Evolution Simulator (EvoSim) – an A level computer science Project.

By Aran Campbell.

Analysis

Introduction

In this coursework I am going to create a better way for children and adults to learn about evolution. Currently, evolution is taught the same way most other things are, through a textbook. I believe this is not the optimal way for the fundamentals of evolution to be taught/learned, and I am therefore going to attempt to solve this problem via a computational approach.

This is for a number of reasons:

* Computers can make the learning interactable, which will boost a person’s interest in the given topic.
* Computers are readily accessible in all classrooms, if not phones, meaning that they can access the resources anywhere they are, perhaps for revision materials too.
* Computers can render images very easily, which would help those with visual memories to remember the fundamentals of evolution.
* Computers allow everyone in the classroom to run their own simulation, which will make the entire experience more memorable.
* Computers allow the user to see the actual process behind evolution and can actually show how all the different factors can have an effect on the outcome.

Stakeholders

The following people will be asked to test and give feedback/insight on my program.

* **Mr Lewis**, Mr Lewis is a biology teacher currently working at Brighton College, and he teaches the pupils about all aspects of biology, including evolution. Mr Lewis currently teaches students via power points and then sets students work about it to consolidate it within their minds. He will be able to make use of my proposed solution (an evolution simulator) by using it in his classes as an engaging aide to the curriculum, therefore helping him achieve his goal of consolidating knowledge within his student’s minds.
* **Mrs Duffield**, Mrs Duffield is also a biology teacher at Brighton College who works alongside Mr Lewis, and teaches A level students biology, which requires a higher level of understanding than GCSE, and therefore students need to be very comfortable with the fundamentals of evolution, and a visual/audible approach is proven to be more engaging than reading from a textbook. Mrs Duffield will be able to use my solution as it will provide a visual approach to teaching evolutions fundamentals, and students of hers will be able to see how the effects of different attributes being changed can impact evolution.
* **Lucinda Slack,** Lucinda is currently a biology student at Brighton college who plans to go on and work as a doctor in future, evolution is one of her weakest topics and it will be very useful to get an actual student’s feedback in terms of ease of use, memorability, and how well it shows the fundamentals of evolution.

Research:

First, a look into:

**A pre-existing solution: ps://openprocessing.org/sketch/205807/**

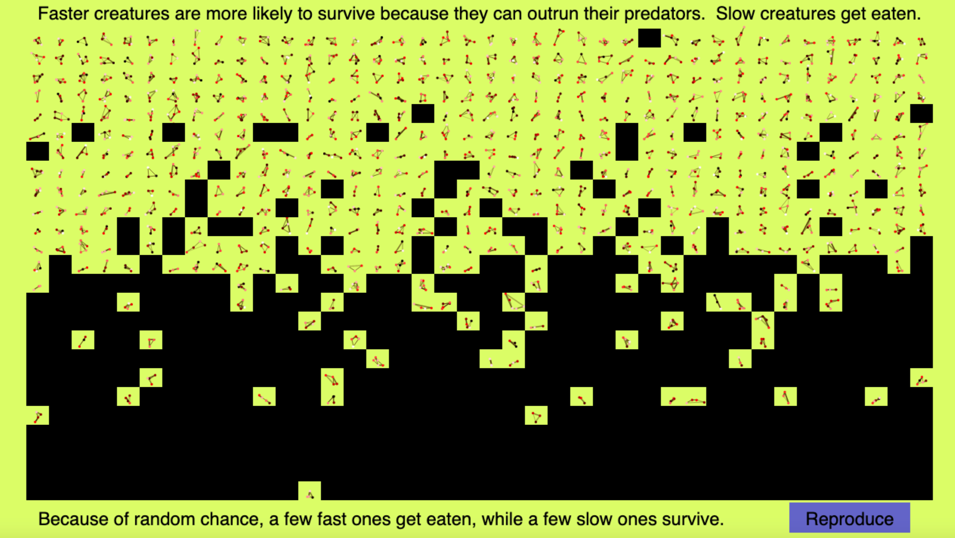
**Evolution Simulator [FIXED] by Cary Huang.**

This is a side-on view 2D evolution simulator with some fairly complex mechanics, it makes good use of randomly generated creatures, however in terms of variation that is the extent of what it does. The end result is always the same creature (after a few ‘generations’) as the ‘survival condition’ is always the same, and there are no variables you can change yourself and there is almost nothing to interact with.

