University of Nottingham

Designing Intelligent Agents

# How can Turtle-Hatchlings Evolve to Develop Low Mortality Rates?

Word Count: 4000 - Excluding Headers and Captions

20370898

*Jesal Vadgama*

# Contents

# 1 Introduction and Overview

This research will focus on developing a simulation that accurately represents the traits and environmental challenges Hatchlings(Baby Turtles) encounter to lower mortality rates. An evolutionary approach through the use of Genetic Selection methods such as Tournament Selection 3.5.1, Steady State Selection 3.5.2 and a Variation of Elitist Selection3.5.3, will be used to find optimal values for these traits mentioned in 2.2.

# 2 Subject Area Research

## 2.1 Prey and Predator

### 2.1.1 Hatchlings

Baby Turtle (known as Hatchings) eruptions occur at various beach locations early in the night. The low visibility offered by the darkness of the night helps the Hatchlings avoid being eaten by predators. Also, Hatchlings can be led astray by artificial lighting[1]. Hatchling eruption signifies when many Hatchlings have undergone subterranean incubation in their nests to navigate their way to the beach surface.
Once on the beach surface, the Hatchlings exhibit a range of typical behaviours, including:

- **A propensity to navigate towards sources of light to reach the sea**[2]. After emerging from their nests, typically at night, they are naturally drawn towards the brightest direction, usually, the moonlight reflecting off the ocean surface. This behaviour, known as 'seafinding', guides them towards the sea, which is crucial for survival, however, this instinct can be disrupted by artificial lighting, leading to misorientation and potential danger such as predators.
- **A propensity to navigate around rocks, rather than climbing them**: The propagation of light can be altered by obstructions in the form of terrestrial features such as rocks which can misguide Hatchings from their intended path towards the sea. When approaching such obstructions, Hatchlings cannot surmount them and thus must deviate from their path by navigating around them[3]. This can decelerate their speed making them vulnerable to predators, notably Coragyps atratus, thereby increasing the mortality rate among Hatchlings.

Figure 1: Hatchling

It is noteworthy that in the empirical reality of Hatchlings' lives, around 25% manage to endure the initial days of their existence before reaching the ocean[4]. The survival rate could increase over time by finding the optimal traits the hatchings have which is mentioned in 2.2.

[1] https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0275088
[2] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9605334/
[3] https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0275088
[4] https://www.ecologyproject.org/post/underdog-to-icon-a-sea-turtle-hatchling-s-journey

### 2.1.2 Coragyps atratus

It is known that predators such as Coragyps atratus target hatchlings at beaches[5]. These predators typically establish their habitats in terrestrial features near the beach (commonly rocks), and strategically wait for hatchlings' nests to hatch to seize the opportunity to feed on the newly emerged baby turtles.

Figure 2: Coragyps atratus

Coragyps atratus display a range of typical predatory behaviours, including:

- **Working in Groups**: Coragyps atratus typically work in coordinated groups[6]. This allows them to locate food sources together more effectively and when food sources are found, multiple vultures will converge on the location leading to a feeding frenzy.
- **Increases in Speed and Accuracy**: Upon sighting their prey, they show an immediate increase in speed and a directional adjustment[7] towards the hatchlings making it almost impossible to escape death. This strategy ensures a high success rate in their hunts, making them formidable predators.

## 2.2 Essential Evolutionary Traits for Survival

### 2.2.1 Hatchling Traits

Hatchlings must develop traits such as having optimal speeds to reach the sea without being eaten by the Coragyps atratus[8]. If the Hatchings are too slow, due to their small size, they can become vulnerable to prey and fail to reach the sea. Similarly, if they are too fast, due to their size, they will become highly noticeable and be targeted by predators. Therefore, maintaining an optimal speed influenced by their size increases the turtle's chances of safely reaching the sea; this balance is crucial for survival.

The direction the Hatchling faces when surfacing is crucial to its survival too, because if the Hatchlings face the opposite direction, they may be led astray due to artificial light sources which can lead them in danger of being eaten by predators[9]. To mitigate this the Hatchlings will have to oscillate in all directions to locate light sources to move in the correct direction. Additionally, when terrestrial features appear in the Hatchling's path they must oscillate towards the direction of the light so that they can move around the rocks.

### 2.2.2 Coragyps atratus Traits

The Coragyps atratus have to develop fast speeds to catch Hatchlings before they make it to the sea. Additionally, these vultures will have to locate hatchlings within the direction it is facing so that they can feed on multiple turtles. It is known that they tend to move towards larger hatchlings, even when smaller ones are closer and visible to them[10]. The reason for this behaviour is that larger hatchlings are easier to spot from a distance. In contrast, noticing smaller turtles would require these creatures to get significantly closer, which they seem to avoid.

[5] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5494172/

[6] https://www.academia.edu/14851627/Observation_of_Nocturnal_Feeding_in_Black_Vultures_Coragyps_atratus_

[7] https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0179819

[8] https://doi.org/10.2744/CCB-1088.1

[9] https://doi.org/10.1093/icb/icaa044

[10] https://bioone.org/journals/chelonian-conservation-and-biology/volume-13/issue-2/CCB-1088.1/Factors-Affecting-Locomotion-in-Olive-Ridley-Lepidochelys-olivacea-Hatchlings-Crawling/10.2744/CCB-1088.1.full

# 3 Genetic Algorithm

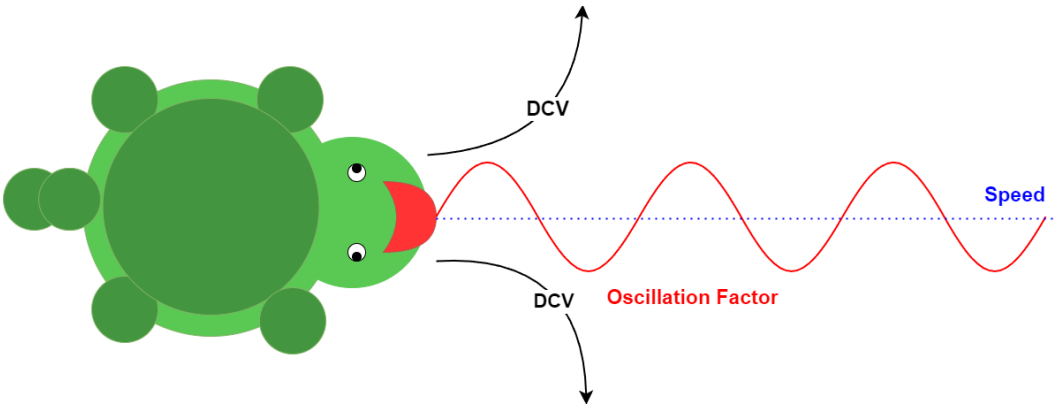
## 3.1 What is a Genetic Algorithm?

