CPSC 565 Term Project Conference-style Paper

Neil Tallim

December 2, 2008

Contents

Part I

Summary

1 What this project was supposed to do

In addition to, naturally, being an attempt at securing a passing grade in this course to make graduation a possibility, this project was conceived of as a means of blending an introduction to emergent concepts with a fun idea that is accessible to almost everyone: ants and conquest.

My vision was to build not just an engine that would allow for those of us with overactive imaginations to make their sugar-high-induced daydreams of ants with swords a virtual reality (yes, I did mentally picture that while trying to come up with an idea for this project, and it was too hilarious to pass up), but to do so in a manner that would allow for colonies to evolve like empires, all without violating the basic curiosity of emergence: the actions and decisions of individuals move the collective whole. And, because these individuals were ants, I had to find a way to do it that allowed them to have no memory of what they did before, while dealing with having very little in the way of communications infrastructure; needless to say, I found this to be a very fascinating objective.

2 What has been prepared

This project has resulted in a system that makes use of a custom simulation engine and uses breve as a canvas. Existing as a Python 2.3-based implementation of the specifications found in Neil Tallim's "CPSC 565 Term Project Development Guide", it offers the following features:

- Agent reproduction
- Agent death
- Colony expansion
- Decision-making
- Memory-free behaviour logic
- Memory-free pathfinding
- Resource competition
- Stateful environments
- Survival logic

- Swarm logic
- Task-switching

For an in-depth look at how these features work, please consult the implementation's source code (published under the GPLv2), or see the afore-mentioned "CPSC 565 Term Project Development Guide" (provided as public domain material). Note: Because all of the math upon which this project is based is presented in the Development Guide, it will not be reiterated here in any form, which will cause this paper to be comparatively shorter than that narrow-margined, 24-page behemoth, to say nothing of the size of the code or its accompanying documentation.

For ideas about what can be done with this system (it's far, **far** more flexible than this feature list makes it out to be), see Section 8 on page 7.

3 What the system looks like

The system actually holds pretty true to the first concept sketch that was prepared:

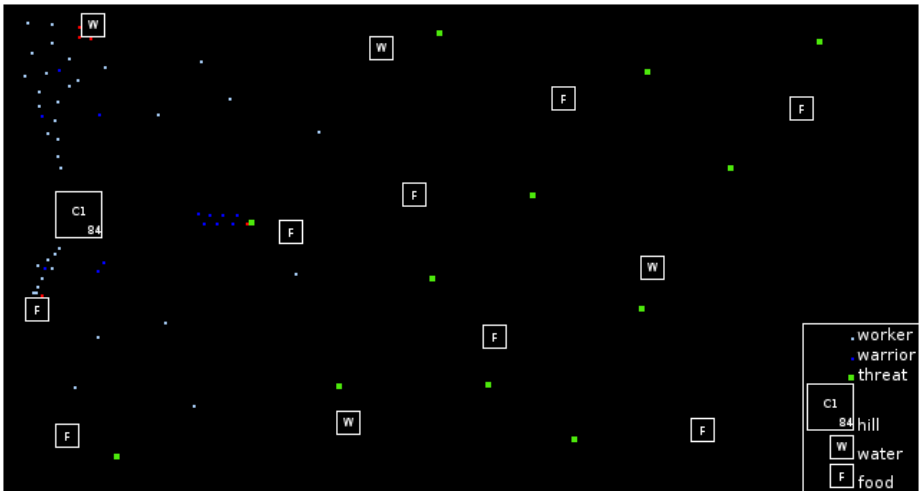

Figure 1: Early conceptual rendering of the system

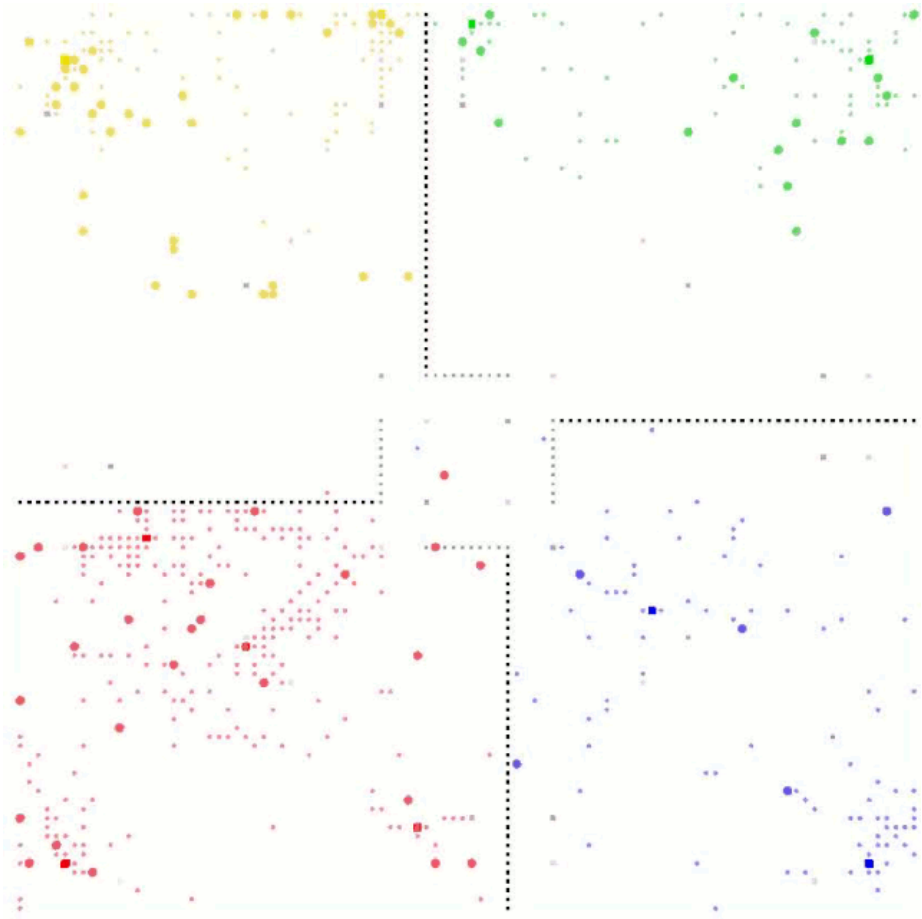

Figure 2: Screenshot of the delivered system

In the latter diagram, the small circles are workers, the larger circles are warriors, the coloured squares are hills, and the grey squares are food and water.

The fullness of each resource is indicated by transparency: those that are mostly drained will be nearly invisible, while those that are full will be opaque.

Pictured between each colony (colour)'s zones are walls (the black squares) and sponges (the grey squares). These offer this particular simulation a way of preventing contact between colonies until they've had time to mature.

4 What the system looks like in action

A video from which the above screenshot was taken was provided with this document (please do not redistribute this paper without making the video available).

In it, you will see a number (but certainly not all) of the behaviours this system has to offer as the four colonies grow, die, and grow again.

Unfortunately, however, the video is based on a slightly broken version of the code: agents were not able to properly detect pheromones that were within their base sense radius because of a mistake during optimization (this has, of course, been corrected); additionally, once red has gathered enough resources to try growing again after dying, it produces architects instead of workers and warriors because of a typographical error (this, too, was fixed). Neither of these issues lessens the sense of scale presented by the video, so it is still considered a valuable reference.

Part II

What changed during implementation

5 Behaviour and design

A number of little tweaks and behavioural enhancements were made during implementation because a cool idea popped into my head or I realized that my original specs were too limited in some way. Among these alterations are the following highlights:

- Stochastic workers no longer avoid pheromone trails. Rather, they just pretend that they do not exist. The net effect is similar – they will discover untouched resources – but they will take less time to process and they're easier to maintain, code-wise.
- As any agent appears on the map, either through creation or by being dispatched from a hill, their orientation will be randomized; this will result in much more uniform dispersion patterns than always having them face in their previous direction (or, in the original implication, facing up, which had an ugly fountain effect that was very quickly altered)
- Things were gradually unified under a class hierarchy to make operations cleaner and faster. While not really significant to the functionality of the system, it does make it very easy to define new agents and objects with the expectation of having them "just work", which may be very useful to other developers.