Background pattern

Description automatically generated

A visual representation of the random ‘1000 creatures generated’, before they have been put through the 10 second race.



After the race results are completed, and the sorting has taken place (which took a big toll on my computers core i7 CPU, which is more powerful than an average CPU) the bottom half are generally culled and a few ones in the top 500 are culled due to ‘random chance’.

Graphical user interface, application

Description automatically generated

After running through 93 generations (again, slowed my computer down to a halt), it becomes apparent that there is no longer anything going on, and the ‘best creature’ has been the best creature for 15 generations now, due to a lack of change in the environment.

There is lots of info presented, but it is not presented in a clear or readable way.

Overall, while the process *did* show insight into very basic evolution, it did *not* take into account: environment, reproducing prerequisites, actual fitness level of the creatures (as the ‘worst’ creature could sometimes go further from the starting point than the ‘best creature’, but due to it being in the opposite direction, it was disregarded and culled), there was little to no interactivity as the user did not really have any control over the final outcome, after running through the evolution simulator and getting to generation 90 (approximately) about 5 times, it became apparent that no matter what the starting creatures were, the end result was always the same.

There was no information available on the best creature of each generation, other than how far it got from the finish line. It did not show any genetic stats or physical attributes that it had, which is not useful for teaching evolution due to physical attributes being very important in determining whether or not something would survive. There was also no ‘survive’ conditions other than the initial one, which does not accurately reflect the real world.

The thing I liked about this solution was the idea of including graphs to help the user see how much change has occurred over time, although I would like to make use of them in a simpler manner, with labelled axis. I believe a top down approach would be a lot more suitable for showing the simulation in progress to the user, as it allows you to display every entity on the screen at once, as opposed to the previous solutions ‘1 at a time’ approach, which gets very boring to watch through when you have to go through 1000 members of each generation.

Other research:

I personally sat in on a lower fifth (year 10) Brighton College biology lesson in which evolution was being taught, and after the lesson I asked a few of the students about how they felt about the lesson and how involved they felt. Many of the students that I asked said that they did not feel very involved in the lesson and as a result they paid slightly less attention. I then asked them how they would make changes to the way evolution is taught, and the vast majority mentioned that if there was an interactive experience included with the lesson, they would be a lot more likely to pay attention and to retain the knowledge they picked up in the lesson.

I looked into how students are actually taught about evolution, and the textbook resources, while informative, display information in a boring manner and do not provide many visual aids such as diagrams or images. There are no interactive resources that could be widely used, and Online resources did not prove to be very engaging and were all outdated.

Text

Description automatically generated

As shown in *Figure 1 (right)*, the content accessible to students is very plain and there is no variation in the presentation of information, there is simply a text description that informs them of all the facts they must remember. There is interactable content on the page, and there is a very limited set of images to explain how humans evolved, but these images did not contribute to any of the content that the students are meant to know and understand the principles behind.

*Figure 1. A screenshot taken on BBC bitesize page.*

<https://www.bbc.co.uk/bitesize/guides/zcqbdxs/revision/1>

I conducted an interview with Lucinda (one of my stakeholders) so that I could find out her thoughts on how its taught to her in lessons. A summary of the things I learned in the interview are as follows:

* Most of the content available that isn’t in the textbook is not actually useful information and can not really be applied to exams.
* A good amount of students still are confused about certain evolutionary systems such as survival of the fittest.
* She would like to be able to see ‘evolution in action’ to help her grasp and understand what actually happens throughout the process itself.
* She does not understand the concept of evolving to adapt to the environment that a particular species/race is.

Requirements.

