**solution identification: the solution: natural selection simulation**

**how is this solvable computationally?**

The breakdown of the solution computationally can be viewed here: https://lucid.app/lucidchart/739eb9ba-7366-45a5-ab4a-4cb1512891e8/edit?viewport\_loc=18%2C148%2C2219%2C1065%2C0\_0&invitationId=inv\_d36397bb-83fc-47f3-b007-08c4abb60d0b#

Table of Contents

[Its solvable computationally due to 1](#_Toc130299445)

[Libraries 2](#_Toc130299446)

[Why did I choose pygame? 2](#_Toc130299447)

[why did I choose python? 2](#_Toc130299448)

[Stakeholders 2](#_Toc130299449)

[Interviews 3](#_Toc130299450)

[Questions 3](#_Toc130299451)

[Interview responses 3](#_Toc130299452)

[Limitations 4](#_Toc130299453)

[Essential features 4](#_Toc130299454)

[Desired 6](#_Toc130299455)

[market research 7](#_Toc130299456)

[other solution 1: (natural\_selection written in python by owen\_davis\_bower) 7](#_Toc130299457)

[Other solution 2: species artificial life real evolution 8](#_Toc130299458)

[Hardware and software requirements 10](#_Toc130299459)

[External dependencies (libraries) 10](#_Toc130299460)

# Its solvable computationally due to

Problem recognition:

An example of this would be the vision cone, I knew I had to not only find a way to find coordinates inside a triangle efficiently but also had to find a reliable algorithm to pivot the cone around as the creatures turns.

Problem decomposition:

After researching I realised each problem can be decomposed such as this

Rotation:

A function to convert relative radians to degrees essentially finding a creatures relative angle to food:

A virtual triangle is made

Cosine rule is applied to find the angle of the triangle facing the food

The radian offset is applied

A way of converting the primitive angle to classic degrees:

Experimented and found that I could just multiply it by 0.1784

Then to stop it going over 360\* a simple clamp

A function to find the relative angle around a point

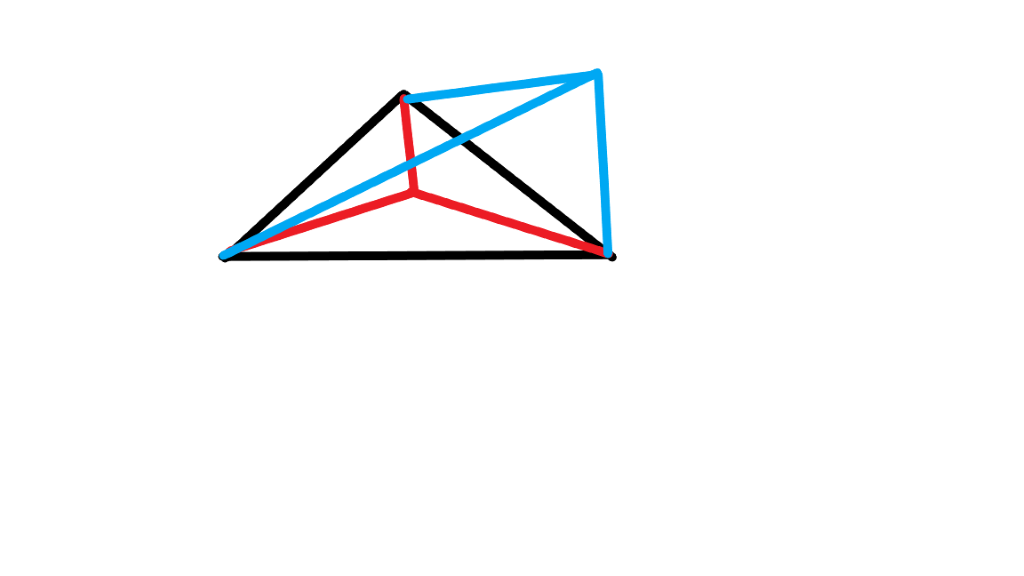
Using a tangent

Math is applied and the cos and sin of the angle is applied to find the ends of the tangent

The middle of the tangent is the coordinate

Coordinates inside a triangle:

By logic, the area of triangles inside a triangle should equal the area of the triangle unless its outside the triangle



Above shows an example of how this works

The area of sub triangles are calculated and added up

Within a margin of error the areas of the 3 sub triangles are compared to the host triangle

Abstraction:

Abstraction **has** to be applied to a problem such as this one, the amount of detail required to accurately simulate real life would definitely not sustainable and by abstracting non affecting/less affecting factors it allows me to maintain efficiency and readability of the final product

# Libraries

## Why did I choose pygame?

to show the simulation being run (can be toggled) the pygame api is rather primitive however highly versatile as I’ve demonstrated with my custom made graphics objects for the project including checkboxes, sliders, colour selection medium etc.

I much prefer pygame to tkinter due to it being comparably as primitive as openGL which I have had experience with in the past however due to pythons syntax I can work a lot faster on pygame.

## why did I choose python?

Many of the complicated backbone algorithms such as, vision and genes can be written using the language python as the library {math} contains the mathematics functions I require to do trigonometry and cosine rule + also contains a neural networking module that can be installed using pip (TensorFlow and NumPy).

