# System 1: the ui element system

## sub class 1: buttons

### iteration 1

**Text

Description automatically generated**

Changes from iteration 1 -> 2:

* Changes to constructor method:

**What’s changed?**

less attributes to define by default ( colours set to defaults unless specified),

**why?**

makes adding a new object easier, if a custom colour is needed for instance the exit button, the colour can be overridden.

**New code:**

**Text

Description automatically generated**

* Changes to visual style:

**Old:**

**Graphical user interface

Description automatically generated with medium confidence**

**New:**

****

**What’s changed?**

Smaller button body, centred text in body, darker theme, no longer bounded for visual appeal

**Why?**

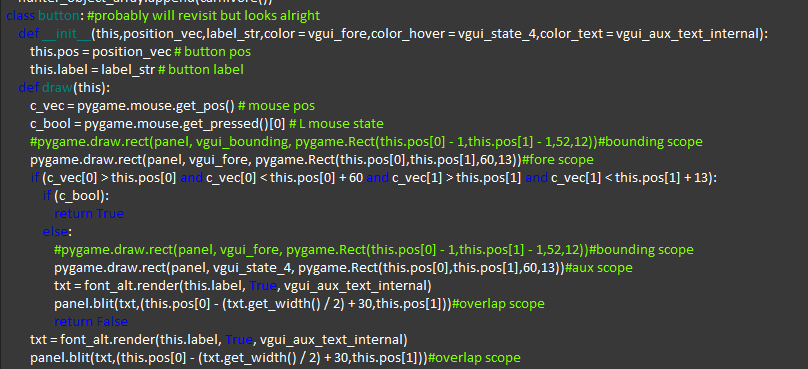
Fits darker background better, more visually appealing, looks more professional

* Changes to draw method:

**What’s changed?**

Mousepos and moustate now no longer passed as arguments, allows for better program maintainability, allows vgui objects to be called with the same method in an array (due to each draw method now only needing the object as context)

### iteration 2



Changes from iteration 2 -> final:

* **Change 1: changes to the text inside the button:**
  + I disliked how the text was displayed inside the button, previously it would centre the text using a constant for text of about 4-5 characters in length however I knew I had to develop a long term solution so the new system grabs the dimensions of the text using pygame.rect and adds the text offset dynamically instead
* **Change 2: changes to the mouse detection:**
  + Previously, there was no response to hovering over the button and instead would return a value when clicked with no visual indication so in the new system I removed c\_bool from the selection statement and instead separated them to allow for the colour of the button to change when its being hovered.

Result:



### Testing:

**Test 1: invalid string label (int)**

Text

Description automatically generated

Result: fail

Text

Description automatically generated

Amendments:

Text

Description automatically generated

the constructor method will now define the label explicitly as string preventing an error

result:

Graphical user interface, application

Description automatically generated

**Test 2: invalid position vector (impossible coordinate)**

Text

Description automatically generated

Result: pass

Passing an invalid coordinate will not throw an error, however since the coordinates -500,-500 don’t exist this code will not draw anything to the screen.

## **Sub class 2: sliders**

### Iteration 1

**Text

Description automatically generated**

**Changes from iteration 1 -> 2:**

* Attributes:

**Whats changed?**

Previously, the slider worked by the mouse ‘grabbing’ a dynamic rect and sliding it across to change the value, the new version is far more simplified and works by just using the relative x position of the mouse to update the slider position meaning it requires less attributes for this function

**Iteration 1:**

**Graphical user interface, text

Description automatically generated**

**Iteration 2:**

**Text

Description automatically generated**

* Possible return value:

**Whats changed?**

Iteration 1 could only manipulate and return values from 1 to 100 since it doesn’t use a formula to convert the slider position and instead uses the difference of the current slider position and the start of the slider

**Iteration 1:**

****

**Iteration 2:**

****

**Why?**

By using a formula to calculate an accurate value it allows for a much wider range of data to be manipulated using sliders such as floating points and values above 100, this is needed in the simulation as some of the new iteration 2 variables require floating point values

* Visual style

**Whats changed?**

Iteration 2 has a much smaller body, the visual representation of the slider is now different with a rectangle being drawn up to the selected point instead of a dynamic rectangle, the design is also a lot darker and user friendly

**Iteration 1:**

**Text

Description automatically generated**

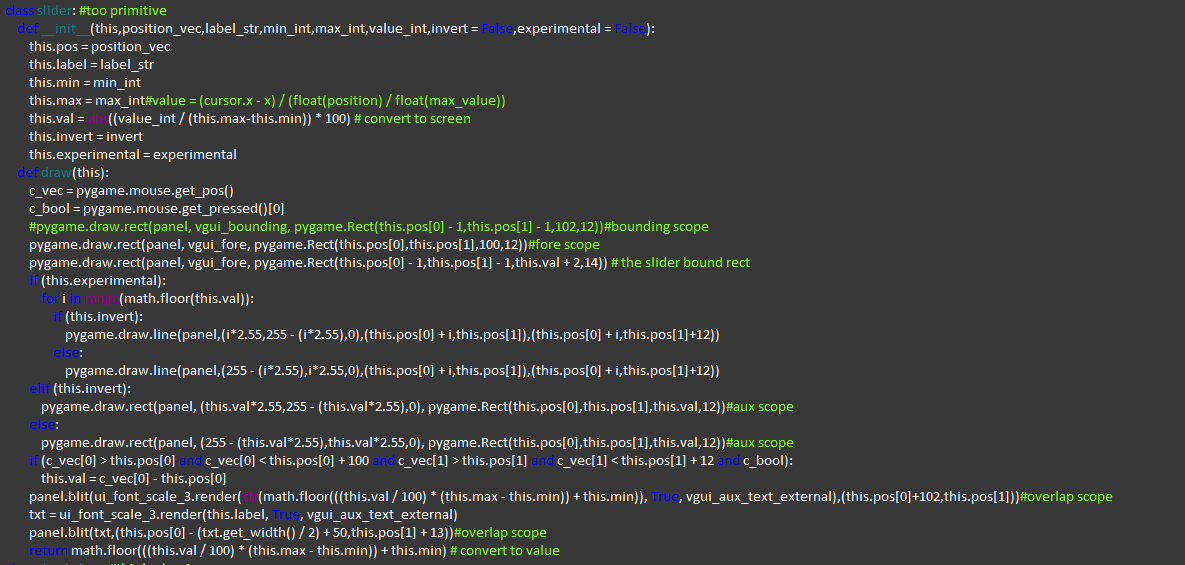
**Iteration 2:**

****

**Why?**

In response to user feedback, the design was adjusted for visual appeal, changing a value is now also a lot easier as the user can press anywhere inside the slider to change its value instead of grabbing the slider rect

### Iteration 2



Changes from iteration 2 -> final:

* **Change 1: abstraction**
  + I disliked how the old sliders worked so decided to port over a technique Ive used in other projects in where instead of a grabbing a slider and moving it across a border I decided to just check for clicking inside the element then grab the x value of the mouse relative to the start, then draw a rectangle corresponding to the position pressed and then updating the value.
* **Change 2: the value system**
  + The previous version worked by measuring the distance between the mouse and the start then returning that raw value, In the revision of this I made the minimum and maximum values get passed as parameters to the constructor method, this allowed for a much wider degree of things to be modified for instance GPP is a value from 0 to 1000 and using the old system it would be very impractical to have a slider that’s a thousand pixels long and so a math equation is used to convert it whilst keeping the same size.

Result:





### Testing

Due to previous testing on the button class, the label attribute has been amended to accept all data types already.

**Test 1: min value larger than max value**

Text

Description automatically generated

Result: pass

output:



**Test 2: set initial value larger than max value.**

Result: fail

Output:

Text

Description automatically generated

Reason:

Text

Description automatically generated

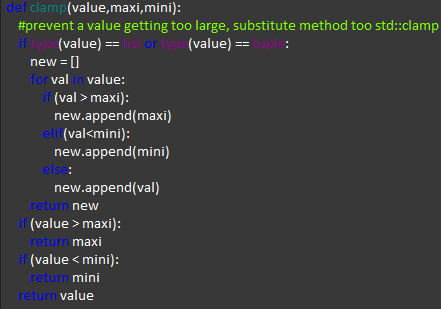
The colour displayed on the slider element is derived from the current value and the max value, values larger than the max value will raise a colour value larger than 255, causing this error.

Amendments:

Text

Description automatically generated

The colour values have been clamped to prevent their values going above 255, therefore preventing the error.



Changes to the clamp function had to be changed for efficiency when clamping tuples or lists, the new code iterates through lists or tuples and amends the violating values back as either the maximum or minimum value back as a list.

Output:

A picture containing graphical user interface

Description automatically generated

## Sub class 3: checkboxes

### Iteration 1

**Text

Description automatically generated**

Changes from iteration 1-> 2:

* Improved maintainability

**Old:**

****

**New:**

****

**What’s changed?**

Mouse position is now sampled inside the method instead of pulling from global, the mouse pressed data now only gets the left mouse state to prevent confusion when trying to check for a mouse press

**Why?**

As shown on the above source it will reduce the numbers of arguments taken by the draw method, this allows ui elements to be iterated through in a list and called as now each UI element will only take their object as context.

It also improves the readability of the code and allows me to maintain it easier as there are no longer confusing dereferences such as the one below



* Improvements to visual style

**Old:**

****

**New:**



**What’s changed?**

Text is now aligned better on the UI element, instead of drawing a cross to show state it will now change the colour of the inner Rect from red to green

**Why?**

Mostly for visual appeal as text now appears much more seamless with the element, its also now more distinguishable for people with visual impairments

### Iteration 2

Text

Description automatically generated

Changes from iteration 2 -> final:

* **Change 1: complete revamp of the visual style** 
  + The previous version used a cross to illustrate whether the Boolean value is true, to simplify this for people to understand it faster I made it so on a Boolean value of false it draws a rectangle in the middle as red and when its True it will draw a green rectangle.
* Really simple design, not much else to change





### Testing

From previous tests, the constructor method was changed to accept any data type.

Test 1: invalid state (could be caused by corrupted JSON)

Graphical user interface, text

Description automatically generated

Result: pass

Output:



Why does this work?



The code does not explicitly check for True or False values and therefore this.state does not have to be a Boolean value.

## Sub class 4: colour selection

### Iteration 1

**Text

Description automatically generated**

**Changes from iteration 1 -> 2:**

* Improvements to visual style:

**Old:**

**Shape

Description automatically generated**

**New:**

**A picture containing shape

Description automatically generated**

**Whats changed?**

Both red and green are now scaled by default on the x and y axis to show a more full colour range, smaller design, there is now a preview position for the currently selected colour. The interface will now stay open unless clicked off.

**Why?**

Visual appeal, professional look, more user friendly as the interface doesn’t close and the user can refine their choice. Smaller design to allow for more space on the screen for other elements.

* **Conversion to OOP:**

**Whats changed?**

Instead of calling a function, the colour selector is now contained inside an object allowing for other UI elements to be contained within

**Why?**

Improves maintainability of code, blocks of code are now in clearly defined methods inside the class instead of calling random functions, improved efficiency of code as handling the opening and closing of the element is now self contained and wont require multiple lines of code to handle for each element.

### Iteration 2

Text

Description automatically generated

**Changes:**

* **Change 1: addition of modes**
  + The first version included one color setting looking something like this:

**![A picture containing text, electronics, display, screenshot

Description automatically generated](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAADgAAAAzCAYAAADCQcvdAAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAJcEhZcwAADsMAAA7DAcdvqGQAAACnSURBVGhD7ZoxDsIwFMU+SL1BjphDZIalh83M0gGs0gGpQmI1el78hmSIvObSWnvWB733WtdbLeylNrXv41FXfGY/tfkNXx74w2WDIQXVhhRUG1JQbUhBtSEF1YYUVBtSUG1IQbUhBdWGFFQbUlBtSEG1IQXVhhRUG1JQbUhBtSEF1YYUVBtSUG1IQbUhBdWGFFQbUlBt+PuCpx+/c85jvRljHMtI1QvVLgVroGG0CQAAAABJRU5ErkJggg==)**

However, some colours where inaccessible such as black or red and so I added a system where the blue value could be offset to provide more of a range of colours which are as follows:

No blue:

![A picture containing text, electronics, monitor, display

Description automatically generated](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAADkAAAA6CAYAAAAKjPErAAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAJcEhZcwAADsMAAA7DAcdvqGQAAAECSURBVGhD7ZshjsJAGEZnSdCYMZg1XAfRQ/QQeEwP0UNUcCIMpmYNdvmZsMkjS4pZMd/s95KXTvKn4mVmXPuRc/5OjbN6PJvmaSf7vn+s2uJX5DAMaR1rdY+HQ5qmKVavjuurNxQFjvy6pjSHl/Accla9YHkng5xPxQJntQveRj7BWe2Ct5HzvC8WOKtdsBi52cVxDbfhZ8hZ9YLlnVQWOFLaoOu6YtOR4zgWfVylBY6UFjhSWuBIaYEjpQWOlBY4UlrgSGmBI6UFjpQWOFJa4EhpgSOlBY6UFjhSWuBIaYEjpQWOlBY4UlrQ9OfZ989b7vhvglb4F5FV3cmfO/S3pHQDHHpZm+t9rt4AAAAASUVORK5CYII=)

Inverted blue:

A picture containing text, monitor, electronics, display

Description automatically generated

These new settings allow for more of a range of colours and can be changed using a selection interface (see below)

* **Change 2: addition of a button to activate the interface**
  + The previous system just had the colour selection system being constantly rendered, this was both space consuming and hit performance as by looking at the code you can see it contains unfortunately a large amount of iteration and which tonnes of these rendering it would significantly reduce the performance of the simulation. The change can be viewed below.

Result:

Graphical user interface

Description automatically generated with low confidence

### Testing

From previous tests on other UI classes I realised the following changes had to be made: clamping of the colour vectors and string redefinition of the label vector

Therefore the code has been amended as so:

Text

Description automatically generated

Besides this, there is no room for human error in this.

## Sub class 5: both selection interfaces

Text

Description automatically generated

Text

Description automatically generated

### Changes

* **Change 1: selection interface singular preview**
  + The previous system would have following the drop down menu being activated the currently selected point be previewed on the header instead of in the drop down which was admittedly confusing and inefficient and so in the next revision I made the currently selected interface listed on the drop down but highlighted (see below)

Graphical user interface, text, application

Description automatically generated

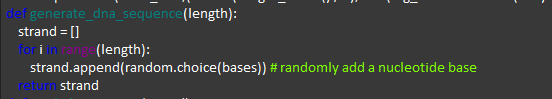


Graphical user interface, text, application

Description automatically generated

# System 2: the mutation system

## Function 1: generation of a dna sequence



Explanation:

A function that generates a dna sequence taking in the length of the sequence as a parameter.

The bases are added from an array called bases containing

A – adenine

T – thymine

G – guanine

C – cytosine

Where A and T and G and C are complementary (bind together)

**Example usage:**

Generate\_dna\_sequence(8)

**Return value:**

“ATGCATAA”

### Testing:

test 1: invalid length

Result: success



Output:

[]

## Function 2: read\_dna\_protein [REDUNDANT]

Text

Description automatically generated

Explanation:

The simulation uses a very simplified version of proteins, proteins are the fabricated and folded result of a dna sequence that perform a function and in this case will modify values such as an organisms sight, speed etc

The function returns an active site which is the area it will bind too e.g sight and also an opcode which is the effect it has on the area e.g +5 or -7 with the system using sign and magnitude

**Example usage:**

Read\_dna\_protein(“ATGCATAA”)

**Return value:**



In this case it would bind to active site 1 and have no effect, this is intended as proteins are not meant to always be helpful and disadvantageous alleles are part of natural selection

### Testing

**Test 1: invalid dna strand**

Text

Description automatically generated

Result: Fail

Text

Description automatically generated

**Amendments:**

**Graphical user interface, text

Description automatically generated**

**Test 2: dna strand that isn’t 8 long (a mutated strand)**

**Text

Description automatically generated**

Result: Fail

Text

Description automatically generated

**Amendments:**

**Graphical user interface, text

Description automatically generated**

## Function 3: read\_dna\_binary

Graphical user interface, text

Description automatically generated

Explanation: This is not true to life at all as dna is never read as a binary value and instead as a triplet codon, this is one of the limitations of the design as its such a complicated system I had to abstract processes to fit the needs of the project.

**example usage:**

read\_dna\_binary(“ATGCATAA”)

**return value:**

****

### Testing

**Test 1: Invalid dna strand:**

**Text

Description automatically generated**

Result: Fail

**Text

Description automatically generated with medium confidence**

**Amendments:**

**Text

Description automatically generated**

## Function 4: creating an actual mutation

Text

Description automatically generated

**Explanation:**

The strand to be mutated is passed as a parameter and a copy is made, an integer value is then generated to dictate the type of mutation, once the strand is mutated the mutated version is returned and the code is modified.

**Changes:**

* **Change 1: new mutation types**
  + I researched new types of mutation adding duplication and inversion mutation, where duplication involves a duplication of a base and inversion where the entire strand is flipped.
* **Change 2: fix for 1 base strands**
  + There was previously a bug where strands of 1 base could be deleted and therefore completely delete a section of the genes, this was fixes by randomising a different range when the length Is 1 to make deletion mutations impossible, I also added a system where different mutations are more rare/frequent as to reflect real life.

**Example taken from the log**



### Testing

**Test 1: invalid dna strand**

**Text

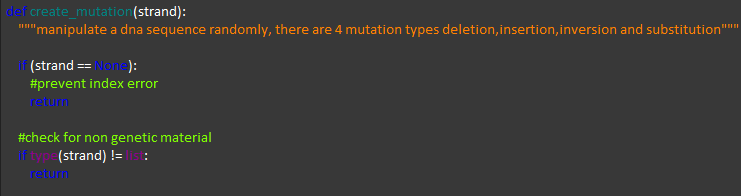
Description automatically generated**

Result: Fail

Text

Description automatically generated with medium confidence

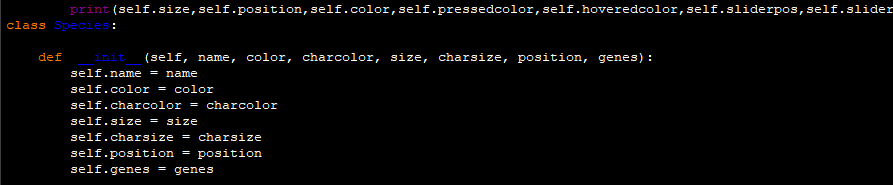
**Amendments**:



# System 3: the main herbivore class

## Sub System 1: the constructor method

**Iteration 1:**

****

**Major changes from iteration 1 -> 2:**

* Removal of all arguments

**Why?**

all arguments in iteration 1 where randomised internally anyway, to improve readability and allow for easy management of the herbivore class I removed all arguments to generate a completely random organism by default, if needed then modifications can be made before addition to the entitiy lists e.g when birthing a new creature

**old:**

****

**New:**

****

* Changes to how genetic code is stored

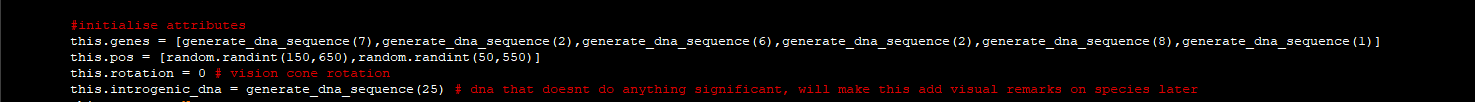
**Whats changed:**

Before, genes where passed as an argument as a sequence of integers, now they are stored as lists of nucleotide bases represented by ATGC characters, the dna is now also stored in an array of genes instead of one long list.

**Old:**

****

**New:**

****

**Why?**

To improve the accuracy of the dna, dna is not stored as numbers and so to increase

Accuracy I converted to using nucleotide bases which (somewhat) mimic dna

**Iteration 2:**

**Text

Description automatically generated**

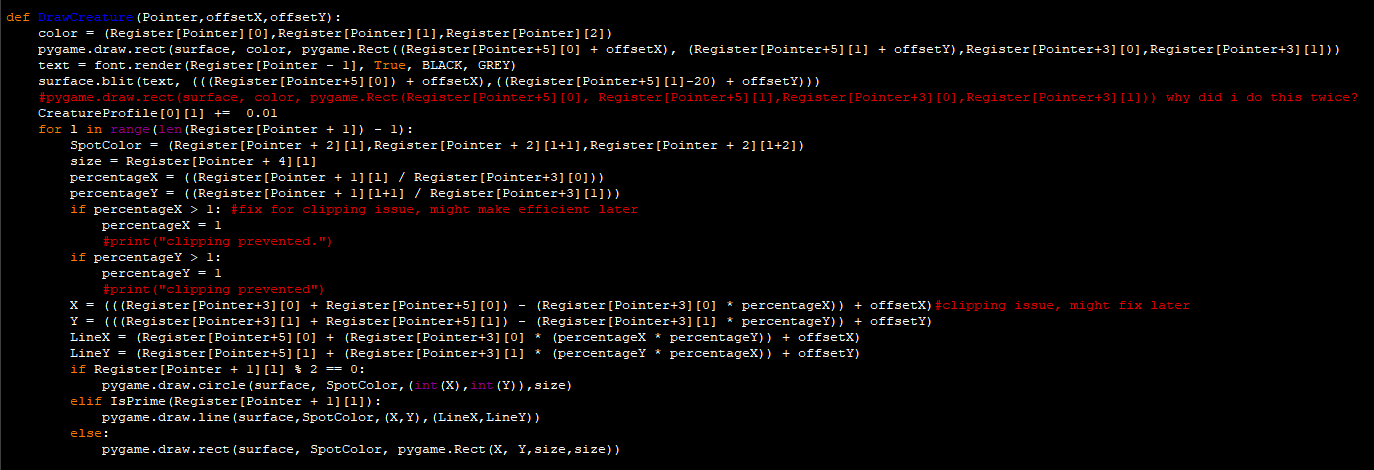
Explanation: the constructor method determines the attributes of the organism for instance its genetic traits, position etc. it also contains the Boolean values as to whether the organism is decaying and releasing its biomass back into the ground

**Changes:**

* **Change 1: change to how genetic traits are handled**
  + The previous system would have all the traits determined in the constructor, unfortunately this meant as the creatures genes changed due to mutations the traits would not change and could only be modified on birth. The new system edits the traits with the refold() method meaning the traits are determined at set intervals to change them from mutations

## sub system 2: the draw method

### iteration 1



**Major changes from iteration 1 -> 2:**

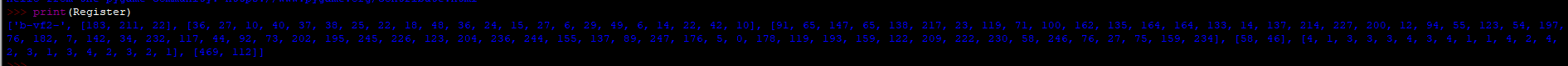
* removal of visual remarks:

**why?**

Due to my decision to make the organisms smaller to fit more on screen, I decided to remove visual remarks on creatures since its barely visible on small organisms and takes up unnecessary cpu time.

* Addition of OOP instead of a highly inefficient list of organism data

**Old:**

****

**New:**

****

**What was changed?**

Organism data now contained in a class, with self-contained methods for drawing the organism and storing of the data inside attributes

**Why?**

The use of storing the raw data in a list meant that the simulation ran quite slowly due to it needing to traverse upwards of 20 entries per organism, with the organism being contained in a class that drops to 1 and reduces the need for for loops elsewhere to keep cpu time low.

### Iteration 2

**Text

Description automatically generated**

Explanation:

I decided to make the draw method separate from create\_move(), this is to ensure that the creature is not redrawn multiple times creating unnecessary processing time so instead there is a separate loop to draw the creatures first and then compute the logic behind them. The creatures can be drawn with either a grey or green body depending on whether they are decaying due to being dead.

**Changes from iteration 2 -> final:**

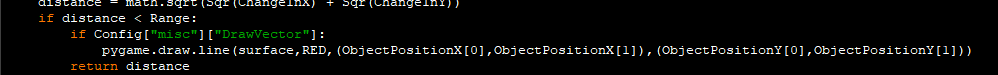
* **Change 1: addition of nose logic separately in a variable:**
  + I recently optimised performance and to do that I filtered through objects behind the sight cone and to do this the nose position is sampled, this could not be done efficiently if the nose position was computed inside the pygame function and so I now store the value in a variable to prevent it being called twice and creating a performance drop.

### Testing

There are no points of failure for the OOP draw method (there are no arguments to pass)

## sub system 3: sight check

#### iteration 1



Major changes from iteration 1->2:

Addition of triangular vision cone:

**What was changed?**

Instead of a simple distance check for a square shaped vision cone the vision cone

Now uses a complex algorithm too detect whether the target is inside its vision.

**Why?**

For realism, organisms do not see in squares and so for accuracy the vision cone was

Created to mimic normal eye vision areas

Movement of function to a class method:

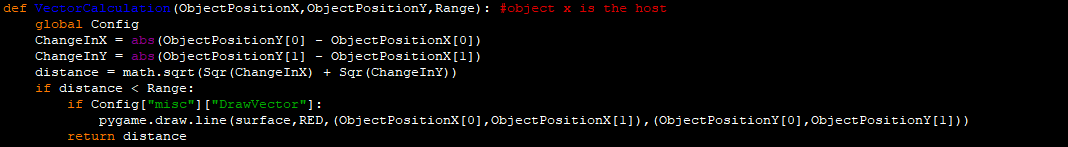
**What was changed:**

sight\_check is now contained inside the organism class

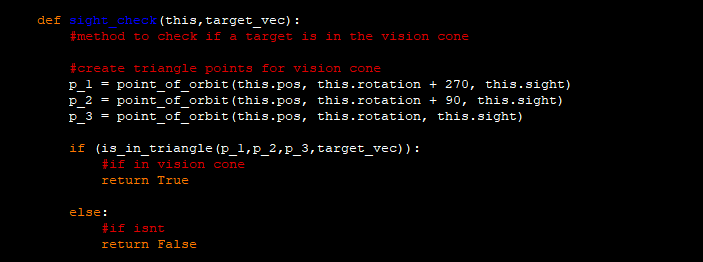
**why?**

Easier maintainability of code, since only organisms need to access it it makes sense to have it inside only there class

**Old:**

****

**New:**

****

#### iteration 2

**Text

Description automatically generated**

Explanation: this function can be broken down into many more sub procedures which will follow however the basis of this is, the area of the triangles resulting from the points of the triangle like this,

Shape

Description automatically generated

The premise is that the triangles resulting from this will have areas summing up to the area of the triangle if they are inside the triangle and wont if they aren’t.

### Sub system 1 of sub system 3 for system 3: getting an orbit point

**Text

Description automatically generated**

Explanation:

The purpose of this function is to get a point relative to the argument “host vec” from an angle and a radius of orbit. It first creates a tangent off the point like this,

Shape, circle

Description automatically generated

using sin and cos rule of the angle which is converted from degrees into an integer useable by sin and cos.

An offset is then added like this,

Shape

Description automatically generated

**How does this apply to sub system 3?**

the vision cone is made up of a triangle and its points are generated using this function as so,

Shape

Description automatically generated A screenshot of a computer

Description automatically generated with medium confidence

Where the offsets on the angle are what makes the triangle.

#### Testing

Test 1: negative radius

Text

Description automatically generated

Result: pass



Test 2: negative rotation ( could occur as a result of the rad\_to\_deg function)

Text

Description automatically generated

Result: pass



### Sub sub system 2 of sub system 3 of system 3: the “linking” function

Explanation:

Essentially the function that links all the sub procedures together and does the math to decide whether the target is in the cone.

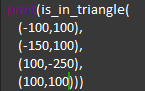


Area\_of\_triangle is pretty much just base x height / 2 so no need to explain.

The function is checking as mentioned earlier whether the sum of the areas of the different triangles are equal to the vision cone triangle.

#### Testing

Test 1: negative coordinates



Result: pass



## sub system 4: create move the “brain” of the organism

**Text

Description automatically generated**

the internal logic of the herbivore, the rest is computed in the external simulation loop.

First it checks if the target is actually still in the object array as to prevent an error,

It will then calculate the organisms next move using rad\_to\_deg or radian to degrees and think\_next\_move which I have been using to modulize the path algorithms and eventually port to machine learning. There will later be an explanation of rad\_to\_deg following this.

The only other notable logic in this algorithm is the one that checks for whether the organism has reached its target in which it will force it to either decay or consume it whole depending on whether it’s a herbivore or carnivore. The organism will then take in the nutrients derived from the carbs and protein of the creature and any excess is converted to biomass.

### sub sub system 1 of sub system 4 of system 3: radian to degrees.

Text

Description automatically generated

Explanation:

This function uses cosine rule to calculate an unknown angle from a fake triangle. An illustration of this can be seen here.

Shape

Description automatically generated

The values of the lengths of the triangle are calculated using pythagoras and the opp and adj can be deduced easily using the x and y values of the point

An offset is then added to account for what quadrant the point is in e.g,

Diagram

Description automatically generated with low confidence

#### Recode:

Text

Description automatically generated

Explanation:

I recently looked into pygames math libraries function to increase the efficiency of my code, since rad\_to\_deg is called multiple times a tick sometimes even hundreds I knew I could free up cpu time by making it more efficient.

First, the math atan function is used to measure the angle from a radian to the hypotonuse much like this section of redundant code,



Then, the angle is converted to degrees using math.degrees once again like the second half of the redundant method.

