CPSC 565: Emergent Computing Winter 2026

Christian Jacob
Dept. of Computer Science
University of Calgary

Assignment 3: Antymology

Due Date: Wednesday, February 11, 2026.

As we've seen in class, ants exhibit very interesting behaviour. From finding the shortest path to building bridges out of bodies ants have evolved to produce complex emergents from very simple rules. For your assignment you will need to create a species of ant which is capable of generating the biggest nest possible.

The base code for the assignment has been provided by Cooper Davies. Currently the simulation environment is devoid of any dynamic behaviour and exists only as a landscape. You will need to extend the functionality of what has been written in order to produce "intelligent" behaviour. Absolutely no behaviour has been added to this project, so you are free to implement whatever you want however you want, with only a few stipulations.

Goal

The only goal you have is to implement some sort of evolutionary algorithm which maximises nest production. You are in complete control over how your ants breed, make choices, and interact with the environment. Because of this, your mark is primarily going to be reflective of how much effort it looks like you put into this vs. how well your agents maximise their fitness (I.e. don't worry about having your ants perform exceptionally well).

Current Code

The code is currently broken into 4 components (found within the components folder)
1. Agents
2. Configuration
3. Terrain
4. UI

You are able to experience it generating an environment by simply running the project once you have loaded it into unity (make sure SampleScene is loaded). Feel free to modify existing files to your needs

Agents

The agent's component is currently empty, this is where you will place most of your code. The component will be responsible for moving ants, digging, making nests, etc. You will need to come up with a system for how ants interact within the world, as well as how you will be maximising their fitness (see ant behaviour).

Configuration
This is the component responsible for configuring the system. For example, currently there exists a file called ConfigurationManager which holds the values responsible for world generation such as the dimensions of the world, and the seed used in the RNG. As you build parameters into your system, you will need to place your necessary configuration components in here.

Terrain
The terrain memory, generation, and display all take place in the terrain component. The main WorldManager is responsible for generating everything.

UI
This is where all UI components will go. Currently only a fly camera, and a camera-controlled map editor are present here.

Requirements

Admin
- This assignment must be implemented using Unity 6000.3.* (minor patches aren't relevant)
- Your code must be maintained in a GitHub (or other similar git environment such as UofC GitLab) repository.
- You must fork from the provided repo [1] to start your project.
- All project documentation should be provided via a Readme.md file found in your repo. Write it as if I was an employer who wanted to see a portfolio of your work. By that I mean write it as if I have no idea what the project is. Describe it in detail. Include images/gifs. (make sure to double check on a different computer to ensure images work properly)

Interface
- The camera must be usable in play-mode so as to allow the grader the ability to look at what is happening in the scene.
- You must create a basic UI which shows the current number of nest blocks in the world

Ant Behaviour
- Ants must have some measure of health. When an ant's health hits 0, it dies and needs to be removed from the simulation.
- Every timestep, you must reduce each ant's health by some fixed amount
- Ants can refill their health by consuming Mulch blocks. To consume a mulch block, and ant must be directly on top of a mulch block. After consuming, the mulch block must be removed from the world.
- Ants cannot consume mulch if another ant is also on the same mulch block
- When moving from one block to another, ants are not allowed to move to a block that is greater than 2 units in height difference
- Ants are able to dig up parts of the world. To dig up some of the world, an ant must be directly on top of the block. After digging, the block is removed from the map and the ant should move to the lower block.

- Ants cannot dig up a block of type ContainerBlock
- Ants standing on an AcidicBlock will have the rate at which their health decreases multiplied by 2.
- Ants may give some of their health to other ants occupying the same space (must be a zero-sum exchange)
- Among your ants must exists a singular queen ant who is responsible for producing nest blocks
- Producing a single nest block must cost the queen 1/3rd of her maximum health.
- The queen ant must look distinctly different than the normal ants
- No new ants can be created during each evaluation phase (you are allowed to create as many ants as you need for each new generation though).

Tips

Initially you should first come up with some mechanism which each ant uses to interact with the environment. For the beginning phases your ants should behave completely randomly, at least until you have gotten it so that your ants don't break the pre-defined behaviour above.

Once you have the interaction mechanism nailed down, begin thinking about how you will get your ants to change over time. One approach might be to use a neural network to dictate ant behaviour. The WorldController can be modified to allow multiple 'worlds' to run in parallel, this may necessitate making some other object such as a 'SimulationController' to manage them.

https://youtu.be/zIkBYwdkuTk

another approach might be to use pheromone deposits (I've commented how you could achieve this in the code for the AirBlock) and have your genes be what action should be taken for different pheromone concentrations, etc.

Submission

Export your project as a Unity package file. Submit your Unity package file and additional document using the D2L system under the corresponding entry in Assessments/Dropbox. Include in the message a link to your git repo where you did your work.

References:

[1] GitHub - DaviesCooper/Antymology

Rubric:

A +	All requirements for an A as well as: • Evolutionary Behaviour is evident • Documentation is complete and shows thorough understanding of topic and method chosen
A	• All requirements implemented as described • Documentation and analysis are complete and describes method chosen • Simulation runs as expected
A -	• One or two minor errors in implementation, or • Documentation lacks depth
B+	• One or two minor errors in implementation, and documentation is lacking, or • A significant implementation error but no other problems
B	• A significant implementation error • One or two minor issues **or** docs or analysis lacking
B-	• A significant implementation error • one or two minor issues **and** docs or analysis lacking.
C+	• A little better than a C
C	• Multiple errors of note, documentation, or analysis completely inadequate or missing, but an effort was made
C-	• Multiple errors of note, documentation, or analysis completely inadequate or missing. • Code compiles but nothing of merit is produced
D	• Solution does not compile. *IMPORTANT NOTE: If your program is otherwise amazing after fixing compiling errors it does not matter. The T.A. is instructed not to fix compile errors!*
F	• Submission missing or non-functional