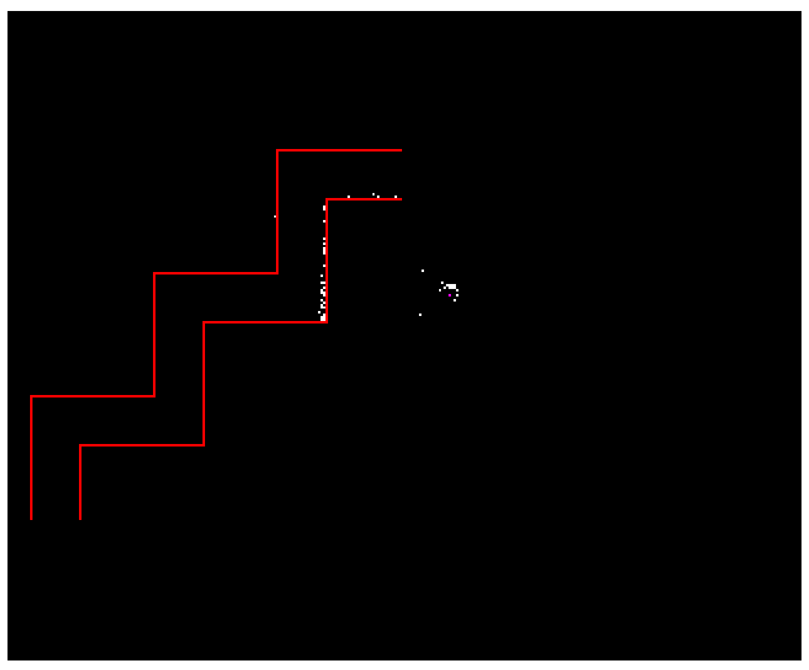
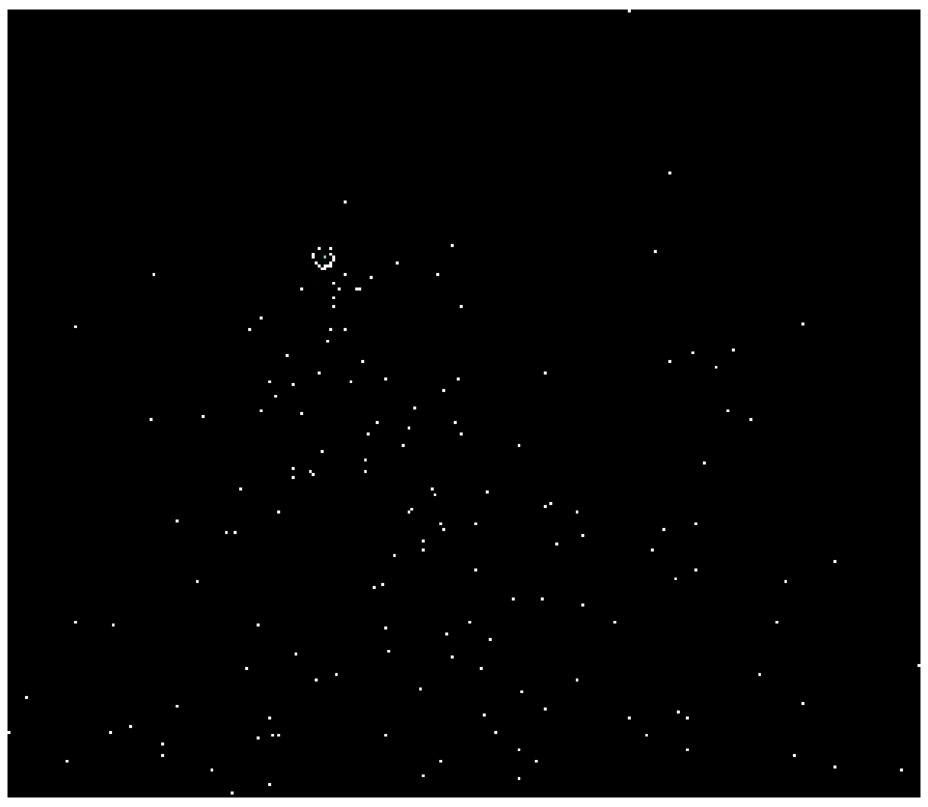
Crowd Dynamics Artificial Intelligence with Particle Swarm Optimization

TJHSST Computer Systems Lab, 2007-2008
By: Keith Ainsworth

Introduction

Articial Intelligence has for long been an important aspect of computer science, but unfortunately articial intelligence is usually computed from a single agent perpective or with multiple, but highly omniscent agents. I have created an articial intelligence engine, which works by having multiple agents, each with a highly limited perspective. In order to solve tasks, they need to communicate their portions with each other through a network. Using that scheme, it will much more accurately simulate crowd dynamics, as seen in real life, using particle swarm optimization to optimize the calculations.