Genetic algorithms are optimisation techniques[11] that simulate the natural evolution process through the use of **selection**, **crossover**, and **mutation** operations. A population of individuals is evolved using this strategy, and its genetic properties are fine-tuned towards an optimal solution.

## 3.2 Genetic Representation

The genetic properties of a turtle were chosen to mimic its real-life tendencies. The genetic information chosen to optimise through iterations of the genetic iteration was chosen as follows, based on the research discussed in Section 2. There are 4 main evolutionary values: speed, size, dcv and oscilation factor.

- **Turtle Speed** is the speed at which the turtle moves forward.
- **Turtle Size** is the size of the turtle.




### 3.2.1 Direction Change

The direction change value is the amount to turn (in degrees), and is multiplied by direction_scaler - which scales the amount to turn by so (1.0 * 7) would mean turn 7 degrees.

### 3.2.2 Oscilation Factor

The magnitude to which turtles' movement oscilates. (i.e. the intensity of the *sin* wave they move along). Tendency to oscillate based on whether it faces water. self.sensor is 1 when facing water, 0 when facing away - so (1 - self.sensor) is 1 when facing away, 0 when facing towards. The direction change value mentioned in 3.2.1 will then be multiplied by the (1 - self.sensor).
The above properties are represented as floating-point weightings between 0 and 1. These weightings are subsequently explored and manipulated during the crossover and mutation operations at the end of each generation.

## 3.3 Fitness Function

An objective function[12] is commonly used in genetic algorithms to guide individuals towards optimal genetic material and solutions. In the context of this project, the goal of a turtle is to **successfully reach the water as fast as possible**. Turtles should also be discouraged from hitting rocks: it was decided to instantly kill turtles who collide with rocks, to effectively discourage this behaviour. The fitness function is used to benchmark and evaluate the performance of individuals at the end of each generation.
Generations may be terminated from a set of conditions:

[11] https://www.researchgate.net/publication/267369803_Application_of_Genetic_Algorithms_in_Machine_learning#:~:text=This%20Genetic%20Algorithms%20(GAs)%20are,problem%20apt%20for%20applying%20GAs.
[12] https://link.springer.com/article/10.1007/s12065-023-00822-6

- **Every turtle is terminated**: Turtles may die if they collide with rocks or are eaten by birds.
- **The time limit is reached**: A time limit of 2000 ticks was imposed to forcefully terminate a generation.

## 3.4 Initialisation

Random weight initialization was used to set initial evolutionary values for all turtles. This introduces a level of inherent, genetic diversity into the population that can be further refined by the genetic process.

## 3.5 Selection

Once a generation has simulated, a portion of the population is selected to reproduce and pass on their genetic material based on their fitness. Multiple selection methods were explored, mentioned in 3.5.1, 3.5.2 and 3.5.3.

### 3.5.1 Tournament selection

Tournament Selection is a common selection method used in genetic algorithms. To ensure that the fittest individuals are selected, subgroups of individuals are randomly chosen and the winner of these subgroups passes on their genes. This method mimics the process of survival of the fittest.
Tournament selection was chosen mainly because it guarantees that each individual has an equal chance of participating in the same number of tournaments.

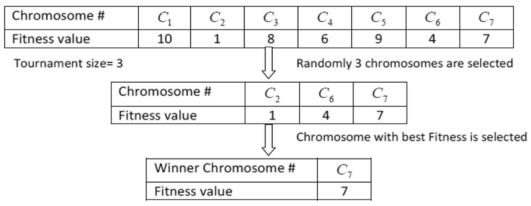



Figure 3: Tournament Selection Genetic Algorithm Process

In this project, tournament selection is approached using the following process:

- The turtles are arranged into 3 random groups with a tournament size of 4. This tournament size was selected to ensure that weak turtles have a small chance of getting selected, while still promoting genetic diversity amongst turtles.
- The turtles are evaluated based on their objective fitness value, determined at the end of a generation. If turtles within the tournament fail to reach the sea, their performance is measured on how long they survived in the simulation. Turtles who reach the sea are automatically preferred (their fitness value is higher), compared to turtles who do not reach the sea.
- The winners of each tournament are used in the crossover process: their genetic material is carried over to the next generation, before being mutated.

### 3.5.2 Steady State Selection

Steady State Selection is a selection method used in genetic algorithms to ensure the genetic material of the fittest individuals in a generation is preserved[13]. An individual's fitness is assessed at the end of a generation, and the fittest individuals undergo mutation mentioned in 3.7. A small subset of the fittest individuals are chosen to replace the least fit individuals in a generation.
Steady State Selection is similar to Tournament Selection as both mimic Survival of the Fittest, however there are no subgroups in this method: the absolute fittest individuals in a generation are guaranteed to be chosen. Although better genes are being preserved, this approach may lack the genetic diversity which is likely in Tournament Selection. This selection method should gradually improve the population's genetic material as the fittest individuals are preserved, while the worst traits are removed.
The approach to developing Steady State Selection in this project is as follows:

[13]https://www.semanticscholar.org/paper/Selective-pressure-in-evolutionary-algorithms%3A-a-of-Back/fda63e0c5f0f38d5c9679eab5f55966c51e47ef2

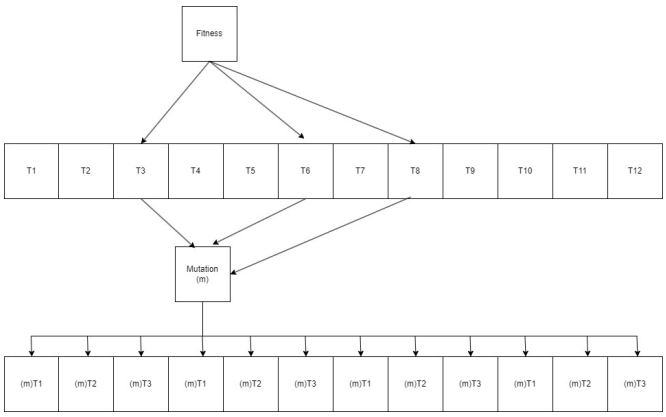
- Find the top three fittest individuals at the end of a generation, out of a total population size of 12 turtles.
- Find the three least fit individuals at the end of a generation. These may be turtles that failed to reach the water, or turtles that reached the water slower than their peers.
- Replace the three best turtles with the three worst turtles, keeping all other turtles the same. Apply mutation to the best turtles.

