Animal Simulator

January 1

2020

Russell de Roeper

Table of Contents

Analysis

Background

High schools across the country learn about the food chain and the ideas of Darwinism in nature. This may be challenging for some students as this might be the first time students are introduced to these ideas. As a student who struggled learning these concepts from GCSE Geography, I always wondered whether there was a better way of learning. If students were able to learn more effectively, topics which I have mentioned would be easier to learn and therefore allow students to gain a deeper knowledge of the criteria for their exam specifications. In Alec Reed Academy especially, all students have to learn geography from Key stage 3 and possibly key stage 4. This means that its crucial for the school that its students are learning with as many resources to help engage them alongside their normal classes.

In order to make learning more effective within the classroom for students and as revision, I strongly believe that an interactive simulation of animal species would perform well to develop understanding for students. As this program is accessible on a school network, teachers across the biology department can use the simulation. This would also mean that students who are signed into the network can access the program as well, which would greatly contribute their learning. By using a simulation rather than making other applications like a quiz game; students can visually see the effects of different factors of the world environment be played out. This would provide students a visual interpretation of how species would behave under certain conditions. A simulation would also improve students' knowledge as it is a more hands-on experience than other learning methods. This would reduce boredom within the classroom as it is a different approach to learning, and so would increase the chance of students remembering topic ideas that have been discussed in class.

Research

Interview

I recorded the conversation on my phone and para-phrased it into this document.

Questions answered by Mrs.Brookes:

Russell: “How do you currently teach the ecosystem in class?”

Mrs. Brookes: “I use three methods to teach Ecosystems to my Key Stage 4 classes. I use the normal textbook styled learning. Online resources such as BBC Bitesize and Youtube videos. And lastly a toy representation were I guide the students in a narrative based way. Fun fact I usually get the toy animals from Mr. Lobbets room.”

Russell: “Do your students learn well in this way?”

Mrs. Brookes: “I hope so. I would definitely say they are more engaged with online resources mainly because they would be sharing the computers in pairs. But obviously they get distracted by each other. The textbook style learning is very effective as it is exactly what the exam board wants the student to know. But I can tell by the students faces that they don’t like it.” Lastly the toy model way is more for the Key Stage three students, nevertheless I sometimes do it for the Key Stage four. I personally believe its good because it takes the student out of the typical classroom lesson. But it might come across as childish and students might still not understand.”

Russell: “Do you feel like online resources can help students learn better?”

Mrs. Brookes: “Absolutely. They seem to retain students concentration. They are able to go back on things they couldn’t understand in the classroom. And I think students prefer to go online in general, they seem to use their phones a lot in school.”

N.B I then discussed my program solution to Mrs. Brookes, giving insights on what is included.

Russell: "I told you this earlier but I'm planning to make an Ecosystem Simulation program for school students to use. In your opinion, do you think this would help your students as a revision tool, and what would you require from this project?"

Mrs. Brookes: "Well from what you were saying earlier I do think it will help students. It will definitely spring interest into students. You said that it will include desert, glacial landscapes and that is part of the GCSE spec. So its definitely tide into the GCSE course. But I do worry how detailed the system could get as a lot of information would be stored. Personally I'd say cut out extra detail, and I really don't want to be replaced by a simulation. By the way I want this to be done around Easter time as that's when I start the topic for the year 10's. Umm. In my personal opinion though I wouldn't say it should be a strict revision tool, but should be an extension of learning for students who get bored. So just focus on making the simulation fun, tide to the GCSE spec. and make the students play around with experimenting in the simulation like your birth rates idea.

Questions with three GCSE geography students (Year 10's):

Russell: Do you revise well with online resources?

James: Yes.

Liwia: Prefer the textbook, but I definitely like using BBC Bitesize for revision.

Shardé: I revise the night before a test, I usually use the textbook though but I feel there is a better way of revision.

Russell: What is your preferred learning type out of reading, visual or listening?

James & **Shardé:** Visual

Liwia: Reading

Russell: Do you think you are learning well during class?

James: Yeah, I'd say Mrs. Brookes is very thorough and I'm getting good grades in my assessments.

Liwia: I honestly learn better by myself, I say I'm slower to learn things so I take most of my learning outside the classroom.

Shardé: When I listen I get it, but I just find my lessons boring.

Russell: Are their improvements you can provide to make you learn better in class?

James: In our geography lessons we rarely use any visual demonstrations. The only thing close to that are school trips but that happens occasionally.

Liwia: I'd say Mrs. Brookes goes through the topics way too fast, I could only catch up by staying behind in class or do extra revision.

Shardé: I'd say my concentration is a really big problem when it comes to the lessons, I just don't feel that interested in the topics.

Questionnaire/Survey

I was given a few minutes to do a questionnaire with Mrs. Brookes Year 10 class. They answered by putting their hand up if they agreed with the statements. (Class size = 27)

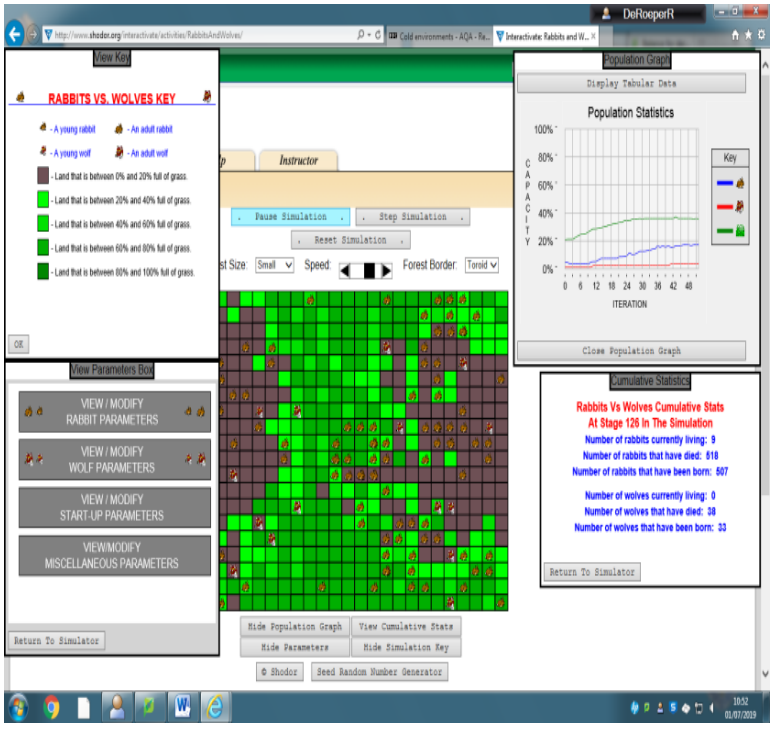
QUESTIONS:	NO	YES

Do you need more revision sources?	**13/27**	**14/27**
Do you learn better with online resources than textbooks?	**9/27**	**18/27**
Would you say you are a visual learner?	**4/27**	**23/27**
If I need help to test my project during development, would you want to be the tester?	**10/27**	**17/27**

From having individual meetings and a class consensus on their thoughts of online learning, I have a stronger feeling that using a simulator is a great idea to spark interest in the topics that students learn in the geography department.

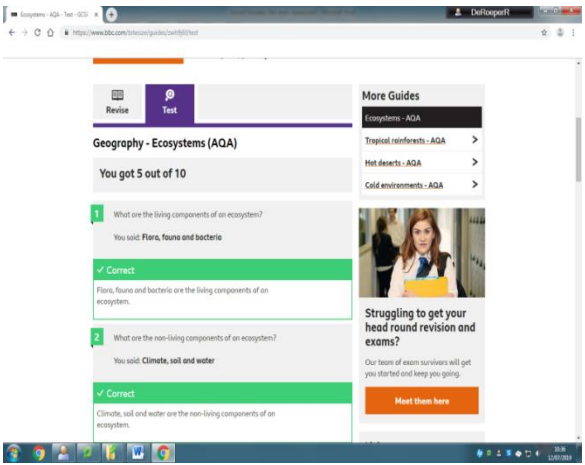
Online Research

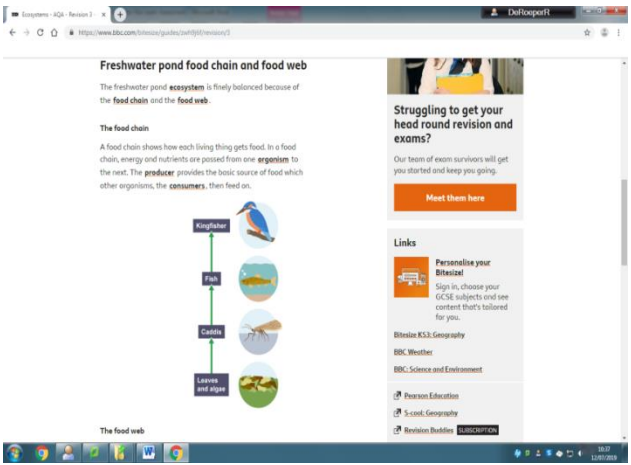
On the internet there are a couple of wildlife simulations were I will draw my inspiration on to create my solution. Firstly a program called “Rabbits and Wolves” is a wildlife simulator that demonstrates how rabbits and wolves interact and behave in a grass environment. This program contains



useful information demonstrated by live graphs like population number of species, which would help aid the user on the population at each step of the program. It also reflects how the rabbits impact the environment by changing the blocks of the grid world from green (representing grass) to brown (representing mud). These insights would help the user understand the impact of wildlife on the environment and also the need of other species to survive. By using the simulator and deriving details as explained, it is a very effective tool in learning these ideas compared to just typing them down. This would therefore provide the user a visual interpretation of the concepts. However this program isn't as complex as needed to understand the ecosystem in detail that the geography students would need. This is because only wolves and rabbits are involved in the environment, when for the specification multiple species are involved. The program also doesn't provide different types of environments, so the effects of the environments are only limited to one area. Furthermore information about birth rate, species gender, diseases are not included making the model of the environment very basic.