Above is a screenshot of a Pnetwork simulation running with ngon obstructions.

Below is a screenshot of a theta-bounded agent field simulation. In the middle of the dense cluster. Towards the upperleft is the target.


Background

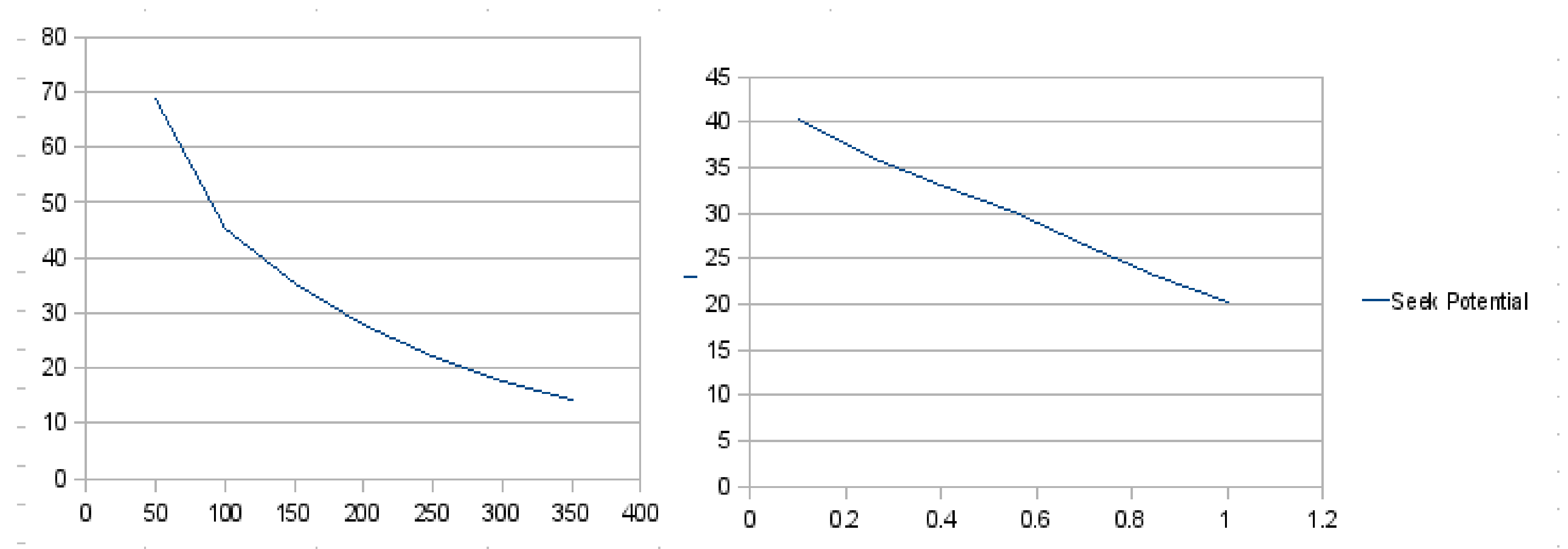
This program relies heavily on object oriented programming and function pointers, two fairly involved programming tasks. Early on I ran into a roadblock attempting to create two simultaneously co-referencing classes. I solved that issue through template classes. I will expect to have to solve many similar issues in the future in the same manner. To program this engine, along with the accompanying game (for graphical output reasons) I've used C++ (and therefore the g++ compiler) along with the SDL (software digital layer) libraries, for keyboard input and graphical output, and I have used OOP programming (therefore the gcc compiler won't be sufficient) and PSO for optimization.

Results

This project has greatly expanded the field of Artificial Intelligence, with this much more modular and isolated AI engine. There are no hints given to the agents in this simulation, they must discover and solve their task themselves, with only their communications to guide them, which also get distorted. Essentially, by increasing the amount of communication, and decreasing the amount of intelligence of each agent, I have made a much more realistic search optimization, crowd dynamics AI engine.

Since time is a factor in the efficiency computations, all of these results are standardized through a rate-restricting class I wrote called rlimit. This down limits the simulations speed so that regardless of what computer it's run on the simulation will run at the same speed. If an iteration of the simulation runs too fast, the library will add a customized delay (accurate to +/-5 milliseconds) to regulate the speed. If more than .5\% of the runs take more than the nominal iteration length, it will discard other statistical output.

With this complex AI system, there were many different variables which had significant results on the simulations efficiency, allowing for customization depending on circumstance. The variables I modified were: sightline distance and communication clarity. As it should be expected, when the sightline distance was decreased or the communication clarity decreases the efficiency of the simulation decreased, while when they increased, the simulation's efficiency increased.



Two Charts Illustrating the relation between the seek efficiency and the sightline distance (left) and the communication efficiency (right).