The birds also may use Steady State Selection:

- Find the best three birds (the birds who have eaten the most turtles).
- Find the worst three birds (the birds who have eaten the least turtles).
- Replace the worst three birds with the best three birds. Apply mutation to the best three birds.

### 3.5.3 Elitist selection

Elitist selection [14] is a selection strategy where a number of individuals with the best fitness values are chosen to pass on to the next generation. However, elitist selection differs from Tournament and Steady State Selection in the fact that it subsequently avoids the mutation operator.




This project proposes a novel variation of elitist selection:

- Find the three turtles with the best fitness values.
- Mutate all three turtles.
- Replace all twelve turtles in the generation with these three turtles.

## 3.6 Crossover

This project uses an asexual approach to crossover[15], which is analogous to cloning. Genetic material of the individuals selected using the chosen selection heuristic is directly copied to individuals of the next generation. This genetic material is subsequently mutated to promote diversity and exploration of the solution space.

## 3.7 Mutation

Mutation is a technique commonly used in genetic algorithms to stochastically alter the genetic material of an individual. The crossover and mutation heuristics are similar to the concept of Exploration vs. Exploitation[16] First, the best genetic material is **exploited** from the previous generation: it is advantageous to retain the properties of optimal solutions found during the run. Next, the genetic material is **explored** using mutation: it is imperative to explore the problem space for better solutions than the local optimum.
The mutation approach used in this project appends a random floating point number to each gene, which is strictly between -0.1 and 0.1.

[14] `https://ieeexplore.ieee.org/document/7927990`
[15] `https://www.sciencedirect.com/science/article/pii/S0020025521000475`
[16] `https://link.springer.com/article/10.1007/s40747-019-0102-7`

# 4 Ray casting with Bresenham Line Algorithm

Ray casting with Bresenham's line algorithm involves drawing a line between two points in a grid-based space[17] and plotting pixels incrementally using the x,y coordinates closest to the direction the line is going.
Adapting this into the light class as a light source in a 2D environment can accurately simulate light rays in real life. The class begins by casting rays in all directions using self.directions and for each ray, it uses self.lines. For each direction, it iterates over each Rock in self.parent.tiles and calls the cast_ray method that calculates the intersection of a ray with a circular Rock. The line can be calculated using:

$$y = mx + c$$

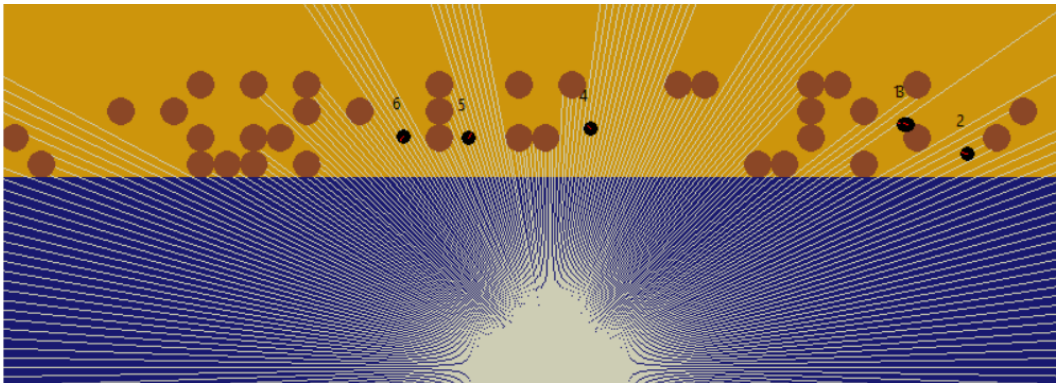
and the circle can be calculated using:

$$(x - h)^2 + (y - k)^2 = r^2$$

.
If the discriminant of the quadratic equation is non-negative, it means there are intersection points, at which it checks if these points are on the line segment from the light source in the direction of the ray and if it is, it returns the intersection point:

$$b^2 - 4ac$$

After finding the closest intersection point, it draws a line from the light source using canvas.create_line and If no intersection point is found, it draws a line in the direction of the ray to a point far away(1000 units). All drawn lines are stored in self.lines and has successfully been implemented in my simulation as shown in Figure 4.




[17] https://www.researchgate.net/publication/26488240_Adapting_Bresenham_Algorithm

# 5 Environment

## 5.1 Tkinter

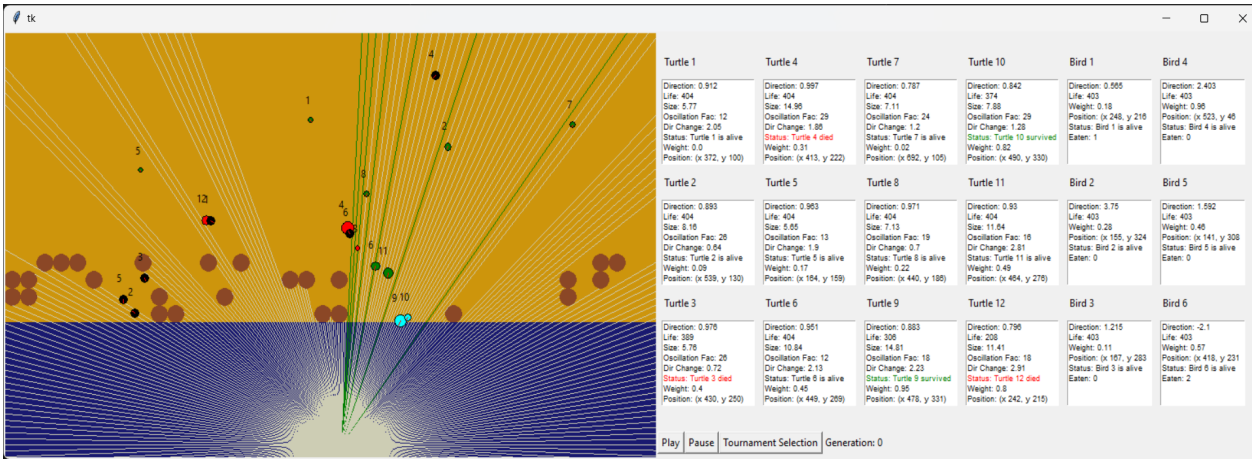
Tkinter has been chosen for this simulation due to its built-in features that handle events such as clicking buttons and its widget variety which is used to evaluate the turtle and bird evolutionary traits.