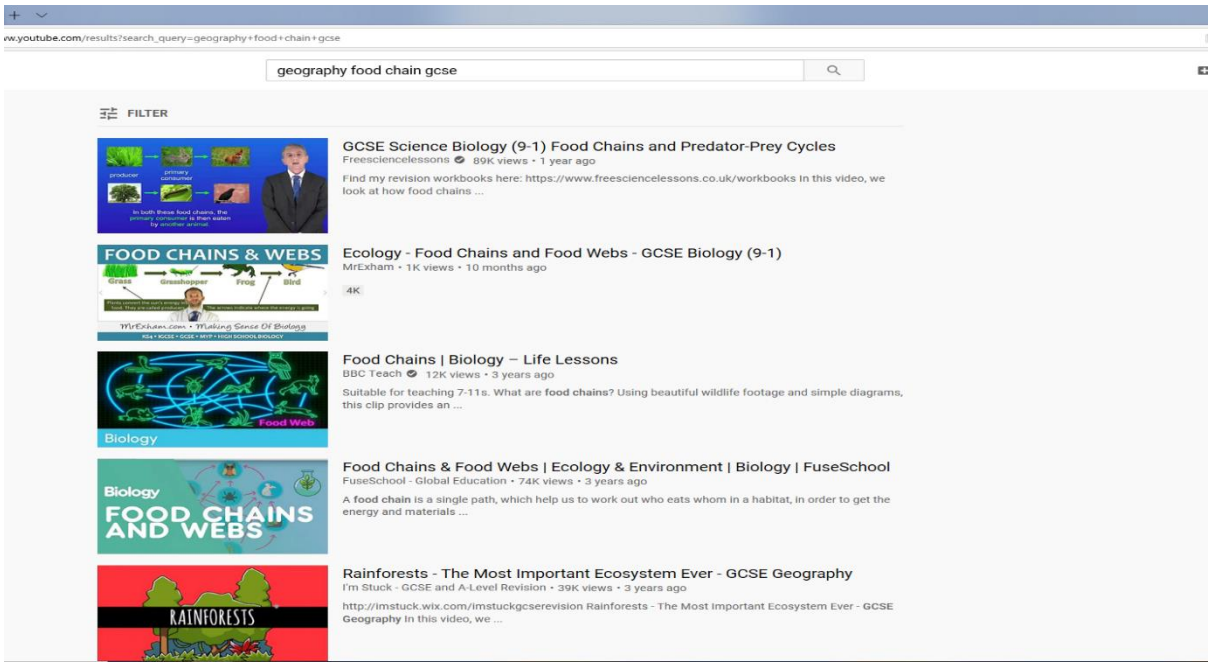
The next resource I would like to note from is BBC Bitesizes' websites, more specifically the Geography revision page on ecosystems and the food chain for the AQA specification. Within the website it contains useful resources like revision notes and a glossary (list of keywords relating to the topic). These are very effective learning tools, and from personal experience whilst doing GCSE geography myself it was a useful revision tool that simplified information to small notes. In addition the website has tests alongside each page of a topic. This means that each topic has several tests concerned with particular ideas regarding the topic. This is beneficial as it strengthens the students' knowledge of the topic, and shows areas of improvement. From doing tests, students are also





engaging more with the course material and are then expanding their knowledge through varied methods. But as this website uses mainly a traditional approach to revision (tests and notes), it may not be the most suitable learning method for all students as it can be too similar to how students revise normally. This means that it would provide less usefulness in comparison to other students who have their own notes and test papers.

My last online resource that is used for students as an effective learning tool is YouTube. YouTube is a video-sharing website with over “8,674,830,769” videos (quoted on Quora), this estimate is most likely inaccurate as the amount of videos published increase immensely per day. From this vast collection of videos, a huge proportion of it is dedicated to educational purposes like past paper walkthroughs, topics, university lectures and more for types of students. As students are connected with these resources it is highly effective as students can work at home and revise. In addition due to the different types of channels which post on the platform. The students have a much greater and in depth knowledge of a topic, which helps them to learn as they have different people teaching in different ways. Furthermore the site is free for anyone with access to the internet can use. This would maximise the number of people using the platform as price is not a deterring factor to use the product, this means that students of rich or poor backgrounds can access it. However YouTube is a source of entertainment for many individuals on the web. Therefore the site can prove to be distracting to the user, and would then obviously steer the user away from the learning experience. Another disadvantage of YouTube is that there is no fact checking tool to check if all the information being displayed in the video is correct. This means that videos on YouTube are not proven to be reliable. This could badly affect learning as revision may be spent on learning wrong material.



Client & Users

My client that I will be developing my project for is Mrs. Brookes of Alec Reed Academy's geography department. At the school there is 2 teachers with 5 classes each. So it is apparent that the ecosystem is taught regularly through the school, with different levels of depth and breath. Furthermore students do not learn concepts as quickly as others as the wide range of students have different learning styles. At Alec Reed Academy they normally teach this through YouTube videos or by textbook, which students may not find suitable to learn from. As my program will be used in classrooms, I must also adapt to the requirements of the students. So when designing my project, I need to weigh in the needs of the teacher and the students. When speaking with Mrs.Brookes I have realised that the ecosystem is taught from year 7 onwards. This mean I have to make a design that would accommodate to both high and low levelled learners in geography. Possible solution to this might be to create different versions for different year groups to access. And within each version a suitable amount of detail for there specific course would be integrated into the versions. The finer details of the users' needs would be found out by conducting regular meetings with the teacher, were I would be addressing updates on the development of the program.

The Current Systems and their review

Teachers within the geography department use physical demonstrations, online resources and school textbooks to teach students at Alec Reed Academy.

Within classrooms teachers have a set of toy animals that represent predators, prey etc. and a plastic green sheet to represent the environment. These items are used to help represent the ecosystems. This style of teaching is largely driven by the teacher as she talks through the workings of the ecosystem. This usually takes places within the middle of the class as the teacher selects a table and sets the pieces and sheet up. This would then allow the whole class to surround the table. As the teacher talks through the ecosystem, the teacher asks the students during it, to keep the students engaged and help the students learn to think for themselves. It shows to be effective as students feel to be concentrated more as they try not to be caught not listening by the teacher. In addition the teacher even ask questions as a recap to what they learnt moments before to instil the concepts into their minds. Another benefit of this method is that it is very simple to setup and pack, this would then mean it would maximise the amount of time within classroom to get other work done such as worksheets and other learning resources. Usually within Alec Reed this style is rarely used as it is a contingency plan in case the school computers don't work. This is very helpful as it doesn't rely on the school network to work and also assures all classroom members have the same knowledge as it is given by the teacher. A disadvantage of this way of teaching is that the class is limited by the teachers knowledge, so in times where a substitute teacher has to do this the quality of teaching drops.

Online resources are widely used across Alec Reed and along other secondary schools. Within Alec Reed we preferably use YouTube and BBC Bitesize as they have tons of information regarding to specific courses taught at school. In this case AQA Geography. These sites are effective tools for revision and in-class resources. Earlier I have mentioned the benefits of these two sites but they

have more benefits within class. As a disadvantage to YouTube I said that the videos might not be reliable. But as a teacher would of reviewed the video before being played within class, the information in the video would be correct and suitable to the level for the students to understand. Another advantage is that these resources would be equally affective even if the teacher is poor at teaching as the need for explanation is gone. However a disadvantage of this method is that not all students learn better this way as you cant question directly the information given unlike questioning a teacher in class. By questioning the teacher it would be much easier for student to have a greater depth of knowledge in the subject material. This also leads onto how the information is limited to what been put up on the resources. So it is unlikely that the online resource would update on new information that could potentially deepen the users understanding.

Lastly textbooks at Alec Reed Academy are used immensely and are critical as to know the entire specification of the course. My client {Teacher Name} gave me a copy of the textbook they use in GCSE Geography called “GCSE 9 – 1 AQA Geography Revision Guide” by Rebecca Tudor, Tim Bayliss, Catherine Hurst and published by Oxford University Press. It is a relatively new book as it is associated with the new GCSE specification and system. The book is very organised with an index, glossary and contents page to help the student learn be organised enough to revise through the book. It covers many topics including are topic on the Ecosystem. The topic covers 19 pages in the book which is a lot regarding the amount of bullet-point facts, case studies and exam style practise they packed into the topic. It is very detailed as it looks into three different ecosystems, tropical rainforests, hot deserts and cold environments. With this book it covers all the necessities needed to achieve high grades in the final exam as it was written directly for the students taken the exam on. In considering disadvantages there aren’t many huge concerns, but as the book can be taken home with teachers acknowledgement the book can easily be damaged or lost. This would mean it can cost the school a lot of money to buy more and so it could halt a students learning as the rest of the class would be learning. Other than that using this revision guide is the standard for all student to learn the course material.

User Requirements

From discussing about how the students best learn about the ecosystem to my client, she gave a list of user requirements that should be implemented to my project.

1. Needs to be user-friendly to the point year 10 students can manage it.
2. Provide information of the topic inside the revision guide to the user, or can easily be used alongside the revision guide.
3. Freely accessible for all students and teachers within the school to use.
4. Have great visuals to give students a better look at how animals behave in an environment rather than just writing.
5. The solution needs to consider aspects of the ecosystem studied in the GCSE (SECTION B 5,6,7,8)
6. The solution needs to be robust ; being able to stay consistent in its utility.

Draft Objectives

The program is easy to manage for students in Key Stage 3:

To achieve this objective I must make a simple main menu as soon as the program is executed. On the main menu it should have an options button, start button and quit button. The interface should be simple and clean so students and teachers can manage setting up the program easily.

Option display:

When the option button is clicked the program should move into a new page were configurations can be made for the simulation. These configurations would consist of landscape size, habitat type, predator and prey selection, birth rate and death rate. These inputs would be clicked in from a drop down menu or typed in for a numerical result.

Animals are able to reproduce, die and age:

To make the system replicate animal reproduction the user would be given a chance to update the birth rate of each specie. If it wasn't filled in, the system would just use the default mode. Likewise, the user can input the death rate accordingly. As of right now I am unsure of what rates are suitable to be the default mode.

Ecosystem simulator components:

A two dimensional grid (with dimensions specified by the user) would be the environment of the simulator. Each block within the grid would be colour coded to represent the density of forestation or plant-life. The lower the level of plant-life there is within the grid the colour would have a lighter shade until there is no plant-life and a dark colour would appear. Over time plant-life would grow back to high plant-life is it isn't used up by animals. Animals would be represented by simple images of the animals. They would move to one new block per step count. In the simulation the user can be able to quicken the speed of each step similar to my online review of the "rabbit's and Wolves" program. Another idea I want to replicate from the rabbits and wolves simulation is the recording of the species population and forestation level. This would be helpful to show the user empirical evidence of how different factors affect each other within different times (steps).

Quit button:

Lastly my program needs a way for the user to exit the program. With this I am thinking of just implementing a quit button at the main menu. For convenience I am going to put back buttons within the simulation screen and options screen to help direct the user back when they are done. So as the user wouldn't need to wait for the whole simulation to return back, users save time.

Prototyping/Feasibility Study(time/technology/cost/resources/operation Legality)

The environment simulator that I will be making will need required technology, time, operation legality and resources.

Firstly as my solution will only acquire resources within school, luckily the project does not have any additional costs to build.

Being guided by my clients needs and timetable; the time we agreed for me to build the simulator is by Easter Holiday. This time would be best as I have been told by her and the geography department that they teach the ecosystem module in the summer. This gives plenty of time for the teacher to understand how to implement and use the program within class. This is also very suitable as it gives plenty of time for geography teachers to plan out classes were the simulator would be used. This is very important to note, as my client is usually stressed from the workload and pressure which is placed upon her.

The technology (resources) that I will be using to build this project is the school computers. {Details of the computer}. As these computers are relatively old the performances on them is quite low. Throughout my time building the project I must be aware of the CPU's performance when running the simulator as it could make the performance of my program worse. In addition to this the computer needs to have Visual Studio installed into the computer. This is a widely used IDE which is used within schools and in businesses across the globe. It is a free application and it is known to be an excellent platform to program on as it supports many programming and mark-up languages. This being said, I will use VB.NET to program the project. It is very suitable for this project as the language can easily implement user interfaces, making it easy to prototype the program and make adjustments when ready. To make this project accessible with the rest of the school it must be put

into the school network. The network consists of many files and applications exclusive to school teachers and students. This would help me achieve one of the user requirements.