Text

Description automatically generated

#### Testing

Test 1: adjacent coordinates (sometimes occurs randomly)

Text

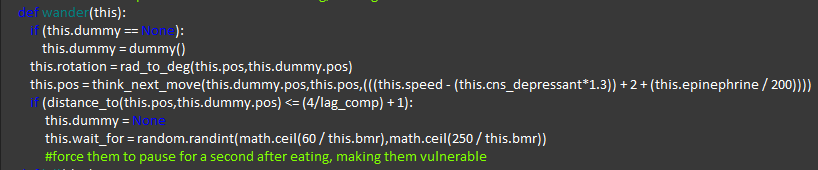
Description automatically generated

Result: pass



The relative angle should be 0, since it’s a straight line.

## Sub system 5: wander method

****

Explanation:

Creates random targets for the organism when it doesn’t have any food to move towards, it uses a dummy() container to store the fake target as to make it overridable by other methods.

Works by first generating a new fake target if need be, then as like before updating rotation and position using rad\_to\_deg and think\_next\_move.

It then checks for collision with the target removing the old one.

### Testing

There are no points of failure in this function.

## Sub system 6: decay

Graphical user interface, text

Description automatically generated

Explanation:

Since organisms store excess energy as biomass (fat,muscle bone etc), it must be partially returned to the environment as it does in real life using decay.

First its checked whether its biomass has been fully depleted in which case they are removed, any remaining biomass is then released into the environment slowly in the form of plant matter (resulting from the fertilized soil).

### Testing

There are no points of faulure in this program

## Sub system 7: run

**Text

Description automatically generated**

Explanation:

The flight or fight response causing vasoconstriction, faster movement and stimulation of bmr. This algorithm works by perfectly inverting the path of the organism chasing it and done so like this



To prevent the creature from running away every time, there is an autonomous system to control hormone levels where after prolonged exposure the variable cns\_depressant is increased simulating exhaustion. There is also deviation in the rotation of path as to make the path it takes slightly random due to ofcourse in real life a creature wouldn’t perfectly run away down to the degree.

### Testing

Test 1: organism being chased off the map (if statement triggered)

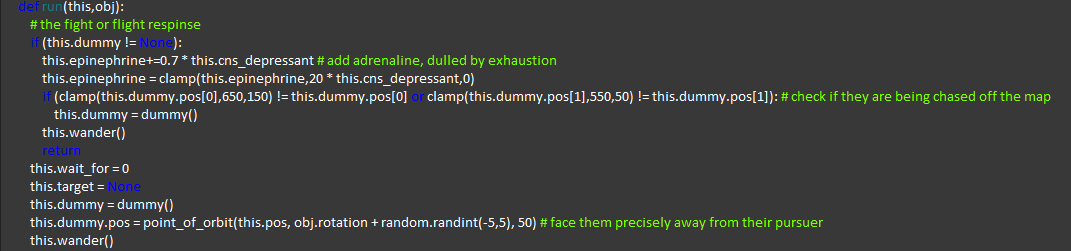
Result: Fail

The organsim will remain stationary due to this line of code



After the if statement is triggered the organism will begin to move to a random coordinate, however since this coordinate no longer triggers the if statement the above line of code will be executed, and since the pursuer hasn’t changed direction the organism will be in a continous loop of triggering a new random coordinate and being chased off the map by the generated coordinate.

Ammendments to code:



Organisms will now fully commit to dummy targets meaning any randomly created targets as they get chased off the map will no longer be overridden, resolving the issue.

## Possible future adaption of think\_next\_move

**Text

Description automatically generated**

Currently, think next move is a very basic function that will return the next place to move towards a target,

Text

Description automatically generated

As can be seen its fairly primitive by design due to me knowing this would possibly be replaced by machine learning model.

I have done research and begun experimenting and gathering skills to implement this feature and have came across two potential solutions,

### Possible model 1: basic keras model

Code for this model can be seen above, at this current point in time it is far from optimised and can so far only move in the + + quadrant and in addition to this the response time of the model is 5ms (CPU) and around 0.3ms on my GPU much slower than what could be implemented right now as a viable solution.

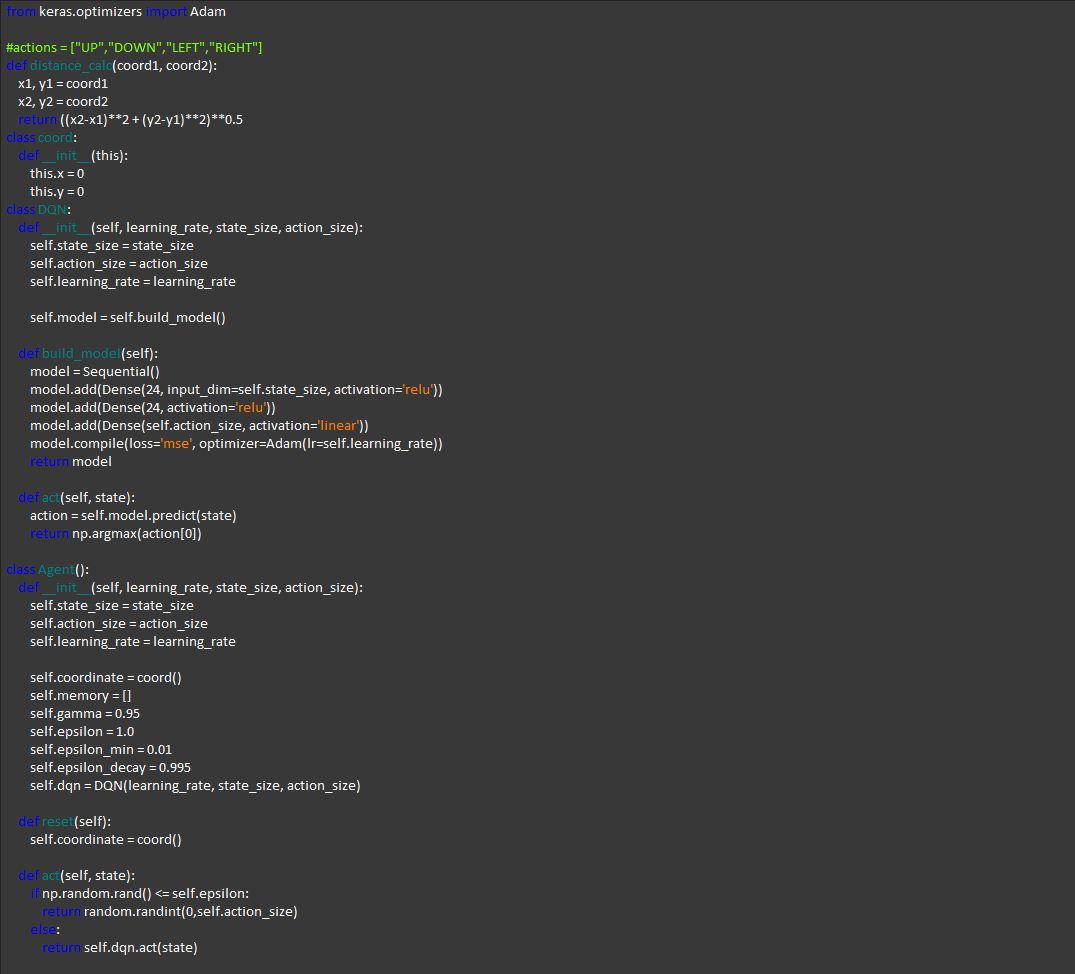
In its current state it uses a sequential keras model with 16 hidden neurons and 2 output neurons to output both the x and y coordinate, the training data demonstrates a pattern that y values should be increased when lower than the target and vice versa.

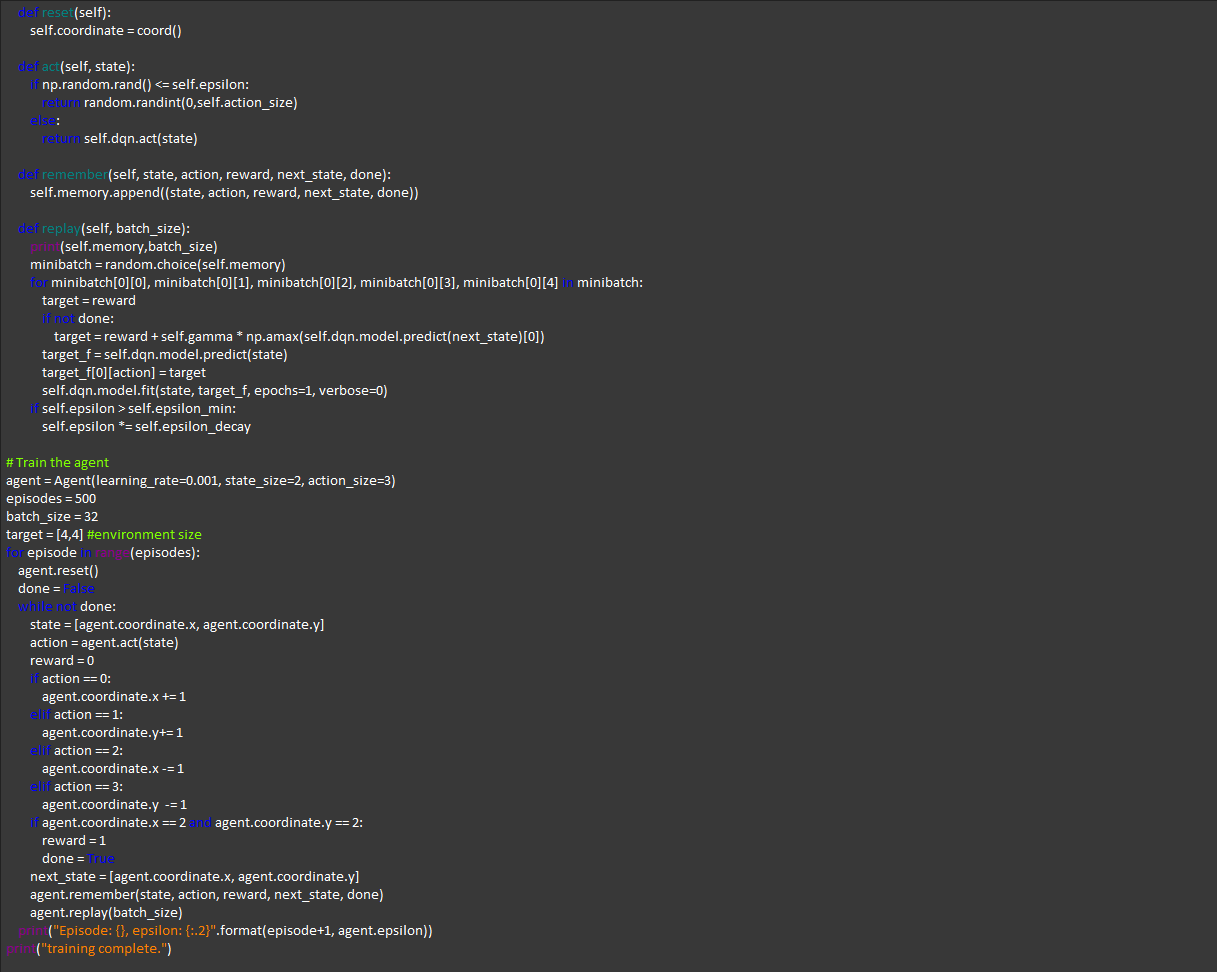
In its current state, I highly doubt I will find a use for it in think\_next\_move and will likely be experimented with in the metrics system as 5ms is perfectly acceptable in that context since its not being called multiple times a tick.

### Possible mode 2: deep q learning

This model consists of a standard keras model with a low neuron count however instead of feeding in training data the model will return an action to control an organism, depending on what it did when it performed that action it the reward will be adjusted and the model will learn, if I managed to implement this without major performance issues this would certainly be a major addition to this project. 2 new major selection pressures could be added including niches and social interaction.

Since writing this ive looked into it and produced this example,





This example works by using a basic keras model to compute an action for the agent to complete and then the DQN algorithm to provide feedback using a reward, typically training data would be pre defined however in a DQN model the training data is learned through first random actions defined by epsilon (random chance) and epsilon\_decay (the depleting chance of a random action)

# System 4: the simulation loop

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

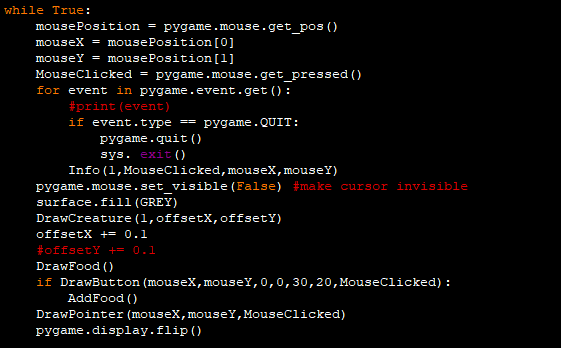
Explanation:

An external loop outside of the organism class to control all other logic,

As mentioned earlier it first draws all the creatures on the board as to prevent overdrawing, from there first herbivores and then carnivore loops are performed to do hormone management and target selection etc which will be dwelled on below.

## Sub system 1: the herbivore loop

#### Iteration 1

****

**Major changes from iteration 1->2**

Addition of target selection:

**What changed**

Iteration was created as the first version, without any organism behaviour and so

Target selection was not needed. Iteration 2 then added a loop to choose the closest

Entity for target selection.

**Why?**

Entity behaviour is vital to simulation for its purpose, without it the organisms would

Not interact with there environment

#### Iteration 2

**Major changes from iteration 2 -> 3**

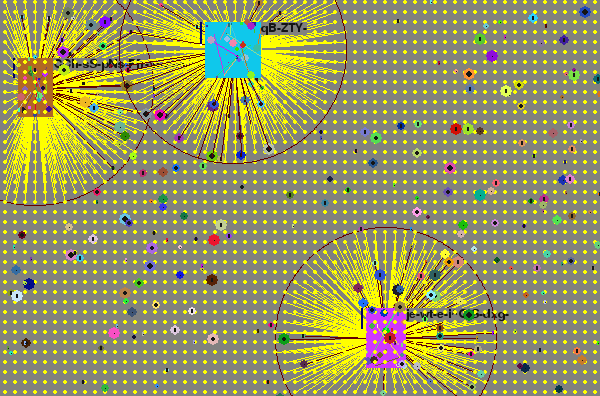
Addition of nodes

**What was added**

Nodes are consistently spaced points on the simulation area, they serve as points for

The organism to move when they cant find any targets

**Visualisation:**

****

Addition of target memory:

**what was added**

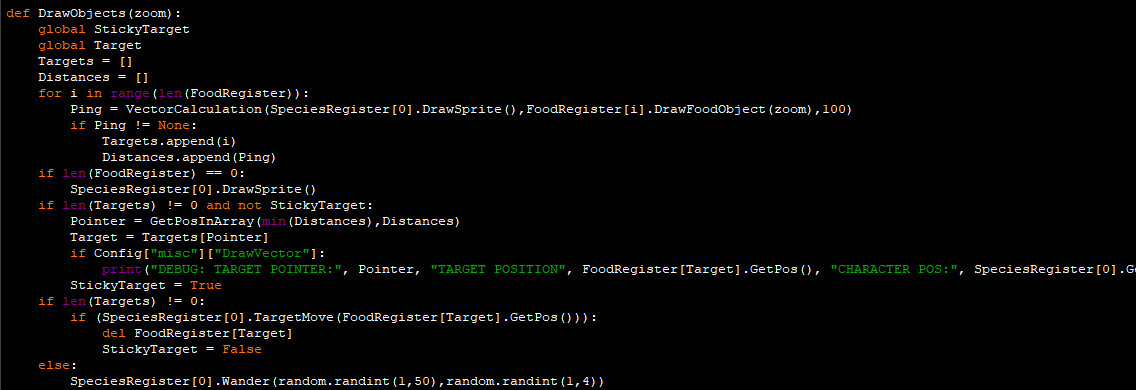
species will now remember previous targets much like a real creature would

previously seen objects will be stored in a memory list contained in the organism class, if needed then a random target will be sampled from memory

**why?**

To further improve the accuracy of the simulation by mimicking some intelligent

Behaviour by the organisms

****

#### Iteration 3

****

**What was changed from iteration 3 to 4:**

* More efficient code usage

**what was added**

removal of pointless function calls that only have one line of code, e.g the method .deletetarget was being called in the simulation loop even though it only sets target to None, instead its now done directly.

**why?**

Free up cpu time for faster simulation speeds

* New usage of radians

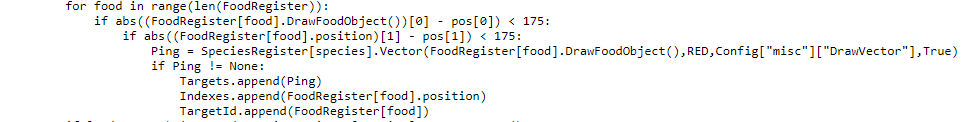
**What was added**

Instead of checking in every direction to sample targets, the organisms only searches infront of it to quickly exclude targets

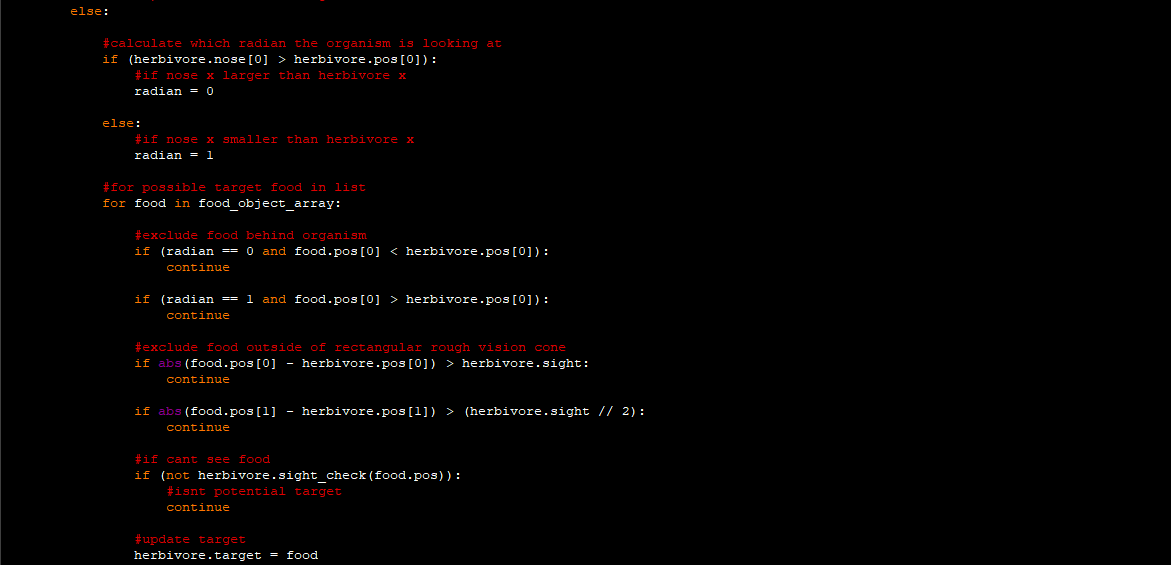
**Why?**

Improve simulation processing time by sight checking less irrelevant targets

**Old:**

****

**New:**

****

* better handling of target collisions:
  + **whats changed**

to prevent collisions (multiple organisms have the same target) (causes an error when trying to delete the food once reached), collisions are now handled every tick by checking for the food In the object lists in create\_move and removing the target if not.

**old**

**Text

Description automatically generated**

**New:**

**Graphical user interface, text

Description automatically generated**

**Why?**

Prevents organisms moving to an already eaten target, to improve the accuracy of the simulation.

#### Iteration 4

Text

Description automatically generated

Firstly, decaying organisms are handled to start releasing biomass into the environment



In which the decay() method is called on the herbivore object (referenced above)

Then, random mutations are handled using a tick based system, when running at the frame cap of 100 2000 ticks is approximately 10-15 seconds, to account for lag the multiplier lag\_comp is applied to make sure it happens at a constant rate

Text

Description automatically generated

The chance is also dictated by the value of the ui element slider labelled as “mutation chance random”

Afterwards a section of code dedicated for cleanup is run where the countdown on waiting organisms is decreased and hormone levels are regulated.

Text

Description automatically generated

As can be seen, as epinephrine levels are depleted then the variable cns\_depressant is incremeneted as to simulate exhaustion, once all epinephrine is depleted the organism begins to recover and so exhaustion levels fall.

Lastly, the herbivore loop is responsible for reproduction and optimisation of target finding and is essentially what feeds information on targets for the herbivore to verify

Text

Description automatically generated

The first system involved in this area of the herbivore loop is reproduction management more specifically hatching old eggs and laying new ones in accordance to both lag\_comp and balance variables in order to maintain ecological balance and a constant rate no matter the level of lag.

**Changes:**

* Optimisation to target selection: addition of radians
  + The vision cone faces one way and so to optimise the algorithm to free up performance I made all food behind the vision cone automatically excluded before the complex vision cone calculation is done, this is done by checking whether the is behind the nose position.

## Sub system 2: gathering metrics

**Text

Description automatically generated**

Metrics are gathered at a constant rate of 15 even ticks not accounting for lag to make sure the metrics arent flooded with useless identical data samples.

Currently the program measures:

* TSC – total stomach content
* NPP – net primary prpduction
* Some genetic traits to show selection like sight,bmr

The metrics are gathered inside the different loops such as the herbivore or food loops and are gathered accordingly for instance food is measured by summing up the total energy content of food and dividing by the length of the food array.

# System 5: the log

Text

Description automatically generated

Explanation:

Most of this is computed inside the log\_entry class which I will explain later, this loop is responsible for iterating through the array of log entries and call the handle\_hover method.

**Changes:**

* Addition of log scroller:
  + A log scroller is drawn when needed after log entries start drawing off the screen, the scroller is drawn inside the log manager function and controls the variable log\_var to add an offset to displayed log entries.

## sub system 1: the log entry class:

Text

Description automatically generated

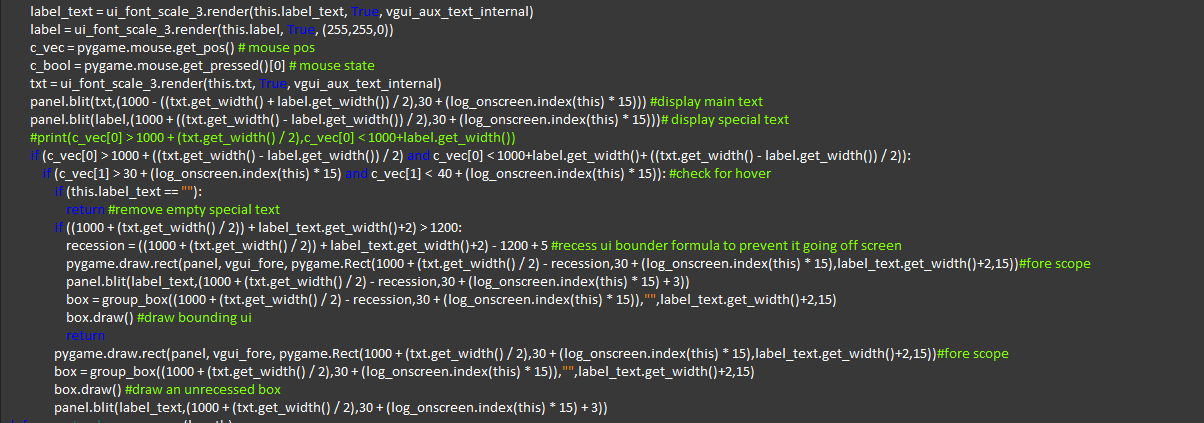
Explanation:

the first system is the application of the log\_var offset, log\_var is returned by the log scroller element and controls visible log entries by applying an offset on this code,

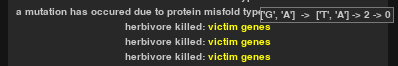


The log\_onscreen is a sub array derived from the main array and will without an offset pick the last 50 entries, by controlling log\_var the entries are pushed back allowing for the viewing of old entries.

The next system is the special text,



**Changes:**

* **Change 1: addition of log scroller:**
  + Previously, I would automatically delete old entries when they started to go off screen however this wasn’t idle for a few reasons as the log would move whilst trying to read an entry as new ones enter and especially as the simulation progresses it could be seriously hard to read anything put in the log, this is why I redesigned the log to be completely static until scrolled meaning you can both read old entries and not be interrupted when reading current ones.
* **change 2: addition of special text:**
  + during the development of the log system I noticed how particularly long information would overflow from the log when rendered on the screen, so I developed a solution where information could be hidden and revealed when hovered to prevent the log from taking up the entire screen.
  + ****

## Testing

**Index out of range**

Derenfeced pointer for non existant data

How it was resolved:

The error was resolved using this statement

Text

Description automatically generated

Since log onscreen is sampled by 50 data points a check is implemented to check if there is 50 data points to sample.

**Index error cant find “this” in log\_onscreen:**

Offscreen entries still being handled

How it was resolved

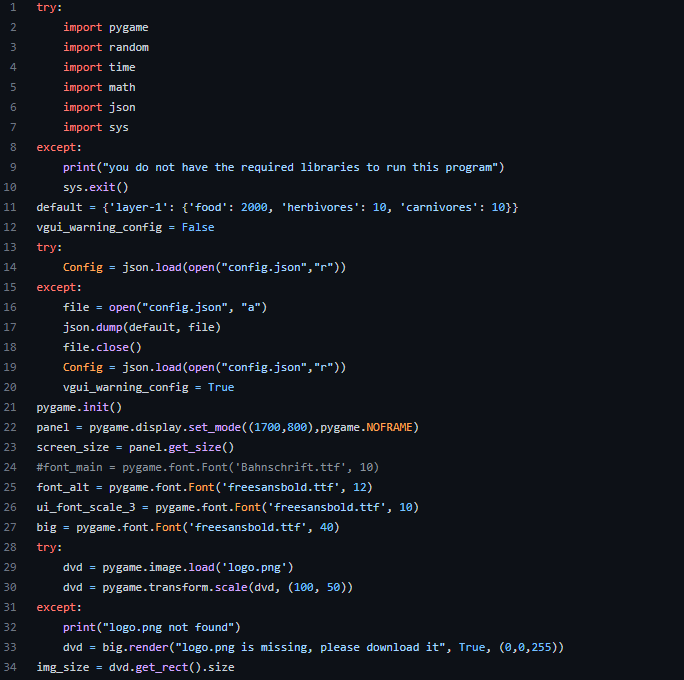
it was resolved using the following code



This prevents onscreen logs being handled and therefore they wont be indexed into the sub array, preventing the error

# pre run checks:

## sub system 1: config and logo file checking and loading



**Explanation:**

Text

Description automatically generated

Since pygame is not packaged with the standard libraries in python it must be installed seperately in order for the program to run, however if the program tries to import the non existant library it will throw an error message that will likely not be understood by most users. In order to fix this issue and to inform users that they need to install the required libraries I implemented a catch statement on imported libraries to print my own error message as to inform users on the pre requisites of the program.



A common issue I had seen with users trying to run the program was they had only downloaded the program file without the config file to store there selected preferences therefore causing an error when python tried to create a handle to a non existent file. To solve this problem I first implemented a catch statement to try and create a handle to config.json and load the data which would work just fine if the file existed, if the file did not exist or config.json wasn formatted correctly it would then create a new empty config.json file within the python open() statement using os.mkdir before dumping the default config as stored below.



Using this solution I was able to solve issues from people loading improperly formatted config files and without config files at all, the solution also informs the user on the gui as shown below after the error is resolved.

Graphical user interface, text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Logo.png is an asset for the GUI and as with the config file I had frequently encountered issues with users not downloading the assets from the repository and therefore would encounter an error. Much like before an attempt is made first to load the image before substituting the image with rendered text informing the user of the missing asset.