## 5.2 Frame One

Frame One consists of the Turtle simulation itself which depicts the events that occur for the turtle to successfully make it to the sea. All the features of the simulation have been made to accurately capture the birth life of Hatchling in real life. The items within the frame are shown below:

### 5.2.1 Map

The 2D map for this simulation shown in Figure 5.2.1 is created using the create_map function where the dimensions of the map are determined by width and height, and is initialised as a 2D array list of zeros. Each zero represents a sand tile, and the bottom eight rows are the water tiles, represented by the number 2, by iterating through these rows and setting each cell to 2. The four rows above the water randomly generate rock tiles, represented by 1, set by iterating through these rows and assigning each cell a 1 with a 25% chance, which makes the maps unique. The rocks are placed stochastically to allow turtles to adapt to various rock formations to reach the sea rather than excelling only in navigating a specific type of rock formation.




### 5.2.2 Sea

The Turtles must reach the top of the sea tracked by self.water, to survive and go through evolutionary methods through the next iteration. Once at the point of the sea, the birds will be unable to eat the turtles.

### 5.2.3 Light

The light is an item positioned in the same location during each simulation that acts as a point of origin for the Bresenhams Line Algorithm mentioned in 4. The light source shown in Figure 4 projects rays across the map which are not obstructed by rocks and can be sensed by the turtles mentioned in 6.1.3 to locate the sea.

### 5.2.4 Location of Turtles and Birds

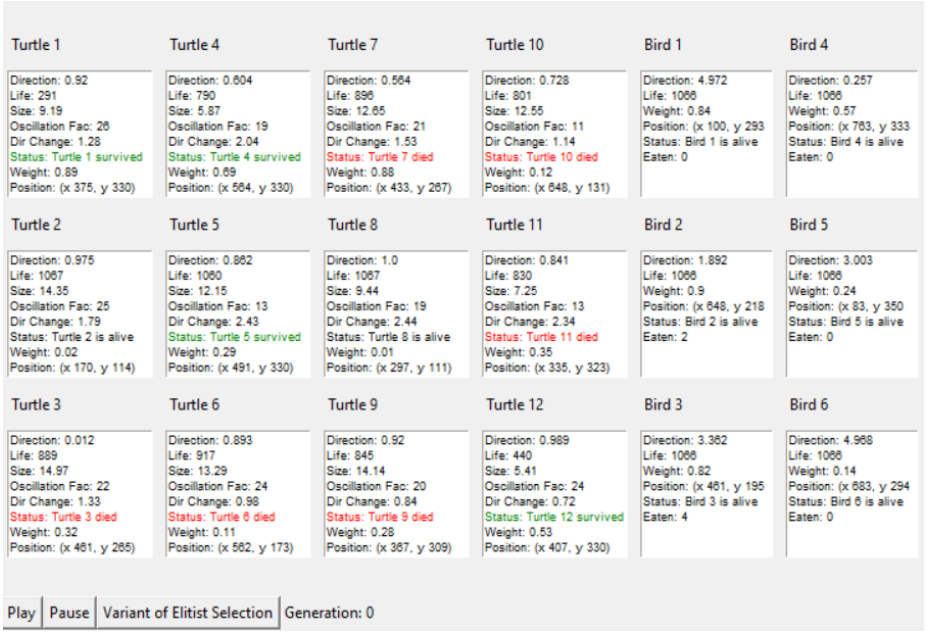
The turtles are spawned in packs at the top of the beach in different beach locations as this is where Hatchling nests are commonly found mentioned in 2.1.1. The turtles are spawned in the simulation on a line and is defined as line_start_x and line_end_x, which is then divided into smaller segments with the width of each segment determined by segment_width where each of the twelve turtles being randomly assigned a spawn location. The same method is used to spawn the Birds in packs which reside near the rocks on a line mentioned in 2.1.2 using bird_line_start_y, bird_line_end_y and bird_segment_width.

## 5.3 Frame Two

Frame Two contains information about the Turtle Simulation in the form of widgets and buttons. The Widgets are updated every tick with several factors in the simulation to show evolutionary change for both Turtles and Birds. This feature is helpful when evaluating the agent's performance within the environment The items within the frame are shown below:

### 5.3.1 Widgets

A Label widget is created for each turtle and bird to display its number, and a Text widget is created to display detailed information about both items. These widgets are stored in the self.turtle_info_texts and self.bird_info_texts list and are placed in the self.info_frame using the grid method, which organises the widgets in a table-like structure with padding and alignment using padx, pady, and sticky options in the grid method.




### 5.3.2 Buttons

The frame also consists of a subframe where the generation is displayed and updated each generation. The buttons are listed below:

- Play and Pause buttons - To Play and Pause to analyse items within the simulation.
- Evolutionary Button - A toggle button to set different evolutionary methods.

### 5.3.3 Export to CSV

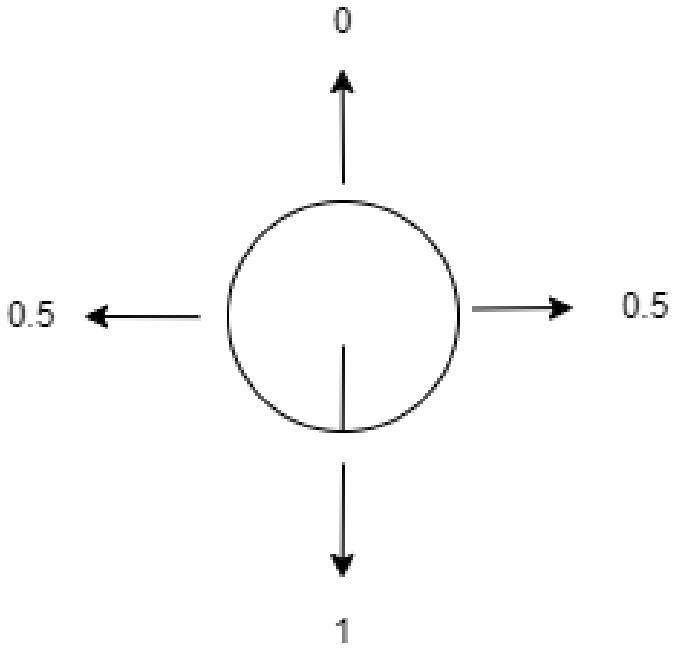
The export_data_to_csv function calculates the average life, size, direction, oscillation, and weight(speed) of the turtles and the average life, weight(speed), and number of turtles eaten by birds in which it exports these averages along with the evolutionary state and iteration number to a CSV file named data.csv.