The following table lists the features that need to be included in the solution. They are represented as user stories, and are not a definitive requirement but a possibility that should be investigated further.

|  |  |  |
| --- | --- | --- |
|  | User Story | Success criteria |
| 1 | As a teacher I must be able to use the program to a basic level and be able to link it in with the lesson. | The program must be easy to use, and have a simple tutorial or help button in case the user does not understand how to make the program function correctly. |
| 2 | As a teacher I must be able to ensure that my students can set up and install the experience onto their devices. | The program must be easy to install, and contained in a single file, while also being less than a gigabyte in file size. |
| 3 | As a student I must be able to run the application on any of the possible choices for a laptop, namely: a Microsoft surface or an Apple MacBook. | The program must run at a minimum of 40 FPS at any given point, on any of the devices that meet the minimum system requirements outlined below. |
| 4 | As a student, the program must show or inform me about some of the fundamental principles involved in evolution. | The program must show how each attribute has changed over time to show which ones lead to survival and are good qualities for the given environment. |
| 5 | As a student, I should be able to have a form of visual aid to learning about evolution rather than just using the simple diagrams in the textbook. | The program must have a visual display output that is update in real time, showing the actions and behaviours of all entities as evolution progresses. |
| 6 | As a student, I should be able to change the starting values of the creatures that are going to evolve. | The program may allow for user input to determine what starting values and stats the creatures begin with. |
| 7 | As a teacher, the entities and processes taking place should be clear to see so that I can explain each behaviour/process to students easily. | The program must be visually simple to look at and understand. Most elements should be explicit, needing no further information to be understood. |
| 8 | As a student, I could be able to go back and look at every step of the evolution sequence so that I can fully understand what changes have happened in each stage. | The program may allow the user to go back in generations to see the changes between each generation and how those changes impacted the ‘success’ score. |
| 9 | As a student, I could be able to see the process of survival of the fittest come into action. | The program may make the preferred genes for ‘offspring’ that of the successful parents, where success is a measured value. |
| 10 | As a student, I could be able to change different factors such as perhaps the environment to see what effect that has on the creatures evolution. | The program may allow the user to select or create an environment for the creatures to be placed in, to determine the effects that environment has on the creatures. |

Limitations

The proposed solution will not have the following features:

* Realistic ray-traced lighting effects – this is due to a lot of computers not having enough power to be able to use raytracing. Raytracing is a rendering method that produces very high quality, realistic lighting and shadows. Realistic lighting and shadows is not a requirement for the program and is not mentioned in any user stories, so it will be abstracted from the program and instead basic lighting will be used.
* The teacher will not be able to see (using his/her own computer) everything that the students are doing on their computer, due to the potential creation of unwanted privacy issues for the students. Despite this, one of the common risks of using computers in classrooms is that it becomes very easy for the students to get distracted without the teachers knowledge. The teacher will have to navigate this issue via their own means.
* A lifelike environment for the simulation to be ran in. Running a lifelike environment, which has many variables and functions with complex and intensive calculations happening at every second, would be very demanding on computers, and especially not the ones used in schools. As a result, many environmental factors will be abstracted out of the final solution.

Hardware and software requirements:

The program must be able to run on the school desktop PC’s, which do not have powerful processors and only have the inbuilt (with the CPU) graphics cards, meaning they are not capable of any heavy rendering. It must also be able to run on MacBook’s and Surface Pro 6’s or newer, with the surface pro 6 being the weaker of the two, as these are the laptops of choice for both the students and teachers at College.

Dell OptiPlex 3060 (school desktop PC)

CPU: Intel core i5-8500.

GPU: Intel integrated HD 610

RAM: 8GB.

Storage: 1TB.

Surface Pro 6:

CPU: Intel core i5-8250U

GPU: Intel UHD Graphics 620 (i5)

Ram: 8GB.

Storage: 256GB.

As the solution will most likely be made using either Unity (2D) or PyGame, the min. requirements for both of those applications are listed as follows:

PyGame:

|  |  |
| --- | --- |
| CPU: | Intel i3 or newer |
| GPU: | DX10 support. (2006+) |
| RAM: | 4GB or more |
| OS: | Win 7/macOS 10.11 or newer. |
| STORAGE: | 3GB |

Unity:

|  |  |
| --- | --- |
| CPU: | Intel i3 or newer |
| GPU: | DX10 support. (2006+) |
| RAM: | 4GB or more |
| OS: | Win 7/macOS 10.12 or newer. |
| STORAGE: | 1GB |

Both of the school desktop and the schools recommended laptop of choice for both students and teachers satisfy the above requirements, and are suitable to run both unity and PyGame on. MacBooks bought after 2011 are also appropriate, and are also widely used by the students at Brighton College.