**Changes:**

**Text

Description automatically generated**

**Text

Description automatically generated**

An issue I had faced witg the previous solution was it being unable to run from visual studio due to the open() function not handling relative paths properly, under normal conditions open(“config”.json) would look for config.json in the solution folder however this was not the case and so I manually added the root path using os.getcwd() to get the solutionfolder then added /config.json to try and locate a config file in the solution folder

# Experimental features

## sub system 1: ray tracing

### iteration 1

**Text

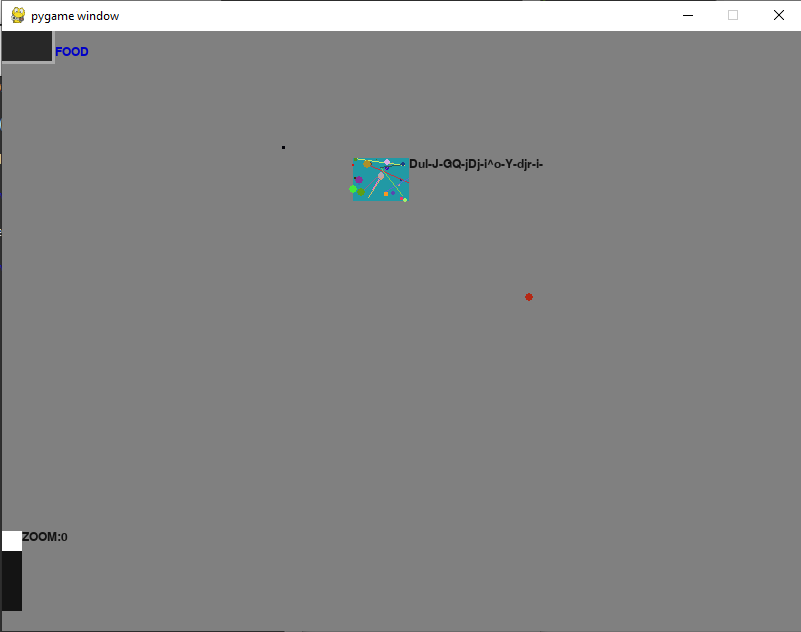
Description automatically generated**

**Explanation:**

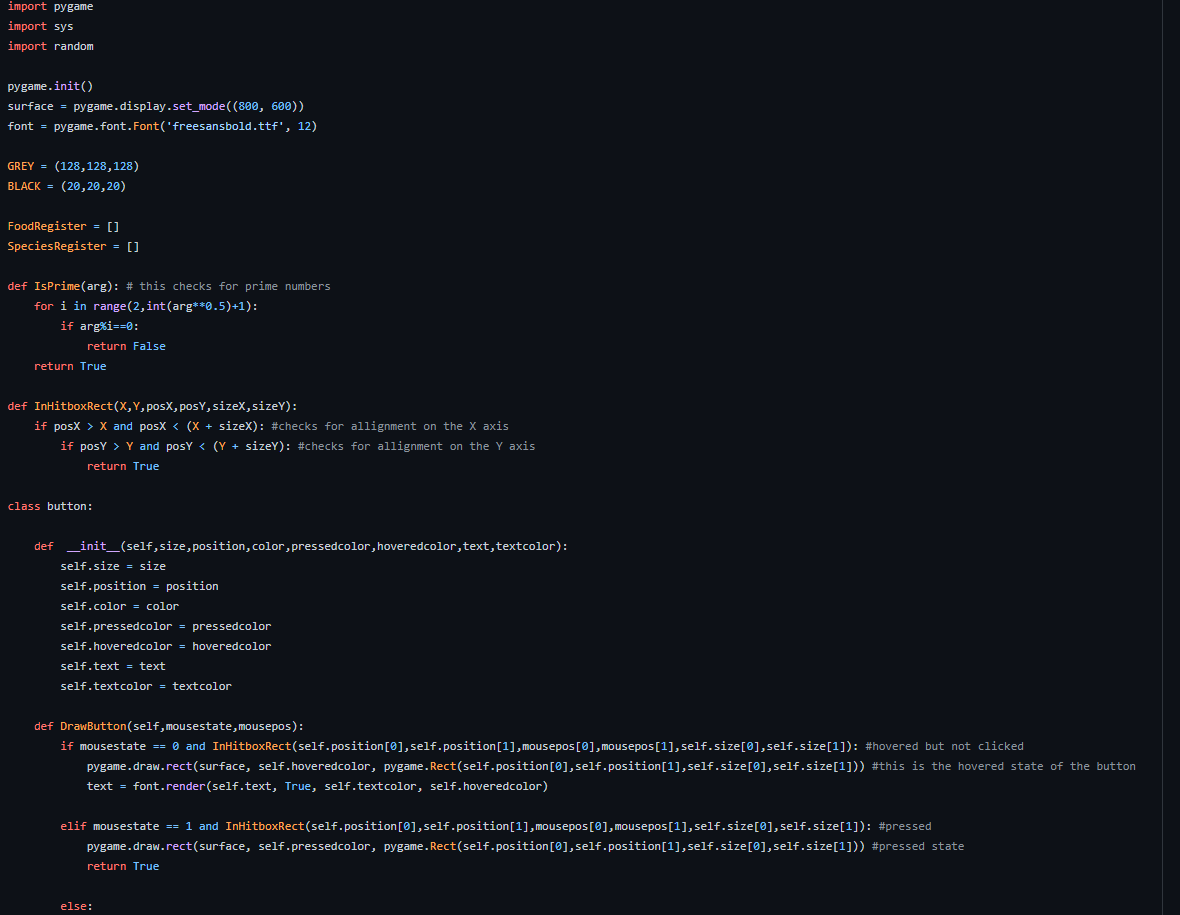
In this program, ray tracing is used for precise vision checks to check if an organism can actually “see” the target its trying to consume. The algorithm works by tracing a ray (just a line) step by step until the ray hits the target, if the ray didn’t hit anything else e.g a food object then the ray hit and the algorithm will return True to indicate it can see the target, the ray can also miss entirely due to floating point inaccuracies in which case it bounces.

# Time line

## Prototype 1



### Code



Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

## Prototype 2

Diagram

Description automatically generated with medium confidence

### Code

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

### Major additions from prototype 1 -> 2

* **Path finding algorithm**
  + **How was this achieved?**
    - **a simple pathing algorithm to increment/decrement the x or y position depending on the position of the target**
  + **why was this done?**
    - **To achieve the intended goal of the simulation for interacting organisms, without movement the organisms cannot interact**
* **Circular vision cone instead of rectangular**
  + **How was this achieved?**
    - **Use of pyhthagoras to find the precise length of a line (by using fake opp and adj), the length can then be compared to the radius of the vision cone**
  + **Why was this done?**
    - **To achieve more precision and accuracy for the simulation, it doesn’t make a lot of sense for an organism to have rectangular vision**
* **Nucleus to food objects**
  + **How was this achieved?**
    - **Drawing a smaller circle inside food objects**
  + **Why was this done?**
    - **More varied appearance of food objects to match that of the remarks on the organisms**
* **Visual bar to show stomach contents** 
  + **How was this achieved?**
    - **Drawing an overlay rect onto the stomach contents rect to show the maximum stomach contents, utilises the same algorithm as the ui sliders.**
  + **Why was this done?**
    - **Visualise the threshold at which stomach contents are converted to biomass instead of just used for respiration and movement.**

## Prototype 3

Background pattern

Description automatically generated

### Code

import pygame

import sys

import random

import math

import json

import time

#json area

default = {'cursor': {'color': [20, 20, 20], 'pressed': [255, 255, 255]}, 'genes': {'lower': 12, 'upper': 48, 'NameSyllablesLower': 1, 'NameSyllablesHigher': 3, 'NameLengthLower': 1, 'NameLengthHigher': 8}, 'food': {'SizeLower': 1, 'SizeUpper': 6}, 'misc': {'DrawVector': False,'DrawNodes': False,'ShowDebug': False, 'Background': True,'DrawNames': True}}

# this is the library data for the default json file, only used for recovery

try:

Json = open("config.json")

Config = json.load(Json)

Json.close()

except:

print("exception occured in parsing/finding config.json... attempting to fix")

file = open("config.json", "a")

json.dump(default, file)

file.close()

Json = open("config.json")

Config = json.load(Json)

Json.close()

print("succesfully restored config.json")

#pygame initiation

pygame.init()

surface = pygame.display.set\_mode((800, 600))

font = pygame.font.Font('freesansbold.ttf', 12)

RED = (100,0,0)

DarkRed = (200,0,0)

GREEN = (0,100,0)

BLUE = (0,0,100)

GREY = (128,128,128)

DarkGrey = (90,90,90)

BLACK = (20,20,20)

WHITE = (255,255,255)

YELLOW = (255,255,0)

pygame.mouse.set\_visible(False)

#misc variable definition

CurrentPage = ""

ButtonToggle1 = False

ButtonToggle2 = False

StickyTarget = False

FoodRegister = []

SpeciesRegister = []

Nodes = []

#utility/math functions

def VectorCalculation(ObjectPositionX,ObjectPositionY,Range,color,state): #object x is the host

global Config

ChangeInX = abs(ObjectPositionY[0] - ObjectPositionX[0])

ChangeInY = abs(ObjectPositionY[1] - ObjectPositionX[1])

distance = math.sqrt(Sqr(ChangeInX) + Sqr(ChangeInY))

if distance < Range:

if state:

pygame.draw.line(surface,color,(ObjectPositionX[0],ObjectPositionX[1]),(ObjectPositionY[0],ObjectPositionY[1]))

return distance

def Sqr(arg):

return (arg \* arg)

def IsPrime(arg): # this checks for prime numbers

for i in range(2,int(arg\*\*0.5)+1):

if arg%i==0:

return False

return True

def InHitboxRect(X,Y,posX,posY,sizeX,sizeY):

if posX > X and posX < (X + sizeX): #checks for allignment on the X axis

if posY > Y and posY < (Y + sizeY): #checks for allignment on the Y axis

#print("object allignment, details below")

#print(X,Y,posX,posY,sizeX,sizeY)

return True

#def GetPosInArray(find,array):

#for thing in range(len(array)):

#if find == array[thing]:

#return thing + 1

#print("couldnt find",find)

def GetPositionOnClick(mousepos,mousestate,posX,posY,sizeX,sizeY):

if mousepos[0] > posX and mousepos[0] < (posX + sizeX):

if mousepos[1] > posY and mousepos[1] < (posY + sizeY):

if mousestate[0]:

return mousepos

#ui functions

def DrawColorPicker(x,y,mousepos,mousestate,offsetX,offsetY):

pygame.draw.rect(surface, BLACK, pygame.Rect(x,y,261,261))

pygame.draw.rect(surface, WHITE, pygame.Rect((x+3),(y+3),255,255))

for color in range(256):

pygame.draw.line(surface,(offsetX,offsetY,color),((x + color + 3),(y + 3)),((x + color + 3),(y + 258)))

if (GetPositionOnClick(mousepos,mousestate,x+3,y+3,255,255)) != None:

Color = list(GetPositionOnClick(mousepos,mousestate,x+3,y+3,255,255))

SelectedColor = (offsetX,offsetY,(Color[0] - x+3))

return SelectedColor

class button:

def \_\_init\_\_(self,size,position,color,pressedcolor,hoveredcolor,text,textcolor):

self.size = size

self.position = position

self.color = color

self.pressedcolor = pressedcolor

self.hoveredcolor = hoveredcolor

self.text = text

self.textcolor = textcolor

self.LastPress = False

def DrawButton(self,mousepos,mousestate):

pygame.draw.rect(surface, DarkGrey, pygame.Rect(self.position[0] - 3,self.position[1] - 3,self.size[0] + 6,self.size[1] + 6))

if mousestate[0] == False and InHitboxRect(self.position[0],self.position[1],mousepos[0],mousepos[1],self.size[0],self.size[1]): #hovered but not clicked

pygame.draw.rect(surface, self.hoveredcolor, pygame.Rect(self.position[0],self.position[1],self.size[0],self.size[1])) #this is the hovered state of the button

elif mousestate[0] == True and InHitboxRect(self.position[0],self.position[1],mousepos[0],mousepos[1],self.size[0],self.size[1]) and mousestate[0] != self.LastPress: #pressed

pygame.draw.rect(surface, self.pressedcolor, pygame.Rect(self.position[0],self.position[1],self.size[0],self.size[1])) #pressed state

self.LastPress = True

return True

else:

self.LastPress = False

pygame.draw.rect(surface, self.color, pygame.Rect(self.position[0],self.position[1],self.size[0],self.size[1])) #this is the static state of the button

text = font.render(self.text, True, self.textcolor, GREY)

surface.blit(text,(self.position[0] + self.size[0] + 3,self.position[1] + (self.size[1] // 2)))

def Output(self):

print(self.size,self.position,self.color,self.pressedcolor,self.hoveredcolor,self.text,self.textcolor)

class slider:

def \_\_init\_\_(self,size,position,color,pressedcolor,hoveredcolor,slidersize,level,text):

self.size = size

self.position = position

self.color = color

self.pressedcolor = pressedcolor

self.hoveredcolor = hoveredcolor

self.sliderpos = [position[0],(position[1] + level)]

self.slidersize = slidersize

self.text = text

def DrawSlider(self,mousepos,mousestate):

pygame.draw.rect(surface, self.color, pygame.Rect(self.position[0],self.position[1],self.size[0],self.size[1]))

text = font.render(self.text + ":" + str(self.sliderpos[1] - self.position[1]), True, BLACK, GREY)

surface.blit(text,(self.position[0] + self.size[0],self.position[1]))

if InHitboxRect(self.sliderpos[0],self.sliderpos[1],mousepos[0],mousepos[1],(self.slidersize[0] + 10),(self.slidersize[1] + 10)) and mousestate[0] == 1:

pygame.draw.rect(surface, self.pressedcolor, pygame.Rect(self.sliderpos[0],self.sliderpos[1],self.slidersize[0],self.slidersize[1]))

if self.sliderpos[1] >= self.position[1]:

if self.sliderpos[1] <= (self.position[1] + self.size[1]):

self.sliderpos[1] = (mousepos[1] - 5)

else:

self.sliderpos[1] = (self.position[1] + (self.size[1] - self.slidersize[1]))

else:

self.sliderpos[1] = self.position[1]

elif InHitboxRect(self.sliderpos[0],self.sliderpos[1],mousepos[0],mousepos[1],(self.slidersize[0]),(self.slidersize[1])):

pygame.draw.rect(surface, self.hoveredcolor, pygame.Rect(self.sliderpos[0],self.sliderpos[1],self.slidersize[0],self.slidersize[1]))

else:

pygame.draw.rect(surface, WHITE, pygame.Rect(self.sliderpos[0],self.sliderpos[1],self.slidersize[0],self.slidersize[1]))

if (self.sliderpos[1] - self.position[1]) < (self.size[1] - self.slidersize[1]):

if (self.sliderpos[1] - self.position[1]) > 0:

return (self.sliderpos[1] - self.position[1])

else:

return 0

else:

return (self.size[1] - self.slidersize[1])

def Output(self):

print(self.size,self.position,self.color,self.pressedcolor,self.hoveredcolor,self.sliderpos,self.slidersize)

class SideSlider:

def \_\_init\_\_(self,size,position,color,pressedcolor,hoveredcolor,slidersize,text):

self.size = size

self.position = position

self.color = color

self.pressedcolor = pressedcolor

self.hoveredcolor = hoveredcolor

self.sliderpos = [position[0],position[1]]

self.slidersize = slidersize

self.text = text

def DrawSlider(self,mousepos,mousestate):

pygame.draw.rect(surface, self.color, pygame.Rect(self.position[0],self.position[1],self.size[0],self.size[1]))

text = font.render(self.text + ":" + str(self.sliderpos[0] - self.position[0]), True, BLACK, GREY)

surface.blit(text,(self.position[0] + self.size[0] + self.slidersize[0],self.position[1]))

if InHitboxRect(self.sliderpos[0],self.sliderpos[1],mousepos[0],mousepos[1],(self.slidersize[0] + 10),(self.slidersize[1] + 10)) and mousestate[0] == 1:

pygame.draw.rect(surface, self.pressedcolor, pygame.Rect(self.sliderpos[0],self.sliderpos[1],self.slidersize[0],self.slidersize[1]))

if self.sliderpos[0] >= self.position[0]:

if self.sliderpos[0] <= (self.position[0] + self.size[0]):

self.sliderpos[0] = (mousepos[0] - 10)

else:

self.sliderpos[0] = (self.position[0] + (self.size[0] - self.slidersize[0]))

else:

self.sliderpos[0] = self.position[0]

elif InHitboxRect(self.sliderpos[0],self.sliderpos[1],mousepos[0],mousepos[1],(self.slidersize[0]),(self.slidersize[1])):

pygame.draw.rect(surface, self.hoveredcolor, pygame.Rect(self.sliderpos[0],self.sliderpos[1],self.slidersize[0],self.slidersize[1]))

else:

pygame.draw.rect(surface, WHITE, pygame.Rect(self.sliderpos[0],self.sliderpos[1],self.slidersize[0],self.slidersize[1]))

if (self.sliderpos[0] - self.position[0]) < (self.size[0] - self.slidersize[0]):

if (self.sliderpos[0] - self.position[0]) > 0:

return (self.sliderpos[0] - self.position[0])

else:

return 0

else:

return (self.size[0] - self.slidersize[0])

def Output(self):

print(self.size,self.position,self.color,self.pressedcolor,self.hoveredcolor,self.sliderpos,self.slidersize)

class CheckBox():

def \_\_init\_\_(self,size,position,color,HoverColor,CheckColor,state,text):

self.size = size

self.position = position

self.color = color

self.HoverColor = HoverColor

self.CheckColor = CheckColor

self.state = state

self.text = text

self.LastPress = False

def Draw(self,mousepos,mousestate):

if InHitboxRect(self.position[0],self.position[1],mousepos[0],mousepos[1],self.size[0],self.size[1]):

pygame.draw.rect(surface, self.HoverColor, pygame.Rect(self.position[0],self.position[1],self.size[0],self.size[1]))

if mousestate[0] and mousestate[0] != self.LastPress:

self.LastPress = mousestate[0]

self.state = not self.state

else:

self.LastPress = mousestate[0]

else:

pygame.draw.rect(surface, self.color, pygame.Rect(self.position[0],self.position[1],self.size[0],self.size[1]))

if self.state:

pygame.draw.line(surface,self.CheckColor,(self.position[0],self.position[1]),(self.position[0] + self.size[0],self.position[1] + self.size[1]))

pygame.draw.line(surface,self.CheckColor,(self.position[0] + self.size[0],self.position[1]),(self.position[0],self.position[1] + self.size[1]))

text = font.render(self.text, True, BLACK, GREY)

surface.blit(text,(self.position[0] + self.size[0],self.position[1]))

return self.state

def DrawCustomCursor(mousepos,mousestate):

global Config

if mousestate[0] == 1:

pygame.draw.rect(surface, Config["cursor"]["pressed"], pygame.Rect((mousepos[0]+5),(mousepos[1]+5),(3),(3)))

else:

pygame.draw.rect(surface, Config["cursor"]["color"], pygame.Rect((mousepos[0]+5),(mousepos[1]+5),(3),(3)))

#simulation functions

def InitiateSpeciesObject():

global Config

name = ""

genes = []

charcolor = []

charsize = []

color = (random.randint(1,255),random.randint(1,255),random.randint(1,255))

size = (random.randint(35,65),random.randint(35,65))

position = [random.randint(100,700),random.randint(100,500)]

Range = random.randint(100,175) # range at which they will look for food

Capacity = random.randint(10,size[1]) # capacity for which they can store food

for i in range(random.randint(Config["genes"]["NameLengthLower"],Config["genes"]["NameLengthHigher"])):

for o in range(random.randint(Config["genes"]["NameSyllablesLower"],Config["genes"]["NameSyllablesHigher"])): #this generates alien like names

name += chr(random.randint(65,122))

name += "-"

for i in range(random.randint(Config["genes"]["lower"],Config["genes"]["upper"])): #genetic code generator, codes for the appearance

X = (random.randint(1,255),random.randint(1,255),random.randint(1,255))

genes.append(random.randint(1,size[0]))

genes.append(random.randint(1,size[1]))

charcolor.append(X)

charsize.append(random.randint(1,4))

SpeciesRegister.append(Species(name,color,charcolor,size,charsize,position,genes,Range,Capacity))

def CreateFoodObject():

global Config

position = [random.randint(100,700),random.randint(100,500)]

color = (random.randint(1,255),random.randint(1,255),random.randint(1,255))

size = random.randint(1,6)

dotsize = random.randint(0,(size // 2))

FoodRegister.append(Food(position,color,size,dotsize))

class Food:

def \_\_init\_\_(self,position,color,size,dotsize):

self.position = position

self.color = color

self.size = size

self.dotsize = dotsize

def DrawFoodObject(self):

pygame.draw.circle(surface, self.color,(self.position[0],self.position[1]),(self.size))

pygame.draw.circle(surface, BLACK,(self.position[0],self.position[1]),(self.dotsize))

return self.position

def Output(self):

print(self.position,self.color,self.size,self.dotsize)

class Node:

def \_\_init\_\_(self,position):

self.position = position

def DrawNode(self):

pygame.draw.circle(surface, YELLOW,(self.position[0],self.position[1]),2)

def InitiateNodesLattice():

x = 100

y = 100

for a in range(40):

for b in range(60):

Nodes.append(Node((x,y)))

x += 10

y += 10

x = 100

if Config["misc"]["ShowDebug"]:

print("succesfully initiated all nodes!")

class Species:

def \_\_init\_\_(self, name, color, charcolor, size, charsize, position, genes,Range,Capacity):

self.name = name

self.color = color

self.charcolor = charcolor

self.size = size

self.charsize = charsize

self.position = position

self.genes = genes

self.range = Range

self.capacity = Capacity

self.saturation = 10

self.RememberedTargetsID = []

self.RememberedTargetsPositions = []

def DrawSprite(self):

pygame.draw.rect(surface, self.color, pygame.Rect(self.position[0],self.position[1],self.size[0],self.size[1]))

pygame.draw.rect(surface, WHITE, pygame.Rect(self.position[0] - 5,self.position[1],2,self.capacity))

pygame.draw.rect(surface, BLUE, pygame.Rect(self.position[0] - 5,self.position[1],2,self.saturation))

if Config["misc"]["DrawNames"]:

text = font.render(self.name, True, BLACK, GREY)

surface.blit(text,((self.position[0] + self.size[0]),(self.position[1])))

factor = 0

for Pointer in range((len(self.genes) // 2) - 2):

Horizontal = (self.position[0] + self.size[0])

Down = (self.position[1] + self.size[1])

X = (Horizontal - self.genes[factor])

Y = (Down - self.genes[factor + 1])

A = (Horizontal - self.genes[factor+2])

B = (Down - self.genes[factor+3])

if self.genes[factor] % 2 == 0: #even check

pygame.draw.circle(surface, self.charcolor[Pointer],(X,Y),self.charsize[Pointer])

elif IsPrime(self.genes[factor]):

pygame.draw.line(surface,self.charcolor[Pointer],(X,Y),(A,B))

else:

pygame.draw.rect(surface, self.charcolor[Pointer], pygame.Rect(X,Y,self.charsize[Pointer],self.charsize[Pointer]))

factor += 2

if Config["misc"]["DrawVector"]:

pygame.draw.circle(surface, RED, (self.position[0] + (self.size[0] // 2),self.position[1] + (self.size[1] // 2)), self.range, 1)

return self.position

def TargetMove(self):

global Config

#print(TargetPos,self.position)

X = self.position[0] + (self.size[0] // 2)

Y = self.position[1] + (self.size[1] // 2)

if X != 0 and X < 800:

if self.TargetPos[0] < math.floor(X):

self.position[0] -= 1

elif self.TargetPos[0] != math.floor(X):

self.position[0] += 1

if Y != 0 and Y < 600:

if self.TargetPos[1] < math.floor(Y):

self.position[1] -= 1

elif self.TargetPos[1] != math.floor(Y):

self.position[1] += 1

if self.TargetPos[0] == math.floor(X) and self.TargetPos[1] == math.floor(Y):

if Config["misc"]["ShowDebug"]:

print("reached food at:" ,self.TargetPos)

self.saturation += 1

#print(self.eaten)

return True

if Config["misc"]["DrawVector"] and self.TargetType == "Root":

pygame.draw.line(surface,BLUE,(X,Y),(self.TargetPos[0],self.TargetPos[1]))

elif Config["misc"]["DrawVector"]:

pygame.draw.line(surface,GREEN,(X,Y),(self.TargetPos[0],self.TargetPos[1]))

if self.TargetId not in FoodRegister:

self.DeleteTarget()

def ReturnTarget(self):

try:

return self.target

except:

return None

def DeleteTarget(self):

del self.RememberedTargetsID[self.RememberedTargetsID.index(self.TargetId)]

del self.RememberedTargetsPositions[self.RememberedTargetsPositions.index(self.TargetId.position)]

del self.target

del self.TargetPos

del self.TargetId

del self.TargetType

def GoTo(self,TargetPos):

global Config

#print(TargetPos,self.position)

X = self.position[0] + (self.size[0] // 2)

Y = self.position[1] + (self.size[1] // 2)

if X != 0 and X < 800:

if TargetPos[0] < math.floor(X):

self.position[0] -= 1

elif TargetPos[0] != math.floor(X):

self.position[0] += 1

if Y != 0 and Y < 600:

if TargetPos[1] < math.floor(Y):

self.position[1] -= 1

elif TargetPos[1] != math.floor(Y):

self.position[1] += 1

if TargetPos[0] == math.floor(X) and TargetPos[1] == math.floor(Y):

return True

if Config["misc"]["DrawVector"]:

pygame.draw.line(surface,YELLOW,(X,Y),(TargetPos[0],TargetPos[1]))

def RandomizePos(self):

self.position[0] = random.randint(100,700)

self.position[1] = random.randint(100,500)

def Vector(self,TargetPos,color,state,UseRange):

if UseRange:

Ping = VectorCalculation((self.position[0] + (self.size[0] // 2), self.position[1] + (self.size[1] // 2)),TargetPos,self.range,color,state)

else:

Ping = VectorCalculation((self.position[0] + (self.size[0] // 2), self.position[1] + (self.size[1] // 2)),TargetPos,1000000,color,state)

return Ping

def GetTarget(self,array,index,ID):

if len(array) != 0:

self.target = min(array)

self.target = array.index(self.target)

self.TargetPos = index[self.target]

self.TargetId = ID[self.target]

self.TargetType = "Root"

for MemoryTargets in ID:

if MemoryTargets not in self.RememberedTargetsID:

self.RememberedTargetsID.append(MemoryTargets)

self.RememberedTargetsPositions.append(MemoryTargets.position)

#if Config["misc"]["ShowDebug"]:

#print(MemoryTargets,"already in target memory")

#print("selected",self.target,"from",array,index)

return True

def RetrieveTargetFromMemory(self):

Targets = []

for PotentialTarget in self.RememberedTargetsID:

if PotentialTarget not in FoodRegister:

del self.RememberedTargetsID[self.RememberedTargetsID.index(PotentialTarget)]

del self.RememberedTargetsPositions[self.RememberedTargetsPositions.index(PotentialTarget.position)]

else:

Targets.append(self.Vector(PotentialTarget.position,GREEN,Config["misc"]["DrawNodes"],False))

if len(Targets) != 0:

self.target = min(Targets)

self.target = Targets.index(self.target)

self.TargetPos = self.RememberedTargetsPositions[self.target]

self.TargetId = self.RememberedTargetsID[self.target]

self.TargetType = "Memory"

return True

def GetNodeTarget(self):

try:

return self.NodeTarget

except:

return None

def SetNodeTarget(self,NodeTarget):

self.NodeTarget = NodeTarget

def DeleteNodeTarget(self):

del self.NodeTarget

def Output(self):

self.name,self.color,self.charcolor,self.size,self.charsize,self.position,self.genes

def DrawAll():

for species in range(len(SpeciesRegister)):

Targets = []

Indexes = []

TargetId = []

ReachableNodes = []

pos = SpeciesRegister[species].position

#print(Nodes)

for Node in Nodes:

if Config["misc"]["DrawNodes"]:

Node.DrawNode()

if abs(Node.position[0] - pos[0]) < 175:

if abs(Node.position[1] - pos[1]) < 175:

NodeCheck = SpeciesRegister[species].Vector(Node.position,YELLOW,Config["misc"]["DrawNodes"],True)

if NodeCheck != None:

ReachableNodes.append(Node.position)

for food in range(len(FoodRegister)):

if abs((FoodRegister[food].DrawFoodObject())[0] - pos[0]) < 175:

if abs((FoodRegister[food].position)[1] - pos[1]) < 175:

Ping = SpeciesRegister[species].Vector(FoodRegister[food].DrawFoodObject(),RED,Config["misc"]["DrawVector"],True)

if Ping != None:

Targets.append(Ping)

Indexes.append(FoodRegister[food].position)

TargetId.append(FoodRegister[food])

if len(Targets) != 0 and SpeciesRegister[species].ReturnTarget() == None:

SpeciesRegister[species].GetTarget(Targets,Indexes,TargetId)

if Config["misc"]["ShowDebug"]:

print("SetFoodTarget at", SpeciesRegister[species].TargetPos)

elif SpeciesRegister[species].ReturnTarget() != None:

if (SpeciesRegister[species].TargetMove()):

try:

if Config["misc"]["ShowDebug"]:

print("deleted:", FoodRegister.index(SpeciesRegister[species].TargetId), "from food register")

del FoodRegister[FoodRegister.index(SpeciesRegister[species].TargetId)]

SpeciesRegister[species].DeleteTarget()

except:

if Config["misc"]["ShowDebug"]:

print("prevented an exception: overlapping food target")

SpeciesRegister[species].DeleteTarget()

SpeciesRegister[species].RandomizePos()

elif SpeciesRegister[species].RetrieveTargetFromMemory():

if Config["misc"]["ShowDebug"]:

print("retrieved a target from memory")

elif SpeciesRegister[species].GetNodeTarget() == None:

SpeciesRegister[species].SetNodeTarget(random.choice(ReachableNodes))

if Config["misc"]["ShowDebug"]:

print("SetNodeTarget at", SpeciesRegister[species].GetNodeTarget())

else:

if SpeciesRegister[species].GoTo(SpeciesRegister[species].GetNodeTarget()):

if Config["misc"]["ShowDebug"]:

print("reached node target at", SpeciesRegister[species].GetNodeTarget())

SpeciesRegister[species].DeleteNodeTarget()

SpeciesRegister[species].DrawSprite()

#loop functions

def OptionPage(mousepos,mousestate):

global CurrentPage

global ButtonToggle1

global ButtonToggle2

if back.DrawButton(mousepos,mousestate):

CurrentPage = ""

if save.DrawButton(mousepos,mousestate):

json.dump(Config,open("config.json","w"))

Unsaved = Config

if DrawVector.Draw(mousepos,mousestate) != Config["misc"]["DrawVector"]:

Config["misc"]["DrawVector"] = not Config["misc"]["DrawVector"]

if DrawNodes.Draw(mousepos,mousestate) != Config["misc"]["DrawNodes"]:

Config["misc"]["DrawNodes"] = not Config["misc"]["DrawNodes"]

if Debug.Draw(mousepos,mousestate) != Config["misc"]["ShowDebug"]:

Config["misc"]["ShowDebug"] = not Config["misc"]["ShowDebug"]

if DrawNames.Draw(mousepos,mousestate) != Config["misc"]["DrawNames"]:

Config["misc"]["DrawNames"] = not Config["misc"]["DrawNames"]

if BackGround.Draw(mousepos,mousestate) != Config["misc"]["Background"]:

Config["misc"]["Background"] = not Config["misc"]["Background"]

if (GeneRangeUpper.DrawSlider(mousepos,mousestate)) > Config["genes"]["lower"]:

Config["genes"]["upper"] = (GeneRangeUpper.DrawSlider(mousepos,mousestate))

if (GeneRangeLower.DrawSlider(mousepos,mousestate) < Config["genes"]["upper"]):