# 6 Sensors and Behaviours

## 6.1 Turtle Sensor's

### 6.1.1 Direction Sensor

Figure 6.1.1 shows how the Direction Sensor functions for the turtle on the map. The 'self.sensor' is used to calculate the turtle's direction, which is converted from radians to degrees, adjusted to align with conventional compass directions, and then normalised to a range between 0 and 2, and inverted to indicate the turtle's facing direction with 1 being South and 0 being North.




The direction sensor influences the turtle's behaviour as it provides readings it needs to oscillate specifically when facing the direction of the sand and the sea. Additionally, it helps the turtle interact with its environment effectively influencing its navigation and decision-making in detecting rocks and rays of light.

### 6.1.2 Rock Sensor

The rock sensor is used so that turtles can evolve to navigate to the sea without being obstructed by a rock mentioned in 2.1.1. The sensor calculates the angle between the turtle and the rock using the math.atan2 function to calculate the angle using Euclidean distance between the turtle and each rock:

$$d = \sqrt{(x_{\text{turtle}} - x_{\text{rock}})^2 + (y_{\text{turtle}} - y_{\text{rock}})^2}$$

This affects the turtle's behaviour by adjusting the turtle's direction to be opposite of this angle which makes the turtle move away from the rock if the distance is between 13 and 17. The turtle adjusts its direction clockwise or counterclockwise depending on the Direction sensor which also determines the current rotation direction allowing for dynamic and responsive navigation around rocks in its environment.

### 6.1.3 Lights Sensor

The light sensor gives the turtle its ability to oscillate towards the light which leads it to its source which is the centre of the sea. The light sensor works by finding the equation (y = mx + c) of the light ray line using the cartesian equation of the circle:

$$Ax + By + Cz + D = 0$$

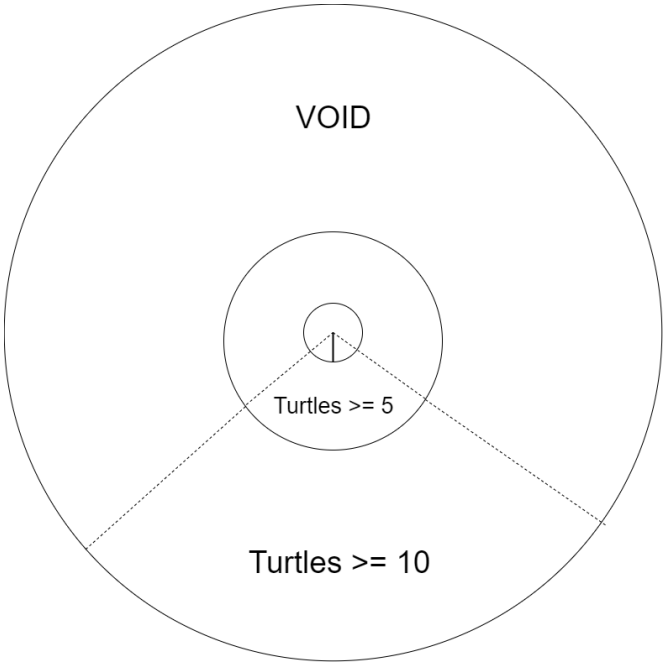
Then the sensor finds the perpendicular normal from the centre of the turtle to the closest part of the line(the shortest distance between the turtle and the ray of light). If this distance is less than the turtle's width, the turtle intersects with the ray of light which triggers the light sensor green.

A timer has been implemented to prevent the turtle from exploiting this behaviour by using subsumption architecture: If the turtle approaches a rock, the rock avoidance behaviour is prioritised.

## 6.2 Bird Sensor's

### 6.2.1 Turtle Sensor

The Diagram in figure 6.2.1 shows the behaviour of a bird when it is within proximity of a turtle.



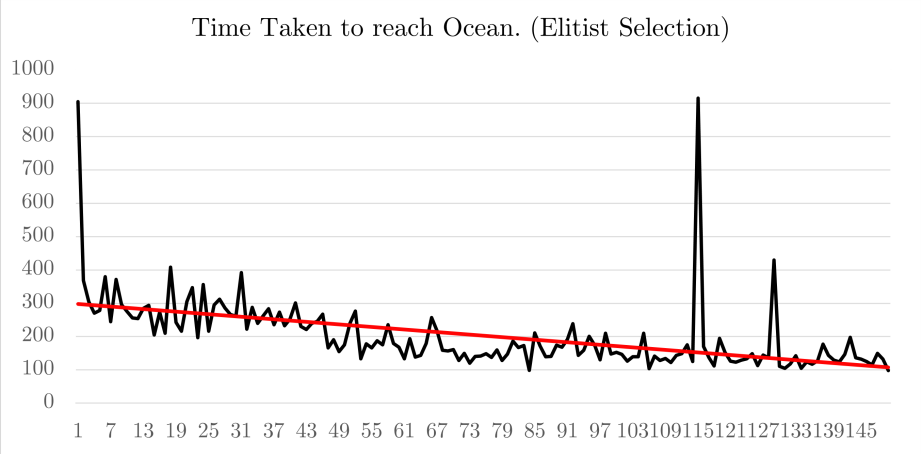

The turtle sensor is used by the birds to eat the turtles by looking for turtles that are alive. For each Turtle, it calculates the distance and angle_to_turtle using the bird's (self.x, self.y) and turtle's (item.x, item.y) to normalise the angle_difference to be within 0 to $\pi$. The sensing_range is a parameter used to detect a turtle by distance and is adjusted based on the size of the turtle. If the turtle is within the adjusted_sensing_range and facing the bird's direction, the bird changes its direction towards the turtle and updates its position (self.x, self.y) using the angle_changer and speed(weight). If the distance is less than or equal to the sum of their radii(-collides), the turtle's alive and water attributes are set to False, the turtle is removed from self.detected_turtles, and the bird's eaten count is incremented.