Design

I have used the flowchart design to break the main solution down, into small, computable problems. The reasoning for this is so that I can solve each problem by coding it independently, allowing me to finally be able to combine all the finished parts of code into one, so that I can get each feature/function working to its fullest before starting another one. This will also aid with the construction of different classes, gameObjects (within Unity), scripts, and passing references between scripts.Diagram

Description automatically generated

Justification of decomposition:

I have chosen ‘update environment’ to be a key procedure that is called at the start of every new frame. This is because each frame can be used to signify time passing in the simulation. The update environment procedure will perform checks on the entities, namely to: see how much food has been consumed by each entity, to calculate energy used since the last frame and the distance travelled by each entity, update the timer left on each cycle (each round). All boxes that end with a question mark are problems that can be solved with simple logic statements within the code.

Another artefact I created to aide my coding process is a simple checklist that names and lists the steps I must have occur in the program, it can be seen below.

* Application start.
* Round 0 begins.
* 20 creatures are spawned in as generation 0, along the sides of the square.
* The same 20 creatures are given starter attributes at the specified base values.
* Round starts - creatures randomly move around and eat food in their view cone
* Round ends, and all creatures who didn’t eat any food are de-spawned.
* All creatures that ate 1 food at the end of the round are allowed to continue to the next round
* All creatures that ate 2 food (the maximum amount) at the end of the round continue to the next round with a single offspring.
* Offspring must copy their parent creatures traits/attributes with a random ~0.5 variation in attributes in order to allow for mutation
* After many rounds, optimal species attributes will be made apparent.

Development Environment.

This project will be coded in C#, using the unity development environment. C# is being used as it’s a high-level programming language that I am familiar with, and has a large community online that I can use for support throughout the project. Unity is being used as it has a built-in physics engine that can handle collisions and other physics calculations without me having to code them in myself as the time requirements for that would not fit within the constraints of the time limit given for this project. Unity also has a built in 3D model renderer, which will be very useful for meeting user story 5’s requirements. I will be using GitHub as my version control software, and also as a progress tracker to see the different phases of development over time. This is because there are a lot of irreversible actions that can be made on Unity, which, if I misuse, could contribute to lost files and lost work. The version control that GitHub provides can remedy this. Furthermore, GitHub can seamlessly integrate with Unity, to allow for security and stability throughout the development process.