6 Optimizations

Naturally, the math that was prepared to serve as the basis of this system was written in a manner that assumed instantaneous evaluation in a parallel manner. Because computers exist outside of the realm of theory, in the land of reality, it was necessary to change some things that were stated in the original specs to faster alternatives that sacrificed some precision for better performance.

First and foremost among the changes is the manner in which pheromones are handled. While, previously, they were defined as gasses that would drift over time (with compounding effects and choking properties in confined spaces), they were reimplemented as singular entities that worked based on a modified inverse-square principle: if the source was beyond the agent's sensing range, the radius used in calculations would be modified to subtract the sensing range; otherwise, it would be intensified on a linear scale through reciprocal division. This sounds like a major difference and, indeed, it does change some things dramatically – sensing pheromones around walls, for example – but it produces comparable results for the most part, and it is so much faster that the decision to sacrifice precision isn't even regrettable.

While not really an optimization because it was not formally defined before, the spaces-in-range look-up functions underwent a ton of revisions; the changes in algorithms used could probably be used as a small paper in a complexity theory course. Suffice it to say that I am still not satisfied, but I think I got it down to $4n(n-1)$. Because that's still pretty high, given that there are additional checks and shadow-mapping techniques required on top of that, when it seems like it would be more efficient to do so, rather than look at all nearby spaces to find out if they contain anything interesting, all objects are asked if they're near enough to be interesting instead; at least the distance algorithm works in constant time, so it's just n – albeit, a much larger n in many cases.

Affected by the previous optimization, however, is agent interaction. Because agents can speak directly, they will sometimes have access to knowledge that's newer than what those that came earlier in the tick cycle will have been privy to. Previously, this issue was addressed by having all agents work off of the previous tick's environment-state, but that's no longer possible (though the speedups easily offset that regret). To deal with this, the turn order of each agent is uniformly randomized each tick, which should ultimately minimize the negative effects of this shortcut. Threats, not needing to know what their peers are doing, are not affected by this, and do not need to be randomized.

Part III

The value of this project

7 Research value

This system only loosely models the behaviour of ants, making it, in its current form, unusable for simulating anything that actually happens in nature. However, it does provide an architecture that could be retooled to cover the basics of Deborah Gordon's harvester ant research if a sufficiently interested individual feels like taking on that task.

8 Academic value

As a curiosity, this system could be the basis of interesting evolutionary algorithm work, with the addition of some simple statistics-gathering and reporting functions. Tuning the behaviour of each individual class of agent or dividing a field with walls and growing multiple colonies in the same situations in parallel could provide for interesting projects in finding patterns of equilibrium and different ways it can come about. Classes can also be easily disabled, so if someone just happens to like the framework I've built and wants to use it to model gathering patterns, warriors, architects, builders, and reproduction itself can be entirely disabled; alternatively, if someone wants to make a survival scenario in which they try to find out how long workers in a maze can survive being hunted by a couple of predators, the code to support that is already done – they just need to set the initial state and pheromone parameters (and, perhaps, write a script that designates an area as a safe zone by placing pheromones to lure the workers before sealing the corridor with walls and opening a new path for those who made it in time).

Additionally, this project may provide a future Python-familiar developer with some ideas about how things can be handled in breve to more readily produce entirely different projects without having to look for the same resources I had to find.

Part IV

Conclusion

Overall, I feel that this project has been highly worthwhile. It is incredibly rare that I will actually have fun while working on anything for school, yet the

freedom to explore afforded by this class, coupled with the interesting scope it asked us to examine, which led to the creation of the idea that resulted in the system presented to you today, has kept me interested for an entire semester.

While the scale of this idea may have been excessively massive for an undergrad term project, I feel I have delivered on the promises I made at the onset of this course. My only lamentation is that the code is not nearly as optimized as it would have been had I not lost a week of planned work time, which causes it to run far too slowly to handle complex scenarios on modern desktops; small-scale simulations, however, are very snappy, often running in near-real-time.

As a final statement, I'd just like to say thank you for taking the time to follow along with this project, and I hope you find a use for it in the future.