However to allow the project to be put into the network I must have authority to do so. This would be in the form of technicians or client as they have direct authority to create, read, update and delete any file or application on the network. As I am a student of the school I alone can't achieve this, so it is safe for me to ask permission so the program can operate legally with notice from people with higher levels of access to the network. Though it can seem as a barrier for developing the application, it can be used as a measure to show if my application is good enough to be used in the school environment. Furthermore it is also beneficial for the security of this program as only members of authority can manage the programs file on the network or possible be a safeguard from anyone changing the code of the program.

In considering my original objectives I must say that my feasibility study does not conflict with my objectives in the design of the program.

Final Objectives

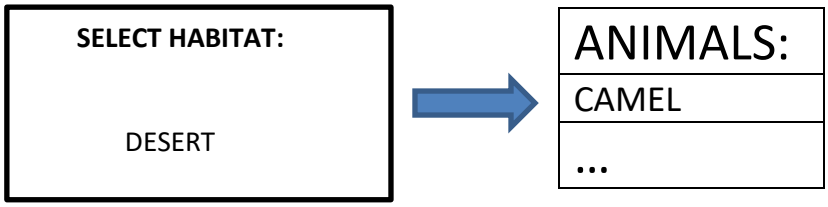
By taking into account of my feasibility study I have updated my objectives in a more detailed way for approaching the design of the system.

The program is easy to manage for students in Key Stage 3:

I plan to achieve this by providing a simple interface made in VB.NET's forms and connecting the options to a database for an easy selection. Students will be able to study the results of the simulator from a population line-graph and from exporting information from a text file. These added components will allow students to understand the dependence of species coexisting in nature.

Options display allows easy configuration for simulations:

In my options display it would contain simple headings such as birth rate, population size and board size to easily help the user identify the type of simulation they want. As certain habitat types won't contain animals for example the desert won't have a giraffe in it, the options would configure to the certain options selected. Put simply an example like:

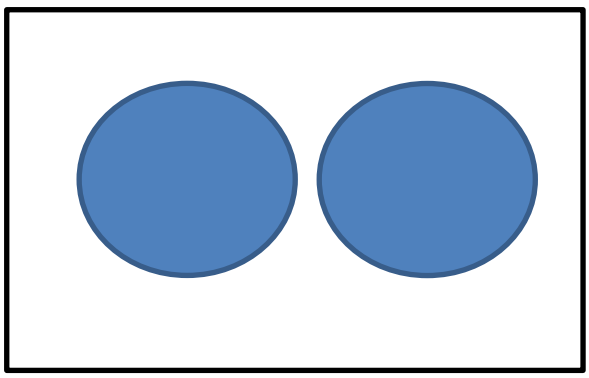


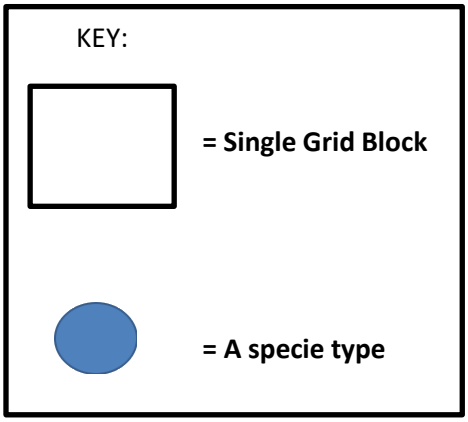
As you can see if a user selects a certain option like "desert", only an appropriate animal will show up in the animals drop down menu. This would make the system easy to use for students as the system would adjust itself.

Animals reproduce, die and age:

Animal reproduction

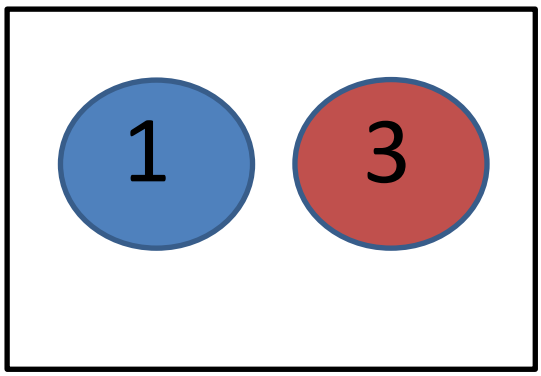
Animal reproduction is based largely on the birth rate dictated by the user in the options menu. In the simulation the birth rate is a probability that provides a chance for the population to grow. It get triggered when two of the same specie meet in the same grid block. This activates the birth rate probability and by a random number from 0 to 1, will determine if a new birth occurs. This represent how not all members in the same specie want to mate, are of different genders or just so happened to be in the same grid block by passing by.

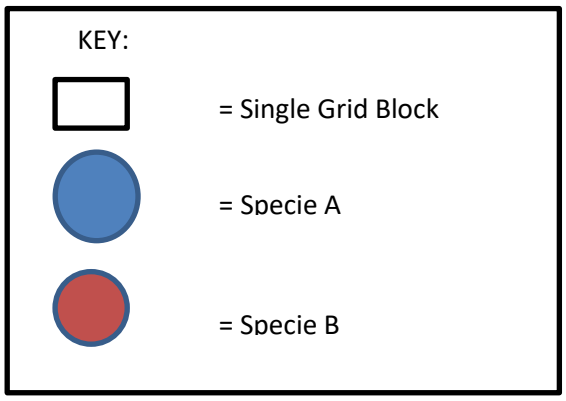




Animal Death

In addition to this there are multiple ways animals can die. The first way is dictated by the user again of controlling the death rate of specific species. The death rate is the same as the birth rate but reduces the population by one when activated in contact of another predatorial specie in the same block.





As you can see the numbers displayed on the species represent whether if the specie is a prey or predator. The method here is that species with higher numbers are able to eat animals of lower numbers, but they cannot eat species of the same numbers. In this case 3 represents predator, and 1 represents prey. Like the birth rate the death rate is a chance within a random that a prey dies or survives and encounter with a predator. If the random lands within the death rate probability the population number of specie A decreases by 1. This also decreases specie B's hunger meter that I will explain now.

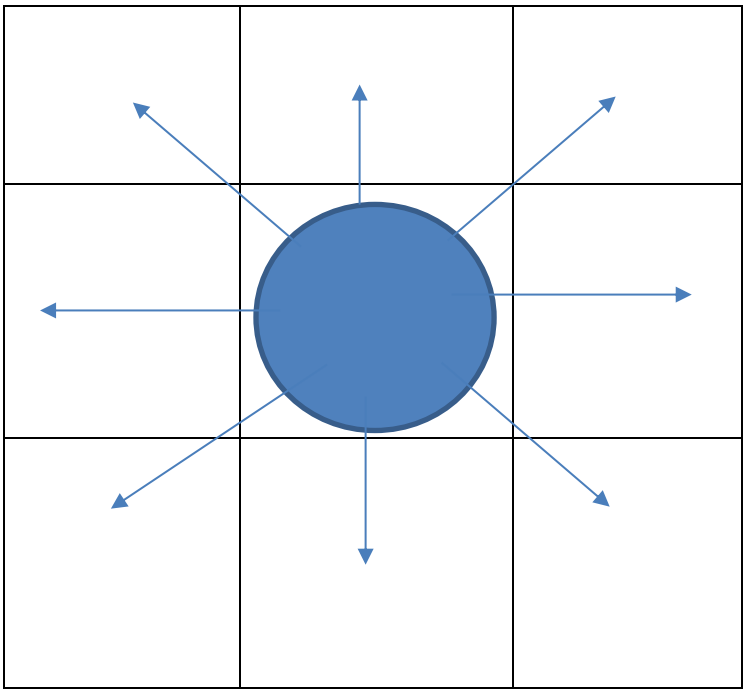
Hunger meter

Each specie is given a hunger meter that can be made variable by configuring the options menu of a specie. In the simulation time is represented by the number of steps taken in the simulation. As the number of simulation steps occur without eating; the hunger meter increases. When the hunger meter increases to its maximum the particular member of the specie dies. Thus decreasing its population by 1. This meter can replenish by eating vegetation or prey depending on the species food type.

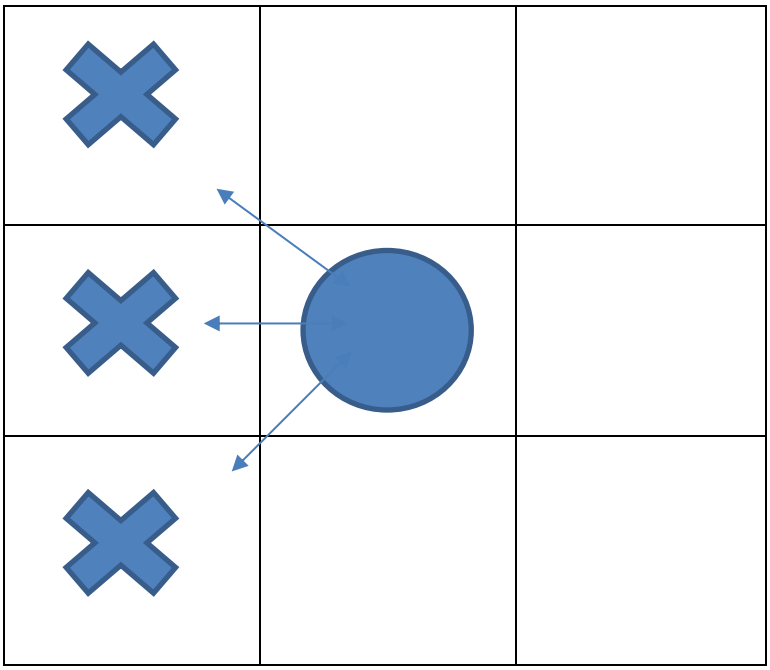
Animal Life Span

Like how all species have particular hunger meters; all species have particular life spans. In the simulation a life span means the total number of steps a specie can live for. This is not affected by the hunger meter or any other factor. As in the real world, all agents in the simulation will age as time goes onward. This balances the number of species within the simulation. As soon as the life span count is reached, the population is reduced appropriately.

Animals are able to move freely:



Animals or species within the simulation would move in total of eight positions. What would dictate the animal moving into a certain position is the probability of a random function. This would only move a specie one block away from them per step.



In case of the agent being next to the grid border and the random selects a move past the border then the agent will then have another random chance to move. Eventually the agent will move away from the edge.

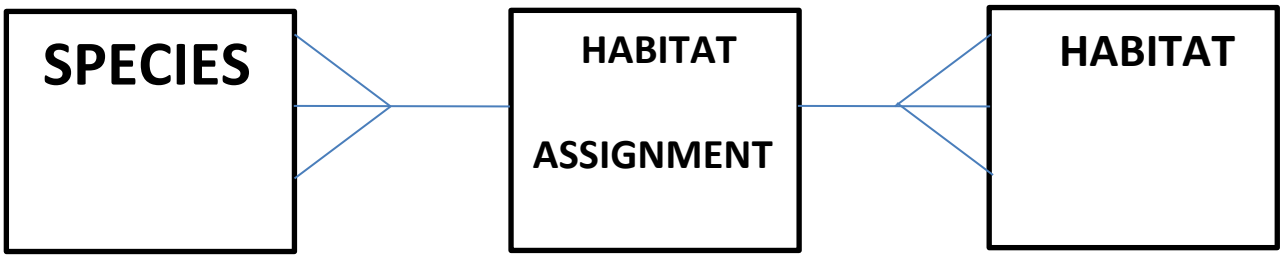
Ecosystem simulator components need to work efficiently:

Simply put, the components that I have explained before need to be working well without any slow down in running the program. However things such as the real time graph of population and vegetation within a certain step count need to be running whilst the simulation occurs. It is absolutely paramount that to make the simulation run smoothly; information must be reduced. This means that even the plans I have created thus far may not be in the final stage of development as it can harm the performance of the program as a whole.

