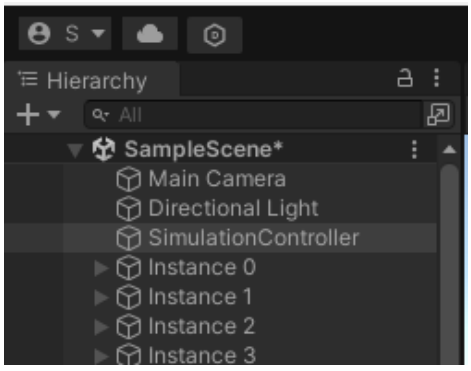
Introduction

This program aims to evolve programs that control ants to maximize nestBlock production. Each of the worlds in a generation contains a number of worker ants and a single queen, worker ants all share the same program and the queen has a unique program. Only the queen can produce nest blocks at the expense of her health, but workers can give her their health, can they evolve to make big nests?

Usage Instructions

Warning: When first launching program it WILL take a while depending on values of 'Selection_pool_size' and 'Selected_offspring_count'. Larger values will take longer to load since the world generation is NOT optimized. To speed this up at the cost of no graphics check 'Simulation Only' in the SimulationController.



The SimulationController is the object that controls everything and contains the most important controls, make sure to select it to view it in the inspector.

The ConfigurationManager component contains values that should only be changed at the very beginning of the program before running the simulation. Some variables of note are:

1. Number_of_ants: How many ants will be created in each world instance
2. Selection_pool_size: After a generation has died how many of the best worlds will be selected to reproduce
3. Selected_offspring_size: For each selected world how many offspring will it produce
4. Simulation_only: Enabling this will disable ALL graphics (mesh creation, rendering, etc) and override any time step delay to 0. I encourage using this to rapidly run the simulation and let evolution happen.

The Simulation manager contains most of the readouts of the simulation as well as runtime simulation options.

1. Dt: The minimum time that should pass between simulation steps
2. Run: Check to run the simulation
3. Step: Check to run one step of the simulation. Unchecks itself after the step is complete

Controls

WASD = camera movement

Middle Mouse + Drag = Camera Rotate

Viewing stats

Stats about the simulation can be viewed in the SimulationManager component.

1. Disp_TotalAntCount: total ants across all worlds in the simulation
2. Disp_WorldAntCounts: Per world count of ants
3. Disp_WorldNestBlocks: Per world count of nest blocks

You can also view an individual ant but it is a bit more weird. I recommend making sure 'run' is unchecked before doing this. In the Hierarchy panel you can go to an instance, select ants then click on an ant to view it in the inspector. You can then click on 2 checkboxes to view info about the ant. NOTE: The first ant is ALWAYS the queen

1. Print Brain: Prints the AST the ant is using
2. Print Mutation History: Prints the mutation that the AST has undergone since the start of the program

NOTE: These print in the Console of the editor

Simulation

The simulation uses a Abstract Syntax Tree (AST) with basic types and type casting to create dynamic programs for ants to follow. Each generation consists of (Selected_offspring_size + 1) * Selection_pool_size worlds each having its own ASTs (one for workers, one for queens), the simulation then runs until all ants are dead. When all the ants die Selection_pool_size worlds are chosen, they then produce Selected_offspring_size children where each child has a single mutation applied to it. These children and the original parent are then simulated again and repeat as long as you want.

Ants

Ants have a brain that defines instructions for them to follow. These instructions can be viewed in the instruction tables at the bottom. Ants also have energy that when it reaches 0 will cause them to die. More details about specific actions are in the table

The queen ant has a light on it and is colored red for easy identification

Blocks

The world is made up of different blocks

1. ContainerBlock: Cant be mined by ants, defines boundaries of the world
2. MulchBlock: Can be consumed by ants to regain energy back to max
3. GrassBlock, StoneBlock: Can be dug by ants
4. NestBlock: Can only be produced by a Queen for 1/3 of max health. Can be dug by other ants

Pheromones

Pheromones are a substance that can be created and sensed by ants. There are 16 types of pheromones in the world and pheromones only work on a 2D grid on the XZ plane, this reduces computation costs. Ants can sense pheromone concentrations either at their location or at one of the 8 blocks around them

Abstract Syntax Tree (AST)

The AST is a logical tree structure that is used a lot in compilers to parse program code. This AST is a modification since it must operate as a program and not just a logical parser. The AST has a type system with 4 data types defined, however under the hood everything is a byte with different interpretations which allows for great flexibility and a basic type casting system. Nodes can specify return types by implementing certain interfaces which define what the user needs to add for logic. I designed this to be very modular and extendable because I hate myself for some reason and can't let an assignment be too easy…

Types

NONE

BOOL
HBYTE
BYTE

All of these are technically bytes under the hood allowing for possible casting between types if desired. Nodes can have returns or slots compatible with multiple types

Node Properties

IBoolReturn: Returns a bool

IByteReturn: Returns a Byte

IHByteReturn: Returns a Half Byte

IAction: This node will cause an action to be triggered, this causes the evaluator to return to the calling agent.