# 7 Evaluation & Data Analysis

## 7.1 Time Taken to reach Ocean

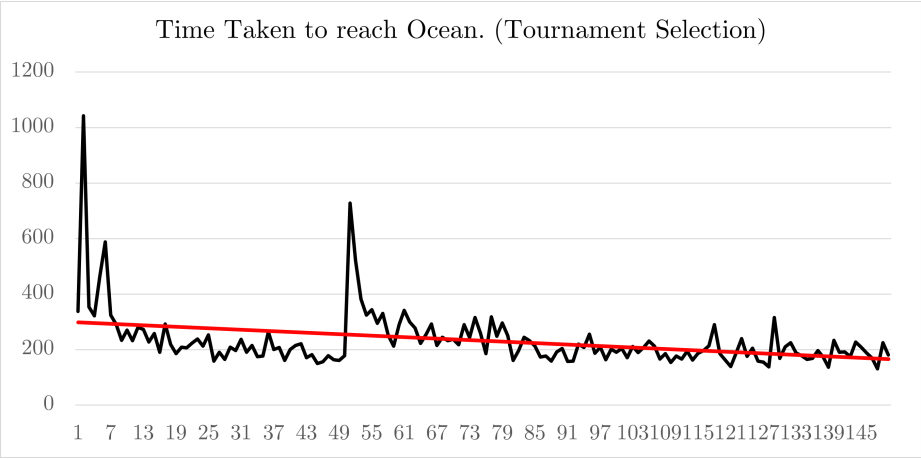
The graphs below show the time it takes turtles to reach the ocean. For the turtles to develop low mortality rates, the time taken for the turtles to reach the sea should decrease over time. Throughout the simulation, measuring in ticks is chosen over measuring in time because it allows for accurate tracking of each generation where the frequency of generation doesn't align with real-world time.

### 7.1.1 Elitiest Selection Time for Turtles



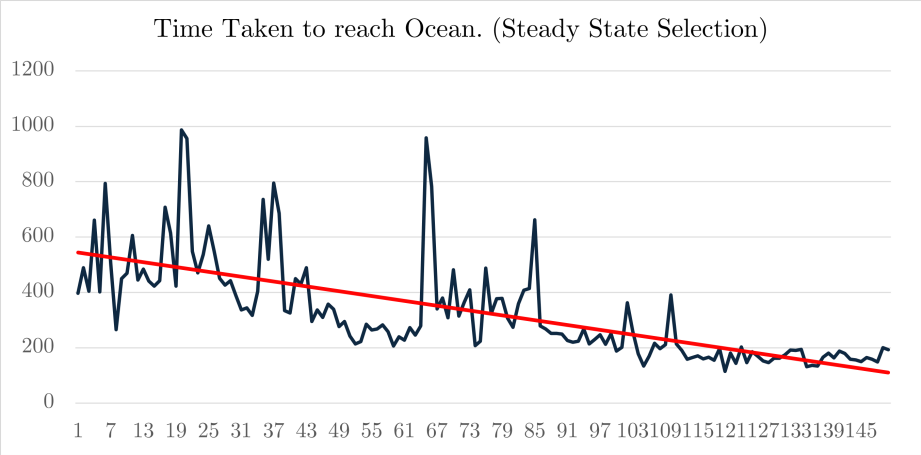

Upon using the Elitist Selection method, the time taken decreases over time which means the turtles are successful at making it to the sea without being detected and eaten by the birds. The spikes over time show the stochastic anomalies that occur during the simulation, showing turtle difficulty when surpassing rock formations resulting in birds detecting and eating turtles. The Elistest Selection started with an average of 905 ticks taken to reach the sea and then finished with 98.

### 7.1.2 Tournament Selection Time for Turtles



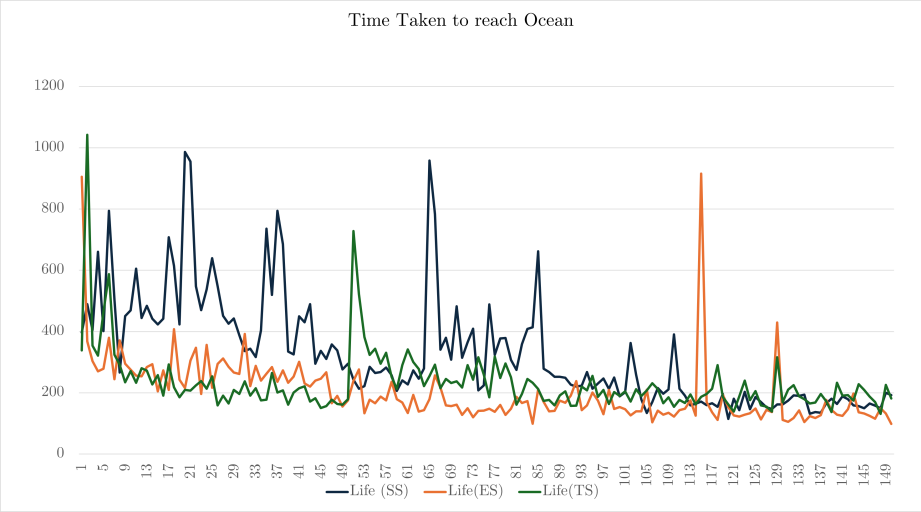

Comparatively, Tournament Selection showed similar effects on turtles reaching the sea as Elitist Selection, however, on each generation of turtles, this is shown by the fluctuations after every 10 generations. There were not many spikes which show anomalies in each generation which is good as they were still efficient. Tournament Selection started with an average of 339 ticks taken to reach the sea and then finished with 182.

### 7.1.3 Steady State Selection Time for Turtles




In the context of Steady State Selection, its influence on the efficiency of turtles' ability to reach the sea was found to be the least effective because each generation consistently produced turtles that took the longest to reach the sea. Notably, the graph shows significant spikes in the time taken for turtles to reach the ocean, indicating periods of decreased efficiency when the turtles tried to reach the sea which is represented by the spikes. Steady State selection started with an average of 397 ticks taken to reach the sea and then finished with 193.