Documented Design

E-R model

A Many-to-Many Relational Enity Diagram. For matching certain Species to its dedicated Habitat.



User will be allowed to manipulate the data found in these tables to allow addition of new animals and habitats be used in the simulator. This would allow the students to have greater control and customizability of the simulator, resulting in the program to be more engaging.

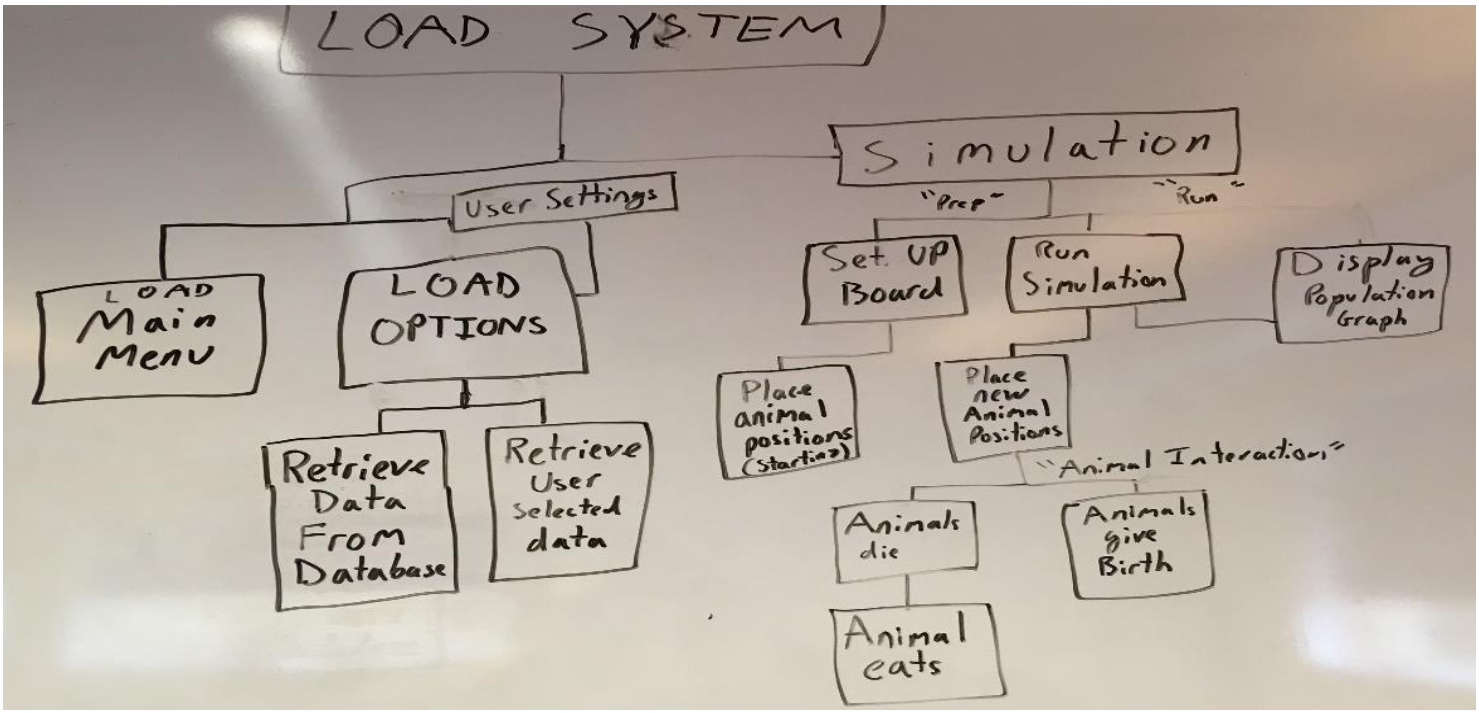
Data Volume

My simulator program may have data volume concerns as the simulation itself would calculate hundreds of factors every single step in the simulation. This means that whenever the simulation is running a lot of processes and calculations are happening in a short space of time. As the school computers are not of the best quality it may also be a cause of concern as there is a potential that the CPU performance may slow down when the simulation runs.

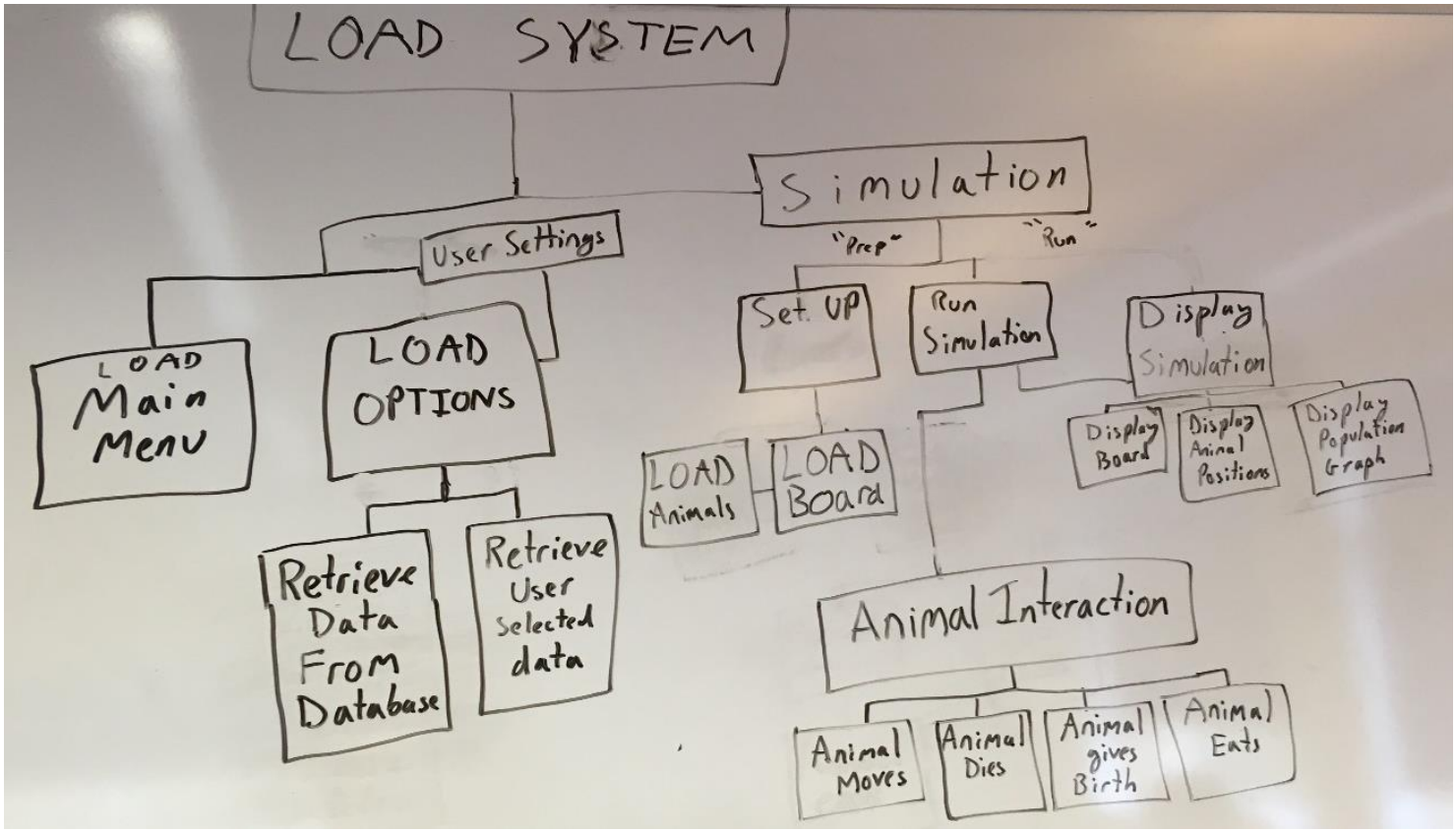
However as we know of the number of users on the program (the geography department), it is safe to say that there is no need to worry about scaling up the school servers. Furthermore I predict that storing the program won't be an issue as I overestimate the finished program won't be greater than a megabyte.

Hierarchy Chart

Draft 1:

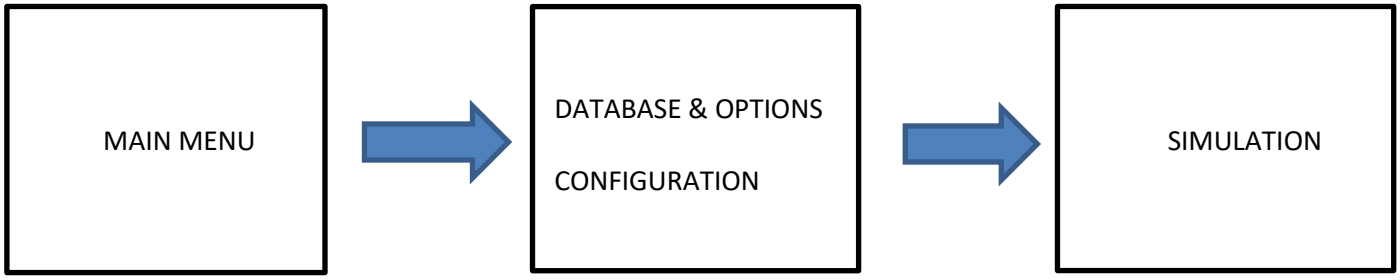


Draft 2:



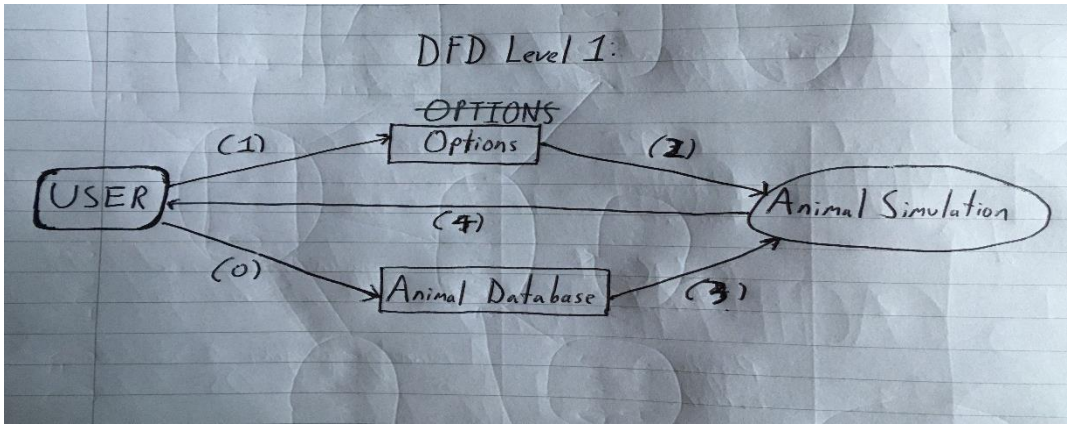
Data Flow Diagram

Data flow diagram 0:



After the user configures all the simulation options, the simulation would begin and record population and vegetation level per step. This would result in sequences of data for each factor over the period of steps which have taken place in the simulation. This data would then be saved into the data if wanted, otherwise would be removed in the likelihood of another simulation data.