Config["genes"]["lower"] = (GeneRangeLower.DrawSlider(mousepos,mousestate))

if not ButtonToggle2:

if (CustomCursorClicked.DrawButton(mousepos,mousestate)) or ButtonToggle1:

ButtonToggle1 = True

offsetX = colorR1.DrawSlider(mousepos,mousestate)

offsetY = colorG1.DrawSlider(mousepos,mousestate)

if DrawColorPicker(500,185,mousepos,mousestate,offsetX,offsetY) != None:

ButtonToggle1 = False

Config["cursor"]["pressed"] = list(DrawColorPicker(500,185,mousepos,mousestate,offsetX,offsetY))

if not ButtonToggle1:

if (CustomCursor.DrawButton(mousepos,mousestate)) or ButtonToggle2:

ButtonToggle2 = True

offsetX = colorR2.DrawSlider(mousepos,mousestate)

offsetY = colorG2.DrawSlider(mousepos,mousestate)

if DrawColorPicker(500,235,mousepos,mousestate,offsetX,offsetY) != None:

ButtonToggle2 = False

Config["cursor"]["color"] = list(DrawColorPicker(500,235,mousepos,mousestate,offsetX,offsetY))

def InfoPage(mousepos,mousestate):

global CurrentPage

if back.DrawButton(mousepos,mousestate):

CurrentPage = ""

text = font.render("this is a project produced by henry frodsham in 12C, its a simulation of natural selection", True, BLACK, GREY)

surface.blit(text,(150,400))

def Simulation(mousepos,mousestate):

DrawAll()

#ui elements

SpawnFood = button((50,30),(0,0),(40,40,40),DarkRed,RED,"FOOD",BLACK)

play = button((50,30),(360,500),(40,40,40),DarkRed,RED,"start simulation",BLACK)

back = button((50,30),(3,567),(40,40,40),DarkRed,RED,"BACK",BLACK)

save = button((50,30),(103,567),(40,40,40),DarkRed,RED,"SAVE",BLACK)

options = button((50,30),(200,500),(40,40,40),DarkRed,RED,"OPTIONS",BLACK)

information = button((50,30),(510,500),(40,40,40),DarkRed,RED,"INFO",BLACK)

CustomCursor = button((50,30),(500,200),(40,40,40),DarkRed,RED,"change cursor color",BLACK)

CustomCursorClicked = button((50,30),(500,150),(40,40,40),DarkRed,RED,"change pressed cursor color",BLACK)

GeneRangeLower = slider((20,80),(40,300),BLACK,DarkRed,RED,(20,20),Config["genes"]["lower"],("CHANGE GENE LOWER BOUND"))

GeneRangeUpper = slider((20,80),(40,200),BLACK,DarkRed,RED,(20,20),Config["genes"]["upper"],("CHANGE GENE UPPER BOUND"))

colorR1 = SideSlider((255,20),(500,450),BLACK,DarkRed,RED,(20,20),("RED"))

colorG1 = SideSlider((255,20),(500,490),BLACK,DarkRed,RED,(20,20),("GREEN"))

colorR2 = SideSlider((255,20),(500,500),BLACK,DarkRed,RED,(20,20),("RED"))

colorG2 = SideSlider((255,20),(500,540),BLACK,DarkRed,RED,(20,20),("GREEN"))

DrawVector = CheckBox((20,20),(40,40),BLACK,DarkRed,(0,200,0),Config["misc"]["DrawVector"],"visualie vectors")

DrawNodes = CheckBox((20,20),(40,70),BLACK,DarkRed,(0,200,0),Config["misc"]["DrawNodes"],"visualize nodes")

Debug = CheckBox((20,20),(40,100),BLACK,DarkRed,(0,200,0),Config["misc"]["ShowDebug"],"print debug data to console")

BackGround = CheckBox((20,20),(40,130),BLACK,DarkRed,(0,200,0),Config["misc"]["Background"],"Draw background")

DrawNames = CheckBox((20,20),(40,160),BLACK,DarkRed,(0,200,0),Config["misc"]["DrawNames"],"Draw Names")

InitiateSpeciesObject()

InitiateSpeciesObject()

InitiateSpeciesObject()

InitiateNodesLattice()

for i in range(300):

CreateFoodObject()

#main loop, in a function to allow for partial looping

def main():

timeX = time.time()

global CurrentPage

global Config

mousepos = pygame.mouse.get\_pos()

mousestate = pygame.mouse.get\_pressed()

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

print("json dumped",Config)

json.dump(Config,open("config.json","w")) #more optimised as it saves everytime the user quits not everytime there is a change

sys. exit()

if Config["misc"]["Background"]:

surface.fill(GREY)

else:

surface.fill(WHITE)

if CurrentPage == "":

if information.DrawButton(mousepos,mousestate):

CurrentPage = "InfoPage"

elif options.DrawButton(mousepos,mousestate):

CurrentPage = "OptionPage"

elif play.DrawButton(mousepos,mousestate):

CurrentPage = "Simulation"

elif CurrentPage == "InfoPage":

InfoPage(mousepos,mousestate)

elif CurrentPage == "OptionPage":

OptionPage(mousepos,mousestate)

elif CurrentPage == "Simulation":

Simulation(mousepos,mousestate)

DrawCustomCursor(mousepos,mousestate) #the cursor should be drawn after everything else

timeY = time.time()

try:

fps = math.ceil(1 / (timeY - timeX))

text = font.render("fps: " + str(fps), True, BLACK, GREY)

surface.blit(text,(0,0))

except:

text = font.render("fps: 1000", True, BLACK, GREY)

surface.blit(text,(0,0))

pygame.display.flip()

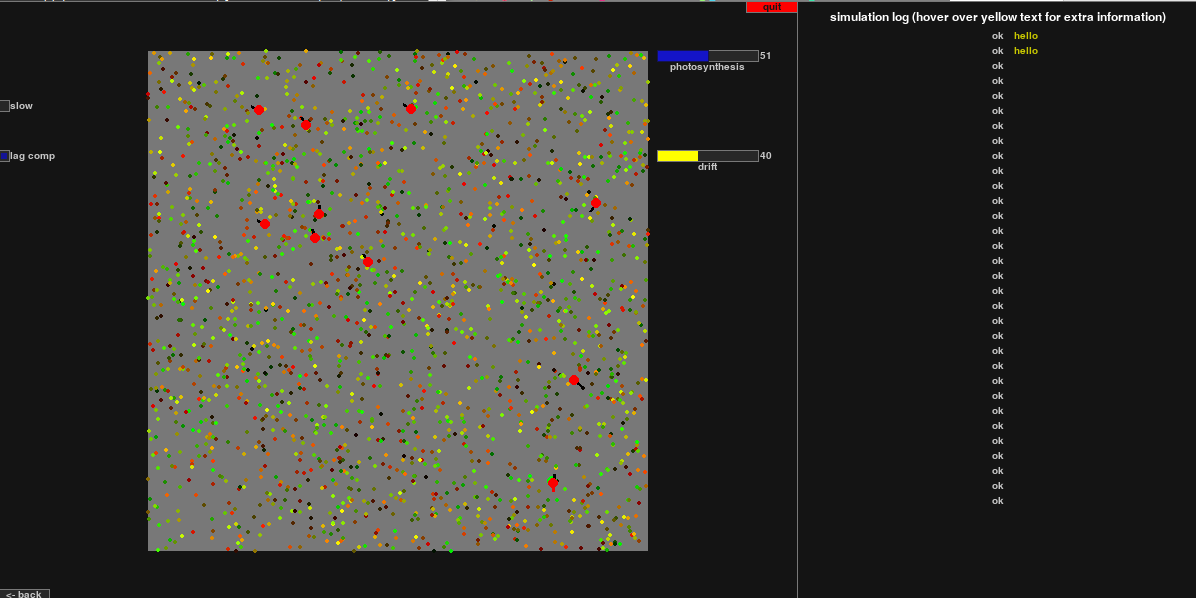
while True:

main()

### Major additions from prototype 2 -> 3

* **Efficiency changes (hence more food objects)**
  + **How was this achieved?**
    - **By excluding obviously out of range food objects from the pathing algorithm, also split entity object array and food object array to stop looping other entities**
  + **Why was this done?**
    - **At this stage, I had realised the simulation would need multiple ui elements and graphs needing cpu time, in order to prevent the simulation having poor performance and ultimately hindering its usefullness I implemented efficiency changes.**
* **Randomisation of genetic traits through new genetic code system**
  + **How was this achieved?**
    - **Genetic code functions to read and create dna strands to proteins and binary**
  + **Why was this done?**
    - **I needed a centralised system to manage mutations and reading of genetic material in order to achieve the simulations main purpose of simulating changes in genomes.**
* **OOP**
  + **How was this achieved?**
    - **Ui elements converted to classes, use of object arrays to contain entity objects to be looped through**
  + **Why was this done?**
    - **Much more efficient and makes the code more readable, it also allows the simulation loop to be simplified as the organisms behaviour is now controlled inside their classes.**
* **Multiple species objects can be on screen at once**
  + **How was this achieved?**
    - **Use of an entity object array to loop through each element and call draw() and create\_move()**
  + **Why was this achieved?**

## First version



### Code

try:

import pygame

import random

import time

import math

import json

import sys

except:

print("you do not have the required libraries to run this program, required libraries are [pygame,tensorflow,numpy] rest are native (or should be)")

sys.exit()

default = {'layer-1': {'food': 2000, 'herbivores': 10, 'carnivores': 10}}

vgui\_warning\_config = False

try:

Config = json.load(open("config.json", "a"))

except:

Config=default

#file = open("config.json", "a")

#json.dump(default, file)

#file.close()

#Config = json.load(open("config.json", "r"))

#vgui\_warning\_config = True

pygame.init()

panel = pygame.display.set\_mode((1200,600),pygame.NOFRAME)

screen\_size = panel.get\_size()

#font\_main = pygame.font.Font('Bahnschrift.ttf', 10)

font\_alt = pygame.font.Font('freesansbold.ttf', 12)

ui\_font\_scale\_3 = pygame.font.Font('freesansbold.ttf', 10)

foreground = (20,20,20)

vgui\_fore = (40,40,40)

vgui\_bounding = (120,120,120)

vgui\_state\_0 = (10,10,220)

vgui\_state\_1 = (20,20,200)

vgui\_state\_2 = (20,20,180)

vgui\_state\_3 = (20,20,150)

vgui\_state\_4 = (20,20,120)

vgui\_state\_5 = (20,20,80)

vgui\_important = (0,0,0)

vgui\_aux\_text\_internal = (200,200,200)

vgui\_aux\_text\_external = (200,200,200)

vgui\_warning\_1 = (150,20,20)

vgui\_ray\_broken = (250,0,0)

vgui\_ray\_beam = (0,255,0)

vgui\_group\_fore = (190,190,190)

vgui\_entity\_herbivore = (0,250,0)

vgui\_entity\_carnivore = (250,0,0)

vgui\_entity\_nose = (0,0,0)

entity\_object\_array = []

hunter\_object\_array = []

food\_object\_array = []

producer\_object\_array = []

tab = 0

lag\_comp = 0

bases = ["G","A","T","C"]

def generate\_dna\_sequence(length):

strand = []

for i in range(length):

strand.append(random.choice(bases))

return strand

def read\_dna\_protein(strand):

active\_site = read\_dna\_binary(strand[0:3])

op\_code = read\_dna\_binary(strand[4:7])

if bases.index(strand[len(strand) - 1]) == 0:

sign = True

else:

sign = False

if (sign):

op\_code -= op\_code \* 2

return [active\_site,op\_code]

def read\_dna\_binary(strand):

binary = ""

for base in strand:

if bases.index(base) == 1 or bases.index(base) == 2:

binary += "0"

else:

binary += "1"

return int(binary,2)

def create\_mutation(strand):

if (strand == None):

return

if (len(strand) == 1):

return

mutation\_type = random.randint(1,3)

#frame shift aka deletion

if (mutation\_type == 1):

del strand[random.randint(0,(len(strand) - 1))]

#substitution

if (mutation\_type == 2):

strand[random.randint(0,(len(strand) - 1))] = random.choice(bases)

#insertion

if (mutation\_type == 3):

strand += random.choice(bases)

return strand

class genetic\_code:

def \_\_init\_\_(this,length):

this.strand = generate\_dna\_sequence(length)

this.numerical\_value = read\_dna\_binary(this.strand)

def update(this):

this.numerical\_value = read\_dna\_binary(this.strand)

def mutation(this):

this.strand = create\_mutation(this.strand)

this.update()

def Sqr(num):

return num\*num

def distance\_to(vec\_point\_a,vec\_point\_b):

change\_in\_x = abs(vec\_point\_a[0] - vec\_point\_b[0])

change\_in\_y = abs(vec\_point\_a[1] - vec\_point\_b[1])

return math.sqrt(Sqr(change\_in\_x) + Sqr(change\_in\_y))

def rad\_to\_deg(vec\_point\_a,vec\_point\_b):

hyp = distance\_to(vec\_point\_a,vec\_point\_b)

opp = distance\_to((vec\_point\_b[0],vec\_point\_a[1]),vec\_point\_b)

raw\_angle = pythag(opp,hyp)

X = vec\_point\_b[0] - vec\_point\_a[0]

Y = vec\_point\_b[1] - vec\_point\_a[1]

if X < 0 and Y < 0:

angle = raw\_angle + 180

elif X < 0:

angle = -raw\_angle + 180

elif Y < 1:

angle = -raw\_angle

else:

angle = raw\_angle

return angle

def pythag(opp,hyp):

try:

sine = (opp/hyp)

except:

return 0

return (math.asin(sine) \* 57.296)

def point\_of\_orbit(host\_vec,rotation\_float,radius\_int):

c = math.cos(rotation\_float \* 0.0174)

s = math.sin(rotation\_float \* 0.0174)

return [math.ceil(host\_vec[0] + radius\_int \* c), math.ceil(host\_vec[1] + radius\_int \* s)]

def trace\_ray(from\_vec,to\_vec,array):

lazy\_trace\_threshold = 0

ray\_skip\_mult = 1

ray\_visualize = True

ray\_ignore\_groups = [False,False]

ray\_hitbox\_add = 0

ray\_angle = rad\_to\_deg(from\_vec,to\_vec)

for ray in range(500 - lazy\_trace\_threshold):

RayPos = point\_of\_orbit(from\_vec,ray\_angle,ray\*ray\_skip\_mult)

if (not ray\_ignore\_groups[0]):

for food\_obj in array:

if (distance\_to(RayPos,food\_obj.pos) > 3):

continue

if (food\_obj.pos == to\_vec):

pygame.draw.line(panel,vgui\_ray\_beam,from\_vec,to\_vec)

return True

else:

#if (ray\_visualize):

#pygame.draw.line(panel,vgui\_ray\_beam,from\_vec,to\_vec)

#pygame.draw.line(panel,vgui\_ray\_broken,RayPos,to\_vec)

return False

def get\_next\_move\_to(from\_vec,rotation\_float,radius = 2):

if (not vgui\_checkbox\_sim\_lag\_comp.state):

return point\_of\_orbit(from\_vec,rotation\_float,radius)

return point\_of\_orbit(from\_vec,rotation\_float,radius / lag\_comp + 1)

def is\_in\_triangle(point\_a\_vec,point\_b\_vec,point\_c\_vec,target\_vec):

if (area\_of\_triangle(target\_vec,point\_a\_vec,point\_b\_vec) < 0 and area\_of\_triangle(target\_vec,point\_b\_vec,point\_c\_vec) < 0 and area\_of\_triangle(target\_vec,point\_c\_vec,point\_a\_vec) < 0):

return True

def area\_of\_triangle(point\_a\_vec,point\_b\_vec,point\_c\_vec):

return ((point\_a\_vec[0] \* (point\_b\_vec[1] - point\_c\_vec[1])) + (point\_b\_vec[0] \* (point\_c\_vec[1] - point\_a\_vec[1])) + (point\_c\_vec[0] \* (point\_a\_vec[1] - point\_b\_vec[1]))) / 2

def clamp(value,max,min):

if (value > max):

return max

if (value < min):

return min

return value

class vector2:

def \_\_init\_\_(this,x,y):

this.x = x

this.y = y

class vector3:

def \_\_init\_\_(this,x,y,z):

this.x = x

this.y = y

this.z = z

class food:

def \_\_init\_\_(this):

this.pos = [random.randint(150,650),random.randint(50,550)]

this.carbs = random.randint(0,255)

this.protein = random.randint(0,255)

def draw(this):

pygame.draw.circle(panel,(this.carbs,this.protein,0),this.pos,2)

class herbivore:

def \_\_init\_\_(this):

this.strand\_stomach = genetic\_code(7)

this.strand\_bmr = genetic\_code(3)

this.strand\_sight = genetic\_code(6)

this.strand\_speed = genetic\_code(2)

this.strand\_reprod = genetic\_code(8)

this.strand\_litter\_size = genetic\_code(3)

this.strand\_total\_genotype = this.strand\_stomach.strand+this.strand\_bmr.strand+this.strand\_sight.strand+this.strand\_speed.strand+this.strand\_reprod.strand+this.strand\_litter\_size.strand

this.species\_code = read\_dna\_binary(this.strand\_total\_genotype)

#print("generated, genome:",this.strand\_stomach.strand,this.strand\_bmr.strand,this.strand\_sight.strand,this.strand\_speed.strand,this.strand\_reprod.strand,this.strand\_litter\_size.strand)

this.speed = this.strand\_speed.numerical\_value - 1

this.pos = [random.randint(150,650),random.randint(50,550)]

this.reproduction\_chance = this.strand\_reprod.numerical\_value

this.litter\_size = this.strand\_litter\_size.numerical\_value + 1

this.rotation = 0

this.sight = this.strand\_sight.numerical\_value + 30

this.stomach = 10

this.stomach\_max = this.strand\_stomach.numerical\_value + 50

this.bmr = this.strand\_bmr.numerical\_value + 1

this.target = None

this.dummy = None

this.reproduction\_progress = 0

this.memory = []

this.strands = []

this.nutrition = this.strand\_stomach.numerical\_value + 10

this.dormant = False

this.satiated = False

this.eating = False

this.being\_eaten = False

this.habitat\_radius = random.randint(10,150)

this.habitat = [random.randint(150 + (this.habitat\_radius // 2),650 - (this.habitat\_radius // 2)),random.randint(50 + (this.habitat\_radius // 2),550 - (this.habitat\_radius // 2))]

this.nose = 0

#print(this.sight,this.stomach\_max,this.bmr)

def draw(this,custom = False,custom\_pos = None):

this.nose = point\_of\_orbit(this.pos,this.rotation,10)

if (not this.dormant):

if (this.eating):

pygame.draw.line(panel,vgui\_ray\_broken,this.pos,this.nose,3)#nose layer

else:

pygame.draw.line(panel,vgui\_entity\_nose,this.pos,this.nose,3)#nose layer

pygame.draw.circle(panel,vgui\_entity\_herbivore,this.pos,5)#body layer

def sight\_check(this,target\_vec):

p\_1 = point\_of\_orbit(this.pos, this.rotation + 270, this.sight)

p\_2 = point\_of\_orbit(this.pos, this.rotation + 90, this.sight)

p\_3 = point\_of\_orbit(this.pos, this.rotation, this.sight)

pygame.draw.line(panel,vgui\_warning\_1,p\_1,p\_2)

pygame.draw.line(panel,vgui\_warning\_1,p\_1,p\_3)

pygame.draw.line(panel,vgui\_warning\_1,p\_3,p\_2)

if (is\_in\_triangle(p\_1,p\_2,p\_3,target\_vec)):return True

else:return False

def create\_move(this):

#print(this.strands[0].protein[1])

try:food\_object\_array.index(this.target)

except:

this.target = None

return

if (this.stomach >= this.stomach\_max-1):

this.satiated = True

if (this.satiated and this.stomach>this.stomach\_max / 2):

this.wander()

return

for hunter in entity\_object\_array:

if hunter.target == this:

this.wander()

return

this.satiated = False

this.dormant = False

if (vgui\_checkbox\_sim\_lag\_comp.state):factor = lag\_comp

else:factor = 1

if (distance\_to(this.pos,this.target.pos) <= (4 / factor) + 1):

this.eating = True

this.stomach += 0.08 \* lag\_comp

this.target.carbs -= 0.25 \* lag\_comp

this.target.protein -= 0.25 \* lag\_comp

this.nutrition += 0.2

if (this.target.carbs<=0 or this.target.protein<=0):

del food\_object\_array[food\_object\_array.index(this.target)]

this.target = None

else:

this.eating = False

this.rotation = rad\_to\_deg(this.pos,this.target.pos)

this.pos = get\_next\_move\_to(this.pos,this.rotation,2+this.speed)

this.stomach -= (this.bmr / (100 \* lag\_comp))

this.stomach = clamp(this.stomach,this.stomach\_max,0)

def wander(this):

this.stomach -= (this.bmr / (100 \* lag\_comp))

this.stomach = clamp(this.stomach,this.stomach\_max,0)

if (this.stomach < 10):

this.nutrition -= 0.5

if (this.nutrition < 5 and this.stomach <= 1):

del entity\_object\_array[entity\_object\_array.index(this)]

#print(this.stomach\_max,this.stomach,this.reproduction\_progress)

if (this.stomach > this.stomach\_max \* 0.8):

if (distance\_to(this.pos,this.habitat) <= 4):

this.dormant = True

if (random.randint(1,256) < this.reproduction\_chance and this.reproduction\_progress > 99):

this.reproduce()

this.reproduction\_progress+=(0.5\*lag\_comp)

else:

this.rotation = rad\_to\_deg(this.pos,this.habitat)

this.pos = get\_next\_move\_to(this.pos,this.rotation,2+this.speed)

else:

this.load\_from\_memory()

this.dormant = False

if (this.dummy == None):

this.dummy = dummy()

this.rotation = rad\_to\_deg(this.pos,this.dummy.pos)

this.pos = get\_next\_move\_to(this.pos,this.rotation,2+this.speed)

if (distance\_to(this.pos,this.dummy.pos) <= (4/lag\_comp) + 1):

this.dummy = None

def load\_from\_memory(this):

for ent in this.memory:

try:

food\_object\_array.index(ent)

except:

del this.memory[this.memory.index(ent)]

if (len(this.memory) != 0):this.target = random.choice(this.memory)

return False

def reproduce(this):

for i in range(this.litter\_size):

offspring = herbivore()

offspring.pos = this.pos

offspring.strand\_stomach = this.strand\_stomach

offspring.strand\_bmr = this.strand\_bmr

offspring.strand\_sight = this.strand\_sight

offspring.strand\_speed = this.strand\_speed

offspring.strand\_reprod = this.strand\_reprod

offspring.strand\_litter\_size = this.strand\_litter\_size

#print("generated, genome:",offspring.strand\_stomach.strand,offspring.strand\_bmr.strand,offspring.strand\_sight.strand,offspring.strand\_speed.strand,offspring.strand\_reprod.strand,offspring.strand\_litter\_size.strand)

entity\_object\_array.append(offspring)

this.stomach = 10

this.reproduction\_progress = 0

class primary\_producer:

def \_\_init\_\_(this):

this.richness = random.randint(1,255)

this.amount = random.randint(1,80)

this.radius = random.randint(80,200)

this.pos = [random.randint(150 + (this.radius // 2),650 - (this.radius // 2)),random.randint(50 + (this.radius // 2),550 - (this.radius // 2))]

this.max = random.randint(55,450)

this.associated\_objects = []

def tick(this):

if (random.randint(1,math.ceil(300 \* lag\_comp)) < this.amount and len(this.associated\_objects) < this.max):

food\_obj = food()

food\_obj.carbs = random.randint(this.richness // 2,this.richness)

food\_obj.protein = random.randint(this.richness // 2,this.richness)

food\_obj.pos = point\_of\_orbit(this.pos,random.randint(1,360),random.randint(1,(this.radius // 2)))

food\_object\_array.append(food\_obj)

this.associated\_objects.append(food\_obj)

class carnivore:

def \_\_init\_\_(this):

this.pos = [random.randint(150,650),random.randint(50,550)]

this.rotation = 0

this.strand\_stomach = genetic\_code(7)

this.strand\_bmr = genetic\_code(3)

this.strand\_sight = genetic\_code(6)

this.strand\_speed = genetic\_code(2)

this.strand\_reprod = genetic\_code(8)

this.strand\_litter\_size = genetic\_code(1)

this.sight = this.strand\_sight.numerical\_value + 30

this.target = None

this.satiated = False

this.dummy = None

this.stomach = 20

this.stomach\_max = this.strand\_stomach.numerical\_value + 70

this.speed = this.strand\_speed.numerical\_value - 1

this.bmr = this.strand\_bmr.numerical\_value+1

this.litter\_size = this.strand\_litter\_size.numerical\_value+1

this.memory = []

this.nose = 0

this.reproduction\_progress = 0

def draw(this):

this.nose = point\_of\_orbit(this.pos,this.rotation,10)

pygame.draw.line(panel,vgui\_entity\_nose,this.pos,this.nose,3)#nose layer

pygame.draw.circle(panel,vgui\_entity\_carnivore,this.pos,5)#body layer

def sight\_check(this,target\_vec):

p\_1 = point\_of\_orbit(this.pos, this.rotation + 270, this.sight)

p\_2 = point\_of\_orbit(this.pos, this.rotation + 90, this.sight)

p\_3 = point\_of\_orbit(this.pos, this.rotation, this.sight)

if (is\_in\_triangle(p\_1,p\_2,p\_3,target\_vec)):

return True

else:

return False

def create\_move(this):

try:

entity\_object\_array.index(this.target)

except:

this.target = None

return

if (this.stomach<140):

this.satiated=False

if (this.stomach > 140 and this.satiated):

this.wander()

return

if (vgui\_checkbox\_sim\_lag\_comp.state):

factor = lag\_comp

else:

factor = 1

if (distance\_to(this.pos,this.target.pos) <= (4 / factor) + 1):

this.target.being\_eaten = True

this.stomach+=1

this.target.nutrition-=1

if (this.target.nutrition<=0):

del entity\_object\_array[entity\_object\_array.index(this.target)]

if (this.stomach>199):

this.satiated = True

this.target = None

else:

if (not this.sight\_check(this.target.pos)):

this.target = None

return

this.rotation = rad\_to\_deg(this.pos,this.target.pos)

this.pos = get\_next\_move\_to(this.pos,this.rotation,2+this.speed)

this.stomach -= (this.bmr / (100 \* lag\_comp))

this.stomach = clamp(this.stomach,this.stomach\_max,0)

def wander(this):

if (this.dummy == None):

this.dummy = dummy()

this.rotation = rad\_to\_deg(this.pos,this.dummy.pos)

this.pos = get\_next\_move\_to(this.pos,this.rotation,2+this.speed)

this.stomach -= (this.bmr / (100 \* lag\_comp))

this.stomach = clamp(this.stomach,this.stomach\_max,0)

this.reproduction\_progress += 0.05

if (distance\_to(this.pos,this.dummy.pos) <= (4/lag\_comp) + 1):

this.dummy = None

if (this.stomach > this.stomach\_max-1 and this.reproduction\_progress>99):

this.reproduce()

def reproduce(this):

for i in range(this.litter\_size):

offspring = carnivore()

offspring.pos = this.pos

offspring.strand\_stomach = this.strand\_stomach

offspring.strand\_bmr = this.strand\_bmr

offspring.strand\_sight = this.strand\_sight

offspring.strand\_speed = this.strand\_speed

offspring.strand\_reprod = this.strand\_reprod

offspring.strand\_litter\_size = this.strand\_litter\_size

print("generated, genome:",offspring.strand\_stomach.strand,offspring.strand\_bmr.strand,offspring.strand\_sight.strand,offspring.strand\_speed.strand,offspring.strand\_reprod.strand,offspring.strand\_litter\_size.strand)

hunter\_object\_array.append(offspring)

this.stomach = 10

this.reproduction\_progress = 0

class dummy:

def \_\_init\_\_(this):

this.pos = [random.randint(150,650),random.randint(50,550)]

this.pos[0] = clamp(this.pos[0],650,150)

this.pos[1] = clamp(this.pos[1],550,50)

for i in range(Config["layer-1"]["food"]):

food\_object\_array.append(food())

for i in range(Config["layer-1"]["herbivores"]):

entity\_object\_array.append(herbivore())

for i in range(Config["layer-1"]["carnivores"]):

hunter\_object\_array.append(carnivore())

class button: #probably will revisit but looks alright

def \_\_init\_\_(this,position\_vec,label\_str,color = vgui\_fore,color\_hover = vgui\_state\_4,color\_text = vgui\_aux\_text\_internal):

this.pos = position\_vec

this.label = label\_str

this.col = color

this.col\_hov = color\_hover

this.text\_cl = color\_text

def draw(this):

c\_vec = pygame.mouse.get\_pos()

c\_bool = pygame.mouse.get\_pressed()[0]

pygame.draw.rect(panel, vgui\_bounding, pygame.Rect(this.pos[0] - 1,this.pos[1] - 1,52,12))#bounding scope

pygame.draw.rect(panel, this.col, pygame.Rect(this.pos[0],this.pos[1],50,10))#fore scope

if (c\_vec[0] > this.pos[0] and c\_vec[0] < this.pos[0] + 50 and c\_vec[1] > this.pos[1] and c\_vec[1] < this.pos[1] + 10):

if (c\_bool):

return True

else:

#pygame.draw.rect(panel, vgui\_fore, pygame.Rect(this.pos[0] - 1,this.pos[1] - 1,52,12))#bounding scope

pygame.draw.rect(panel, this.col\_hov, pygame.Rect(this.pos[0],this.pos[1],50,10))#aux scope

txt = ui\_font\_scale\_3.render(this.label, True, this.text\_cl)

panel.blit(txt,(this.pos[0] - (txt.get\_width() / 2) + 25,this.pos[1]))#overlap scope

return False

txt = ui\_font\_scale\_3.render(this.label, True, this.text\_cl)

panel.blit(txt,(this.pos[0] - (txt.get\_width() / 2) + 25,this.pos[1]))#overlap scope

class slider: #not done yet

def \_\_init\_\_(this,position\_vec,label\_str,min\_int,max\_int,value\_int):

this.pos = position\_vec

this.label = label\_str

this.min = min\_int

this.max = max\_int#value = (cursor.x - x) / (float(position) / float(max\_value))

this.val = ((value\_int / (this.max-this.min)) \* 100)

def draw(this):

c\_vec = pygame.mouse.get\_pos()

c\_bool = pygame.mouse.get\_pressed()[0]

pygame.draw.rect(panel, vgui\_bounding, pygame.Rect(this.pos[0] - 1,this.pos[1] - 1,102,12))#bounding scope

pygame.draw.rect(panel, vgui\_fore, pygame.Rect(this.pos[0],this.pos[1],100,10))#fore scope

pygame.draw.rect(panel, vgui\_state\_5, pygame.Rect(this.pos[0] - 1,this.pos[1] - 1,this.val + 2,12))

pygame.draw.rect(panel, vgui\_state\_1, pygame.Rect(this.pos[0],this.pos[1],this.val,10))#aux scope

if (c\_vec[0] > this.pos[0] and c\_vec[0] < this.pos[0] + 100 and c\_vec[1] > this.pos[1] and c\_vec[1] < this.pos[1] + 10 and c\_bool):

this.val = c\_vec[0] - this.pos[0]

panel.blit(ui\_font\_scale\_3.render(str(math.floor(((this.val / 100) \* (this.max - this.min)) + this.min)), True, vgui\_aux\_text\_external),(this.pos[0]+102,this.pos[1]))#overlap scope

txt = ui\_font\_scale\_3.render(this.label, True, vgui\_aux\_text\_external)

panel.blit(txt,(this.pos[0] - (txt.get\_width() / 2) + 50,this.pos[1] + 11))#overlap scope

return math.floor(((this.val / 100) \* (this.max - this.min)) + this.min)

class check\_box: #this looks nice

def \_\_init\_\_(this,position\_vec,label\_str,state):

this.pos = position\_vec

this.label = label\_str

this.state = state

this.c\_state = False

def draw(this):

c\_vec = pygame.mouse.get\_pos()

c\_bool = pygame.mouse.get\_pressed()[0]

pygame.draw.rect(panel, vgui\_bounding, pygame.Rect(this.pos[0] - 1,this.pos[1] - 1,12,12))#bounding scope

pygame.draw.rect(panel, vgui\_fore, pygame.Rect(this.pos[0],this.pos[1],10,10))#fore scope

if (this.state):

pygame.draw.rect(panel, vgui\_state\_3, pygame.Rect(this.pos[0] + 2,this.pos[1] + 2,6,6))#aux scope

if (c\_vec[0] > this.pos[0] and c\_vec[0] < this.pos[0] + 10 and c\_vec[1] > this.pos[1] and c\_vec[1] < this.pos[1] + 10):

if (c\_bool != this.c\_state and c\_bool):

this.state = not this.state

elif(this.state):

pygame.draw.rect(panel, vgui\_state\_1, pygame.Rect(this.pos[0] + 2,this.pos[1] + 2,6,6))#overlap scope

else:

pygame.draw.rect(panel, vgui\_state\_2, pygame.Rect(this.pos[0] + 2,this.pos[1] + 2,6,6))#overlap scope

this.c\_state = c\_bool

panel.blit(ui\_font\_scale\_3.render(this.label, True, vgui\_aux\_text\_external),(this.pos[0]+11,this.pos[1]))#overlap scope

return this.state

def draw\_visual\_bar(value,minim,maxim,pos,label,color = vgui\_state\_1):

pygame.draw.rect(panel, vgui\_bounding, pygame.Rect(pos[0] - 1,pos[1] - 1,102,12))#bounding scope

pygame.draw.rect(panel, vgui\_fore, pygame.Rect(pos[0],pos[1],100,10))#fore scope

pygame.draw.rect(panel, color, pygame.Rect(pos[0],pos[1],((value / (maxim-minim)) \* 100),10))#fore scope

panel.blit(ui\_font\_scale\_3.render(str(math.floor(((value / 100) \* (maxim - minim)) + minim)), True, vgui\_aux\_text\_external),(pos[0]+102,pos[1]))#overlap scope

txt = ui\_font\_scale\_3.render(label, True, vgui\_aux\_text\_external)

panel.blit(txt,(pos[0] - (txt.get\_width() / 2) + 50,pos[1] + 11))#overlap scope

class color\_selector: #this also looks nice

def \_\_init\_\_(this,position\_vec,current\_col\_vec,label\_str):

this.pos = position\_vec

this.col = current\_col\_vec

this.state = False

this.c\_state = False

this.label = label\_str

def draw(this):

c\_vec = pygame.mouse.get\_pos()

c\_bool = pygame.mouse.get\_pressed()[0]

if (this.state):

pygame.draw.rect(panel, vgui\_bounding, pygame.Rect(this.pos[0] - 1,this.pos[1] - 1,52,52))#bounding scope

for red in range(50):

for green in range(50):

blue = (50 - ((red + green) // 2))

pygame.draw.rect(panel, (red \* 5,green \* 5,blue \* 5), pygame.Rect(this.pos[0] + red,this.pos[1] + green,1,1))#aux scope

if (c\_vec[0] > this.pos[0] and c\_vec[0] < this.pos[0] + 50 and c\_vec[1] > this.pos[1] and c\_vec[1] < this.pos[1] + 50 and c\_bool and c\_bool != this.c\_state):

this.col = ((c\_vec[0] - this.pos[0]) \* 5,(c\_vec[1] - this.pos[1]) \* 5,(50 - (((c\_vec[0] - this.pos[0]) + (c\_vec[1] - this.pos[1])) // 2)) \* 5)

elif (c\_bool and c\_bool != this.c\_state):

this.state = not this.state

pygame.draw.rect(panel, vgui\_important, pygame.Rect(this.c2s()[0] - 1,this.c2s()[1] - 1,2,2))#overlay scope

pygame.draw.rect(panel, vgui\_bounding, pygame.Rect(this.pos[0] + 50,this.pos[1] - 1,12,12))#overlay scope

pygame.draw.rect(panel, this.col, pygame.Rect(this.pos[0] + 51,this.pos[1],10,10))#overlay scope

else:

pygame.draw.rect(panel, vgui\_bounding, pygame.Rect(this.pos[0] - 1,this.pos[1] - 1,12,12))#bounding scope

pygame.draw.rect(panel, this.col, pygame.Rect(this.pos[0],this.pos[1],10,10))#aux scope

if (c\_vec[0] > this.pos[0] and c\_vec[0] < this.pos[0] + 10 and c\_vec[1] > this.pos[1] and c\_vec[1] < this.pos[1] + 10 and c\_bool and c\_bool != this.c\_state):

this.state = not this.state

this.c\_state = c\_bool

panel.blit(ui\_font\_scale\_3.render(this.label, True, vgui\_aux\_text\_external),(this.pos[0]+11,this.pos[1]))#overlap scope

this.c\_state = c\_bool

def c2s(this): #color to screen

reversed\_red = (this.col[0] / 5) + this.pos[0]

reversed\_green = (this.col[1] / 5) + this.pos[1]

return (reversed\_red,reversed\_green)

class selection\_interface\_s: #meh

def \_\_init\_\_(this,position\_vec,selections\_array,selected\_pointer):

this.pos = position\_vec

this.selections = selections\_array

this.selected = selected\_pointer

this.state = False

this.c\_state = False

def draw(this):

c\_vec = pygame.mouse.get\_pos()

c\_bool = pygame.mouse.get\_pressed()[0]

pygame.draw.rect(panel, vgui\_bounding, pygame.Rect(this.pos[0] - 1,this.pos[1] - 1,52,12))#bounding scope

pygame.draw.rect(panel, vgui\_fore, pygame.Rect(this.pos[0],this.pos[1],50,10))#fore scope

txt = ui\_font\_scale\_3.render(this.selections[this.selected], True, vgui\_aux\_text\_internal, vgui\_fore)

panel.blit(txt,(this.pos[0] - (txt.get\_width() / 2) + 25,this.pos[1]))#overlap scope

if (c\_vec[0] > this.pos[0] and c\_vec[0] < this.pos[0] + 50 and c\_vec[1] > this.pos[1] and c\_vec[1] < this.pos[1] + 10 and c\_bool and c\_bool != this.c\_state):

this.state = not this.state

this.c\_state = this.state

if (this.state):

for selection in range(len(this.selections)):

pygame.draw.rect(panel, vgui\_bounding, pygame.Rect(this.pos[0] - 1,this.pos[1] - 1 + 10 + (selection \* 10),52,12))#bounding scope

pygame.draw.rect(panel, vgui\_fore, pygame.Rect(this.pos[0],this.pos[1] + 10 + (selection \* 10),50,10))#fore scope

txt = ui\_font\_scale\_3.render(this.selections[selection], True, vgui\_aux\_text\_internal, vgui\_fore)

panel.blit(txt,(this.pos[0] - (txt.get\_width() / 2) + 25,this.pos[1] + 10 + (selection \* 10)))#overlap scope

if (selection == this.selected):

pygame.draw.rect(panel, vgui\_state\_1, pygame.Rect(this.pos[0],this.pos[1] + 10 + (selection \* 10),50,10))#fore scope

txt = ui\_font\_scale\_3.render(this.selections[selection], True, vgui\_fore, vgui\_state\_1)

panel.blit(txt,(this.pos[0] - (txt.get\_width() / 2) + 25,this.pos[1] + 10 + (selection \* 10)))#overlap scope

if (c\_vec[0] > this.pos[0] and c\_vec[0] < this.pos[0] + 50 and c\_vec[1] > this.pos[1] + 10 + (selection \* 10) and c\_vec[1] < this.pos[1] + 20 + (selection \* 10) and c\_bool and c\_bool != this.c\_state):

this.selected = selection

this.state = not this.state

this.c\_state = c\_bool

class selection\_interface\_m: #better but still meh

def \_\_init\_\_(this,position\_vec,selections\_array,selected\_array,label\_str):

this.pos = position\_vec

this.selections = selections\_array

this.selected = selected\_array

this.state = False

this.c\_state = False

this.label = label\_str

def draw(this):

c\_vec = pygame.mouse.get\_pos()

c\_bool = pygame.mouse.get\_pressed()[0]

pygame.draw.rect(panel, vgui\_bounding, pygame.Rect(this.pos[0] - 1,this.pos[1] - 1,52,12))#bounding scope

pygame.draw.rect(panel, vgui\_fore, pygame.Rect(this.pos[0],this.pos[1],50,10))#fore scope

txt = ui\_font\_scale\_3.render(this.label, True, vgui\_aux\_text\_internal)

panel.blit(txt,(this.pos[0] - (txt.get\_width() / 2) + 25,this.pos[1]))#overlap scope

if (this.state):

for selection in range(len(this.selections)):

pygame.draw.rect(panel, vgui\_bounding, pygame.Rect(this.pos[0] - 1,this.pos[1] - 1 + 10 + (selection \* 10),52,12))#bounding scope

pygame.draw.rect(panel, vgui\_fore, pygame.Rect(this.pos[0],this.pos[1] + 10 +(selection \* 10),50,10))#fore scope

txt = ui\_font\_scale\_3.render(this.selections[selection], True, vgui\_aux\_text\_internal)

panel.blit(txt,(this.pos[0] - (txt.get\_width() / 2) + 25,this.pos[1] + 10 +(selection \* 10)))#overlap scope

if (this.selected[selection]):

pygame.draw.rect(panel, vgui\_state\_4, pygame.Rect(this.pos[0],this.pos[1] + 10 + (selection \* 10),50,10))#fore scope

txt = ui\_font\_scale\_3.render(this.selections[selection], True, vgui\_aux\_text\_internal)

panel.blit(txt,(this.pos[0] - (txt.get\_width() / 2) + 25,this.pos[1] + 10 + (selection \* 10)))#overlap scope

if (c\_vec[0] > this.pos[0] and c\_vec[0] < this.pos[0] + 50 and c\_vec[1] > this.pos[1] + 10 + (selection \* 10) and c\_vec[1] < this.pos[1] + 20 + (selection \* 10) and c\_bool and c\_bool != this.c\_state):

this.selected[selection] = not this.selected[selection]

if ((c\_vec[0] > this.pos[0] and c\_vec[0] < this.pos[0] + 50 and c\_vec[1] > this.pos[1] and c\_vec[1] < this.pos[1] + 10 and c\_bool and c\_bool != this.c\_state)):

this.state = not this.state

this.c\_state = c\_bool

class warning:

def \_\_init\_\_(this,pos,label,label2 = "",label3 = ""):

this.label = label

this.label2 = label2

this.label3 = label3

this.pos = pos

this.button = button([this.pos[0] + 50 ,this.pos[1] + 40],"ok")

def draw(this):

pygame.draw.rect(panel, vgui\_bounding, pygame.Rect(this.pos[0] - 1,this.pos[1] - 1,102,52))#bounding scope

pygame.draw.rect(panel, vgui\_warning\_1, pygame.Rect(this.pos[0],this.pos[1],100,50))#fore scope

panel.blit(ui\_font\_scale\_3.render(this.label, True, vgui\_aux\_text\_internal, vgui\_warning\_1),(this.pos[0],this.pos[1] + 5))#overlap scope

panel.blit(ui\_font\_scale\_3.render(this.label2, True, vgui\_aux\_text\_internal, vgui\_warning\_1),(this.pos[0],this.pos[1] + 15))#overlap scope

panel.blit(ui\_font\_scale\_3.render(this.label3, True, vgui\_aux\_text\_internal, vgui\_warning\_1),(this.pos[0],this.pos[1] + 25))#overlap scope

if (this.button.draw()):

return True

return False

class group\_box:

def \_\_init\_\_(this,pos,label,base,height):

this.pos = pos

this.label = ui\_font\_scale\_3.render(label, True, vgui\_aux\_text\_external,foreground)

#this.label.set\_alpha(127)

this.b = base

this.h = height

def draw(this):

#pygame.draw.rect(panel, vgui\_group\_fore, pygame.Rect(this.pos[0],this.pos[1],this.b,this.h))#fore scope

pygame.draw.line(panel,vgui\_bounding,this.pos,(this.pos[0] + this.b,this.pos[1]))

pygame.draw.line(panel,vgui\_bounding,this.pos,(this.pos[0],this.pos[1] + this.h))

pygame.draw.line(panel,vgui\_bounding,(this.pos[0] + this.b,this.pos[1]),(this.pos[0] + this.b,this.pos[1] + this.h))

pygame.draw.line(panel,vgui\_bounding,(this.pos[0] + this.b,this.pos[1] + this.h),(this.pos[0],this.pos[1] + this.h))

panel.blit(this.label,(this.pos[0] + (this.b // 5),this.pos[1] - 5))

class log\_entry:

def \_\_init\_\_(this,text,label):

this.txt = text

this.label = label

txt = ui\_font\_scale\_3.render(this.txt, True, vgui\_aux\_text\_internal)

this.label\_dimensions = [txt.get\_width() + 20,40]

def handle\_hover(this):

c\_vec = pygame.mouse.get\_pos()

c\_bool = pygame.mouse.get\_pressed()[0]

#pygame.draw.rect(panel, vgui\_bounding, pygame.Rect(this.pos[0] - 1,this.pos[1] - 1,this.label\_dimensions[0]+2,this.label\_dimensions[1]+2))

#pygame.draw.rect(panel, vgui\_fore, pygame.Rect(this.pos[0],this.pos[1],this.label\_dimensions[0],this.label\_dimensions[1]))

vgui\_button\_back = button((1,589),"<- back")

vgui\_button\_exit = button((749,1)," quit ", (255,0,0),(180,0,0),(20,20,20))

vgui\_button\_start = button((375,300),"start")

vgui\_button\_options = button((375,315),"options")

vgui\_button\_theme = button((375,330),"theme")

vgui\_button\_entity\_list\_manager = button((375,345),"edit ents")

vgui\_slider\_food = slider((100,75),"food amount",0,5000,Config["layer-1"]["food"])

vgui\_slider\_herb = slider((100,100),"herbivore amount",0,20,Config["layer-1"]["herbivores"])

vgui\_slider\_carn = slider((100,125),"carnivore amount",0,20,Config["layer-1"]["carnivores"])

vgui\_slider\_muta = slider((100,150),"mutation chance per 10 seconds",0,100,40)

vgui\_slider\_ray\_lazy = slider((500,75),"lazy tracing",0,400,20)

vgui\_slider\_ray\_mult = slider((500,100),"speed multiplier",1,10,1)

vgui\_slider\_ray\_add = slider((500,125),"drunk ray",1,10,5)

vgui\_slider\_photosynth = slider((660,50),"photosynthesis",1,100,50)

vgui\_slider\_sim\_slow\_val = slider((36,100),"slow amount",1,100,5)

vgui\_checkbox\_sim\_slow\_bool = check\_box((1,100),"slow",False)

vgui\_checkbox\_sim\_lag\_comp = check\_box((1,150),"lag comp",True)

vgui\_checkbox\_ray\_visualise = check\_box((500,160),"visualise",False)

vgui\_checkbox\_visualise\_math = check\_box((100,180),"visualise math",False)

vgui\_checkbox\_ray\_master = check\_box((100,195),"ray tracing",False)

vgui\_checkbox\_ray\_ehhh = check\_box((100,210),"ehhh",False)

vgui\_color\_ray\_visualise\_1 = color\_selector((570,160),vgui\_ray\_beam,"beam")

vgui\_color\_ray\_visualise\_2 = color\_selector((570,212),vgui\_ray\_broken,"broken")

vgui\_slc\_ray\_ignore = selection\_interface\_m((500,212),["food","creatures"],[False,False],"ignore")

vgui\_warning\_conf = warning([300,250],"default was loaded", "because didnt find","config")

environment = group\_box([60,60],"environment",180,200)

raytracing = group\_box([460,60],"ray tracing",180,200)

log\_index = [log\_entry("ok","hello"),log\_entry("ok","hello"),log\_entry("ok",""),log\_entry("ok",""),log\_entry("ok",""),log\_entry("ok",""),log\_entry("ok",""),log\_entry("ok",""),log\_entry("ok",""),log\_entry("ok",""),log\_entry("ok",""),log\_entry("ok",""),log\_entry("ok",""),log\_entry("ok",""),log\_entry("ok",""),log\_entry("ok",""),log\_entry("ok",""),log\_entry("ok",""),log\_entry("ok",""),log\_entry("ok",""),log\_entry("ok",""),log\_entry("ok",""),log\_entry("ok",""),log\_entry("ok",""),log\_entry("ok",""),log\_entry("ok",""),log\_entry("ok",""),log\_entry("ok",""),log\_entry("ok",""),log\_entry("ok",""),log\_entry("ok",""),log\_entry("ok","")]

def sim\_thread():

for food\_obj in food\_object\_array:

food\_obj.draw()

for herbivore in entity\_object\_array:

herbivore.draw()

if (herbivore.being\_eaten):

continue

if (herbivore.target != None):

herbivore.create\_move()

else:

if (herbivore.nose[0] > herbivore.pos[0]):radian = 0

else:radian = 1

distance = 100000000000

for food in food\_object\_array:

if (radian == 0 and food.pos[0] < herbivore.pos[0]):

continue

if (radian == 1 and food.pos[0] > herbivore.pos[0]):

continue

if abs(food.pos[0] - herbivore.pos[0]) > herbivore.sight:

continue

if abs(food.pos[1] - herbivore.pos[1]) > (herbivore.sight // 2):

continue

if (herbivore.sight\_check(food.pos)):

if (distance\_to(herbivore.pos,food.pos) < distance):

herbivore.target = food

distance = distance\_to(herbivore.pos,food.pos)

if food not in herbivore.memory:

herbivore.memory.append(food)

if (herbivore.target == None):

herbivore.wander()

for carnivore in hunter\_object\_array:

carnivore.draw()

if (carnivore.target != None):

carnivore.create\_move()

if (carnivore.nose[0] > carnivore.pos[0]):radian = 0

else:radian = 1

distance = 100000000000

if (carnivore.target==None):

for food in entity\_object\_array:

if (food.dormant):

continue

if (radian == 0 and food.pos[0] < carnivore.pos[0]):

continue

if (radian == 1 and food.pos[0] > carnivore.pos[0]):

continue

if abs(food.pos[0] - carnivore.pos[0]) > carnivore.sight:

continue

if abs(food.pos[1] - carnivore.pos[1]) > (carnivore.sight // 2):

continue

if (carnivore.sight\_check(food.pos)):

if (distance\_to(carnivore.pos,food.pos) < distance):

carnivore.target = food

distance = distance\_to(carnivore.pos,food.pos)

if (carnivore.target == None):

carnivore.wander()

def vgui\_thread():

if (vgui\_checkbox\_sim\_slow\_bool.draw()):

time.sleep(0.1 \* (vgui\_slider\_sim\_slow\_val.draw() / 100))

vgui\_checkbox\_sim\_lag\_comp.draw()

vgui\_slider\_photosynth.draw()

draw\_visual\_bar(40,1,100,(660,150),"drift",(255,255,0))

def simulation():

global tab

if (random.randint(1,math.ceil(100 \* lag\_comp)) < vgui\_slider\_photosynth.val):

food\_object\_array.append(food())

pygame.draw.rect(panel, vgui\_bounding, pygame.Rect(150,50,500,500))#sim area

if (vgui\_button\_back.draw()):

tab = 0

sim\_thread()

vgui\_thread()

def options():

global tab

global vgui\_ray\_beam

global vgui\_ray\_broken

if (vgui\_button\_back.draw()):

tab = 0

environment.draw()

raytracing.draw()

Config["layer-1"]["food"] = vgui\_slider\_food.draw()

Config["layer-1"]["herbivores"] = vgui\_slider\_herb.draw()

Config["layer-1"]["carnivores"] = vgui\_slider\_carn.draw()

vgui\_slider\_muta.draw()

vgui\_checkbox\_visualise\_math.draw()

vgui\_checkbox\_ray\_master.draw()

vgui\_checkbox\_ray\_ehhh.draw()

vgui\_slider\_ray\_lazy.draw()

vgui\_slider\_ray\_mult.draw()

vgui\_slider\_ray\_add.draw()

vgui\_checkbox\_ray\_visualise.draw()

vgui\_color\_ray\_visualise\_1.draw()

vgui\_color\_ray\_visualise\_2.draw()

vgui\_slc\_ray\_ignore.draw()

vgui\_ray\_beam = vgui\_color\_ray\_visualise\_1.col

vgui\_ray\_broken = vgui\_color\_ray\_visualise\_2.col

def ent\_manager():

global tab

if (vgui\_button\_back.draw()):

tab = 0

for ent in entity\_object\_array:

ent.draw(True,(400,entity\_object\_array.index(ent) \* 15 + 50))

def main\_menu():

global tab

if vgui\_button\_entity\_list\_manager.draw():

tab = 3

if vgui\_button\_start.draw():

tab = 2

if vgui\_button\_options.draw():

tab = 1

vgui\_button\_theme.draw()

def log\_manager():

txt = font\_alt.render("simulation log (hover over yellow text for extra information)", True, (255,255,255))

panel.blit(txt,(1000 - (txt.get\_width() / 2),10))

for entry in log\_index:

txt = ui\_font\_scale\_3.render(entry.txt, True, vgui\_aux\_text\_internal)

panel.blit(txt,(1000 - (txt.get\_width() / 2),30 + (log\_index.index(entry) \* 15)))

if (entry.label != ""):

panel.blit(ui\_font\_scale\_3.render(entry.label, True, (200,200,0)),(1000 + (txt.get\_width() / 2) + 10,30 + (log\_index.index(entry) \* 15)))

def main():

global lag\_comp

lag\_comp = clamp(lag\_comp,3,0.01)

sample\_time = time.time()

for event in pygame.event.get():

if event.type == pygame.QUIT:

json.dump(Config,open("config.json", "w"))

pygame.quit()

sys. exit()

panel.fill(foreground)

if (tab == 0):

main\_menu()

elif (tab == 1):

options()

elif (tab == 2):

simulation()

elif (tab == 3):

ent\_manager()

if (vgui\_button\_exit.draw()):

json.dump(Config,open("config.json", "w"))

pygame.quit()

sys. exit()

log\_manager()

master.draw()

log.draw()

pygame.display.flip()

if ((time.time() - sample\_time) != 0):

lag\_comp = ((math.ceil(1 / (time.time() - sample\_time))) / 100) #based off 100 ticks,

while True:

screen\_size = panel.get\_size()

vgui\_button\_back = button((1,screen\_size[1]-11),"<- back")

master = group\_box((0,0),"",screen\_size[0] - 401,screen\_size[1] - 1)

log = group\_box((799,0),"",399,599)

while vgui\_warning\_config:

for event in pygame.event.get():

if event.type == pygame.QUIT:

json.dump(Config,open("config.json", "w"))

pygame.quit()

sys. exit()

panel.fill(foreground)

if (vgui\_warning\_conf.draw()):

vgui\_warning\_config = False

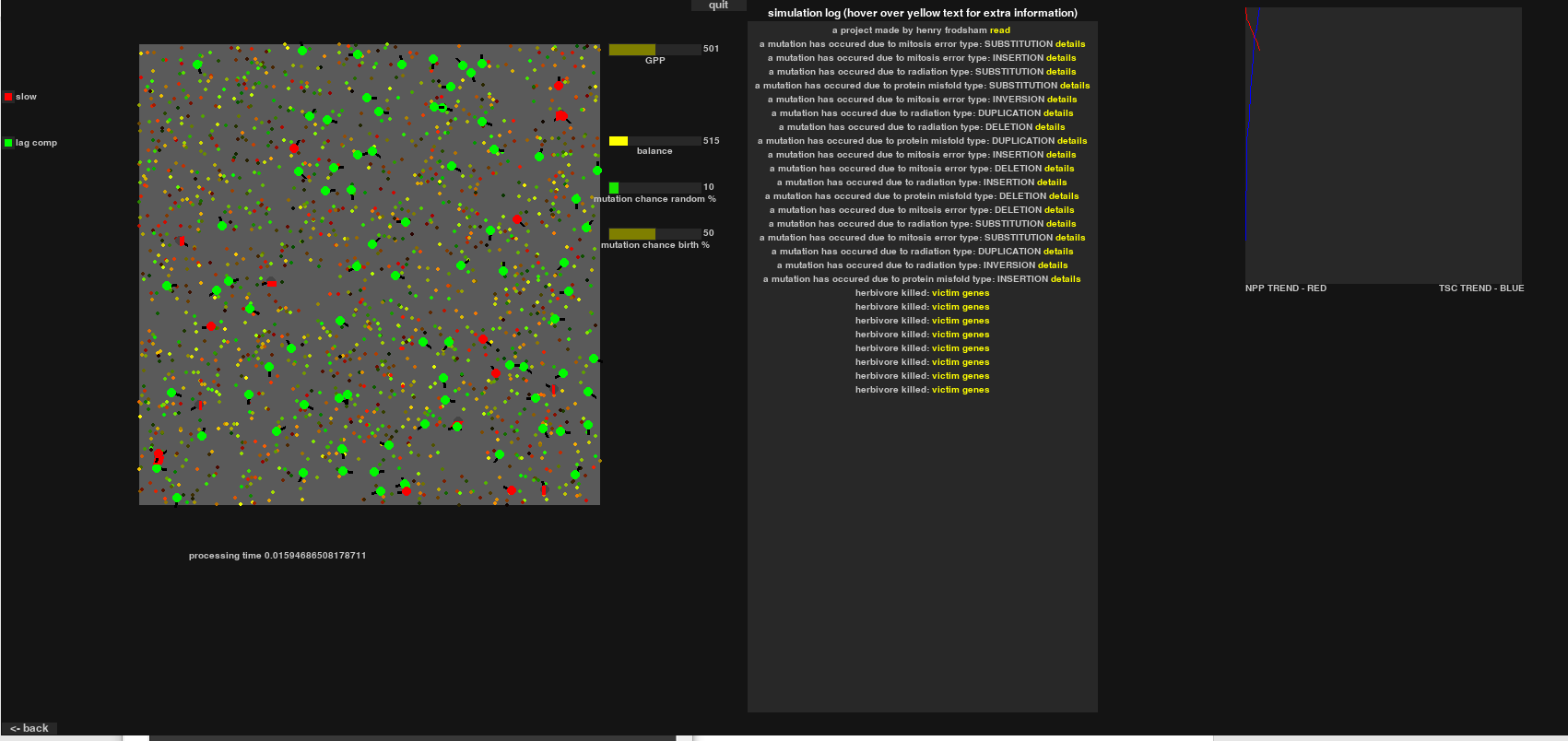
pygame.display.flip()

main()

### major additions from prototype to first version

* **Multilayer ui system** 
  + **How was this achieved?**
    - **Functions in main to call option\_page() etc**
  + **Why was this done?**
    - **To allow disctinct sections of the program to be allocated to their function, e.g a theme page to change the menu colors and an option page to manipulate the config file**
* **Simulation log backbone**
  + **How was this achieved?**
    - **Creating of a simulation\_log class to create log entries**
  + **Why was this done?**
    - **To allow easy addition of log elements to the log manager and class methods to manage individual elements e.g handling of special text**
* **Ability to change photosynthetic rate in the simulation**
  + **How was this achieved?**
    - **Silider element which manipulates the rate at which food objects are randomly added**
  + **Why was this done?**
    - **To allow the user easy access to selection pressures within the simulation page**
* **New ui theme**
  + **How was this achieved?**
    - **Darker background, removal of bounding from some elements to provide a cleaner feel**
  + **Why was this achieved?**
    - **Improve user friendliness, is less accessible however with the theme editor the user can change the theme in order to meet individual requirements.**