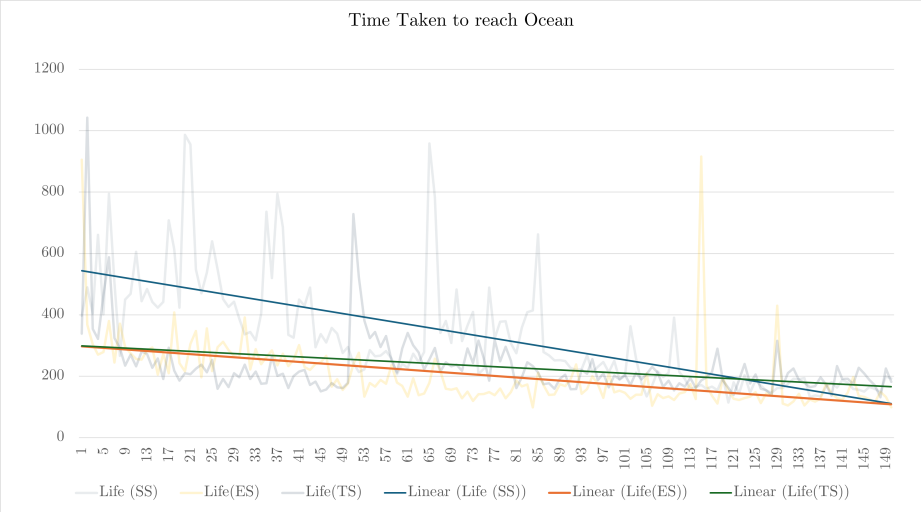
## 7.2 Combined Selection Methods Evaluation




This graph plots Elitist, Tournament and Steady State Selection on the same area, however due to stochastic anomaly - several peaks make the data hard to read. Linear interpolation was performed on the graph to provide a more accurate reading (see next section).

## 7.3 Combined Selection Methods Linear Interpolation

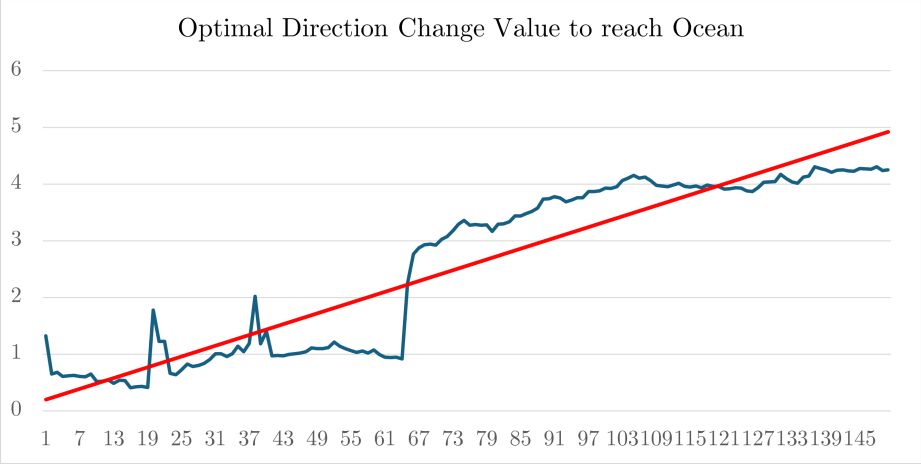
The following graph linearly interpolates the time taken to reach the ocean for all 3 selection functions. The strongest gradient here is Stable-State Selection, but over 150 generations it overtakes Elitist and Tournament - meaning that depsite taking longer it can yield stronger results due to the maintained diversity of the population.




## 7.4 Optimal Trait Values

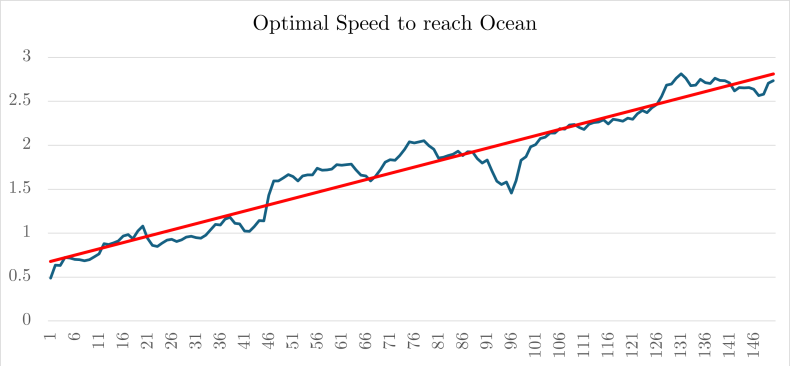
### 7.4.1 Direction Change Value (DCV)

The optimal DCV value, which controls how much a turtle turns - especially when encountering a rock, has grown from around 1 to stabilizing at around 4.



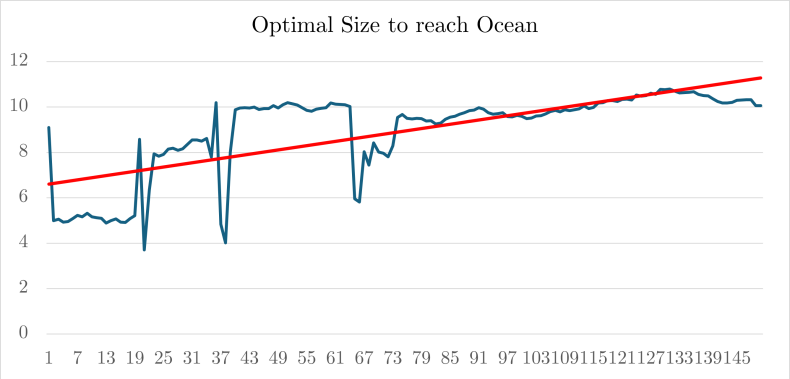
### 7.4.2 Optimal Turtle Speed

The optimal speed value grew over time, from approximately 0.75 to around 2.75 units/tick. This value didn't stabilize towards 150 generations, and prospectively could continue along the upwards trend - getting bigger and bigger with more generations.