Data Flow Diagram 1:



(0) The database can be configured before the program begins, which is why the process is marked with 0. This data determines the attributes each animal has, and possibly add further animals to the system.

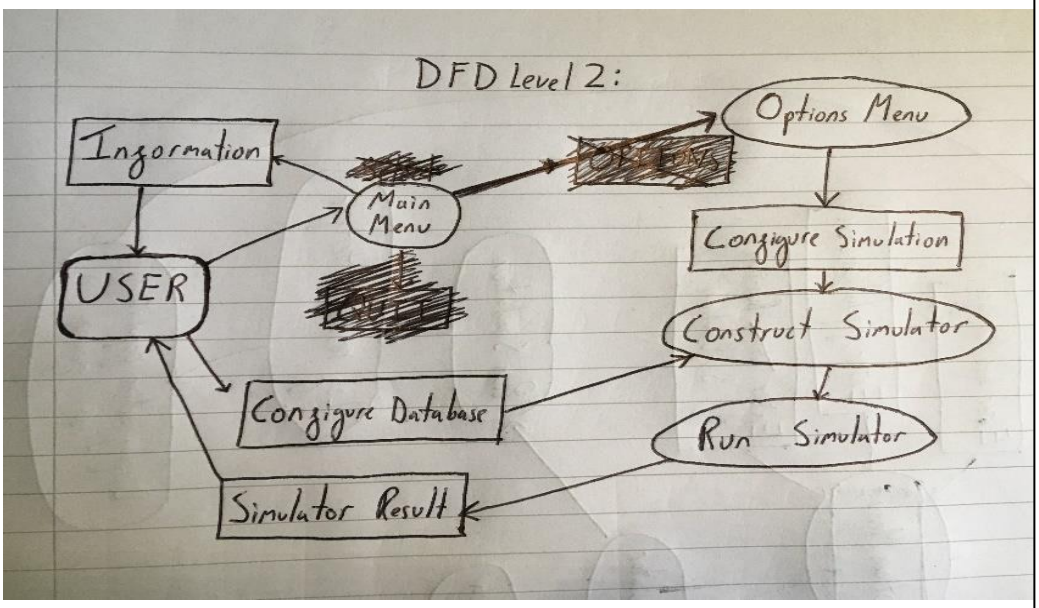
(1) When the options menu is opened, the user can determine the populations and types of animals, type of habitat and the run-time of the simulation.

(2) The stored data from the options menu is then sent to the simulation, where the simulation constructs itself from.

(3) Likewise to (2) the animal database's stored data is sent to the simulation, which helps construct the simulation.

(4) The results of the simulation would be projected back from the simulation to the user in a graphical form and text file.

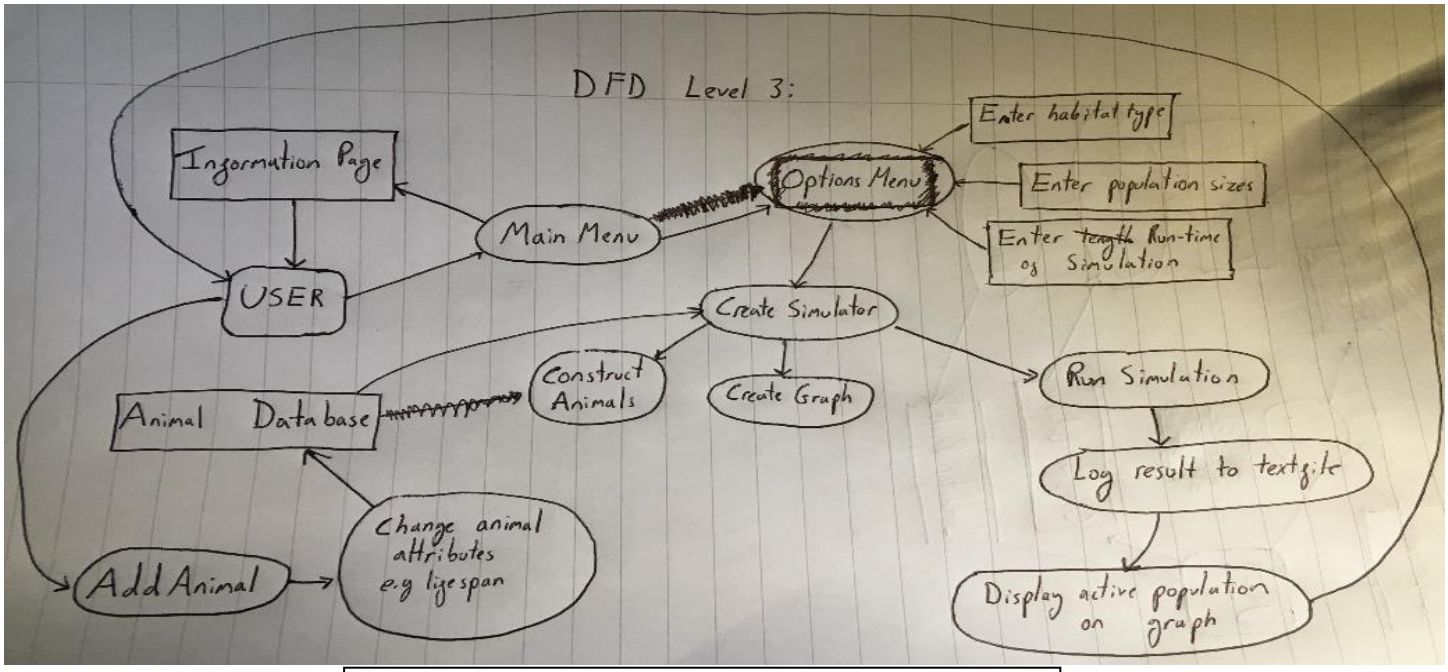
Data flow Diagram Level 2:



The user selection of the main menu determines how the information form and the options menu processes affect the flow of data.

*At this point of development I chose to put the simulation and options forms together, to adjust to Mrs Brooke's preference found in HCI.

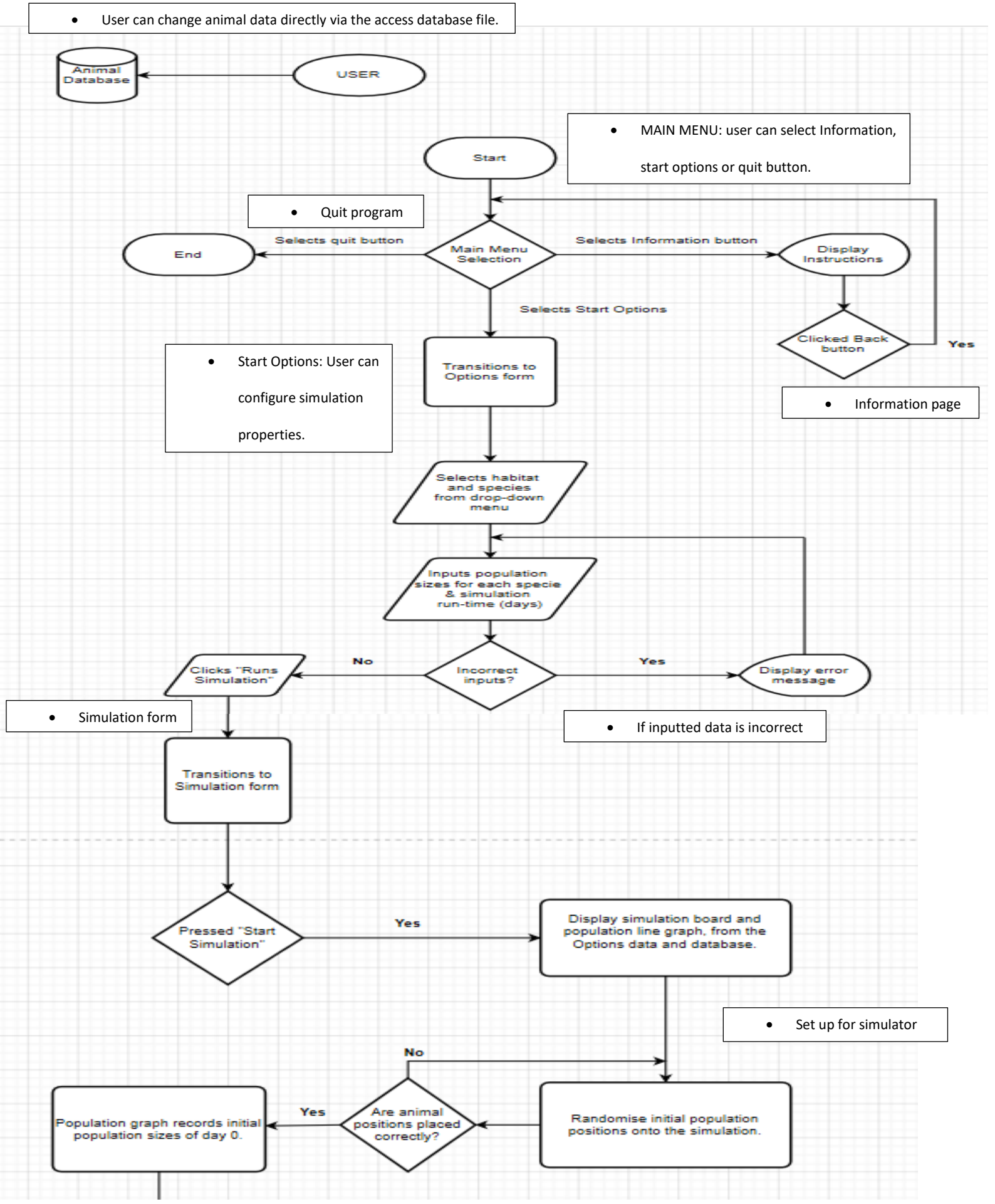
Data flow Diagram Level 3:

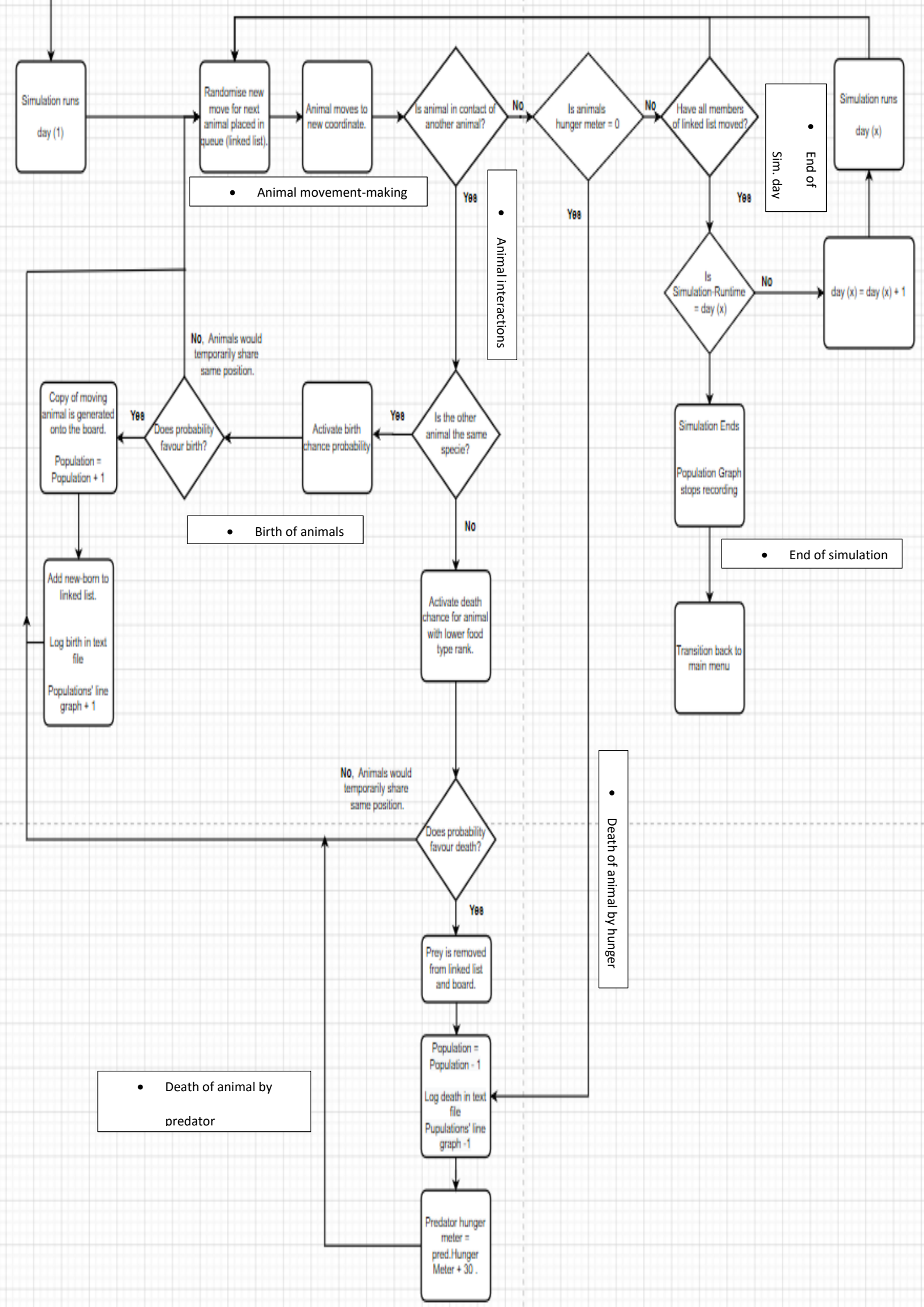


DFD level 3 breaks the process down even further.

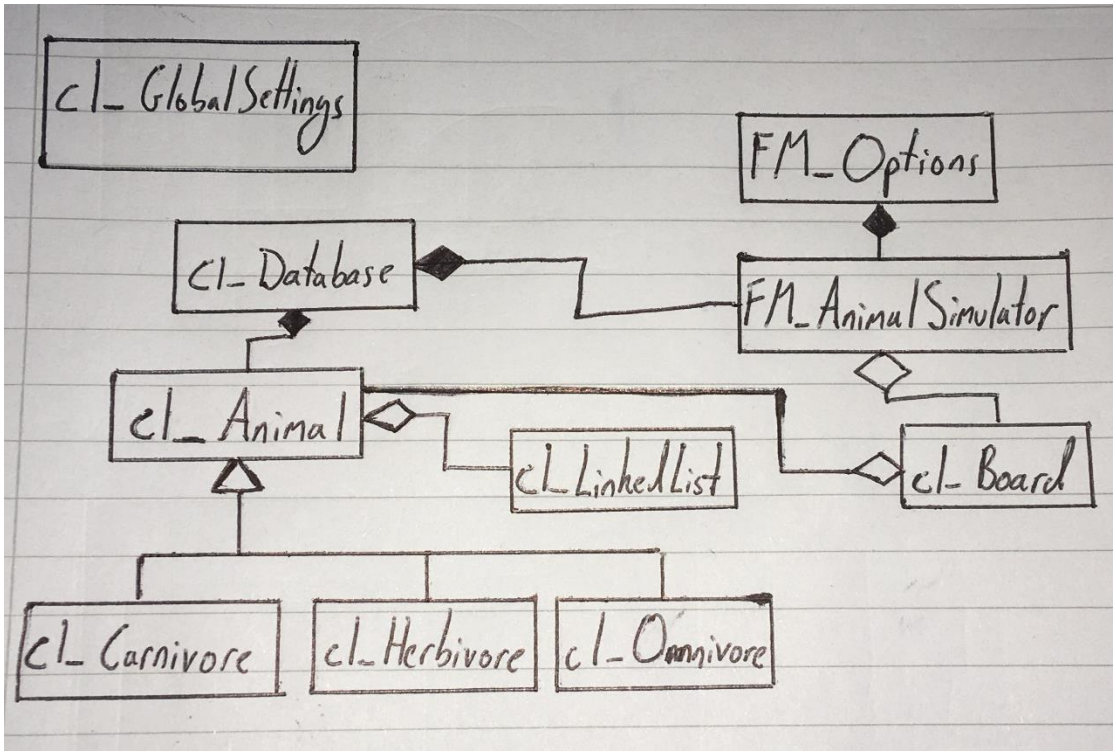
This is the system flowchart of the Animal Simulator. I have annotated parts of the flowchart to show what the flowchart is representing to prevent confusion.

System Flowchart




Simulation runs day (1)
Randomise new move for next animal placed in queue (linked list).
Animal moves to new coordinate.
Is animal in contact of another animal?
No
Is animals hunger meter = 0
No
Have all members of linked list moved?
Yes
End of Sim. day
Simulation runs day (x)
Animal movement-making
Yes
Yes
Animal interactions
Is Simulation-Runtime = day (x)
No
day (x) = day (x) + 1
No, Animals would temporarily share same position.
Copy of moving animal is generated onto the board.
Population = Population + 1
Yes
Does probability favour birth?
Activate birth chance probability
Yes
Is the other animal the same specie?
Simulation Ends
Population Graph stops recording
Birth of animals
No
End of simulation
Add new-born to linked list.
Log birth in text file
Populations' line graph + 1
Activate death chance for animal with lower food type rank.
Transition back to main menu
No, Animals would temporarily share same position.
Does probability favour death?
Yes
Death of animal by hunger
Prey is removed from linked list and board.
Population = Population - 1
Log death in text file
Pupulations' line graph -1
Death of animal by predator
Predator hunger meter = pred.Hunger Meter + 30 .

Class Diagram



This is the class diagram of the Animal Simulation system. Carnivores, herbivores and omnivore classes all inherit from cl_Animal. The cl_LinkedList is a class that generalises the method for a queued linked list. This means that it is not dependent on another class. and has an associated aggregation relationship with cl_Animal and cl_Board as the linked list contains data of the animals and simulation activities within the cl_board class occur from the linked list. FM_AnimalSimulator needs the data of FM_Options and CL_Database to operate and therefore has a composition aggregation relationship. Cl_GlobalSettings is a static class and holds values that is important to how a particular simulation works. Further details of how each class works is below, but greater annotation is found in the source code by implementation.

Cl_Animal is the parent class of Cl_Herbivore, Cl_Omnivore and Cl_Carnivore as it contains all the characteristics each of these animal types need to operate in the simulation. Things like movement, age, hunger and also confirmation if they die from hunger or old age.

Class cl_Animal
+ AnimalName As String "All Animals have a name to be contained. + FoodType As Integer "[1] = herbivore, [2] = omnivore, [3] = carnivore + LifeSpan As Integer "Each animal has a certain life span unique to them. + DeathRate As Double "Each animal has a certain Death Rate unique to them. + Birth_Rate As Double "Each Animal has a certain birth rate unique to them. + HungerMeter As Integer "Each Animal has a length of time they can live without food." + AnimalSpeed As Integer "The number of grid blocks an animal can move in a sim. day. + FoodChainRank As Integer "Compares animals are able to be eaten or not when met. + AnimalSize As Integer = 10 "Constant that helps sizing of the animal on to the board." + isDead As Boolean "Confirm if animal is dead or not from age or hunger. + age As Integer "Each animal ages as simulation continues. + hungerlevel As Integer "The current hunger rate after being increased or decreased.
+getAnimalSize() As Integer "All animals fit the board by 10x10. +GetMove() As String "Random function picks directions for animals to move to. +GiveBirth() As cl_Animal "Function that is overridden as each food source has different method. +GiveDeath() As Boolean "Function that is overridden as each food source has different method. +GetFood() As Boolean "Function that is overridden as each food source has different method. +DayStatus() "Increases or decreases hunger and age when called and declares if animal is alive or dead

Cl_Board is a class that helps an animal know where to move on the board. With this location information this class also determines whether animals breed or can kill each other depending on the animals involved. It also holds all the habitat information and size which all animals are in.

+ Class cl_Board
+ BOARDSIZEX As Integer = 500 "The width of the board." +BOARDSIZEY As Integer = 500 "The height of the board." + BoardType As String "Glacial, Desert, Greenland." + Board(,) As cl_Animal "List of animals which are on the board for animal interaction." + animalList As New cl_LinkedList "List of animals that are removed and added to linked list." + EatenFood As Boolean "Check if an animal has been eaten or not."
+New(Habitat) " Sets up a new board with habitat and 50X50 size." +returnBOARDSIZEX() As Integer " Returns boards width." +returnBOARDSIZEY() As Integer " Returns boards height." +RunDay() " Subroutine that runs each day and handles animal interactions (death & birth." +returnCor(direction As String, x As Integer, y As Integer) As Integer() " Returns new animal position coordinates." +checkBoundary(coordinate() As Integer) As Boolean

"Checks if new animal position is within the board." +checkEmpty(coordinate() As Integer) As Boolean "Checks if a possible animal position is free." +addAnimal(animal As cl_Animal, amount As Integer) "Adds initial amount of populations and animals types into the simulation." +checkAvailableSpace(x As Integer, y As Integer) As Integer() "Check free space surrounded by an animal."

Cl_Herbivore is sub-class to cl_Animal, and has all the methods specific for herbivore species.

+ Class cl_Carnivore
Inherits cl_Animal "Carnivore inherits animal methods and attributes."
+New(animalName As String, animalLifeSpan As Integer, animalDeathRate As Double, animalBirthRate As Double, animalHungerMeter As Integer, AnimalSpeed As Integer, animalFoodChainRank As Integer) "Database values are given to carnivore." + Overrides GiveBirth() As cl_Animal "Dictates odd of giving birth and the attributes given to new born carnivore." -GetFood(ByVal food As cl_Animal) As Boolean "Dictates odd of getting food and if the animal is able to be eaten."