IValue: This node is a constant value

Nodes

These node types are abstract classes and the user must subclass them and implement their desired functionality. The types are required by the ASTEvaluator and mutators.

1. Internal_node: These nodes have children and must define what return type they expect their children to have
2. Leaf_node: These nodes have no children

ASTEvaluator

The AST evaluator is what agents use to iterate through the AST and get the next action node. The evaluator will keep evaluating the AST and only return once an action has been found. This allows for complex logic to be used at no cost of time (or energy) to the agent. My hope was that ants would be able to evolve the ability to make smart decisions such as only transferring energy when the queen is on the same block as them or detecting gradients of morphogens.

Mutations

The AST currently has 4 mutations it supports

1. Expand: Takes a leaf node and replaces it with an internal node with leaf children. If a sequence node is chosen the new node is added as a new child in the sequence
2. Shrink: Replaces an internal node with a leaf node
3. Hoist: Chooses a child of an internal node and replaces itself with the child. Other children are removed
4. Point: Finds a node on the AST and replaces it with a different node of the same child count and returns and is compatible with the parents expected child return.

Expand and shrink seem to be the most commonly used mutations, likely because of their simplicity and high likelihood of success. However which is chosen is always random

Advice

1. Before starting the program make sure to turn on Simulation_Only as while the visuals are nice they aren't really needed until evolution has occurred.
2. If you are not using Simulation_Only mode make sure to reduce the selection pool values otherwise your computer will chug hard!
3. Simulation_only Is the ONLY variable you can change in Configuration Manager once you hit run or step for the first time. Changing any values after this point may cause a crash!!

AST Default node implementations

[x] means “Value returned by the child x”

Name	Type	Properties	Child Types	Description
Constant bool	Leaf	IBoolReturn IValue		Returns a constant bool value
Constant byte	Leaf	IByteReturn IValue		Returns a constant byte value
Constant hbyte	Leaf	IHByteReturn IValue		Returns a constant hbyte value
NOOP	Leaf	IAction		Action node that will yield the evaluator but no action will be performed
Sequence	Internal		NONE* *Sequence can have n children, not just 1	Contains n children nodes that will be executed in first->last order
IfElse	Internal		1. BOOL | HBYTE | BYTE 2. NONE 3. NONE	Branching statement. Depending on [child 1] only one of the 2 branches will be performed
And	Internal	IBoolReturn	1. BOOL | HBYTE | BYTE 2. BOOL | HBYTE | BYTE	Logical AND
OR	Internal	IBoolReturn	1. BOOL | HBYTE | BYTE 2. BOOL | HBYTE | BYTE	Logical OR
NOT	Internal	IBoolReturn	1. BOOL | HBYTE | BYTE 2. BOOL | HBYTE | BYTE	Logical NOT
GT	Internal	IBoolReturn	1. HBYTE | BYTE 2. HBYTE | BYTE	Mathematical > operator

Ant node implementations

Name	Type	Properties	Child Types	Description
AntHealth	Leaf	IByteReturn		Returns the current health of the ant
AntsHere	Leaf	IByteReturn		Returns how many ants are on the same block as the ant
MoveForward	Leaf	IAction		Moves the ant forward 1 space if the height difference is <= 2
TurnRight	Leaf	IAction		Turns the ant right
TurnLeft	Leaf	IAction		Turns the ant left
Consume	Leaf	IAction		If there are no other ants on the same block as the ant and the block below is mulch, remove the block and regain all energy
Dig	Leaf	IAction		Remove the block below the ant
TransferEnergy	Internal	IAction	1. BYTE	Give each ant on the same block as this ant [child 1]/#ants energy. Remove that much energy from this ant
GetValue	Internal	IBoolReturn IHByteReturn IByteReturn	1. HBYTE	Get the value stored in memory at the location of child 1s value
SetValue	Internal	IAction	1. HBYTE 2. BOOL | HBYTE | BYTE	Set the value stored in memory at the location of child 1s value to the value of child 2.
DepositPheromoneArond	Internal	IAction	1. HBYTE 2. HBYTE	Deposit pheromone of the type given by child 1 at the ants location
SensePheromoneHere	Leaf	IByteReturn	1. HByte	Return the amount of the pheromone type specified by [child 1] at the ants location
SensePheromoneAround	Internal	IByteReturn	2. HBYTE 3. HBYTE	Return the amount of the pheromone type specified by [child 1] at a location around the ant specified

				by [child 2]. See **Pheromones** for more details
SenseBlockBelow	Leaf	IHByteReturn		Returns a value corresponding to the block type below
SenseBlockAhead	Leaf	IHbyteReturn		Returns a value corresponding to the block type ahead
QueenHere*	Leaf	HBoolReturn		Returns true if the queen is on the same block as this ant
CreateNest**	Leaf	IAction		Creates a nest block at the ants location. Decreases health by 1/3 max value

* Specific to worker ants
**Specific to queen ants