Python also supports XML and JSON both of which I will be using to store the simulation metrics.

I evaluated my options from the programming languages I am most used too (python, c++) and python seemed the most viable option due to c++’s hard to understand syntax from an examiners standpoint and its rather annoying implementation of arrays and datatypes. openGL is also a bit more complex than pygame and my working speed is better using pygame.

# Stakeholders

Main client:

My stakeholders include any individuals of groups with an interest in biology and or natural selection in general, it aims to educate people on not only the process of natural selection but also the invisible biological processes behind it such as genes and how they are handled and synthesized into amino acids, mutations and alleles.

How does the project fit the stakeholders needs?

The project provides both a visual GUI and a CLI to analyse the process of natural selection including but not limited too:

Species relation diagrams

Real gene and amino acid storage

Enzyme binding

Receptor binding (and the effect of the released proteins)

Mutations

Alleles (more specifically favourable alleles)

The product is also tailored towards the client as it includes features to perform their desired use cases and is presented neatly and clearly.

# Interviews

## Questions

* Would a visual example of natural selection help you understand biology?
* Who do you think the product should be tailored towards?
* What do you think the main reason is that someone would use this type of product?
* Do you have any suggestions to make the product better for people not currently studying biology?

## Interview responses

Holly Harrison 13G, interest in biology

Question 1: would a visual example of natural selection help you understand biology?

Response: yes, one of the reasons I achieved an 8 in GCSE was due to free science lessons, an interactive version of something like this would have been helpful

Question 2: Who do you think the product should be tailored towards?

Response: A level students, I believe university students already have high level equipment to experiment with. From what I’ve heard, a level students don’t have the right equipment or time to experiment properly

Question 3: What do you think the main reason is that someone would use this type of product?

I think someone would use this out of interest and for amusement with tampering with certain conditions.

Question 4: Do you have any suggestions to make the product better for people not currently studying biology?

Perhaps a simple explanation after each heading? Also, if someone isn’t particularly interested in biology then a means of disabling them entirely.

# Limitations

Cant be 100% true to life since that would require vast amounts of computer power and simulation of every single factor effecting selection even the extremely abstract details. The simulation cannot also account for the workings of a brain since the brain is not yet fully understood. This wont really affect my solution as reaching an environmental equilibrium without forcing it is the main goal, to achieve this goal not every factor has to be simulated especially considering that the environment its being simulated has been abstracted from real world anyway and so complex behaviours are unnecessary

Pygame doesn’t handle 3d environments very well at all so the project is limited to 2d (initialising a view matrix would be insanely hard). This limits the project as a 3d project would be more understandable to users and more initiative to use.

Other limitations include:

* Python doesn’t support threading (essentially limiting the game to one cpu)
* pythons awkward support for OOP + unions/structs. Cant employ techniques I’m used to using in C
* I predominantly use compiled libraries in my work (its faster), however some of these aren’t native libraries and aren’t on school computers so anyone wanting to run the program would have to download them

# Essential features

* Path finding:
  + Vision
    - Cone: size, length, obscurity
  + Rotation
    - Affects where its “looking”
  + Preferred sites (visual representation is purple circles)
    - Feeding zones (where its reliably found the most food)
  + Smooth movement in straight lines instead of zig zagging
    - Step manipulation, the amount that they move per step (this is elaborated on in the algorithms section)

**Why is this relevant to the solution?**

To improve the accuracy of the simulation, I implemented a system whereas each creature has a set of eyes and the vision of such is dependant on their genetic code,

The vision cone Is necessary to this goal as it allows the simulation to automatically exclude any food it cannot see, rotation refers to the direction they are looking in and therefore the position of the vision cone.

* Simulation:
  + Visual differentiation using genes
    - Spots etc
  + Receptors and their binders
    - For instance epinephrine and norepinephrine
  + Enzyme binding
  + External substances that can be diffused into for a good/bad effect#
    - Stimulants
  + Amino acid sequencing
* Metrics:
  + Graphs
  + Species relation diagrams
  + Genetic diagrams
  + A different pygame screen that shows such metrics (if this isn’t possible ill hook another program onto it)

**Why is this relevant to the solution?**

A form of logging the results of the solution is a way of showing the results over time e.g the change in frequency of one species and also a change in efficiency of the different energy levels. It also provides an interface for viewing what’s actually happening behind the display of the simulation

* Settings:
  + Accessibility page
  + User preference page
  + Simulation settings page
  + Species manager page
  + Information page

**Why is this relevant to the solution:**

In the success criteria it is stated that the solution should accommodate for disabilities such as colour blindness, different menus before the simulation is started allows the user to adjust any preferences or accommodations they need before starting.

* Environment control:
  + Spawn a new species on the mouse position when the mouse is clicked in the simulation area
  + Spawn food:
    - Range manipulation
    - Amount per sec
    - Distributed across the whole environment
  + Temperature:
    - Affects external enzyme efficiency
    - Thermoregulation, (water loss)
  + Humidity:
    - Hindered thermoregulation
  + Food richness

**Why is this relevant to the solution?**

The user may want to simulate conditions under specific circumstances and measure the effect they would have on the final environment. This also adds a degree of control to the user to use the simulation for a wide range of uses.