CL_Database is the class that allows the program to gain access to the access file containing all the data of every specie involved with the program. Because of this, the simulation is dependent on accessing the Microsoft access file and it also allows the user to select the animals to be used in the simulation.

+ Class cl_Database
+ Shared connectionString As String "Locates where the database file is in the directory." - +connection As OleDbConnection "Object that creates a connection between the program and Microsoft access database."
+New() "Whenever a new cl_Database object gets called it must identify the database directory and establish the vb library can manipulate file via the connection variable. +selectAnimal() As DataTable "Select all the animals stored within the database, to then be used in the simulation." +selectAnimalTerrain(Habitat As String) As DataTable "Select a list of animals that a user can pick that are assigned to a certain habitat." +selectTerrain() As DataTable "Create a list of possible habitats that can be used in the simulation." +selectAnimalData(AnimalName As String) As DataTable "Selects all the attribute data an animal has, to then pass into the simulation."

Cl_GlobalSetting allows many values for a specific simulation to be accessed across the program.

+ Class cl_GlobalSetting
+ Shared Terrain As String "Stores the habitat name being used for the simulation." + Shared AnimalONE As String "First specie picked for the simulation." + Shared AnimalTWO As String "Second specie picked for the simulation." + Shared AnimalTHREE As String "Third specie picked for the simulation." + Shared AnimalFOUR As String "Fourth specie picked for the simulation." + Shared AnimalFIVE As String "Fifth specie picked for the simulation." + Shared AnimalONEpopulation As Integer "First species' population size." + Shared AnimalTWOpopulation As Integer "Second species' population size." + Shared AnimalTHREEpopulation As Integer "Third species' population size." + Shared AnimalFOURpopulation As Integer "Fourth species' population size." + Shared AnimalFIVEpopulation As Integer "Fifth species' population size." + Shared TotalSimulationTime As Integer "The total runtime of the simulation in days." + Shared Board As cl_Board "Object Board is allowed to be accessed across the program." + Shared log As cl_log "Object log is given access to all parts of the program."

Cl_Herbivore is sub-class to cl_Animal, and has all the methods specific for herbivore species.

+ Class cl_Herbivore
Inherits cl_Animal "Possesses all the attributes and methods as every other animal."
+New(animalName As String, animalLifeSpan As Integer, animalDeathRate As Double, animalBirthRate As Double, animalHungerMeter As Integer, AnimalSpeed As Integer, animalFoodChainRank As Integer) "Gives all the data a specific herbivore has from the herbivores database data." -GetFood() As Boolean "Herbivores individual method of eating described fully in source code." + GiveBirth() As cl_Animal "Odd of herbivore giving birth.

Cl_linkedList helps the simulation gain structure in knowing what number of animals are alive, dead or gave birth.

+ Class cl_LinkedList
+ head As cl_Node "Head node of the linked list, stores an active animal on the board that's first." + tail As cl_Node "tail node of the linked list, stores an active animal on the board that's last." + current As cl_Node "Current animal being looked at in the linked list." -previous As cl_Node "The animal that has been looked at before the current animal." + populationNum As Integer "The number of nodes in the linked list which represents active animals in the simulation."
+New() "Linked List is established with no nodes." +addQueue(thisAnimal As cl_Animal X As Integer, Y As Integer) "How nodes are added into the First-in-First-Out form of linked list (Queue)." +NextAnimalNode() As Boolean "Moves the current node to the next node, so we can go through the list." +reset() "Resets the linked list back to the head node." +delete(x As Integer, y As Integer) "Deletes certain nodes within the linked list."

Cl_Log is used to log all the simulation data (The activities of animals per turn) into a text file. The text file is then saved in the same directory as the program and is accessed after the simulation is finished. When they close the simulation the data is lost for the new simulation data to be recorded.

+ Class cl_log
+wfile As StreamWriter "Class helps log all simulation activity into a text file."
+ New() "Renews the textfile to hold the data of a new simulation." + write(text As String) "Write simulation data into the textfile."

Cl_node contains all properties of a node to be used in a linked list.

+ Class cl_Node
+ piece As cl_Animal "The animal on the board which corresponds to its node." + X As Integer "The animals horizontal position on the board." + Y As Integer "The animals vertical position on the board." + nextNode As cl_Node "An node object to for the next node in the linked list."

Cl_Omnivore is sub-class to cl_Animal, and has all the methods specific for Omnivore species.

+ Class cl_Omnivore
Inherits cl_Animal "All omnivores inherit the methods and attributes of the animal class."
+New(animalName As String, animalLifeSpan As Integer, animalDeathRate As Double, animalBirthRate As Double, animalHungerMeter As Integer, animalSpeed As Integer, animalFoodChainRank As Integer) "All new omnivores are given specific values of there species contained in the database." +GiveBirth() As cl_Animal "Assigns new born carnivore the same attribute values as the parent." **NOTE:** Get food isn't needed as class board will handle omnivores getfood. Only carnivore and herbivore have unique getfood() methods.

The FM_AnimalSimulation class is the form and processes of the simulation. This contains the graph function which visualises the populations of each specie. As this holds the simulation it also means to carry out all simulation activities of all active species on the board.

+ Class FM_AnimalSimulation

+ b As cl_Board "An object used to have all the properties of the simulation board."
+Sub BTBack_Click() "Allows the user to go back to the options form." +Sub AnimalSimulation_Load() "Assigns the object b to contain the value of the board in global settings." +Sub RunSimulationBT() "Button that runs the simulation when clicked. It contains the real-time graph which shows the current population alive on the board. Also creates the board, animals and runs the activities on the board." + AddNewBornToBoard() "Subroutine that creates new born species and adds them to the simulation."

The FM_Options class is the form in which the user configure the species, the population sizes and habitat type for a particular simulation.

+ Class FM_Options
+FM_Options_Load() "Initialise the options menu by setting the population and specie types to empty." +ListAnimalNames(terrain As String) "Subroutine to allow combo box to contain all the available species, to then be selected by the user." +ListHabitatNames() "Subroutine to select the right specie which goes to a specific habitat type" +Animaldata(AnimalName As String) As cl_Animal "Returns animals data from its database attributes for the simulation." + BTback_Click() "Allows user to go back to the main menu form." + BTGoToSimulation_Click() "Button that's allows the program to transition from options form to the simulation form." +setup() "When all the wanted species and population sizes are selected, they are declared into variables to then be used for the simulation." +CBterrainType () "Combo Box that allows user to select the terrain of the simulation." +CBanimalONEpop() "First animal combo box which selected animals can't be selected again in the other options." +CBanimalTWOpop()"Second animal combo box which selected animals can't be selected again in the other options."

+CBanimalTHREEpop()"Third animal combo box which selected animals can't be selected again in the other options."

+ CBanimalFOURpop()"Four animal combo box which selected animals can't be selected again in the other options."

+CBanimalFIVEpop()"Five animal combo box which selected animals can't be selected again in the other options."

FM_Start is the form which contains the main menu for the program.

+ Class FM_Start
+ btOption_Click() "Button used to transition from main menu to options form." + btQuit_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btQuit.Click "Close the whole program." + InstructionPage_Click() "Button used to transition from main menu form to the instruction form."

Prototypes and its proofs of concepts

1. **Create the simulation environment :**

To give the user a visual representation of the simulation, I will be using a grid world that would be displayed within a form. To create this I will need to be able to draw squares to represent the habitat and species in different colours.

```
Dim myGraphics As Graphics = Me.CreateGraphics
Dim myPen As Pen
myPen = New Pen(Brushes.Black, 1)
Dim Brush As SolidBrush
Dim rect As New RectangleF(0.0, 0.0, 500, 500)

' Create rectangle.  ' Fill rectangle to screen.

Brush = New SolidBrush(Color.Green)
myGraphics.FillRectangle(Brush, rect)
myGraphics.FillRectangle(Brush, rect)

Dim ANimal As Pen
ANimal = New Pen(Brushes.Black, 1)
Brush = New SolidBrush(Color.Blue)
Dim r As New RectangleF(458, 155, 10, 10)
myGraphics.FillRectangle(Brush, r)
```

To draw squares, visual basic has a library called SolidBrush that help draw graphics.

1. **Proof of Concept:**

Output of the above code. A green square size 500x500 and a blue square at (458,155) from the top left with a size 10x10).

2. **<u>Create a moving line graph to record data:</u>**

For my animal simulation program, I want the user to know how the populations of

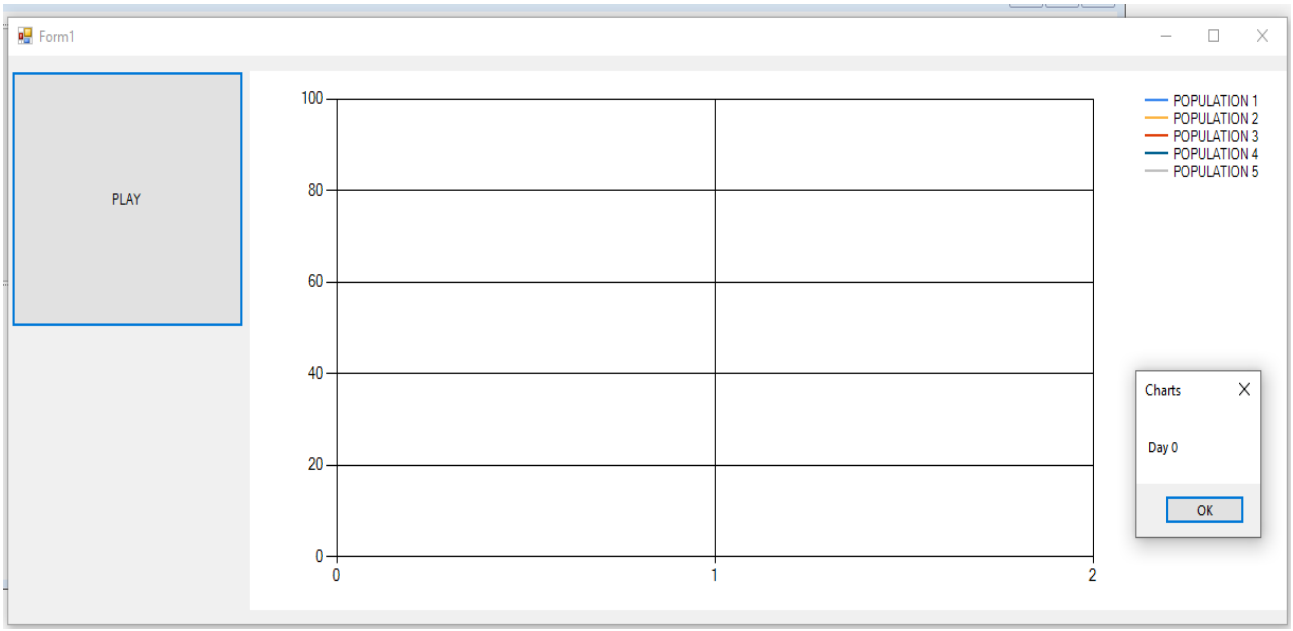
```
Public Class Form1

    Dim days As Integer = 0
    Dim totaldays As Integer = 200


    Private Sub BTplay_Click(sender As Object, e As EventArgs) Handles BTplay.Click

        For days = 0 To totaldays

            'To make the graph animate

            Me.Chart1.Series("POPULATION 1").Points.AddXY(days, GetPopulation)
            Me.Chart1.Series("POPULATION 2").Points.AddXY(days, GetPopulation)
            Me.Chart1.Series("POPULATION 3").Points.AddXY(days, GetPopulation)
            Me.Chart1.Series("POPULATION 4").Points.AddXY(days, GetPopulation)
            Me.Chart1.Series("POPULATION 5").Points.AddXY(days, GetPopulation)


            MsgBox("Day " & days)

        Next days

    End Sub

    Public Function GetPopulation() As Integer

        Dim number As Integer
        Randomize()

        number = Rnd() * 100
        Return number

       End Function
```

To make the charts have the appearance of “real-time”, I placed the drawing aspect of the chart function inside a for loop. This would animate the line graph reveal the constant changes in the simulation. For data inputs for the chart I set up a random number generator to demonstrate the “real-time graph”.