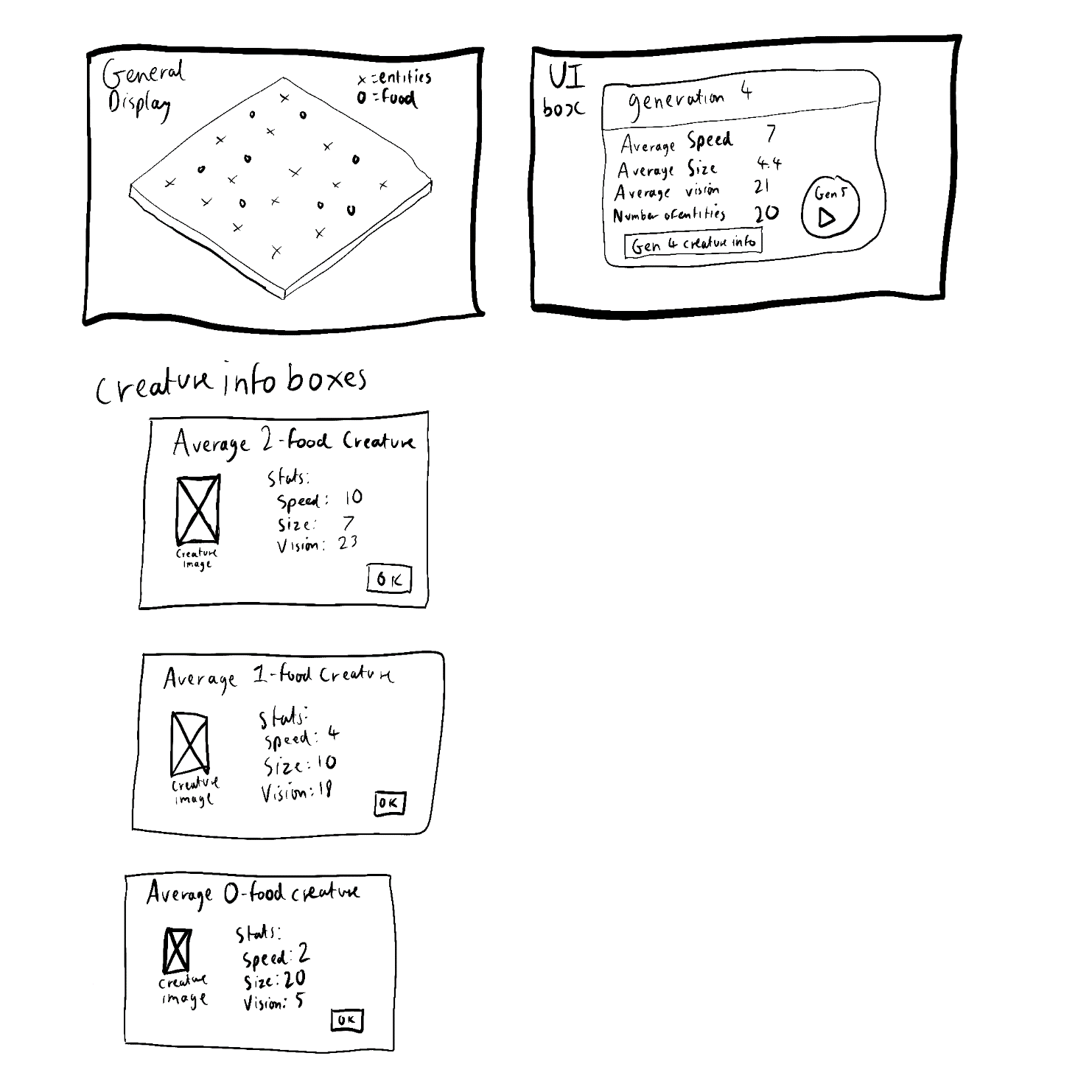
*Links to both Unity and GitHub can be found below:*

*Unity - https://unity.com*

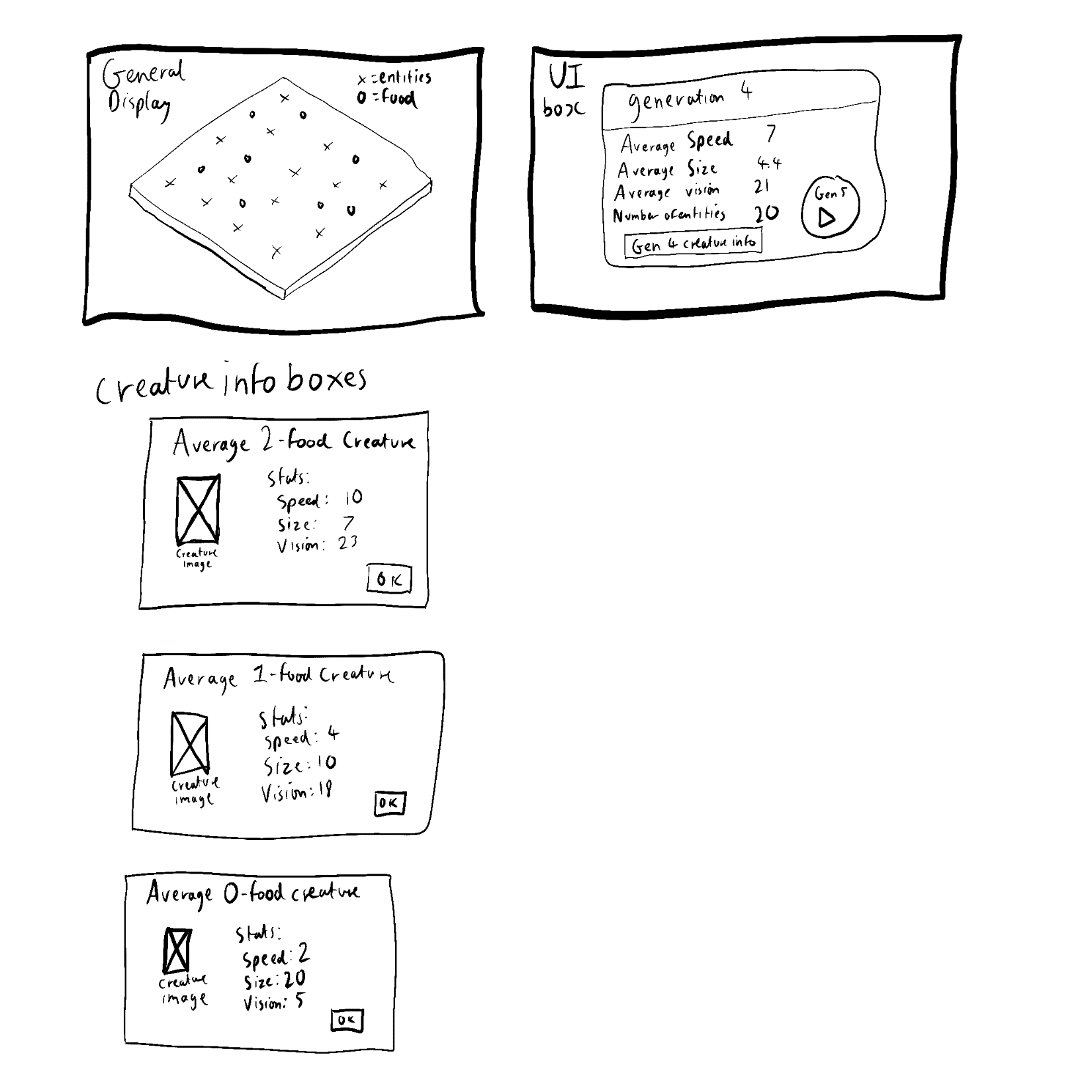
*Unity desktop client download - https://unity3d.com/get-unity/download*

