TJHSST Computer Systems Lab Senior Research Project Paper Crowd Dynamics Artificial Intelligence with Particle Swarm Optimization 2006-2007

Keith Ainsworth

January 23, 2008

Abstract

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 plan on creating an articial intelligence engine, which works by having multiple agents, each with highly limitted 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, using particle swarm optimization to optimize the calculations.
Keywords: AI, PSO, C++, OOP, SDL

1 Introduction

The AI engine I'm programming is implemented through C++'s object orientation. I have programmed several classes which interact in order to make a completed end project. As my engine is an agent based networking engine, naturally the first two classes are the agent, Person, and network, PNetwork classes. The main program must include an array of Persons, which is passed to the PNetwork class on instantiation. Then the main program only needs to talk to the PNetwork, as its managing the list of people from instantiation on, and will take care of the movement of the Persons.

The Person and PNetwork also utilize another class I've written, the weightlist class. This class is a set of two array based, fixed size, looping lists; one for data, the other for the data's relative weighting. The important feature of the weightlist that other prewritten container classes don't offer is a summation function. This function effectively averages the data list, based on the relative weightings, and a decay weighting that favors the more recent entries in the list. This is crucial because the communication aspect of the PNetwork has to have a way of keeping track of each Person's communications. Therefore each Person in the PNetwork is assigned a weightlist.

2 Background

Artificial Intelligence programming always requires a task at hand for functionality and relative significance. In my implementation, the initial task for the agents in my network based artificial intelligence engine will be target detection and convergence. The agents will have to locate and converge upon a random target (eventually to become a user controlled target) using methods streamlined with particle swarm optimization methods. Additionally, as they attempt to accomplish their motives, they should become increasingly more efficient at it. This would have to be as a result of the optimizations found in the paper by Kennedy and Eberhart.

3 Procedures and Methods

3.1 Overview

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'll need C++ (and therefore the g++ compiler) along with the SDL (software digital layer) libraries, for keyboard input and graphical output, and I'll be using OOP programming (therefore the gcc compiler won't be sufficient) and PSO for optimization.

The way the engine works is by: instantiate a PNetwork with a list of Persons, and run the PNetwork in a non-terminating loop (except by exit). The PNetwork class has a step() method in which the communicator methods from each Person in its list are called. The communicator methods simply return whatever the Person can see that he wishes to inform the others about. Then those messages are added into weightlists associated with the other people in the list. When everyone has communicated what they have to communicate, the PNetwork calls the summation function on the weightlists and instructs the Persons the summation is relevant to, to head in that direction. That simple process can repeat endlessly, with obvious variations in the behavior created through different communicator methods (each Person only has a pointer to a function of the proper parameters and return type, it can be defined on instantiation to be whatever the programmer wishes) and different communication distances and message resilience (currently they decay in accuracy as a function of the distance they travel).

3.2 Testing

I've programmed this project so far with several debugging features and text based outputs for constant error checking. While for the final project these would be commented out for the final compilation, I plan to continue programming with those features to allow for ease of code writing and testing. Right now I'm using a series of testing shells to assess the resilience of my AI system. I have shells which print out pixels for each agent in the simulation, using SDL, which work for the regular PNetwork and ngon world classes. These automated tests inform me whether the program is doing what it should be.

Eventually I will apply this AI engine to a game I've already programmed in Java. Using this game with enemy AI, there will be a very clear visual display as to whether or not the engine is effective and fast. If the game runs smoothly and is challenging, I'd consider it a success. That will be the final testing.

4 Expected Results

This project will greatly expand 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 will make a much more realistic search optimization, crowd dynamics AI engine.

This project requires plenty of additional research however. I could program the engine simply as written above, but my real intent is to create such an engine that can be used in real time environments, such as but not limitted to, game environments. By implementing particle swarm optimization, which in and of itself will require much additional research, I should be able to accomplish the desired feat of real time run speed.