# Desired

* Machine learning pathing

Why?

Utilising ai in this solution would allow the creatures to learn over time, simulating adolescence e.g a creature wouldn’t be able to function at the same level as its parents as soon as its been born.

* Launch manager written in c# (probably just a windows forms program)

Why?

This would add a degree of professionality to the program instead of running from a ide, I believe that by hiding the console and source it would make the program more user friendly as someone less experienced with computer science may not know how an ide works or how to run the program, this feature would solve that problem

* Ray tracing too ensure food isn’t obscured (on the more likely end of these) –done

Why?

This feature is necessary for adding new environments as to prevent creatures seeing through the environment, ray tracing allows me to discard anything a creature cant see and also improve performance as less objects are being passed through the vision cone

# market research

## other solution 1: (natural\_selection written in python by owen\_davis\_bower)

Chart, bubble chart

Description automatically generated

Pros against my product:

* --No identifiable pros--

cons (identified by looking into source code):

* The blits are essentially randomly appearing and so its not really simulating anything and instead illustrating how the process works in a highly simplified way. These can be deduced by:
  + No genotype or phenotype (individual characteristics) of each creature therefore meaning none of these are simulated
    - No mutations
    - No speciation
    - No genetic drift
  + No reproduction as can be seen in the source as new creatures randomly appear

how can I learn from this solution?

nothing technically, this is due to the simplicity of the above solution without any technical details. However, I have taken away from this that a key to identify the herbivores/omnivores/carnivores of the solution will be helpful.

## Other solution 2: species artificial life real evolution

A picture containing text, grass

Description automatically generated

Pros against my product:

* 3 dimensional environment and creatures
* Allows for splicing of genes of two species to allow the user to make their own species with visible limbs
* Allows control of the landscape

Cons:

* Omnivores and herbivores etc are not easily identifiable
* Mutations are limited to randomness instead of that presented during reproduction meaning speciation is much less complicated and genetic diagrams would appear to be completely random instead of branching

What can I learn from this solution:

Investigating this solution has allowed me to realise how the landscape can affect the selection pressures during natural selection, it has also made be consider adding visible remarks on creatures to make them more easily identifiable as different to the user.

|  |  |  |
| --- | --- | --- |
| **Success criteria** | **How it will be met** | **evidence** |
| A logging system able to expand certain elements for more detail | A CLI style interface on the right of the GUI with plain text marked as white and expandable as yellow |  |
| A species diagram showing frequency and evolution chain? | Each species allocated a code corresponding to the genome, if the genome is outside a range of 50 bases then it will have speciated and will be shown as a species stemming from the ancestor species |  |
| An interface to control all selection pressures | A top bar with slider elements, each labelled and coloured to control a specific element e.g mutation chance or temperature |  |
| An accessibility/theme page to change colours of UI elements or make them bigger/smaller | A page branching from the start page that contains a list of all colour variables and a suitable interface to control them  Sub criteria:   * Accessibility for those with colour blindness * **How this will be met:** * A drop down menu for different kinds of colour blindness * Accessibility for those with poor eyesight in general * **How this will be met:** * a font size slider and the ability to change colour of text |  |
| a means of slowing down or speeding up the simulation | slider elements off to the side appropriately labelled. The slow will work by sleeping inside the thread and the speed up will add a value to the speed of the creatures moving and performing actions |  |
| does the simulation perform as expected and reach ecosystem equilibrium (assuming the right conditions) | In order for this requirement to be met the simulation must:   * React accordingly to changes made to the environment * **How this will be done:**   This will be done by using a formula to change aspects of the creatures e.g  Digestion of food using (1\*bmr) \* (temp \*0.2)   * Follow the expected route in that the most adapted survives (this isn’t always the case even in real life) * **How this will be done:**   This will be done by creating an environment favouring better traits such as speed and higher litter sizes   * Reach an environmental equilibrium in that the proportions of creatures are balanced perfectly * **how this will be done:**   This will be done by making sure the population of a consumer is proportionate to the population of the food it eats   * Illustrates that natural selection is a continuous process * **How this will be done:**   This will be done using a species diagram showing the genotypes over time |  |
|  |  |  |

# Hardware and software requirements

* Python 3.x (as long as it isn’t version 2 it should be fine)
* at least pygame 3.x
* Doesn’t interact with an operating system so any of (mac, windows, Linux)
* Recommend a powerful CPU such as gen 2+ ryzen or intel 9000 +
* CUDA (modern nvidia card) GPU (I’m using the gpu version of TensorFlow to make sure it doesn’t run terribly)
* Windows operating system, I will be using a windows library for this project mainly for exporting the results however some basic memory management could apply if I’m forced to work around a limitation in pygame

# External dependencies (libraries)

* Math
* Pygame
* Random
* Json
* Sys
* Time
* TensorFlow
* NumPy
* Keras