## Version 2



### Code

|  |
| --- |
| try: |
|  |  | import pygame |
|  |  | import random |
|  |  | import time |
|  |  | import math |
|  |  | import json |
|  |  | import sys |
|  |  | except: |
|  |  | print("you do not have the required libraries to run this program") |
|  |  | sys.exit() |
|  |  | default = {'layer-1': {'food': 2000, 'herbivores': 10, 'carnivores': 10}} |
|  |  | vgui\_warning\_config = False |
|  |  | try: |
|  |  | Config = json.load(open("config.json","r")) |
|  |  | except: |
|  |  | file = open("config.json", "a") |
|  |  | json.dump(default, file) |
|  |  | file.close() |
|  |  | Config = json.load(open("config.json","r")) |
|  |  | vgui\_warning\_config = True |
|  |  | pygame.init() |
|  |  | panel = pygame.display.set\_mode((1700,800),pygame.NOFRAME) |
|  |  | screen\_size = panel.get\_size() |
|  |  | #font\_main = pygame.font.Font('Bahnschrift.ttf', 10) |
|  |  | font\_alt = pygame.font.Font('freesansbold.ttf', 12) |
|  |  | ui\_font\_scale\_3 = pygame.font.Font('freesansbold.ttf', 10) |
|  |  | big = pygame.font.Font('freesansbold.ttf', 40) |
|  |  | try: |
|  |  | dvd = pygame.image.load('logo.png') |
|  |  | dvd = pygame.transform.scale(dvd, (100, 50)) |
|  |  | except: |
|  |  | print("logo.png not found") |
|  |  | dvd = big.render("logo.png is missing, please download it", True, (0,0,255)) |
|  |  | img\_size = dvd.get\_rect().size |
|  |  |  |
|  |  | x = random.randint(150, 1700-160) |
|  |  | y = random.randint(150, 800-160) |
|  |  | x\_speed = 1.5 |
|  |  | y\_speed = 1.5 |
|  |  |  |
|  |  | foreground = (20,20,20) |
|  |  | vgui\_fore = (40,40,40) |
|  |  | vgui\_bounding = (120,120,120) |
|  |  | vgui\_state\_0 = (10,10,220) |
|  |  | vgui\_state\_1 = (20,20,200) |
|  |  | vgui\_state\_2 = (20,20,180) |
|  |  | vgui\_state\_3 = (20,20,150) |
|  |  | vgui\_state\_4 = (20,20,120) |
|  |  | vgui\_state\_5 = (20,20,80) |
|  |  | vgui\_important = (0,0,0) |
|  |  | vgui\_aux\_text\_internal = (200,200,200) |
|  |  | vgui\_aux\_text\_external = (200,200,200) |
|  |  | vgui\_warning\_1 = (150,20,20) |
|  |  |  |
|  |  | vgui\_ray\_broken = (250,0,0) |
|  |  | vgui\_ray\_beam = (0,255,0) |
|  |  |  |
|  |  | vgui\_color\_ON = (0,255,0) |
|  |  | vgui\_color\_OFF = (255,0,0) |
|  |  | vgui\_group\_fore = (190,190,190) |
|  |  | vgui\_entity\_herbivore = (0,250,0) |
|  |  | vgui\_herbivore\_egg = (221,100,221) |
|  |  | vgui\_entity\_dead = (70,70,70) |
|  |  | vgui\_entity\_carnivore = (250,0,0) |
|  |  | vgui\_entity\_nose = (0,0,0) |
|  |  | entity\_object\_array = [] |
|  |  | hunter\_object\_array = [] |
|  |  | food\_object\_array = [] |
|  |  | producer\_object\_array = [] |
|  |  | egg\_object\_array = [] |
|  |  | log\_index = [] |
|  |  | tab = 0 |
|  |  | lag\_comp = 0 |
|  |  | bases = ["G","A","T","C"] |
|  |  | GPP = 0 |
|  |  | NPP\_SAMPLES = [] |
|  |  | TSC\_SAMPLES = [] |
|  |  | NPP = 0 |
|  |  | GSP = 0 |
|  |  | TSC = 0 |
|  |  | BMI = 0 |
|  |  | PE = 0 |
|  |  | tick = 0 |
|  |  | balance = 0 |
|  |  | sight\_average = [] |
|  |  | log\_var = 0 |
|  |  | def biodiversity\_index(): |
|  |  | num\_organisms = len(entity\_object\_array) |
|  |  | register = [] |
|  |  | for e in entity\_object\_array: |
|  |  | if e.introgenic\_dna not in register: |
|  |  | register.append(e.introgenic\_dna) |
|  |  | num\_species = len(register) |
|  |  | N = num\_organisms \* (num\_organisms - 1) |
|  |  | n = 0 |
|  |  | for r in register: |
|  |  | n+=entity\_object\_array.count(r) \* (entity\_object\_array.count(r) - 1) |
|  |  | return N / n |
|  |  | class log\_entry: |
|  |  | def \_\_init\_\_(this,text = "",label = "",label\_text = ""): |
|  |  | global log\_var |
|  |  | this.txt = text #main text displayed on log |
|  |  | this.label = label #special hoverable text |
|  |  | this.label\_text = label\_text #text displayed on the hover |
|  |  | txt = ui\_font\_scale\_3.render(this.txt, True, vgui\_aux\_text\_internal) #rect of text for display |
|  |  | this.label\_dimensions = [txt.get\_width() + 20,40] |
|  |  | #if (log\_var != 0): |
|  |  | # log\_var+=1 |
|  |  | def handle\_hover(this): |
|  |  | # if (30 + (log\_index.index(this) \* 15)) > 760: |
|  |  | #del log\_index[0] #remove off the screen logs from back to front |
|  |  | if len(log\_index) > 51: |
|  |  | log\_onscreen = log\_index[len(log\_index) - 51 - log\_var : len(log\_index) - 1 - log\_var] |
|  |  | else: |
|  |  | log\_onscreen = log\_index |
|  |  | if this not in log\_onscreen: |
|  |  | return |
|  |  | label\_text = ui\_font\_scale\_3.render(this.label\_text, True, vgui\_aux\_text\_internal) |
|  |  | label = ui\_font\_scale\_3.render(this.label, True, (255,255,0)) |
|  |  | c\_vec = pygame.mouse.get\_pos() # mouse pos |
|  |  | c\_bool = pygame.mouse.get\_pressed()[0] # mouse state |
|  |  | txt = ui\_font\_scale\_3.render(this.txt, True, vgui\_aux\_text\_internal) |
|  |  | panel.blit(txt,(1000 - ((txt.get\_width() + label.get\_width()) / 2),30 + (log\_onscreen.index(this) \* 15))) #display main text |
|  |  | panel.blit(label,(1000 + ((txt.get\_width() - label.get\_width()) / 2),30 + (log\_onscreen.index(this) \* 15)))# display special text |
|  |  | #print(c\_vec[0] > 1000 + (txt.get\_width() / 2),c\_vec[0] < 1000+label.get\_width()) |
|  |  | if (c\_vec[0] > 1000 + ((txt.get\_width() - label.get\_width()) / 2) and c\_vec[0] < 1000+label.get\_width()+ ((txt.get\_width() - label.get\_width()) / 2)): |
|  |  | if (c\_vec[1] > 30 + (log\_onscreen.index(this) \* 15) and c\_vec[1] < 40 + (log\_onscreen.index(this) \* 15)): #check for hover |
|  |  | if (this.label\_text == ""): |
|  |  | return #remove empty special text |
|  |  | if ((1000 + (txt.get\_width() / 2)) + label\_text.get\_width()+2) > 1200: |
|  |  | recession = ((1000 + (txt.get\_width() / 2)) + label\_text.get\_width()+2) - 1200 + 5 #recess ui bounder formula to prevent it going off screen |
|  |  | pygame.draw.rect(panel, vgui\_fore, pygame.Rect(1000 + (txt.get\_width() / 2) - recession,30 + (log\_onscreen.index(this) \* 15),label\_text.get\_width()+2,15))#fore scope |
|  |  | panel.blit(label\_text,(1000 + (txt.get\_width() / 2) - recession,30 + (log\_onscreen.index(this) \* 15) + 3)) |
|  |  | box = group\_box((1000 + (txt.get\_width() / 2) - recession,30 + (log\_onscreen.index(this) \* 15)),"",label\_text.get\_width()+2,15) |
|  |  | box.draw() #draw bounding ui |
|  |  | return |
|  |  | pygame.draw.rect(panel, vgui\_fore, pygame.Rect(1000 + (txt.get\_width() / 2),30 + (log\_onscreen.index(this) \* 15),label\_text.get\_width()+2,15))#fore scope |
|  |  | box = group\_box((1000 + (txt.get\_width() / 2),30 + (log\_onscreen.index(this) \* 15)),"",label\_text.get\_width()+2,15) |
|  |  | box.draw() #draw an unrecessed box |
|  |  | panel.blit(label\_text,(1000 + (txt.get\_width() / 2),30 + (log\_onscreen.index(this) \* 15) + 3)) |
|  |  | def generate\_dna\_sequence(length): |
|  |  | strand = [] |
|  |  | for i in range(length): |
|  |  | strand.append(random.choice(bases)) # randomly add a nucleotide base |
|  |  | return strand |
|  |  | def read\_dna\_protein(strand): |
|  |  | active\_site = read\_dna\_binary(strand[0:3]) |
|  |  | op\_code = read\_dna\_binary(strand[4:7]) |
|  |  | if bases.index(strand[len(strand) - 1]) == 0: |
|  |  | sign = True # handle negative sign bases |
|  |  | else: |
|  |  | sign = False |
|  |  | if (sign): |
|  |  | op\_code -= op\_code \* 2 # make the opcode negative |
|  |  | return [active\_site,op\_code] |
|  |  | def read\_dna\_binary(strand): |
|  |  | binary = "" |
|  |  | for base in strand: |
|  |  | if bases.index(base) == 1 or bases.index(base) == 2: # checks for a non complementary base e.g A or G as A and T are complementary |
|  |  | binary += "0" |
|  |  | else: |
|  |  | binary += "1" |
|  |  | return int(binary,2) #convert bases to binary |
|  |  | def create\_mutation(strand): |
|  |  | if (strand == None): |
|  |  | return |
|  |  | mutated = strand[0 : len(strand)] #make a copy of the argument |
|  |  | mutation\_type = random.randint(1,300) # insertion and base deletion mutations are rarer so their chances are reduced |
|  |  | if (len(strand) == 1): |
|  |  | mutation\_type = random.randint(1,249) |
|  |  | #frame shift aka deletion |
|  |  | if (mutation\_type <= 300 and mutation\_type > 250): |
|  |  | del mutated[mutated.index(random.choice(mutated))] # remove a base |
|  |  | return (mutated,"DELETION") |
|  |  | #substitution |
|  |  | if (mutation\_type <= 100): |
|  |  | mutated[mutated.index(random.choice(mutated))] = random.choice(bases) #change a base |
|  |  | return (mutated,"SUBSTITUTION") |
|  |  | #inversion |
|  |  | if (mutation\_type <= 200 and mutation\_type > 150): |
|  |  | return(mutated[::-1],"INVERSION") |
|  |  | #duplication |
|  |  | if (mutation\_type <= 150 and mutation\_type > 100): |
|  |  | mutated.append(mutated[len(mutated) - 1]) |
|  |  | return(mutated,"DUPLICATION") |
|  |  | #insertion |
|  |  | if (mutation\_type <= 250 and mutation\_type > 200): |
|  |  | mutated.append(random.choice(bases)) # add a base |
|  |  | return (mutated,"INSERTION") |
|  |  |  |
|  |  | #class genetic\_code: |
|  |  | #def \_\_init\_\_(this,length): |
|  |  | #this.strand = generate\_dna\_sequence(length) |
|  |  | #this.numerical\_value = read\_dna\_binary(this.strand) |
|  |  | #def update(this): |
|  |  | #this.numerical\_value = read\_dna\_binary(this.strand) |
|  |  | #def mutation(this): |
|  |  | #this.strand = create\_mutation(this.strand) |
|  |  | #this.update() |
|  |  | def Sqr(num): |
|  |  | return num\*num #simple math for cleanliness |
|  |  | def distance\_to(vec\_point\_a,vec\_point\_b): |
|  |  | change\_in\_x = abs(vec\_point\_a[0] - vec\_point\_b[0]) |
|  |  | change\_in\_y = abs(vec\_point\_a[1] - vec\_point\_b[1]) |
|  |  | return math.sqrt(Sqr(change\_in\_x) + Sqr(change\_in\_y)) #pythagoras |
|  |  | def rad\_to\_deg(vec\_point\_a,vec\_point\_b): |
|  |  | x1, y1 = vec\_point\_a |
|  |  | x2, y2 = vec\_point\_b |
|  |  | angle = math.degrees(math.atan2(y2 - y1, x2 - x1)) |
|  |  | return angle |
|  |  | #hyp = distance\_to(vec\_point\_a,vec\_point\_b) |
|  |  | #opp = distance\_to((vec\_point\_b[0],vec\_point\_a[1]),vec\_point\_b) # find unknown sides |
|  |  | #raw\_angle = pythag(opp,hyp) |
|  |  | #X = vec\_point\_b[0] - vec\_point\_a[0] |
|  |  | #Y = vec\_point\_b[1] - vec\_point\_a[1] |
|  |  | #if X < 0 and Y < 0: |
|  |  | # angle = raw\_angle + 180 # add preexisting offset |
|  |  | #elif X < 0: |
|  |  | # angle = -raw\_angle + 180 |
|  |  | #elif Y < 1: |
|  |  | # angle = -raw\_angle |
|  |  | #else: |
|  |  | # angle = raw\_angle |
|  |  | #return angle |
|  |  | def pythag(opp,hyp): |
|  |  | try: |
|  |  | sine = (opp/hyp) # only used for sight cone as second method can cause an error when the object is perfectly adjacent |
|  |  | except: |
|  |  | return 0 |
|  |  | return (math.asin(sine) \* 57.296) |
|  |  | def point\_of\_orbit(host\_vec,rotation\_float,radius\_int): |
|  |  | c = math.cos(rotation\_float \* 0.0174) |
|  |  | s = math.sin(rotation\_float \* 0.0174) |
|  |  | return [(host\_vec[0] + radius\_int \* c),(host\_vec[1] + radius\_int \* s)] # create a tangent and add an offset |
|  |  | def trace\_ray(from\_vec,to\_vec,array): |
|  |  | lazy\_trace\_threshold = 0 |
|  |  | ray\_skip\_mult = 1 # skip a ray step per trace |
|  |  | ray\_visualize = True |
|  |  | ray\_ignore\_groups = [False,False] |
|  |  | ray\_hitbox\_add = 0 # remove ray precision |
|  |  | ray\_angle = rad\_to\_deg(from\_vec,to\_vec) #angle to trace |
|  |  | for ray in range(500 - lazy\_trace\_threshold): |
|  |  | RayPos = point\_of\_orbit(from\_vec,ray\_angle,ray\*ray\_skip\_mult) |
|  |  | if (not ray\_ignore\_groups[0]): |
|  |  | for food\_obj in array: # food group |
|  |  | if (distance\_to(RayPos,food\_obj.pos) > 3): |
|  |  | continue # exclude non collided |
|  |  | if (food\_obj.pos == to\_vec): |
|  |  | pygame.draw.line(panel,vgui\_ray\_beam,from\_vec,to\_vec) # collided with the target |
|  |  | return True |
|  |  | else: |
|  |  | #hit something else so missed (meaning its colluded) |
|  |  | #if (ray\_visualize): |
|  |  | #pygame.draw.line(panel,vgui\_ray\_beam,from\_vec,to\_vec) |
|  |  | #pygame.draw.line(panel,vgui\_ray\_broken,RayPos,to\_vec) |
|  |  | return False |
|  |  | def get\_next\_move\_to(from\_vec,rotation\_float,radius = 0.2): |
|  |  | #old method |
|  |  | if (not vgui\_checkbox\_sim\_lag\_comp.state): |
|  |  | return point\_of\_orbit(from\_vec,rotation\_float,radius) |
|  |  | return point\_of\_orbit(from\_vec,rotation\_float,radius / lag\_comp + 1) |
|  |  | def think\_next\_move(to\_vec,from\_vec,step): |
|  |  | if (to\_vec == None or from\_vec == None): |
|  |  | return |
|  |  | #primitive, will change maybe using tensorflow |
|  |  | step \*= 0.5 |
|  |  | step /= lag\_comp # account for lag |
|  |  | if from\_vec[0] < to\_vec[0]: |
|  |  | from\_vec[0] += step |
|  |  | else: |
|  |  | from\_vec[0] -= step |
|  |  | if (from\_vec[1] < to\_vec[1]): |
|  |  | from\_vec[1] += step |
|  |  | else: |
|  |  | from\_vec[1] -= step |
|  |  | return from\_vec |
|  |  | def is\_in\_triangle(point\_a\_vec,point\_b\_vec,point\_c\_vec,target\_vec): |
|  |  | if (area\_of\_triangle(target\_vec,point\_a\_vec,point\_b\_vec) < 0 and area\_of\_triangle(target\_vec,point\_b\_vec,point\_c\_vec) < 0 and area\_of\_triangle(target\_vec,point\_c\_vec,point\_a\_vec) < 0): # formula checking if coordinates are in a triangle, compares the areas of triangles derived from the coordinates of the target. explained in design |
|  |  | return True |
|  |  | def area\_of\_triangle(point\_a\_vec,point\_b\_vec,point\_c\_vec): |
|  |  | return ((point\_a\_vec[0] \* (point\_b\_vec[1] - point\_c\_vec[1])) + (point\_b\_vec[0] \* (point\_c\_vec[1] - point\_a\_vec[1])) + (point\_c\_vec[0] \* (point\_a\_vec[1] - point\_b\_vec[1]))) / 2 |
|  |  | def clamp(value,maxi,mini): |
|  |  | #prevent a value getting too large, substitute method too std::clamp |
|  |  | if (value > maxi): |
|  |  | return maxi |
|  |  | if (value < mini): |
|  |  | return mini |
|  |  | return value |
|  |  |  |
|  |  | class vector2: |
|  |  | #need to add overloads, will use these next rewrite |
|  |  | def \_\_init\_\_(this,x,y): |
|  |  | this.x = x |
|  |  | this.y = y |
|  |  | class vector3: |
|  |  | def \_\_init\_\_(this,x,y,z): |
|  |  | this.x = x |
|  |  | this.y = y |
|  |  | this.z = z |
|  |  |  |
|  |  | class food: |
|  |  | def \_\_init\_\_(this): |
|  |  | this.pos = [random.randint(150,650),random.randint(50,550)] #random spot in the simulation area |
|  |  | this.carbs = random.randint(0,255) |
|  |  | this.protein = random.randint(0,255) |
|  |  | this.being\_eaten = False |
|  |  | def draw(this): |
|  |  | pygame.draw.circle(panel,(this.carbs,this.protein,0),this.pos,2) |
|  |  | class egg: |
|  |  | def \_\_init\_\_(this,pos,genes,intron): |
|  |  | this.pos = pos |
|  |  | this.genes = genes |
|  |  | this.intron = intron |
|  |  | this.count\_down = 3000 |
|  |  | this.log = log\_entry("egg laid: progress: " + str(this.count\_down)," parent",str(this.genes)) # add a dynamic log entry showing the egg progress |
|  |  | log\_index.append(this.log) |
|  |  |  |
|  |  | def draw(this): |
|  |  | pygame.draw.circle(panel,vgui\_herbivore\_egg,this.pos,3) #draw the egg |
|  |  | def tick(this): |
|  |  | this.count\_down -= (1 / lag\_comp) \* balance |
|  |  | if (this.count\_down < 0): #herbivore being born |
|  |  | infant = herbivore() |
|  |  | infant.egg\_progress = -600 |
|  |  | log\_index.append(log\_entry("herbivore born: ","genome",str(infant.genes))) |
|  |  | infant.introgenic\_dna = this.intron |
|  |  | infant.pos = this.pos |
|  |  | infant.genes = this.genes |
|  |  | if (random.randint(1,100) < vgui\_slider\_birth\_muta\_chance.val): #simulate independant assortment |
|  |  | index\_to\_patch = infant.genes.index(random.choice(infant.genes)) |
|  |  | mutated = create\_mutation(infant.genes[index\_to\_patch]) |
|  |  | old\_strand = infant.genes[index\_to\_patch] |
|  |  | try: |
|  |  | mutated[1] |
|  |  | except: |
|  |  | mutated = [mutated] |
|  |  | mutated.append("SHORT") # temporary measure to stop genetic material being completely deleted |
|  |  | log\_index.append(log\_entry("a mutation has occured on birth! type: " + mutated[1]," details",str(str(old\_strand) + " -> " + str(mutated[0])) + " -> " + str(read\_dna\_binary(old\_strand)) + " -> " + str(read\_dna\_binary(mutated[0])))) |
|  |  | infant.genes[index\_to\_patch] = mutated[0] |
|  |  | infant.refold() #refold proteins |
|  |  | entity\_object\_array.append(infant) |
|  |  | del egg\_object\_array[egg\_object\_array.index(this)] |
|  |  | try: |
|  |  | del log\_index[log\_index.index(this.log)] |
|  |  | except: |
|  |  | return |
|  |  | return |
|  |  | try: |
|  |  | this.new\_log = log\_entry("egg laid: progress: " + str(math.ceil(this.count\_down)),"parent",str(this.genes)) |
|  |  | log\_index[log\_index.index(this.log)] = this.new\_log |
|  |  | this.log = this.new\_log |
|  |  | except: |
|  |  | return |
|  |  | class herbivore: |
|  |  | def \_\_init\_\_(this): |
|  |  | this.genes = [generate\_dna\_sequence(7),generate\_dna\_sequence(2),generate\_dna\_sequence(6),generate\_dna\_sequence(2),generate\_dna\_sequence(8),generate\_dna\_sequence(1)] |
|  |  | this.pos = [random.randint(150,650),random.randint(50,550)] |
|  |  | this.rotation = 0 # vision cone rotation |
|  |  | this.introgenic\_dna = generate\_dna\_sequence(25) # dna that doesnt do anything significant, will make this add visual remarks on species later |
|  |  | this.target = None |
|  |  | this.wait\_for = 0 |
|  |  | this.dummy = None |
|  |  | this.egg\_progress = 0 |
|  |  |  |
|  |  | this.nutrition = 100 |
|  |  | this.dead = False |
|  |  | this.nose = 0 |
|  |  |  |
|  |  | #hormone levels for fight or flight |
|  |  | this.epinephrine = 0 |
|  |  | this.cns\_stimulant = 0 |
|  |  | this.cns\_depressant = 0 |
|  |  | this.tolerance\_factor = 1 |
|  |  | this.refold() |
|  |  | def refold(this): |
|  |  | #reset variables after a dna change |
|  |  | this.stomach = 10 |
|  |  | this.stomach\_max = read\_dna\_binary(this.genes[0]) + 20 |
|  |  | this.bmr = read\_dna\_binary(this.genes[1]) + 1 |
|  |  | this.sight = read\_dna\_binary(this.genes[2]) |
|  |  | this.speed = read\_dna\_binary(this.genes[3]) |
|  |  | this.litter\_size = read\_dna\_binary(this.genes[5]) + 1 |
|  |  | def draw(this,custom = False,custom\_pos = None): |
|  |  | this.nose = point\_of\_orbit(this.pos,this.rotation,10) |
|  |  | pygame.draw.line(panel,vgui\_entity\_nose,this.pos,this.nose,3)#nose layer |
|  |  | pygame.draw.circle(panel,vgui\_entity\_herbivore,this.pos,5)#body layer |
|  |  | if (this.dead): |
|  |  | pygame.draw.circle(panel,vgui\_entity\_dead,this.pos,5)#body layer |
|  |  | #txt = font\_alt.render(str(math.ceil(this.stomach)), True, (0,0,0)) |
|  |  | #panel.blit(txt,this.pos) |
|  |  | def sight\_check(this,target\_vec): |
|  |  | #check if a target is in the vision cone |
|  |  | p\_1 = point\_of\_orbit(this.pos, this.rotation + 270, this.sight) |
|  |  | p\_2 = point\_of\_orbit(this.pos, this.rotation + 90, this.sight) |
|  |  | p\_3 = point\_of\_orbit(this.pos, this.rotation, this.sight) |
|  |  | #pygame.draw.line(panel,vgui\_warning\_1,p\_1,p\_2) |
|  |  | #pygame.draw.line(panel,vgui\_warning\_1,p\_1,p\_3) |
|  |  | #pygame.draw.line(panel,vgui\_warning\_1,p\_3,p\_2) |
|  |  | if (is\_in\_triangle(p\_1,p\_2,p\_3,target\_vec)): |
|  |  | #was in the vision cone |
|  |  | return True |
|  |  | else: |
|  |  | return False |
|  |  | def create\_move(this): |
|  |  | try: |
|  |  | #check if the target was killed by by other means |
|  |  | food\_object\_array.index(this.target) |
|  |  | except: |
|  |  | this.target = None |
|  |  | return |
|  |  | this.rotation = rad\_to\_deg(this.pos,this.target.pos) #calculate the relative angle |
|  |  | this.pos = think\_next\_move(this.target.pos,this.pos,this.speed + 2 + (this.epinephrine /100)) # update the position |
|  |  | if (distance\_to(this.pos,this.target.pos) <= (4/lag\_comp) + 1): |
|  |  | this.stomach+= (this.target.carbs \* 0.1 + this.target.protein \* 0.1) |
|  |  | if (this.stomach > (this.stomach\_max \* 0.95)): |
|  |  | this.nutrition+= ((this.target.carbs \* 0.1 + this.target.protein \* 0.1) \* 0.35) |
|  |  | del food\_object\_array[food\_object\_array.index(this.target)] |
|  |  | this.wait\_for = random.randint(math.ceil(60 / this.bmr),math.ceil(250 / this.bmr)) |
|  |  | #force them to pause for a second after eating, making them vulnerable |
|  |  | def wander(this): |
|  |  | if (this.dummy == None): |
|  |  | this.dummy = dummy() |
|  |  | this.rotation = rad\_to\_deg(this.pos,this.dummy.pos) |
|  |  | this.pos = think\_next\_move(this.dummy.pos,this.pos,(((this.speed - (this.cns\_depressant\*1.3)) + 2 + (this.epinephrine / 200)))) |
|  |  | if (distance\_to(this.pos,this.dummy.pos) <= (4/lag\_comp) + 1): |
|  |  | this.dummy = None |
|  |  | this.wait\_for = random.randint(math.ceil(60 / this.bmr),math.ceil(250 / this.bmr)) |
|  |  | #force them to pause for a second after eating, making them vulnerable |
|  |  | def kill(this): |
|  |  | this.dead = True |
|  |  | #only reason this is a method is because there will be more here in the future |
|  |  | #create\_blood\_effect() |
|  |  | def decay(this): |
|  |  | #visual bleeding, |
|  |  | pygame.draw.line(panel,(255,0,0),this.pos,point\_of\_orbit(this.pos,random.randint(0,360),random.randint(1,10)),10) |
|  |  | if (this.nutrition < 0): |
|  |  | del entity\_object\_array[entity\_object\_array.index(this)] |
|  |  | return |
|  |  | this.nutrition -= 0.1 / lag\_comp |
|  |  | if (random.randint(1,200) != 20): |
|  |  | return |
|  |  | decay = food() |
|  |  | #return their biomass to the ecosytem |
|  |  | decay.pos = [this.pos[0] + 2 + random.randint(-7,5),this.pos[1] + 2 + random.randint(-7,5)] |
|  |  | food\_object\_array.append(decay) |
|  |  | def run(this,obj): |
|  |  | # the fight or flight respinse |
|  |  | this.wait\_for = 0 |
|  |  | this.target = None |
|  |  | this.dummy = dummy() |
|  |  | this.dummy.pos = point\_of\_orbit(this.pos, obj.rotation + random.randint(-5,5), 50) # face them precisely away from their pursuer |
|  |  | this.epinephrine+=0.7 \* this.cns\_depressant # add adrenaline, dulled by exhaustion |
|  |  | this.epinephrine = clamp(this.epinephrine,20 \* this.cns\_depressant,0) |
|  |  | if (clamp(this.dummy.pos[0],650,150) != this.dummy.pos[0] or clamp(this.dummy.pos[1],550,50) != this.dummy.pos[1]): # check if they are being chased off the map |
|  |  | this.dummy = dummy() |
|  |  | this.wander() |
|  |  | class carnivore: |
|  |  | def \_\_init\_\_(this): |
|  |  | this.pos = [random.randint(150,650),random.randint(50,550)] |
|  |  | this.rotation = 0 |
|  |  | this.strand\_stomach = generate\_dna\_sequence(7) |
|  |  | this.strand\_bmr = generate\_dna\_sequence(3) |
|  |  | this.strand\_sight = generate\_dna\_sequence(6) |
|  |  | this.strand\_speed = generate\_dna\_sequence(3) |
|  |  | this.strand\_reprod = generate\_dna\_sequence(8) |
|  |  | this.strand\_litter\_size = generate\_dna\_sequence(1) |
|  |  | this.sight = read\_dna\_binary(this.strand\_sight) + 30 |
|  |  | this.target = None |
|  |  | this.dummy = None |
|  |  | this.stomach\_max = read\_dna\_binary(this.strand\_stomach) + 70 |
|  |  | this.stomach = this.stomach\_max / 2 |
|  |  | this.speed = read\_dna\_binary(this.strand\_speed) - 1 |
|  |  | this.bmr = read\_dna\_binary(this.strand\_bmr)+1 |
|  |  | this.litter\_size = read\_dna\_binary(this.strand\_litter\_size)+1 |
|  |  | this.nose = 0 |
|  |  | this.egg\_progress = 0 |
|  |  | this.dead = False |
|  |  | this.wait\_for = 0 |
|  |  | def draw(this): |
|  |  | this.nose = point\_of\_orbit(this.pos,this.rotation,10) |
|  |  | pygame.draw.line(panel,vgui\_entity\_nose,this.pos,this.nose,3)#nose layer |
|  |  | pygame.draw.circle(panel,vgui\_entity\_carnivore,this.pos,5)#body layer |
|  |  | def sight\_check(this,target\_vec): |
|  |  | p\_1 = point\_of\_orbit(this.pos, this.rotation + 270, this.sight) |
|  |  | p\_2 = point\_of\_orbit(this.pos, this.rotation + 90, this.sight) |
|  |  | p\_3 = point\_of\_orbit(this.pos, this.rotation, this.sight) |
|  |  | if (is\_in\_triangle(p\_1,p\_2,p\_3,target\_vec)): |
|  |  | return True |
|  |  | else: |
|  |  | return False |
|  |  | def create\_move(this): |
|  |  | try: |
|  |  | entity\_object\_array.index(this.target) |
|  |  | except: |
|  |  | this.target = None |
|  |  | return |
|  |  | if (not this.sight\_check(this.target.pos)): |
|  |  | this.target = None |
|  |  | #give the prey a chance to escape, if they are fast enough |
|  |  | return |
|  |  | this.target.run(this) # trigger their fight or flight response |
|  |  | this.rotation = rad\_to\_deg(this.pos,this.target.pos) |
|  |  | this.pos = think\_next\_move(this.target.pos,this.pos,this.speed + 2) |
|  |  | if (distance\_to(this.pos,this.target.pos) <= (4/lag\_comp) + 1): |
|  |  | log\_index.append(log\_entry("herbivore killed: ","victim genes",str(this.target.introgenic\_dna))) |
|  |  | this.target.kill() |
|  |  | this.stomach+=this.target.nutrition \* 0.65 |
|  |  | this.target.nutrition-=this.target.nutrition \* 0.65 |
|  |  | this.target = None |
|  |  | this.wait\_for = random.randint(math.ceil(160 / this.bmr),math.ceil(400 / this.bmr)) # prevent them from killing too many at once in the same area |
|  |  | def wander(this): |
|  |  | if (this.dummy == None): |
|  |  | this.dummy = dummy() |
|  |  | this.dummy.pos = point\_of\_orbit(this.pos,random.randint(1,360),random.randint(40,120)) |
|  |  | while (clamp(this.dummy.pos[0],650,150) != this.dummy.pos[0] or clamp(this.dummy.pos[1],550,50) != this.dummy.pos[1]): |
|  |  | this.dummy.pos = point\_of\_orbit(this.pos,random.randint(1,360),random.randint(40,120)) |
|  |  | this.rotation = rad\_to\_deg(this.pos,this.dummy.pos) |
|  |  | this.pos = think\_next\_move(this.dummy.pos,this.pos,this.speed + 2) |
|  |  | if (distance\_to(this.pos,this.dummy.pos) <= (4/lag\_comp) + 1): |
|  |  | this.dummy = None |
|  |  | this.wait\_for = random.randint(math.ceil(60 / this.bmr),math.ceil(250 / this.bmr)) |
|  |  | def kill(this): |
|  |  | this.dead = True |
|  |  | def decay(this): |
|  |  | x = 2 # as of right now predators cant be killed so this is empty |
|  |  | class dummy: |
|  |  | def \_\_init\_\_(this): #creates a fake creature for the organism to "chase", can be fed into wander and create move |
|  |  | this.pos = [random.randint(150,650),random.randint(50,550)] |
|  |  | this.pos[0] = clamp(this.pos[0],650,150) |
|  |  | this.pos[1] = clamp(this.pos[1],550,50) |
|  |  | for i in range(Config["layer-1"]["food"]): |
|  |  | #add starting food |
|  |  | food\_object\_array.append(food()) |
|  |  | for i in range(Config["layer-1"]["herbivores"]): |
|  |  | #add starting herbivores |
|  |  | entity\_object\_array.append(herbivore()) |
|  |  | for i in range(Config["layer-1"]["carnivores"]): |
|  |  | #add starting carnivores |
|  |  | hunter\_object\_array.append(carnivore()) |
|  |  | class button: #probably will revisit but looks alright |
|  |  | def \_\_init\_\_(this,position\_vec,label\_str,color = vgui\_fore,color\_hover = vgui\_state\_4,color\_text = vgui\_aux\_text\_internal): |
|  |  | this.pos = position\_vec # button pos |
|  |  | this.label = label\_str # button label |
|  |  | def draw(this): |
|  |  | c\_vec = pygame.mouse.get\_pos() # mouse pos |
|  |  | c\_bool = pygame.mouse.get\_pressed()[0] # L mouse state |
|  |  | #pygame.draw.rect(panel, vgui\_bounding, pygame.Rect(this.pos[0] - 1,this.pos[1] - 1,52,12))#bounding scope |
|  |  | pygame.draw.rect(panel, vgui\_fore, pygame.Rect(this.pos[0],this.pos[1],60,13))#fore scope |
|  |  | if (c\_vec[0] > this.pos[0] and c\_vec[0] < this.pos[0] + 60 and c\_vec[1] > this.pos[1] and c\_vec[1] < this.pos[1] + 13): |
|  |  | if (c\_bool): |
|  |  | return True |
|  |  | else: |
|  |  | #pygame.draw.rect(panel, vgui\_fore, pygame.Rect(this.pos[0] - 1,this.pos[1] - 1,52,12))#bounding scope |
|  |  | pygame.draw.rect(panel, vgui\_state\_4, pygame.Rect(this.pos[0],this.pos[1],60,13))#aux scope |
|  |  | txt = font\_alt.render(this.label, True, vgui\_aux\_text\_internal) |
|  |  | panel.blit(txt,(this.pos[0] - (txt.get\_width() / 2) + 30,this.pos[1]))#overlap scope |
|  |  | return False |
|  |  | txt = font\_alt.render(this.label, True, vgui\_aux\_text\_internal) |
|  |  | panel.blit(txt,(this.pos[0] - (txt.get\_width() / 2) + 30,this.pos[1]))#overlap scope |
|  |  | class slider: #too primitive |
|  |  | def \_\_init\_\_(this,position\_vec,label\_str,min\_int,max\_int,value\_int,invert = False,experimental = False): |
|  |  | this.pos = position\_vec |
|  |  | this.label = label\_str |
|  |  | this.min = min\_int |
|  |  | this.max = max\_int#value = (cursor.x - x) / (float(position) / float(max\_value)) |
|  |  | this.val = abs((value\_int / (this.max-this.min)) \* 100) # convert to screen |
|  |  | this.invert = invert |
|  |  | this.experimental = experimental |
|  |  | def draw(this): |
|  |  | c\_vec = pygame.mouse.get\_pos() |
|  |  | c\_bool = pygame.mouse.get\_pressed()[0] |
|  |  | #pygame.draw.rect(panel, vgui\_bounding, pygame.Rect(this.pos[0] - 1,this.pos[1] - 1,102,12))#bounding scope |
|  |  | pygame.draw.rect(panel, vgui\_fore, pygame.Rect(this.pos[0],this.pos[1],100,12))#fore scope |
|  |  | pygame.draw.rect(panel, vgui\_fore, pygame.Rect(this.pos[0] - 1,this.pos[1] - 1,this.val + 2,14)) # the slider bound rect |
|  |  | if (this.experimental): |
|  |  | for i in range(math.floor(this.val)): |
|  |  | if (this.invert): |
|  |  | pygame.draw.line(panel,(i\*2.55,255 - (i\*2.55),0),(this.pos[0] + i,this.pos[1]),(this.pos[0] + i,this.pos[1]+12)) |
|  |  | else: |
|  |  | pygame.draw.line(panel,(255 - (i\*2.55),i\*2.55,0),(this.pos[0] + i,this.pos[1]),(this.pos[0] + i,this.pos[1]+12)) |
|  |  | elif (this.invert): |
|  |  | pygame.draw.rect(panel, (this.val\*2.55,255 - (this.val\*2.55),0), pygame.Rect(this.pos[0],this.pos[1],this.val,12))#aux scope |
|  |  | else: |
|  |  | pygame.draw.rect(panel, (255 - (this.val\*2.55),this.val\*2.55,0), pygame.Rect(this.pos[0],this.pos[1],this.val,12))#aux scope |
|  |  | if (c\_vec[0] > this.pos[0] and c\_vec[0] < this.pos[0] + 100 and c\_vec[1] > this.pos[1] and c\_vec[1] < this.pos[1] + 12 and c\_bool): |
|  |  | this.val = c\_vec[0] - this.pos[0] |
|  |  | panel.blit(ui\_font\_scale\_3.render(str(math.floor(((this.val / 100) \* (this.max - this.min)) + this.min)), True, vgui\_aux\_text\_external),(this.pos[0]+102,this.pos[1]))#overlap scope |
|  |  | txt = ui\_font\_scale\_3.render(this.label, True, vgui\_aux\_text\_external) |
|  |  | panel.blit(txt,(this.pos[0] - (txt.get\_width() / 2) + 50,this.pos[1] + 13))#overlap scope |
|  |  | return math.floor(((this.val / 100) \* (this.max - this.min)) + this.min) # convert to value |
|  |  | class check\_box: #this looks nice |
|  |  | def \_\_init\_\_(this,position\_vec,label\_str,state): |
|  |  | this.pos = position\_vec |
|  |  | this.label = label\_str |
|  |  | this.state = state |
|  |  | this.c\_state = False |
|  |  | this.hovered = False |
|  |  | def draw(this): |
|  |  | this.hovered = False |
|  |  | c\_vec = pygame.mouse.get\_pos() |
|  |  | c\_bool = pygame.mouse.get\_pressed()[0] |
|  |  | #pygame.draw.rect(panel, vgui\_bounding, pygame.Rect(this.pos[0] - 1,this.pos[1] - 1,12,12))#bounding scope |
|  |  | pygame.draw.rect(panel, vgui\_fore, pygame.Rect(this.pos[0],this.pos[1],14,14))#fore scope |
|  |  | pygame.draw.rect(panel, vgui\_color\_OFF, pygame.Rect(this.pos[0] + 3,this.pos[1] + 3,8,8))#aux scope |
|  |  | if (this.state): |
|  |  | pygame.draw.rect(panel, vgui\_color\_ON, pygame.Rect(this.pos[0] + 3,this.pos[1] + 3,8,8))#aux scope |
|  |  | if (c\_vec[0] > this.pos[0] and c\_vec[0] < this.pos[0] + 10 and c\_vec[1] > this.pos[1] and c\_vec[1] < this.pos[1] + 10): |
|  |  | this.hovered = True |
|  |  | if (c\_bool != this.c\_state and c\_bool): |
|  |  | this.state = not this.state |
|  |  | #elif(this.state): |
|  |  | # pygame.draw.rect(panel, vgui\_state\_1, pygame.Rect(this.pos[0] + 3,this.pos[1] + 3,8,8))#overlap scope |
|  |  | #else: |
|  |  | # pygame.draw.rect(panel, vgui\_state\_2, pygame.Rect(this.pos[0] + 3,this.pos[1] + 3,8,8))#overlap scope |
|  |  | this.c\_state = c\_bool |
|  |  | panel.blit(ui\_font\_scale\_3.render(this.label, True, vgui\_aux\_text\_external),(this.pos[0]+15,this.pos[1] + 2))#overlap scope |
|  |  | return this.state |
|  |  | def draw\_visual\_bar(value,minim,maxim,pos,label,color = vgui\_state\_1): |
|  |  | #pygame.draw.rect(panel, vgui\_bounding, pygame.Rect(pos[0] - 1,pos[1] - 1,102,12))#bounding scope |
|  |  | pygame.draw.rect(panel, vgui\_fore, pygame.Rect(pos[0],pos[1],100,10))#fore scope |
|  |  | pygame.draw.rect(panel, color, pygame.Rect(pos[0],pos[1],((value / (maxim-minim)) \* 100),10))#fore scope |