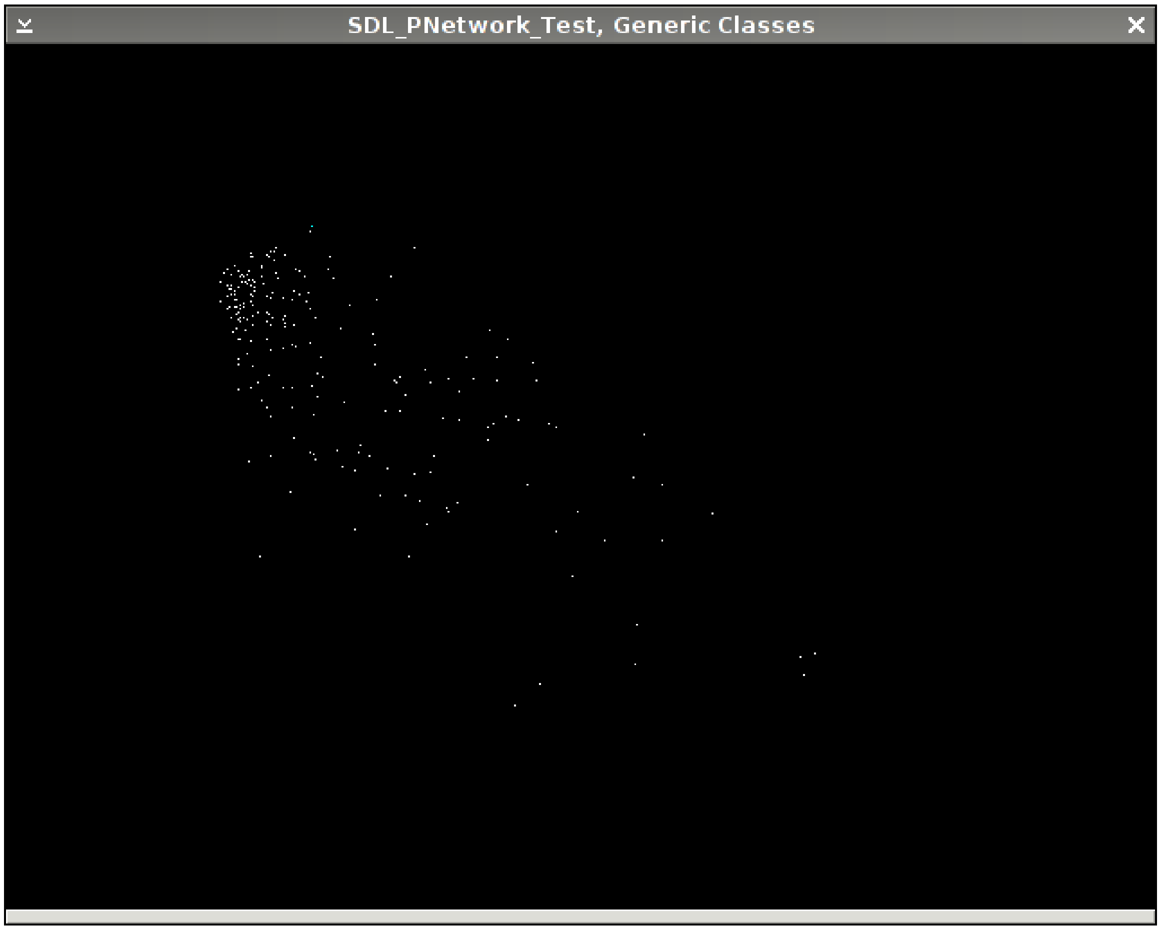

Figure 1: This is the simulation tester shell running

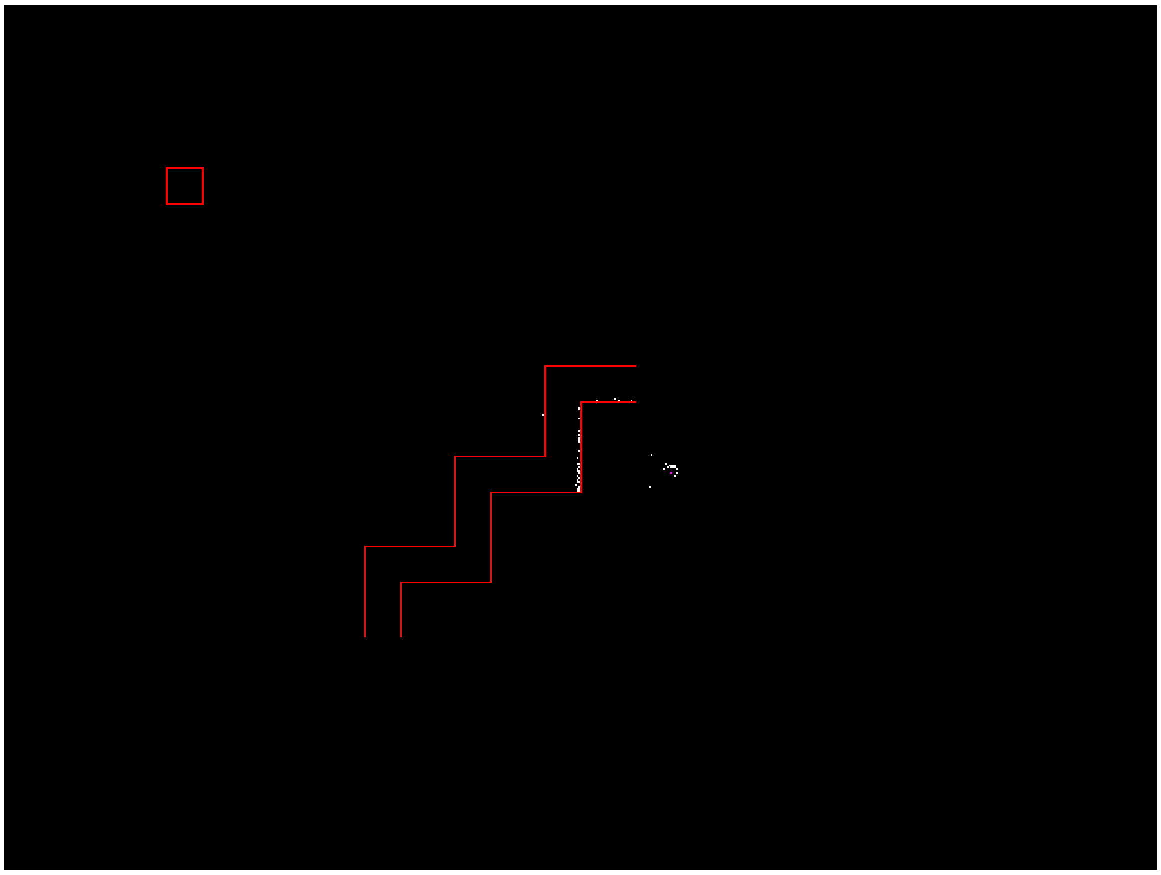

Figure 2: This is the ngon world optimized simulation tester shell.