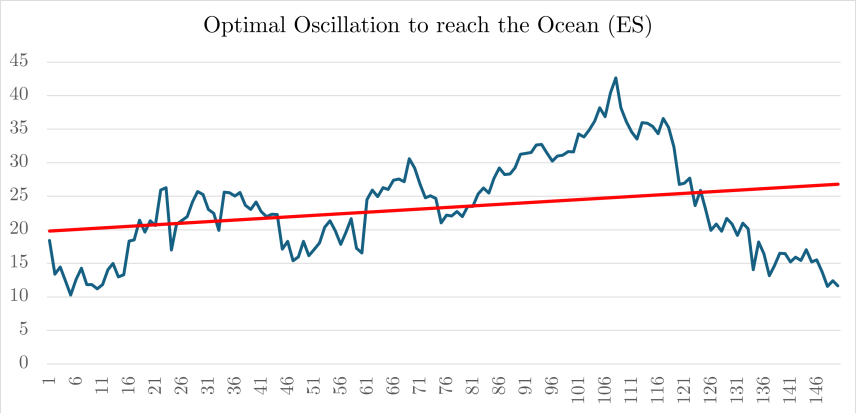

### 7.4.3 Optimal Turtle Size

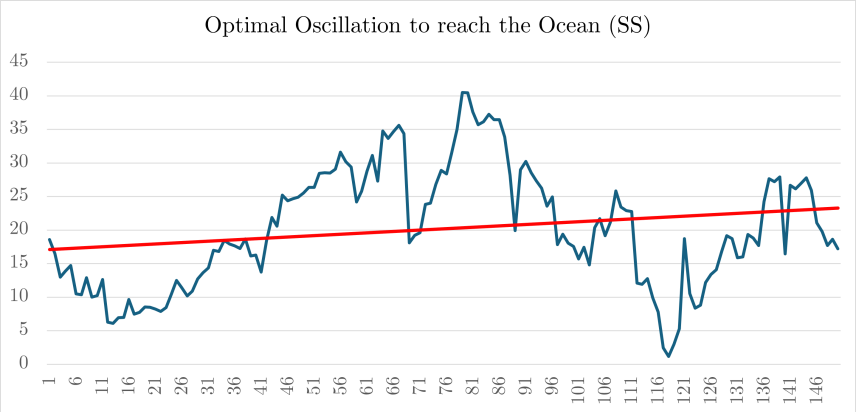
The optimal turtle size fluctuated in the first 80 generations, but seemed to stabilize at around 10. This can be seen as the balance between speed, and being detectable by the birds (predators).

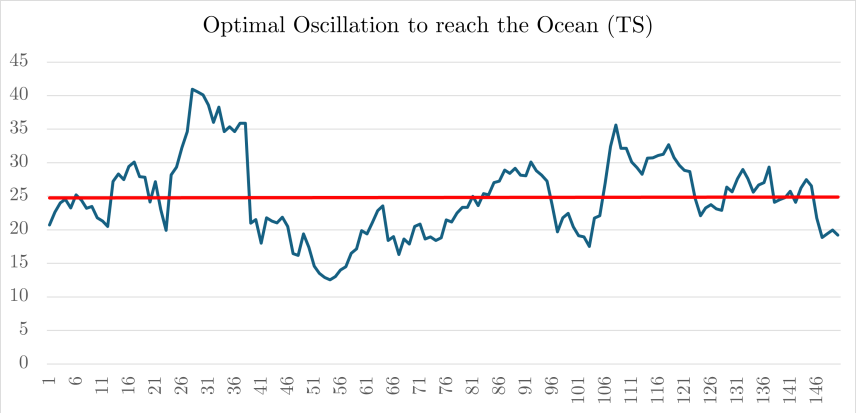


### 7.4.4 Oscillation Factor

Across Elitist, Tournament and Steady-State Selection, the optimal oscillation factor fluctuated, peaking at around 40 on generation 125 before falling down to 10. While these values occupy a large range, the values fluctuate around 25, which can be seen as a medium level of oscillation.



# 8 Conclusion

All three evolutionary algorithms evolve the turtles efficiently and quickly which resulted in them reaching the sea and ultimately developing low mortality rates. However, my novel approach towards Elitiest Selection mentioned in 3.5.3 outperformed Tournament Selection 3.5.1 and Steady State Selection 3.5.2 in terms of reaching the optimal values quicker. Turtles that learned to oscillate their movement managed to use this behaviour to escape rock predicaments however turtles that oscillate too much often went in circles. It was also noticeable that turtles that followed light too often found linear unequalled paths to the sea. It was also known that turtles that evolved with bigger sizes and speeds managed to outpace the birds.

## 8.1 Challenges Faced During Implementation

One of the challenges I faced during the early stages of this project was working with Tkinter because I was new to programming with this module. Upon learning to create items, I faced memory leaks where I was drawing multiple items on Canvas which was only noticeable when drawing substantial items, where the simulatation started to slow down. To resolve this issue, I realised when drawing to the canvas, you need to delete the previous frame. During the later stages of the project, I found that Tkinter was not suitable due to the lack of GUI tools and that I was unable to capture all of the finer details.
After reviewing different methods of capturing higher levels of detail, it was found Perlin noise generation would have best suited this project, however, this method would require a higher level of understanding and bring complex challenges during Implementation. Learning how to use a decorator[18], would be useful to improve performance issues or even rebuilding the simulation in unity would make my simulation more realistic as Tkinter would not be able to handle all the necessary components need to do a highly accurate simulation.
Another Challenge I faced was trying to implement Ray Casting in Tkinter 4 because of the mathematical complexity required to do Bresenham Line Algorithm on a 2D array map. These mathematical functions can easily be executed in Unity in a single line whereas, for Tkinter, multiple lines were used to achieve the same result however it resulted in high levels of coding complexity.

## 8.2 Success Vs Limitations

| **Success** | **Limitations** |
|---|---|
| Implemented the Bresenham Line Algorithm | GUI issues |
| Implemented 3 working Evolutionary Algorithms | Complex code |
| Implemented 4 mathematical complex sensors that positively improve Turtle's behaviour | |

Table 1: Successes and Limitations

## 8.3 Future Implementations

For future implementations, implementing a variety of beach conditions and introducing more predators to this simulation to explore how turtles would react to a specific condition. A specific condition that could be simulated is the effect temperature would have on turtles' survival rate which would require using the Boltzmann selection algorithm. In conjunction with this truncation selection and crossover approaches would be used, to simulate reproduction to identify and propagate the most advantageous traits within the turtle population.

[18] `https://www.youtube.com/watch?v=DnKxKFXB4NQ&t=10s`