|  |  | panel.blit(ui\_font\_scale\_3.render(str(math.floor(((value / 100) \* (maxim - minim)) + minim)), True, vgui\_aux\_text\_external),(pos[0]+102,pos[1]))#overlap scope |
|  |  | txt = ui\_font\_scale\_3.render(label, True, vgui\_aux\_text\_external) |
|  |  | panel.blit(txt,(pos[0] - (txt.get\_width() / 2) + 50,pos[1] + 11))#overlap scope |
|  |  | class color\_selector: #this also looks nice |
|  |  | def \_\_init\_\_(this,position\_vec,current\_col\_vec,label\_str): |
|  |  | this.pos = position\_vec |
|  |  | this.col = current\_col\_vec |
|  |  | this.state = False |
|  |  | this.c\_state = False |
|  |  | this.label = label\_str |
|  |  | #this.invert = check\_box((position\_vec[0] + 51,position\_vec[1]+40),"flip blue",False) |
|  |  | #this.remove = check\_box((position\_vec[0] + 51,position\_vec[1]+30),"no blue",False) |
|  |  | this.extras = selection\_interface\_s((position\_vec[0] + 51,position\_vec[1]),["default","flip blue","no blue"],0) |
|  |  | def draw(this): |
|  |  | c\_vec = pygame.mouse.get\_pos() #mouse pos |
|  |  | c\_bool = pygame.mouse.get\_pressed()[0]# L mouse pos |
|  |  | if (this.state): |
|  |  | pygame.draw.rect(panel, vgui\_bounding, pygame.Rect(this.pos[0] - 1,this.pos[1] - 1,52,52))#bounding scope |
|  |  | for red in range(50): |
|  |  | for green in range(50): |
|  |  | if (this.extras.selected == 1): # different settings for different color spectrums |
|  |  | blue = (red + green) // 2 |
|  |  | elif (this.extras.selected == 2): |
|  |  | blue = 0 |
|  |  | else: |
|  |  | blue = (50 - ((red + green) // 2)) |
|  |  | pygame.draw.rect(panel, (red \* 5,green \* 5,blue \* 5), pygame.Rect(this.pos[0] + red,this.pos[1] + green,1,1))#aux scope |
|  |  | #pygame.draw.rect(panel, (red \* 5,green \* 5,blue \* 5), pygame.Rect(this.pos[0] + red +2,this.pos[1] + green + 2,2,2))#aux scope |
|  |  | if (c\_vec[0] > this.pos[0] and c\_vec[0] < this.pos[0] + 50 and c\_vec[1] > this.pos[1] and c\_vec[1] < this.pos[1] + 50 and c\_bool and c\_bool != this.c\_state): # if hovered, clicked and not double pressed |
|  |  | if (this.extras.selected == 1): # convert the col on the pressed cursor |
|  |  | this.col = ((c\_vec[0] - this.pos[0]) \* 5,(c\_vec[1] - this.pos[1]) \* 5, (((c\_vec[0] - this.pos[0])+(c\_vec[1] - this.pos[1])) // 2) \* 5) |
|  |  | elif (this.extras.selected == 2): |
|  |  | this.col = ((c\_vec[0] - this.pos[0]) \* 5,(c\_vec[1] - this.pos[1]) \* 5,0) |
|  |  | else: |
|  |  | this.col = ((c\_vec[0] - this.pos[0]) \* 5,(c\_vec[1] - this.pos[1]) \* 5,(50 - (((c\_vec[0] - this.pos[0]) + (c\_vec[1] - this.pos[1])) // 2)) \* 5) |
|  |  | elif (c\_bool and c\_bool != this.c\_state and not this.extras.hovered): |
|  |  | this.state = not this.state |
|  |  | pygame.draw.rect(panel, vgui\_important, pygame.Rect(this.c2s()[0] - 1,this.c2s()[1] - 1,4,4))#overlay scope |
|  |  | pygame.draw.rect(panel, this.col, pygame.Rect(this.c2s()[0],this.c2s()[1],2,2))#overlay scope |
|  |  | #pygame.draw.rect(panel, vgui\_bounding, pygame.Rect(this.pos[0] + 50,this.pos[1] - 1,12,12))#overlay scope |
|  |  | #pygame.draw.rect(panel, this.col, pygame.Rect(this.pos[0] + 51,this.pos[1],10,10))#overlay scope |
|  |  | #this.invert.draw() |
|  |  | #this.remove.draw() |
|  |  | this.extras.draw() |
|  |  | else: |
|  |  | pygame.draw.rect(panel, vgui\_bounding, pygame.Rect(this.pos[0] - 1,this.pos[1] - 1,32,12))#bounding scope |
|  |  | pygame.draw.rect(panel, this.col, pygame.Rect(this.pos[0],this.pos[1],30,10))#aux scope |
|  |  | if (c\_vec[0] > this.pos[0] and c\_vec[0] < this.pos[0] + 30 and c\_vec[1] > this.pos[1] and c\_vec[1] < this.pos[1] + 10 and c\_bool and c\_bool != this.c\_state): |
|  |  | this.state = not this.state |
|  |  | this.c\_state = c\_bool |
|  |  | panel.blit(ui\_font\_scale\_3.render(this.label, True, vgui\_aux\_text\_external),(this.pos[0]+31,this.pos[1]))#overlap scope |
|  |  | this.c\_state = c\_bool |
|  |  | def c2s(this): #color to screen |
|  |  | reversed\_red = (this.col[0] / 5) + this.pos[0] |
|  |  | reversed\_green = (this.col[1] / 5) + this.pos[1] |
|  |  | return (reversed\_red,reversed\_green) |
|  |  | class selection\_interface\_s: #meh |
|  |  | def \_\_init\_\_(this,position\_vec,selections\_array,selected\_pointer): |
|  |  | this.pos = position\_vec |
|  |  | this.selections = selections\_array |
|  |  | this.selected = selected\_pointer |
|  |  | this.state = False |
|  |  | this.c\_state = False |
|  |  | this.hovered = False |
|  |  | def draw(this): |
|  |  | c\_vec = pygame.mouse.get\_pos() |
|  |  | c\_bool = pygame.mouse.get\_pressed()[0] |
|  |  | pygame.draw.rect(panel, vgui\_bounding, pygame.Rect(this.pos[0] - 1,this.pos[1] - 1,52,12))#bounding scope |
|  |  | pygame.draw.rect(panel, vgui\_fore, pygame.Rect(this.pos[0],this.pos[1],50,10))#fore scope |
|  |  | txt = ui\_font\_scale\_3.render(this.selections[this.selected], True, vgui\_aux\_text\_internal) |
|  |  | panel.blit(txt,(this.pos[0] - (txt.get\_width() / 2) + 25,this.pos[1]))#overlap scope |
|  |  | this.hovered = False |
|  |  | if (c\_vec[0] > this.pos[0] and c\_vec[0] < this.pos[0] + 50 and c\_vec[1] > this.pos[1] and c\_vec[1] < this.pos[1] + 10): |
|  |  | this.hovered = True |
|  |  | if (c\_bool and c\_bool != this.c\_state): |
|  |  | this.state = not this.state |
|  |  | this.c\_state = this.state |
|  |  | if (this.state): |
|  |  | for selection in range(len(this.selections)): |
|  |  | pygame.draw.rect(panel, vgui\_bounding, pygame.Rect(this.pos[0] - 1,this.pos[1] - 1 + 10 + (selection \* 10),52,12))#bounding scope |
|  |  | pygame.draw.rect(panel, vgui\_fore, pygame.Rect(this.pos[0],this.pos[1] + 10 + (selection \* 10),50,10))#fore scope |
|  |  | txt = ui\_font\_scale\_3.render(this.selections[selection], True, vgui\_aux\_text\_internal) |
|  |  | panel.blit(txt,(this.pos[0] - (txt.get\_width() / 2) + 25,this.pos[1] + 10 + (selection \* 10)))#overlap scope |
|  |  | if (selection == this.selected): |
|  |  | pygame.draw.rect(panel, vgui\_state\_1, pygame.Rect(this.pos[0],this.pos[1] + 10 + (selection \* 10),50,10))#fore scope |
|  |  | txt = ui\_font\_scale\_3.render(this.selections[selection], True, vgui\_fore,) |
|  |  | panel.blit(txt,(this.pos[0] - (txt.get\_width() / 2) + 25,this.pos[1] + 10 + (selection \* 10)))#overlap scope |
|  |  | if (c\_vec[0] > this.pos[0] and c\_vec[0] < this.pos[0] + 50 and c\_vec[1] > this.pos[1] + 10 + (selection \* 10) and c\_vec[1] < this.pos[1] + 20 + (selection \* 10)): |
|  |  | this.hovered = True |
|  |  | if (c\_bool and c\_bool != this.c\_state): |
|  |  | this.selected = selection |
|  |  | this.state = not this.state |
|  |  | this.c\_state = c\_bool |
|  |  | class selection\_interface\_m: #better but still meh |
|  |  | def \_\_init\_\_(this,position\_vec,selections\_array,selected\_array,label\_str): |
|  |  | this.pos = position\_vec |
|  |  | this.selections = selections\_array |
|  |  | this.selected = selected\_array |
|  |  | this.state = False |
|  |  | this.c\_state = False |
|  |  | this.label = label\_str |
|  |  | def draw(this): |
|  |  | c\_vec = pygame.mouse.get\_pos() |
|  |  | c\_bool = pygame.mouse.get\_pressed()[0] |
|  |  | pygame.draw.rect(panel, vgui\_bounding, pygame.Rect(this.pos[0] - 1,this.pos[1] - 1,52,12))#bounding scope |
|  |  | pygame.draw.rect(panel, vgui\_fore, pygame.Rect(this.pos[0],this.pos[1],50,10))#fore scope |
|  |  | txt = ui\_font\_scale\_3.render(this.label, True, vgui\_aux\_text\_internal) |
|  |  | panel.blit(txt,(this.pos[0] - (txt.get\_width() / 2) + 25,this.pos[1]))#overlap scope |
|  |  | if (this.state): |
|  |  | for selection in range(len(this.selections)): |
|  |  | pygame.draw.rect(panel, vgui\_bounding, pygame.Rect(this.pos[0] - 1,this.pos[1] - 1 + 10 + (selection \* 10),52,12))#bounding scope |
|  |  | pygame.draw.rect(panel, vgui\_fore, pygame.Rect(this.pos[0],this.pos[1] + 10 +(selection \* 10),50,10))#fore scope |
|  |  | txt = ui\_font\_scale\_3.render(this.selections[selection], True, vgui\_aux\_text\_internal) |
|  |  | panel.blit(txt,(this.pos[0] - (txt.get\_width() / 2) + 25,this.pos[1] + 10 +(selection \* 10)))#overlap scope |
|  |  | if (this.selected[selection]): |
|  |  | pygame.draw.rect(panel, vgui\_state\_4, pygame.Rect(this.pos[0],this.pos[1] + 10 + (selection \* 10),50,10))#fore scope |
|  |  | txt = ui\_font\_scale\_3.render(this.selections[selection], True, vgui\_aux\_text\_internal) |
|  |  | panel.blit(txt,(this.pos[0] - (txt.get\_width() / 2) + 25,this.pos[1] + 10 + (selection \* 10)))#overlap scope |
|  |  | if (c\_vec[0] > this.pos[0] and c\_vec[0] < this.pos[0] + 50 and c\_vec[1] > this.pos[1] + 10 + (selection \* 10) and c\_vec[1] < this.pos[1] + 20 + (selection \* 10) and c\_bool and c\_bool != this.c\_state): |
|  |  | this.selected[selection] = not this.selected[selection] |
|  |  | if ((c\_vec[0] > this.pos[0] and c\_vec[0] < this.pos[0] + 50 and c\_vec[1] > this.pos[1] and c\_vec[1] < this.pos[1] + 10 and c\_bool and c\_bool != this.c\_state)): |
|  |  | this.state = not this.state |
|  |  | this.c\_state = c\_bool |
|  |  | class warning: |
|  |  | def \_\_init\_\_(this,pos,label,label2 = "",label3 = ""): |
|  |  | this.label = label |
|  |  | this.label2 = label2 |
|  |  | this.label3 = label3 |
|  |  | this.pos = pos |
|  |  | this.button = button([this.pos[0] + 50 ,this.pos[1] + 40],"ok") |
|  |  | def draw(this): |
|  |  | pygame.draw.rect(panel, vgui\_bounding, pygame.Rect(this.pos[0] - 1,this.pos[1] - 1,102,52))#bounding scope |
|  |  | pygame.draw.rect(panel, vgui\_warning\_1, pygame.Rect(this.pos[0],this.pos[1],100,50))#fore scope |
|  |  | panel.blit(ui\_font\_scale\_3.render(this.label, True, vgui\_aux\_text\_internal, vgui\_warning\_1),(this.pos[0],this.pos[1] + 5))#overlap scope |
|  |  | panel.blit(ui\_font\_scale\_3.render(this.label2, True, vgui\_aux\_text\_internal, vgui\_warning\_1),(this.pos[0],this.pos[1] + 15))#overlap scope |
|  |  | panel.blit(ui\_font\_scale\_3.render(this.label3, True, vgui\_aux\_text\_internal, vgui\_warning\_1),(this.pos[0],this.pos[1] + 25))#overlap scope |
|  |  | if (this.button.draw()): |
|  |  | return True |
|  |  | return False |
|  |  | class verticle\_slider: |
|  |  | def \_\_init\_\_(this,pos,mini,maxi,val,length): |
|  |  | this.pos = pos |
|  |  | this.min = mini |
|  |  | this.max = maxi |
|  |  | this.val = length - abs((val / (this.max-this.min)) \* length) |
|  |  | this.len = length |
|  |  | def draw(this): |
|  |  | pygame.draw.rect(panel, vgui\_fore, pygame.Rect(this.pos[0],this.pos[1],13,this.len)) |
|  |  | pygame.draw.rect(panel, vgui\_aux\_text\_internal, pygame.Rect(this.pos[0],this.val-20,13,20)) |
|  |  | #pygame.draw.line(panel,foreground,(this.pos[0],this.val - 10),(this.pos[0] + 13,this.val - 10)) |
|  |  | c\_vec = pygame.mouse.get\_pos() |
|  |  | c\_bool = pygame.mouse.get\_pressed()[0] |
|  |  | if ((c\_vec[0] > this.pos[0] and c\_vec[0] < this.pos[0] + 13 and c\_vec[1] > this.pos[1] and c\_vec[1] < this.pos[1] + this.len)): |
|  |  | pygame.draw.rect(panel, (144,144,144), pygame.Rect(this.pos[0],this.val - 20,13,20)) |
|  |  | if c\_bool: |
|  |  | this.val = c\_vec[1] + this.pos[1] |
|  |  | return this.max - math.floor(((this.val / this.len) \* (this.max - this.min)) + this.min) |
|  |  | class group\_box: |
|  |  | def \_\_init\_\_(this,pos,label,base,height): |
|  |  | this.pos = pos |
|  |  | this.label = ui\_font\_scale\_3.render(label, True, vgui\_aux\_text\_external,foreground) |
|  |  | #this.label.set\_alpha(127) |
|  |  | this.b = base |
|  |  | this.h = height |
|  |  | def draw(this): |
|  |  | #pygame.draw.rect(panel, vgui\_group\_fore, pygame.Rect(this.pos[0],this.pos[1],this.b,this.h))#fore scope |
|  |  | pygame.draw.line(panel,vgui\_bounding,this.pos,(this.pos[0] + this.b,this.pos[1])) |
|  |  | pygame.draw.line(panel,vgui\_bounding,this.pos,(this.pos[0],this.pos[1] + this.h)) |
|  |  | pygame.draw.line(panel,vgui\_bounding,(this.pos[0] + this.b,this.pos[1]),(this.pos[0] + this.b,this.pos[1] + this.h)) |
|  |  | pygame.draw.line(panel,vgui\_bounding,(this.pos[0] + this.b,this.pos[1] + this.h),(this.pos[0],this.pos[1] + this.h)) |
|  |  | panel.blit(this.label,(this.pos[0] + (this.b // 5),this.pos[1] - 5)) |
|  |  |  |
|  |  | def handle\_metrics(): |
|  |  | global NPP\_SAMPLES |
|  |  | #handle all graphs |
|  |  | pygame.draw.rect(panel, vgui\_fore, pygame.Rect(1350,10,300,300))#fore scope |
|  |  | panel.blit(ui\_font\_scale\_3.render("NPP TREND - RED", True, vgui\_aux\_text\_external),(1350,310)) |
|  |  | #pygame.draw.rect(panel, vgui\_bounding, pygame.Rect(1350,350,300,300))#fore scope |
|  |  | panel.blit(ui\_font\_scale\_3.render("TSC TREND - BLUE", True, vgui\_aux\_text\_external),(1560,310)) |
|  |  | try: |
|  |  | NPP\_HEIGHT = max(NPP\_SAMPLES) # find the heights of all data to find, if there is no data their is an error when calling max(), so there is a catch statement |
|  |  | TSC\_HEIGHT = max(TSC\_SAMPLES) |
|  |  | #sight\_height = max(sight\_average) |
|  |  | except: |
|  |  | return |
|  |  | if len(NPP\_SAMPLES) > 300: # recess the graph scroll when it goes off screen |
|  |  | for data\_entry in NPP\_SAMPLES: |
|  |  | if NPP\_SAMPLES.index(data\_entry) % 2 == 0: |
|  |  | del NPP\_SAMPLES[NPP\_SAMPLES.index(data\_entry)] |
|  |  | if len(TSC\_SAMPLES) > 300: |
|  |  | for data\_entry in TSC\_SAMPLES: |
|  |  | if TSC\_SAMPLES.index(data\_entry) % 2 == 0: |
|  |  | del TSC\_SAMPLES[TSC\_SAMPLES.index(data\_entry)] |
|  |  | for data\_point in NPP\_SAMPLES: |
|  |  | pygame.draw.rect(panel, (255,0,0), pygame.Rect(1350 + NPP\_SAMPLES.index(data\_point),310 - math.ceil((data\_point / NPP\_HEIGHT) \* 300),1,1))#bounding scope |
|  |  | #adds a dot for each point |
|  |  | try: |
|  |  | pygame.draw.line(panel,(255,0,0),(1350 + NPP\_SAMPLES.index(data\_point),310 - math.ceil((data\_point / NPP\_HEIGHT) \* 300)),(1350 + NPP\_SAMPLES.index(data\_point) + 1,310 - math.ceil((NPP\_SAMPLES[NPP\_SAMPLES.index(data\_point) + 1] / NPP\_HEIGHT) \* 300))) |
|  |  | #draw a line from data point to data point |
|  |  | except: |
|  |  | continue |
|  |  | for data\_point in TSC\_SAMPLES: |
|  |  | pygame.draw.rect(panel, (0,0,255), pygame.Rect(1350 + TSC\_SAMPLES.index(data\_point),310 - math.ceil((data\_point / TSC\_HEIGHT) \* 300),1,1))#bounding scope |
|  |  | try: |
|  |  | pygame.draw.line(panel,(0,0,255),(1350 + TSC\_SAMPLES.index(data\_point),310 - math.ceil((data\_point / TSC\_HEIGHT) \* 300)),(1350 + TSC\_SAMPLES.index(data\_point) + 1,310 - math.ceil((TSC\_SAMPLES[TSC\_SAMPLES.index(data\_point) + 1] / TSC\_HEIGHT) \* 300))) |
|  |  | except: |
|  |  | continue |
|  |  | #for data\_point in sight\_average: |
|  |  | # pygame.draw.rect(panel, (255,0,0), pygame.Rect(1350 + sight\_average.index(data\_point),650 - math.ceil((data\_point / sight\_height) \* 300),2,2))#bounding scope |
|  |  | #try: |
|  |  | # pygame.draw.line(panel,(255,0,0),(1350 + sight\_average.index(data\_point),650 - math.ceil((data\_point / sight\_height) \* 300)),(1350 + sight\_average.index(data\_point) + 1,650 - math.ceil((sight\_average[sight\_average.index(data\_point) + 1] / sight\_height) \* 300))) |
|  |  | #except: |
|  |  | #continue |
|  |  | vgui\_button\_back = button((1,589),"<- back") |
|  |  | vgui\_button\_exit = button((749,1)," quit ", (255,0,0),(180,0,0),(20,20,20)) |
|  |  | vgui\_button\_start = button((375,300),"start") |
|  |  | vgui\_button\_options = button((375,315),"options") |
|  |  | vgui\_button\_theme = button((375,330),"theme") |
|  |  | vgui\_button\_entity\_list\_manager = button((375,345),"edit ents") |
|  |  |  |
|  |  | vgui\_slider\_food = slider((100,75),"food amount",0,5000,Config["layer-1"]["food"]) |
|  |  | xspeed = slider((20,75),"change logo speed x ",0,10,x\_speed,True,True) |
|  |  | yspeed = slider((20,100),"change logo speed y ",0,10,y\_speed,True,True) |
|  |  | vgui\_slider\_herb = slider((100,100),"herbivore amount",0,100,Config["layer-1"]["herbivores"]) |
|  |  | vgui\_slider\_carn = slider((100,125),"carnivore amount",0,20,Config["layer-1"]["carnivores"],True) |
|  |  |  |
|  |  | #vgui\_slider\_muta = slider((100,150),"mutation chance per 10 seconds",0,100,40) |
|  |  |  |
|  |  | vgui\_slider\_ray\_lazy = slider((500,75),"lazy tracing",0,400,20) |
|  |  | vgui\_slider\_ray\_mult = slider((500,100),"speed multiplier",1,10,1) |
|  |  | vgui\_slider\_ray\_add = slider((500,125),"drunk ray",1,10,5) |
|  |  | vgui\_slider\_photosynth = slider((660,50),"GPP",1,1000,500) |
|  |  | vgui\_slider\_birth\_muta\_chance = slider((660,250),"mutation chance birth %",0,100,50,True) |
|  |  | vgui\_slider\_random\_muta\_chance = slider((660,200),"mutation chance random %",0,100,10,True) |
|  |  | vgui\_slider\_sim\_slow\_val = slider((36,100),"slow amount",1,100,5,True) |
|  |  |  |
|  |  | vgui\_checkbox\_sim\_slow\_bool = check\_box((1,100),"slow",False) |
|  |  | vgui\_checkbox\_sim\_lag\_comp = check\_box((1,150),"lag comp",True) |
|  |  |  |
|  |  | vgui\_checkbox\_ray\_visualise = check\_box((500,160),"visualise",False) |
|  |  | vgui\_checkbox\_visualise\_math = check\_box((100,180),"visualise math",False) |
|  |  | vgui\_checkbox\_ray\_master = check\_box((100,195),"ray tracing",False) |
|  |  | #vgui\_checkbox\_ray\_ehhh = check\_box((100,210),"ehhh",False) |
|  |  | vgui\_color\_ray\_visualise\_1 = color\_selector((570,160),vgui\_ray\_beam,"beam") |
|  |  | vgui\_color\_ray\_visualise\_2 = color\_selector((570,212),vgui\_ray\_broken,"broken") |
|  |  |  |
|  |  | vgui\_color\_state\_0 = color\_selector((50,50),vgui\_state\_0,"vgui\_state\_0") |
|  |  | vgui\_color\_state\_1 = color\_selector((50,105),vgui\_state\_1,"vgui\_state\_1") |
|  |  | vgui\_color\_state\_2 = color\_selector((50,160),vgui\_state\_2,"vgui\_state\_2") |
|  |  | vgui\_color\_state\_3 = color\_selector((50,215),vgui\_state\_3,"vgui\_state\_3") |
|  |  | vgui\_color\_state\_4 = color\_selector((50,270),vgui\_state\_4,"vgui\_state\_4") |
|  |  | vgui\_color\_state\_5 = color\_selector((50,325),vgui\_state\_5,"vgui\_state\_5") |
|  |  | vgui\_color\_bounding = color\_selector((50,380),vgui\_bounding,"vgui\_bounding") |
|  |  | foreground\_color = color\_selector((50,435),foreground,"foreground") |
|  |  | vgui\_fore\_color = color\_selector((50,490),vgui\_fore,"vgui\_fore") |
|  |  | vgui\_herbivore\_color = color\_selector((150,50),vgui\_entity\_herbivore,"vgui\_entity\_herbivore") |
|  |  | vgui\_carnivore\_color = color\_selector((150,105),vgui\_entity\_carnivore,"vgui\_entity\_carnivore") |
|  |  | vgui\_text\_internal\_color = color\_selector((150,215),vgui\_aux\_text\_internal,"vgui\_aux\_text\_internal") |
|  |  | vgui\_text\_external\_color = color\_selector((150,270),vgui\_aux\_text\_external,"vgui\_aux\_text\_external") |
|  |  | vgui\_egg\_color = color\_selector((150,325),vgui\_herbivore\_egg,"vgui\_herbivore\_egg") |
|  |  | vgui\_nose\_color = color\_selector((150,160),vgui\_entity\_nose,"vgui\_entity\_nose") |
|  |  | vgui\_color\_dead = color\_selector((150,380),vgui\_entity\_dead,"vgui\_entity\_dead") |
|  |  | vgui\_slc\_ray\_ignore = selection\_interface\_m((500,212),["food","creatures"],[False,False],"ignore") |
|  |  | vgui\_color\_ON\_state = color\_selector((150,500),vgui\_color\_ON,"vgui\_color\_ON") |
|  |  | vgui\_color\_OFF\_state = color\_selector((150,560),vgui\_color\_OFF,"vgui\_color\_OFF") |
|  |  | vgui\_color\_blindness = selection\_interface\_s((250,50),["normal","deutera","protano","tritano"],0) |
|  |  |  |
|  |  | vgui\_warning\_conf = warning([300,250],"first time setup", "because didnt find","config") |
|  |  | environment = group\_box([60,60],"environment",180,200) |
|  |  | raytracing = group\_box([460,60],"ray tracing",180,200) |
|  |  |  |
|  |  | log\_box = group\_box([799,25],"",400,550) |
|  |  | mut\_reasons = ["radiation","protein misfold","mitosis error"] |
|  |  | def sim\_thread(): |
|  |  | global balance |
|  |  | global NPP\_SAMPLES |
|  |  | global NPP |
|  |  | global TSC |
|  |  | global sight\_average |
|  |  | NPP = 0 |
|  |  | TSC = 0 |
|  |  | for food\_obj in food\_object\_array: |
|  |  | food\_obj.draw() |
|  |  | NPP += food\_obj.carbs + food\_obj.protein |
|  |  | for egg\_obj in egg\_object\_array: |
|  |  | egg\_obj.draw() |
|  |  | egg\_obj.tick() |
|  |  | for herbivore in entity\_object\_array: |
|  |  | herbivore.draw() |
|  |  | if (herbivore.dead): |
|  |  | herbivore.decay() |
|  |  | continue |
|  |  | if ((tick % 2000 \* lag\_comp) == 0) and random.randint(0,100) < vgui\_slider\_random\_muta\_chance.val: # random mutation |
|  |  | index\_to\_patch = herbivore.genes.index(random.choice(herbivore.genes)) |
|  |  | mutated = create\_mutation(herbivore.genes[index\_to\_patch]) |
|  |  | try: |
|  |  | mutated[1] |
|  |  | except: |
|  |  | mutated = [mutated] |
|  |  | mutated.append("UNEDITABLE") # stop empty dna |
|  |  | old\_strand = herbivore.genes[index\_to\_patch] |
|  |  | log\_index.append(log\_entry("a mutation has occured due to " + random.choice(mut\_reasons) + " type: " + mutated[1]," details",str(str(old\_strand) + " -> " + str(mutated[0])) + " -> " + str(read\_dna\_binary(old\_strand)) + " -> " + str(read\_dna\_binary(mutated[0])))) |
|  |  | herbivore.refold() |
|  |  | herbivore.genes[index\_to\_patch] = mutated[0] |
|  |  | TSC += herbivore.stomach |
|  |  | if (herbivore.wait\_for > 0): |
|  |  | herbivore.wait\_for -= 1 / lag\_comp |
|  |  | continue |
|  |  | if (herbivore.stomach <= 0): |
|  |  | herbivore.dead = True |
|  |  | log\_index.append(log\_entry("herbivore starved"," genes",str(herbivore.introgenic\_dna))) |
|  |  | continue |
|  |  | if herbivore.epinephrine > 0: |
|  |  | herbivore.epinephrine -= 0.1 / lag\_comp |
|  |  | herbivore.cns\_depressant += 0.01 |
|  |  | elif (herbivore.cns\_depressant > 0): |
|  |  | herbivore.cns\_depressant -= 0.03 |
|  |  | if (herbivore.egg\_progress > (900 / balance)): |
|  |  | for i in range(herbivore.litter\_size): |
|  |  | egg\_object\_array.append(egg([herbivore.pos[0] + random.randint(1,3),herbivore.pos[1] + random.randint(1,3)],herbivore.genes,herbivore.introgenic\_dna)) |
|  |  | herbivore.stomach -= herbivore.stomach\_max \* 0.2 |
|  |  | herbivore.egg\_progress = -200 / balance |
|  |  | elif herbivore.stomach > (herbivore.stomach\_max \* 0.9): |
|  |  | herbivore.egg\_progress += (1 / lag\_comp) \* balance |
|  |  | if (herbivore.target != None): |
|  |  | herbivore.create\_move() |
|  |  | else: |
|  |  | if (herbivore.nose[0] > herbivore.pos[0]): |
|  |  | radian = 0 |
|  |  | else: |
|  |  | radian = 1 |
|  |  | for food in food\_object\_array: |
|  |  | if (radian == 0 and food.pos[0] < herbivore.pos[0]): |
|  |  | continue |
|  |  | if (radian == 1 and food.pos[0] > herbivore.pos[0]): |
|  |  | continue |
|  |  | if abs(food.pos[0] - herbivore.pos[0]) > herbivore.sight: |
|  |  | continue |
|  |  | if abs(food.pos[1] - herbivore.pos[1]) > (herbivore.sight // 2): |
|  |  | continue |
|  |  | if (not herbivore.sight\_check(food.pos)): |
|  |  | continue |
|  |  | herbivore.target = food |
|  |  | break |
|  |  | herbivore.wander() |
|  |  | herbivore.stomach -= (herbivore.bmr / (100 \* lag\_comp)) |
|  |  | herbivore.stomach = clamp(herbivore.stomach,herbivore.stomach\_max,0) |
|  |  | herbivore.cns\_depressant = clamp(herbivore.cns\_depressant,1,0) |
|  |  | for carnivore in hunter\_object\_array: |
|  |  | carnivore.draw() |
|  |  | if (carnivore.dead): |
|  |  | carnivore.decay() |
|  |  | continue |
|  |  | if (carnivore.wait\_for > 0): |
|  |  | carnivore.wait\_for -= 1 / lag\_comp |
|  |  | continue |
|  |  | #if (carnivore.stomach <= 0): |
|  |  | #carnivore.kill() |
|  |  | #continue |
|  |  | if (carnivore.target != None): |
|  |  | carnivore.create\_move() |
|  |  | else: |
|  |  | if (carnivore.nose[0] > carnivore.pos[0]): |
|  |  | radian = 0 |
|  |  | else: |
|  |  | radian = 1 |
|  |  | for prey in entity\_object\_array: |
|  |  | if (prey.dead): |
|  |  | continue |
|  |  | if (radian == 0 and prey.pos[0] < carnivore.pos[0]): |
|  |  | continue |
|  |  | if (radian == 1 and prey.pos[0] > carnivore.pos[0]): |
|  |  | continue |
|  |  | if abs(prey.pos[0] - carnivore.pos[0]) > carnivore.sight: |
|  |  | continue |
|  |  | if abs(prey.pos[1] - carnivore.pos[1]) > (carnivore.sight // 2): |
|  |  | continue |
|  |  | if (not carnivore.sight\_check(prey.pos)): |
|  |  | continue |
|  |  | carnivore.target = prey |
|  |  | break |
|  |  | carnivore.wander() |
|  |  | carnivore.stomach -= (carnivore.bmr / (100 \* lag\_comp)) |
|  |  | carnivore.stomach = clamp(carnivore.stomach,carnivore.stomach\_max,0) |
|  |  | if tick % 15 == 0: |
|  |  | NPP\_SAMPLES.append(NPP) |
|  |  | TSC\_SAMPLES.append(TSC) |
|  |  | total = 0 |
|  |  | for gene\_holder in entity\_object\_array: |
|  |  | total+=gene\_holder.sight |
|  |  | an\_average = total / len(entity\_object\_array) |
|  |  | sight\_average.append(an\_average) |
|  |  |  |
|  |  |  |
|  |  | def vgui\_thread(): |
|  |  | global NPP |
|  |  | global balance |
|  |  | if (vgui\_checkbox\_sim\_slow\_bool.draw()): |
|  |  | time.sleep(0.1 \* (vgui\_slider\_sim\_slow\_val.draw() / 100)) |
|  |  | vgui\_checkbox\_sim\_lag\_comp.draw() |
|  |  | vgui\_slider\_photosynth.draw() |
|  |  | vgui\_slider\_birth\_muta\_chance.draw() |
|  |  | vgui\_slider\_random\_muta\_chance.draw() |
|  |  | txt = ui\_font\_scale\_3.render("warning! really crazy stuff could happen at high mutation rates ", True, vgui\_aux\_text\_internal) |
|  |  | panel.blit(txt,(400 - ((txt.get\_width() + txt.get\_width()) / 2),600)) #display main text |
|  |  | draw\_visual\_bar(balance \* 100,1,500,(660,150),"balance",(255,255,0)) |
|  |  |  |
|  |  | def simulation(): |
|  |  | global tab |
|  |  | global tick |
|  |  | if (random.randint(1,math.ceil(1000 \* lag\_comp)) < vgui\_slider\_photosynth.val): |
|  |  | food\_object\_array.append(food()) |
|  |  | pygame.draw.rect(panel, (90,90,90), pygame.Rect(150,50,500,500))#sim area |
|  |  | if (vgui\_button\_back.draw()): |
|  |  | tab = 0 |
|  |  | sim\_thread() |
|  |  | vgui\_thread() |
|  |  | tick+=1 |
|  |  | def options(): |
|  |  | global tab |
|  |  | global vgui\_ray\_beam |
|  |  | global vgui\_ray\_broken |
|  |  | if (vgui\_button\_back.draw()): |
|  |  | tab = 0 |
|  |  | environment.draw() |
|  |  | raytracing.draw() |
|  |  | Config["layer-1"]["food"] = vgui\_slider\_food.draw() |
|  |  | Config["layer-1"]["herbivores"] = vgui\_slider\_herb.draw() |
|  |  | Config["layer-1"]["carnivores"] = vgui\_slider\_carn.draw() |
|  |  | #vgui\_slider\_muta.draw() |
|  |  | #vgui\_checkbox\_visualise\_math.draw() |
|  |  | vgui\_checkbox\_ray\_master.draw() |
|  |  | vgui\_slider\_ray\_lazy.draw() |
|  |  | vgui\_slider\_ray\_mult.draw() |
|  |  | vgui\_slider\_ray\_add.draw() |
|  |  | vgui\_checkbox\_ray\_visualise.draw() |
|  |  | vgui\_color\_ray\_visualise\_1.draw() |
|  |  | vgui\_color\_ray\_visualise\_2.draw() |
|  |  | vgui\_slc\_ray\_ignore.draw() |
|  |  | vgui\_ray\_beam = vgui\_color\_ray\_visualise\_1.col |
|  |  | vgui\_ray\_broken = vgui\_color\_ray\_visualise\_2.col |
|  |  | def theme(): |
|  |  | global tab |
|  |  | global vgui\_state\_0 |
|  |  | global vgui\_state\_1 |
|  |  | global vgui\_state\_2 |
|  |  | global vgui\_state\_3 |
|  |  | global vgui\_state\_4 |
|  |  | global vgui\_state\_5 |
|  |  | global vgui\_fore |
|  |  | global foreground |
|  |  | global vgui\_bounding |
|  |  | global vgui\_entity\_herbivore |
|  |  | global vgui\_entity\_carnivore |
|  |  | global vgui\_aux\_text\_internal |
|  |  | global vgui\_aux\_text\_external |
|  |  | global vgui\_entity\_nose |
|  |  | global vgui\_herbivore\_egg |
|  |  | global vgui\_entity\_dead |
|  |  | if (vgui\_button\_back.draw()): |
|  |  | tab = 0 |
|  |  | vgui\_color\_state\_0.draw() |
|  |  | vgui\_color\_state\_1.draw() |
|  |  | vgui\_color\_state\_2.draw() |
|  |  | vgui\_color\_state\_3.draw() |
|  |  | vgui\_color\_state\_4.draw() |
|  |  | vgui\_color\_state\_5.draw() |
|  |  | vgui\_color\_dead.draw() |
|  |  | vgui\_color\_bounding.draw() |
|  |  | foreground\_color.draw() |
|  |  | vgui\_fore\_color.draw() |
|  |  | vgui\_herbivore\_color.draw() |
|  |  | vgui\_carnivore\_color.draw() |
|  |  | vgui\_nose\_color.draw() |
|  |  | vgui\_text\_internal\_color.draw() |
|  |  | vgui\_text\_external\_color.draw() |
|  |  | vgui\_egg\_color.draw() |
|  |  | vgui\_color\_ON\_state.draw() |
|  |  | vgui\_color\_OFF\_state.draw() |
|  |  | vgui\_state\_0 = vgui\_color\_state\_0.col |
|  |  | vgui\_state\_1 = vgui\_color\_state\_1.col |
|  |  | vgui\_state\_2 = vgui\_color\_state\_2.col |
|  |  | vgui\_state\_3 = vgui\_color\_state\_3.col |
|  |  | vgui\_state\_4 = vgui\_color\_state\_4.col |
|  |  | vgui\_state\_5 = vgui\_color\_state\_5.col |
|  |  | vgui\_herbivore\_egg = vgui\_egg\_color.col |
|  |  | vgui\_bounding = vgui\_color\_bounding.col |
|  |  | foreground = foreground\_color.col |
|  |  | vgui\_fore = vgui\_fore\_color.col |
|  |  | vgui\_color\_blindness.draw() |
|  |  | vgui\_entity\_herbivore = vgui\_herbivore\_color.col |
|  |  | vgui\_entity\_carnivore = vgui\_carnivore\_color.col |
|  |  | vgui\_entity\_nose = vgui\_nose\_color.col |
|  |  | vgui\_aux\_text\_internal = vgui\_text\_internal\_color.col |
|  |  | vgui\_aux\_text\_external = vgui\_text\_external\_color.col |
|  |  | vgui\_entity\_dead = vgui\_color\_dead.col |
|  |  | def main\_menu(): |
|  |  | global tab |
|  |  | global x |
|  |  | global y |
|  |  | global img\_size |
|  |  | global x\_speed |
|  |  | global y\_speed |
|  |  | if vgui\_button\_theme.draw(): |
|  |  | tab = 3 |
|  |  | if vgui\_button\_start.draw(): |
|  |  | tab = 2 |
|  |  | if vgui\_button\_options.draw(): |
|  |  | tab = 1 |
|  |  | if (x + img\_size[0] >= 1700) or (x <= 0): |
|  |  | x\_speed = -x\_speed |
|  |  | if (y + img\_size[1] >= 800) or (y <= 0): |
|  |  | y\_speed = -y\_speed |
|  |  | x += x\_speed / lag\_comp |
|  |  | y += y\_speed / lag\_comp |
|  |  | if x\_speed < 0: |
|  |  | x\_speed = -xspeed.draw() |
|  |  | else: |
|  |  | x\_speed = xspeed.draw() |
|  |  | if y\_speed < 0: |
|  |  | y\_speed = -yspeed.draw() |
|  |  | else: |
|  |  | y\_speed = yspeed.draw() |
|  |  | panel.blit(dvd,(x,y)) |
|  |  | def log\_manager(): |
|  |  | global log\_var |
|  |  | pygame.draw.rect(panel,vgui\_fore,pygame.Rect(810,25,380,750)) |
|  |  | txt = font\_alt.render("simulation log (hover over yellow text for extra information)", True, (255,255,255)) |
|  |  | panel.blit(txt,(1000 - (txt.get\_width() / 2),10)) |
|  |  | if (len(log\_index) > 51): |
|  |  | vgui\_slider\_scroll = verticle\_slider([1200,25],0,len(log\_index) - 51,log\_var,750) |
|  |  | #vgui\_slider\_temporary = slider((1200,125),"temporary log scroller",0,len(log\_index) - 51,log\_var,True) |
|  |  | log\_var = abs(vgui\_slider\_scroll.draw()) |
|  |  | for pointer in range(len(log\_index)): |
|  |  | log\_index[len(log\_index) - pointer - 1].handle\_hover() #read backwards to fix overdrawing issue |
|  |  | def main(): |
|  |  | global lag\_comp |
|  |  | lag\_comp = clamp(lag\_comp,3,0.01) |
|  |  | sample\_time = time.time() |
|  |  | for event in pygame.event.get(): |
|  |  | if event.type == pygame.QUIT: |
|  |  | json.dump(Config,open("config.json","w")) |
|  |  | pygame.quit() |
|  |  | sys. exit() |
|  |  | panel.fill(foreground) |
|  |  | handle\_metrics() |
|  |  | log\_manager() |
|  |  | if (tab == 0): |
|  |  | main\_menu() |
|  |  | elif (tab == 1): |
|  |  | options() |
|  |  | elif (tab == 2): |
|  |  | simulation() |
|  |  | elif (tab == 3): |
|  |  | theme() |
|  |  | if (vgui\_button\_exit.draw()): |
|  |  | json.dump(Config,open("config.json","w")) |
|  |  | pygame.quit() |
|  |  | sys. exit() |
|  |  | #master.draw() |
|  |  | #log.draw() |
|  |  | #og\_box.draw() |
|  |  | pygame.display.flip() |
|  |  | if ((time.time() - sample\_time) != 0): |
|  |  | lag\_comp = ((math.ceil(1 / (time.time() - sample\_time))) / 100) #based off 100 ticks, |
|  |  | #for i in range(100): |
|  |  | #log\_index.append(log\_entry("test" + str(100 - i),"wefweufgweifisfdfds","why are you reading this")) |
|  |  | log\_index.append(log\_entry("a project made by henry frodsham"," read ","please familiarise yourself with the ui elements in options before starting the simulation")) |
|  |  | while True: |
|  |  | summ = 0 |
|  |  | for i in entity\_object\_array: |
|  |  | if i.dead: |
|  |  | continue |
|  |  | summ += 1 |
|  |  | balance = (len(hunter\_object\_array) / (((summ + (len(egg\_object\_array) // 2))) \* 4) + 1) |
|  |  | screen\_size = panel.get\_size() |
|  |  | vgui\_button\_back = button((1,screen\_size[1]-14),"<- back") |
|  |  | master = group\_box((0,0),"",screen\_size[0] - 401,screen\_size[1] - 1) |
|  |  | log = group\_box((0,0),"",1299,599) |
|  |  | while vgui\_warning\_config: |
|  |  | for event in pygame.event.get(): |
|  |  | if event.type == pygame.QUIT: |
|  |  | json.dump(Config,open("config.json","w")) |
|  |  | pygame.quit() |
|  |  | sys. exit() |
|  |  | panel.fill(foreground) |
|  |  | if (vgui\_warning\_conf.draw()): |
|  |  | vgui\_warning\_config = False |
|  |  | pygame.display.flip() |
|  |  | main() |

