AT PROJECT on Natural Selection:

-Debaditya Bhattacharya (190254)

Project Description:

Language: C++

The project is to create a simulation that will show natural selection, enabled through genetic drift. How different traits help in survival and how natural selection chooses these organisms which have traits which have enabled them to thrive (Survive and have more progeny). Genetic variation occurs through mainly mutation and sexual interactions and choose each other based on attributes.

Attributes are functions of traits and other attributes, and traits are intrinsic properties of the organism that is determined by the genetic makeup. Example: Speed is an attribute which depends on traits like: feet size, body size, enzyme efficiency. Another example is: Fighting skill: intelligence, body size, feet size, enzyme efficiency, vision, etc. (Intelligence is used in judgement before fighting, and while mates). Yet to completely determine a list of attributes and traits, but they come up automatically.

At the end of the simulation we should be able to observe a general drift towards genes that are more favourable for survival, and this is how natural selection works. Organisms which have more favourable genes are expected to be seen more in number.

The simulation is successful if it is able to show the existence of natural selection and show give us an insight into how species interact with organisms of their own species and with the environment. If this is successful, then it can be expanded to accommodate higher levels of complexity which involves more species and different kinds of environments, and hopefully might be able to show evolution.

New Learnings:

Object Oriented Programming

Typedefs

C++ syntax (vectors)

How to organize a project

How to think (and how not to think)

A lot about natural selection

Gene Class:

Variables:

1) Identification to trait
2) Value of trait
3) Dominance (larger dominance = dominant trait)

Methods:

1) Randomize value

2) Randomize dominance

Genetic Material Class:

Variables:

1) A collection of Gene Classes.

Genome Class:

Variables:

1) A tuple of Genetic Material.

Methods:

1) Meiosis function that will take two genetic material classes and return a randomly mixed haploid genetic material.

Position Class:

Variables:

1) XPos
2) YPos
3) Heading Direction

Methods:

1) Update pos;
2) Update direction;

Organism Class:

Variables:

1) Genome
2) Energy
3) Food Storage
4) Health
5) Attributes
6) Age
7) Position
8) Enzyme Efficiency

Methods:

1) Vision & Judgement – Look at its surroundings and take appropriate actions. If food, head towards food. If mate, head towards mate, if male, either intimidate, run or ignore. Judgement to be made based on present health and energy levels. (incase of multiple elements within reach closest one is given priority).
2) Death – If health goes down, then they die.
3) Consume food – Based on steps taken and enzyme efficiency; if no food, decrease health.
4) Move
5) Bodyily repair – If they have consumed food and haven't fought in a while, increase health.

6) Update health.

Food Class:

Variables:

1) Amount
2) Type
3) Position

Functions:

1) Update amount

World Class:

Variables:

1) Vector of Organisms
2) Vector of Food
3) Size of world (in coordinates) (assume world to be walled in)

Methods:

1) Distance
2) Interactions
3) Update Organisms
4) Food availability changes, food type changes
5) Reproduce
6) Spawn food or organism

Interaction Functions:

Species and Food:

Sight Causes update in direction.

When interact with food, use attributes to consume food.

Store food

If organism within distance, trigger fight.

Species and Species(Sex):

Males – Males:

Males tend to run towards things which it might seem to think that it can kill and they think they have similar enough genes.

Males may avoid fights based on intelligence and power

Call Fight function if their genes are similar enough.

Males – Females:

Males will purse females based on Attributes

Females will also choose based on Attributes of males.

Call Reproduction function.

Main Function:

1) Set world size
2) Set food size
3) Set initial organism count
4) Spawn world; food and organisms spawned at random locations within the boundary. Spawn world based on a single seed. (use a pseudo random generator to generate random numbers for all other purposes) (probably will use Linear congruential generator)(this so that I can re-run simulation and still expect the same result)
5) Call a global update function.

Global Update Function:

1) Calculate distances
2) Call vision method in organisms
3) Calculate interactable distances and call interaction functions
4) Run repair function for organisms
5) Run death function on organisms
6) Run random fluctuation on food (pseudo random) (not sure if I want to do)
7) Update positions of organisms, decrease energy / health.
8) Export data (Data to include instances of world, and all containing data especially the genetic material of the creatures.

Output:

Due to the nature of the data output, I have to invest some time in writing code to plot graphs of the output of the code. This is a potentially feasible way to figure out if I am on the right track.

Plan of action:

Write basic code for all classes & Get a list of attributes and traits made. -> done by 6th March

Write the functions. -> done by 10th March

Test the functions on a small scale. -> Done by 13th march

Write the export code. & Plot the export code. -> done by 18th march

Run many simulations and debug till -24th March