*GitHub - https://github.com*

*GitHub client download - https://desktop.github.com/*

UI Mock-ups

The general display (left) will be what the user sees when a round is in progress, with entities moving around and food appearing in real time. The UI box (right) will be shown at the end of each round, providing round information, and giving the user an option to start the next round/generation, or to provide information on the creatures that performed in the round that has just ended. Should a user press the play button for the generation, the general display will be shown again, this time with the new creatures inside the round.



Should the user press the creature information box, the below UI mock-ups will appear, and when the OK button is pressed on any of them, it will return to the previous UI information box (top right) about the round that has just ended.

It shows the average information for each level of entity performance, with 2 food creatures being successful that will have offspring with genetic variations, 1 food creatures not having any offspring but being placed inside the next round, and 0 food creatures being removed (killed off from an evolutionary perspective).

PHASE 1

I have abstracted the ‘entities’ down to simple green cylinders, that will appear to ‘float’ around the specified environment (in this case, a square plane). The reasoning behind this is to standardise the application for all users so that the application does not become too graphically demanding on slower school computers or student’s Surfaces/MacBook’s. The integrated GPU found within all of these devices cannot handle more than 10 – fairly complex – humanoid models, as shown by the screenshots and graphs displayed below. The target framerate for my application to run at is >40 FPS, as specified by user story 3, on any of the target devices. While a complex model is more lifelike, it doesn’t work in this context.

Link to GitHub commit: <https://github.com/07acampbell/EvoSim/commit/0d31ef124a8e3c3d58784cec8404476bc23bfe45>

Link to Code: <https://github.com/07acampbell/EvoSim/tree/main/EvoSim/Assets/Scriptz>

*A visual comparison between the simple models (left), and the complex models (right).*

*The data shown in the graph was collected on a minimum-spec 2018 Microsoft Surface.*

A picture containing text, stationary, envelope, businesscard

Description automatically generatedA picture containing text, businesscard, envelope

Description automatically generated

Icon

Description automatically generated

The simple model’s results show that it is consistently more than 3x higher performing than the minimum requirement FPS (depicted by the white line on the graph). This means that using this model would meet the requirements mentioned in user story 3, and for even more models than necessary for the application to run, resulting in greater stability across all devices.

The complex model’s results show that it can meet the minimum requirements for FPS but only for up to 14 entities on the screen at any time, which is not suitable for the program as mass species evolution must be shown. Adding walking animations to the character models resulted in a ~65% drop in frames per second, so that does not meet the minimum requirements either at all.

