intelligent sneaking and hiding behaviors in a controlled environment. This algorithm would then be applied to have an agent complete a "capture the flag" task involving navigating through an environment with patrolling guards and obstacles, and utilizing designated hiding places to avoid being seen. My vision for the final product is a game in which a human player takes on the role of the "guard" and tries to find and catch the NPC (non-playable character) before it completes its objective. Needless to say, the NPC should be intelligent enough to provide a sufficient challenge to a human player. I'm hoping to have the guard agents and sneaking NPC have both visual and sound perception abilities and in the case of the sneaking NPC, use these to assess its situation and locations of hiding places and guards to make decisions in real time.

Project blog:
http://bripsdp.wordpress.com

1. INTROUDUCTION

1.1. Significance of Problem or Production/Development Need

There are many stealth games out there, but very few, if any, that have the NPCs using intelligent sneaking and hiding behavior – essentially taking on the role that the player usually assumes in such games. Having an NPC intelligent and skillful enough to evade other NPCs and ultimately, a human player, could be useful in various simulation environments as well as in video games.

1.2. Technology

HiDAC/CAROSA crowd simulator, eventually transplant over to C# and XNA, start with basic pathfinding and behavior techniques and build from there

1.3. Design Goals

1.3.1 Target Audience.

Anyone needing to utilize an agent with intelligent sneaking and evasion in a controlled environment, most likely game developers or those wanting to simulate environments and situations conducive to this behavior for other purposes.

1.3.2 User goals and objectives

In the game, the player will play as one of the “guard” agents. The NPC will start at some random location and attempt to make its way to and capture some object (the “flag”) and return to its starting location without being seen by the guards. The player’s goal is to stop the NPC before it does this using some “catching” mechanic to be decided. This could simply be catching the NPC inside its range of vision, or something more complicated.

1.3.3 Project features and functionality

I want to develop both classes of agents and the environment so that they can function autonomously (without any user-input at run time), but also such that they function the same when a human player controls one the agents, playing either the “predator” or the “prey” role.

2. Prior Work

Hiding and Evasion:
Parasol’s Algorithm & Applications Group’s “Nathrually-Inspired Exploring, Pursuit and Evasion Behaviors” focuses on the group behaviors of predators and prey and have developed a “General Evasion Algorithm,” which can incorporate the purusing agent(s) visible to the evading agent(s), the evader’s memory of pursuing agents and communicating knowledge of pursing agents to other pursuing agents into the evading agent’s behavior. The two implemented strategies of evasion are “flee-and-freeze” and “flee-and-hide.” In the “flee-and-freeze” implementation, the evaders remain in their current location, and flee to a safer location once spotted and resume waiting. The “flee-and-hide” implementation attempts to reduce the likelihood of being or staying visible using hiding locations.

In “Efficient and Dynamic Response to Fire,” the aim is to develop more realistic hiding and covering behaviors in first-person-shooter games. In my opinion, one of the remarkable things about this algorithm is that it takes into account potential visibility at the new hiding location for 3 different postures (“prone”, “crouching,” or “standing”), making it so that hiding spots in which the agent can remain hidden in all 3 postures are favored.

3. PROJECT DEVELOPMENT APPROACH

3.1. Algorithm Details

To start off, goals for each type of agent:
"Predator" Patrol areas in set paths

- When they "hear" a noise, deviate from their patrol path to investigate in the direction the noise came from and after some (tbd) criteria is met (either a set time is passed, a certain area has been checked, etc…), resume their original patrol path
- Should pursue any "prey" in sight

"Prey"

- Objective is to "capture the flag" essentially. Start at a location, move towards an object (the "flag"), capture it and return to the starting location without being seen
- If spotted, flee from the predator and perhaps attempt to disorient them
- Can generate loud noises depending on how fast its moving that could alert nearby predators

Based on the prior work described above:

Naturally-Inspired Exploring, Pursuit and Evasion Behaviors
The "flee-and-hide" model described here seems to be a good starting point for developing an algorithm. The prey scans for hiding points within view (which will most likely be areas just flagged as "hiding spot") and pick the best on based on a number of factors including: the distance to the hiding spot, the distance from and whether it would have to move toward or away from a predator in order to reach the spot, and in addition to this, the distance from the goal is important (we don't want to be moving backwards) and also how fast it could get there without making noise (if predators are nearby, otherwise this will influence the decision very little if at all). Depending on the weighting of these factors the "prey" agent's priorities could range from getting to the goal as fast as possible or staying hidden at all times, regardless of the time it takes to reach the goal.

Efficient and Dynamic Response to Fire
Obviously, I'm not focusing on responding to fire, but parts of the hiding spot-picking algorithm could prove useful, since I plan to be doing a "run-time search in the immediate vicinity of the agent," as it says in the abstract of this paper.

A* Pathfinding
Could use this as basis, and modify the F cost equation to incorporate the desired decision-making factors.

3.2. Target Platforms

3.2.1 Hardware

PC

3.2.2 Software

HiDAC/CAROSA crowd simulator
Visual Studio
XNA Game Studio

4. WORK PLAN

4.1.1. Project Milestone Report (Alpha Version)

I want to have some manner of (at the very least), pathfinding AI that should have the "prey" completing its task successfully. I'm not sure how good at hiding and evasion it should be at this point. I'll probably decide that once I get more into actual coding.

4.1.2. Project Final Deliverables

I'll probably decide this sometime between now and the alpha review, but ideally, the final product will be the XNA game, though depending on how it all goes there may not be time to complete that by the end of one semester.

4.1.3 Project timeline.

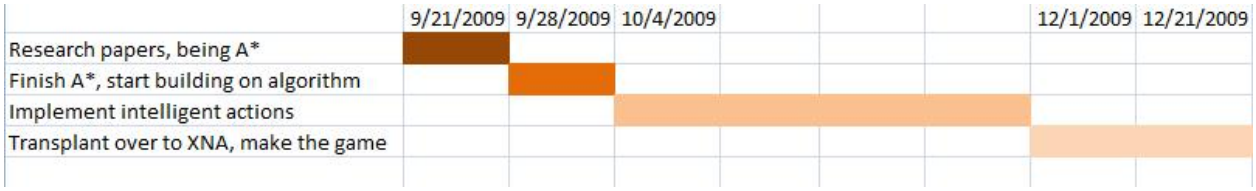
Sept 21 – 27: continue to research (find papers) and gather information. Read up on A* and begin implementation. I'm choosing a simple algorithm to begin with to test out the framework and prove that coding in the chosen environment is feasible.

Sept 28- Oct 4: finish up A*, start incorporating obstacles and "guards," probably as view cones initially

Oct 4 – Dec 1: from here on, iteratively improve intelligent actions such that they produce better paths than simple obstacle avoidance and incorporate sound perception and the all the other decision factors previously mentioned. Every 2 or so weeks I plan to have completed implementation on one of these factors, though at this time I'm not sure of the order in which I'll add them.

Dec 1- Dec 21: Assuming the AI is satisfactorily complete, at this time I'll begin building the game using XNA Game Studio.

4.1.4 Gant Chart



5. REFERENCES

Efficient and Dynamic Response to Fire
A* Pathfinding
Naturally-Inspired Exploring, Pursuit and Evasion Behaviors