### Major additions from version 1 -> 2

* **Graph** 
  + **How was this achieved?**
    - **Log\_manager() function, iterates through NPP and GPP data to position the data as linked points along a line, creating a line graph. Also handles when the lines go offscreen by removing unnecesarry data that doesn’t show a link**
  + **Why was this done?**
    - **Allow the user to see the simulation metrics including the previous entries**
* **Working log**
  + **How was this achieved?**
    - **Handle\_hover() function to check for mouse position over log entries, this will then display special text which cannot go offscreen, a way of scrolling previous entries was then added by initialising log\_onscreen of the first 50 elements and then adding an offset dictated by the log scroller to change visible logs**
* **New ui systems with more complex colours**
  + **How was this achieved?**
    - **Theme page and revamped ui elements, the user can now change all colors by using labelled color selection interfaces**
  + **Why was this done?**
    - **User friendliness, easier readability and interpretability of the simulation**
* **Centered log**
  + **How was this achieved?**
    - **Use of the development tool to reposition ui elements perfectly without trial and error**
  + **Why was this done?** 
    - **Ui looks more professional and therefore more user friendly.**
* **Ability to scroll log**
  + **How was this achieved?**
    - **Initialising log\_onscreen variable of first 50 elements, by adding an offset to see previous logs. The offset is set by the slider which appears when there are more than 50 logs**
  + **Why was this done?**
    - **Ability to see previous log entry to further analyse simulation data**

# System 6: saving the simulations

## Saving a session

### Iteration 1

Text

Description automatically generated

**Developments from iteration 1 to 2:**

* Efficiency changes:
  + the path to change is now only calculated once, instead of using os.path.join the paths are manually added together. Writing bytes is now also specified

**why?**

Improves code readability and maintainability, also sometimes wouldn’t write to the file if WB is not specified

### Iteration 2

Text

Description automatically generated

**Major changes from iteration 2->3:**

* **more saved data**
  + **whats changed?**
    - Addition of log and metrics to be saved
  + **Why?**
    - Improve simulation loading to better replicate the saved session
* **Better formatting of generated session names**
  + **Whats changed?**
    - **Date and time formatting is now less confusing**
  + **Why?**
    - **Makes it easier to tell which simulation is being loaded**

### Iteration 3

Text

Description automatically generated

**Major changes from iteration 3->4:**

* **Modularisation:**
  + **Whats changed?**
    - **Instead of repeating the same of block of code over and over, saving a list of objects is now placed inside a method save\_objects**
  + **Why?**
    - **Easier readability and maintainability**
* **Addition of more graph metrics:**
  + Whats changed?
    - The function has been updated to save the new graph metrics
  + **Why?**
    - to more accurately save the currently loaded session.

### Iteration 4

Text

Description automatically generated

### Save\_objects method

Text

Description automatically generated

## Loading a session

### Iteration 1

Text

Description automatically generated

**Major changes from iteration 1 -> 2:**

* **Now checks for valid file input:**
  + **Whats changed?**

Previously, the program would only check for the basefolders existence, since I added graphs to be loaded in a different folder like below,

Graphical user interface, text

Description automatically generated

However, trying to load simulations generated from a different version would cause an error due to them not having this folder so I have now included a check before loading to ensure it will not try and access a folder that does exist.

### Iteration 2

### Text Description automatically generated

**Major changes from iteration 2->3:**

* **Modularisation:**
  + **Whats changed?**
    - **Instead of repeating the same of block of code over and over, saving a list of objects is now placed inside a function load\_objects**
  + **Why?**
    - **Easier readability and maintainability**
* **Addition of more graph metrics:**
  + Whats changed?
    - The function has been updated to load the new graph metrics
  + **Why?**
    - to more accurately replicate the saved session.

### Iteration 3

Text

Description automatically generated

### Load\_objects function

Text

Description automatically generated

# System 7: displaying metrics

## Plot\_graph

Text

Description automatically generated

Used to draw the graph on the screen, iterates through all data points and joins them up

## Trim\_data

Text

Description automatically generated

Prevents samples going off screen by removing every second data point when the list items goes over the parameter max\_items (usually 300 as that’s the width of the graphs)

## Iteration 0

Text

Description automatically generated

Text

Description automatically generated

**Major changes from iteration 0 -> 1:**

* **visual changes**
* **modularisation:**
* **efficiency changes**

## Iteration 1

Text

Description automatically generated

Text

Description automatically generated