In the first working prototype, the entities are seen to begin at evenly distributed positions on the edges of the plane, and as the ‘round’ starts, food begins to spawn randomly around the plane at a rate of 1 food every 2 seconds. The creatures wander around at random, until they detect food in their vision radius (which has a base value of 20). When they detect the food, they move towards it in a straight line until making contact with it and having their ‘FoodEaten’ stat (starts at 0 for each round) increment by 1. This process continues until the 10 second round is over, and at this point the simulation ends, and all creatures that ate 2 food gather at the top corner of the plane, all creatures that ate 1 food gather at the leftmost corner of the plane, and all creatures that did not eat any food gather at the bottom of the plane.

This first prototype **satisfies user stories 3, 5 and 7.** User story 3 is satisfied as the prototype runs at 140 fps, which is almost 100 more frames per second than required by user story 3 and as a result the program is not intensive to run, and will work on even the slowest of devices. User story 5 is satisfied as a visual display of the creatures within the environment can be seen on screen, in real time, updating around 140 times a second to give a live depiction. User story 7 is satisfied as all elements and models used in the prototype are simple shapes and objects, with a unique colour to differentiate them and to make the type of object that the student is looking at explicit.

Below are some screenshots from the first prototype, before any changes from the user feedback are made.

A picture containing text, pool table, worktable, gambling house

Description automatically generatedA picture containing text, hinge

Description automatically generated

*Simulation before the round has started (left). Round in progress (5s) with food being spawned (right)*

A picture containing text, businesscard, envelope, worktable

Description automatically generated

*End of round, with creatures who have eaten 2 food at the top, creatures who have eaten 1 food at the left corner, and creatures who ate no food at the bottom.*

*This is the end of the first prototyped solution.*

User feedback for phase 1:

I showed the first working prototype to all 3 of my stakeholders, and have summarised their opinions below.

|  |  |  |
| --- | --- | --- |
| **Name of stakeholder** | **Feedback** | **Resulting changes to make** |
| Mr Lewis | I really like the aesthetics and design choices that you have made, they are pleasing to look at without being too simplistic. One thing I’d like to see is the actual evolutionary process occurring, as right now there is only one round so the creatures don’t have a chance to evolve. | * Have multiple rounds * Show evolutionary process occurring within the creatures. |
| Mrs Duffield | To add to Mr Lewis’s comments, it’s got great colours and I like the minimalism, it looks like even my laptop could run the app. I think it could be better if the creatures that ate 2 bits of food would have offspring, and the creatures that didn’t eat any food should disappear. | * Implement food system that changes entities at the start of a new round |
| Lucinda S. | The movement of the creatures looks natural and I think it does a good job at representing the ‘looking for food’ instinct in most animals. One thing I’d change is that the food colour is white, and its quite difficult to differentiate between the food and the environment as they are both light colours. | * Change colour of food to a darker shaded colour so that it is offset from the colour of the environment plane. |

Algorithms, classes, and methods.

Wander method:

Text

Description automatically generated

The above method allows the entity to ‘wander’ around at random, in the chance that a food item enters its ‘vision’ circle. A method used for food detection and then eating any found food items is shown below.

Food detection and consumption method:

Text

Description automatically generated

The above method adds a spherical collider that has a radius equal to the creatures unique ‘vision’ stat. Any ‘food’ items that are detected inside this sphere, as long as they are not already targeted by another entity and the ‘FoodEaten’ stat is below 2, will be marked as ‘targeted’ so other entities do not eat the food before the entity that first detected it has had the chance to eat it.

Text

Description automatically generated

The code on the right shows all variables, public or private, used in the movement and food detection script. There is a separate *agentVariables* script that runs in parallel with the food detection script, which stores all key variables which are subject to ‘evolving’ and changing. The food detection script accesses some variables from the *agentVariables* script to be used in calculations, such as vision range.

Algorithms, classes, and methods (continued).

Text

Description automatically generated

Update method (called every frame):

If no food is found, the entity will wander around randomly (using the wander method), and if food is detected by the hunt Food Method, the entity will stop wandering and will make its way to the food item detected within its vision sphere.