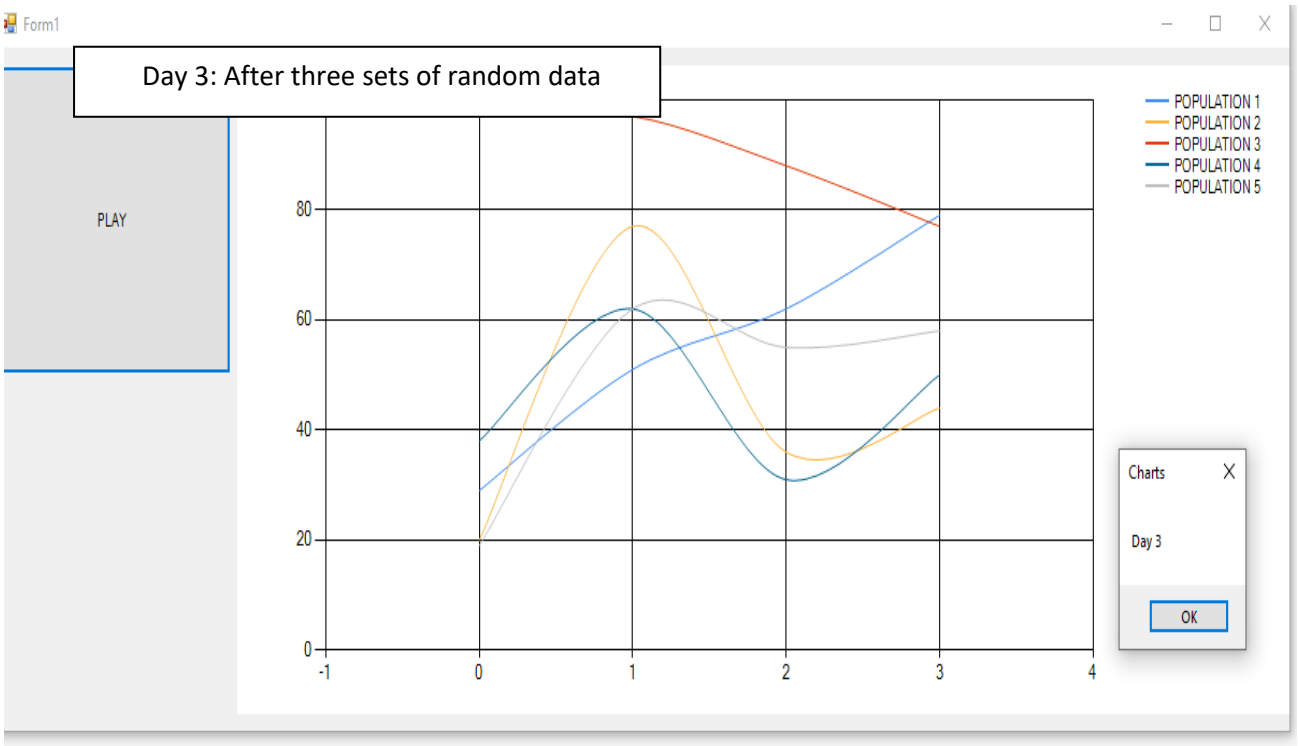
each specie changes over time. To do this, a line graph would be appropriate to represent the population changes within each day of the simulation. Fortunately the VB programming language have in-built functions designed for many graph applications, and in our case the line graph.

2. Proof of Concept:

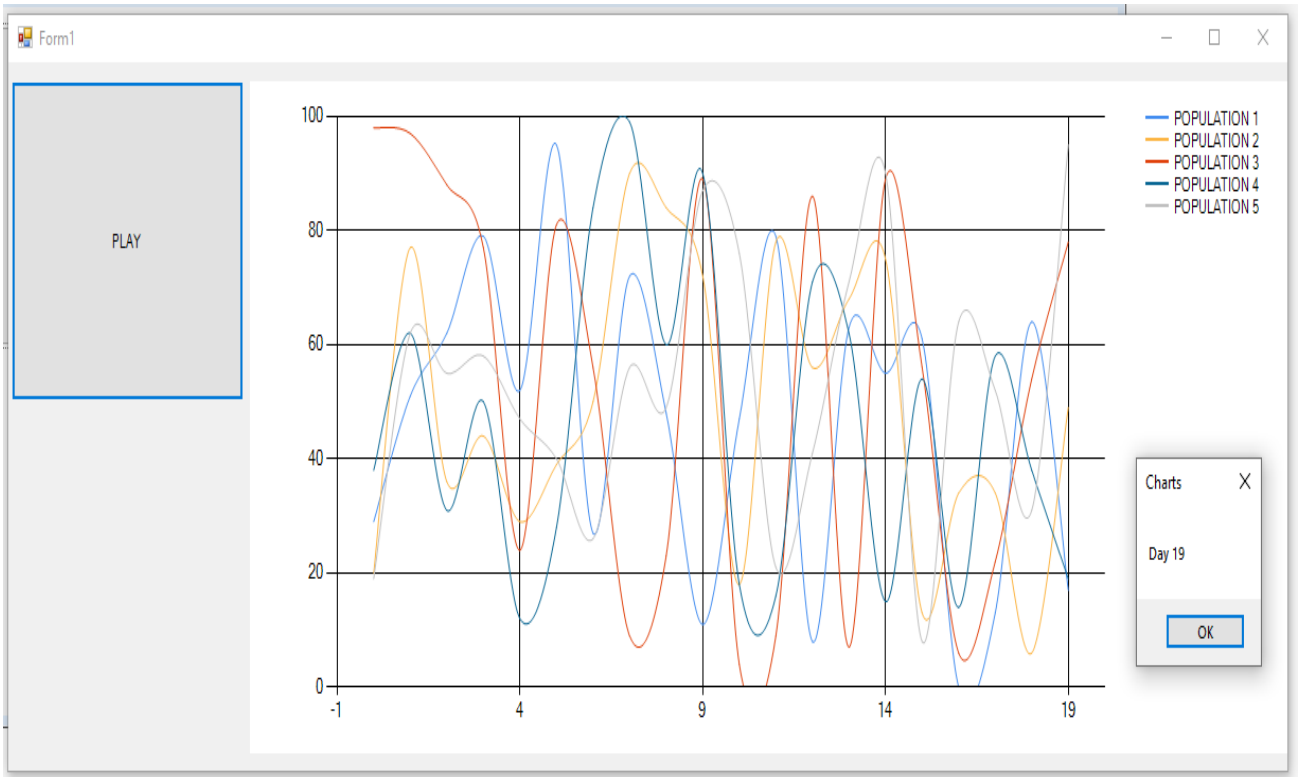
Day 0: With no data



Day 3: After three sets of random data



Day 19: After 19 sets of random data



As you can see the lines change shape as each new day passes. This would help with my population graph as different populations increase, decrease or fluctuate as the simulation runs.

3. Retrieve data from Microsoft Access:

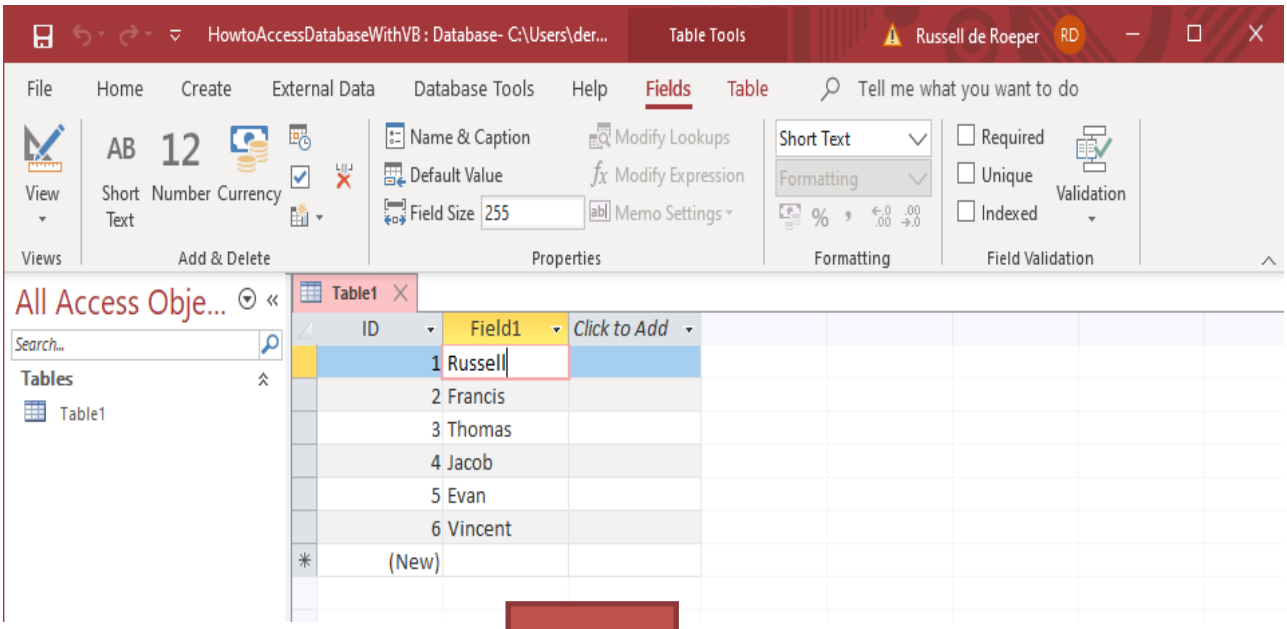
To allow different species to possess different traits, an animal database will store all the data needed to differentiate each specie. I will be using a vb library that can select data from an Access database file. This is suitable as all school computers have Microsoft Access installed.

```
Imports System.Data.OleDb

Public Class Database

    Private connectionString As String

    Private connection As OleDbConnection

    Sub New()

        connectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=N:\HowtoAccessDatabaseWithVB.accdb"

        connection = New OleDbConnection(connectionString)

    End Sub

    Function selectSQL() As DataTable

        Dim selectCommand As String = "SELECT * FROM [Table1]"

        Dim reader As OleDbDataReader

        Dim command As New OleDbCommand(selectCommand, connection)

        Dim dataTable As New DataTable("")

        connection.Open()

        reader = command.ExecuteReader()

        dataTable.Load(reader)

        Return dataTable
```

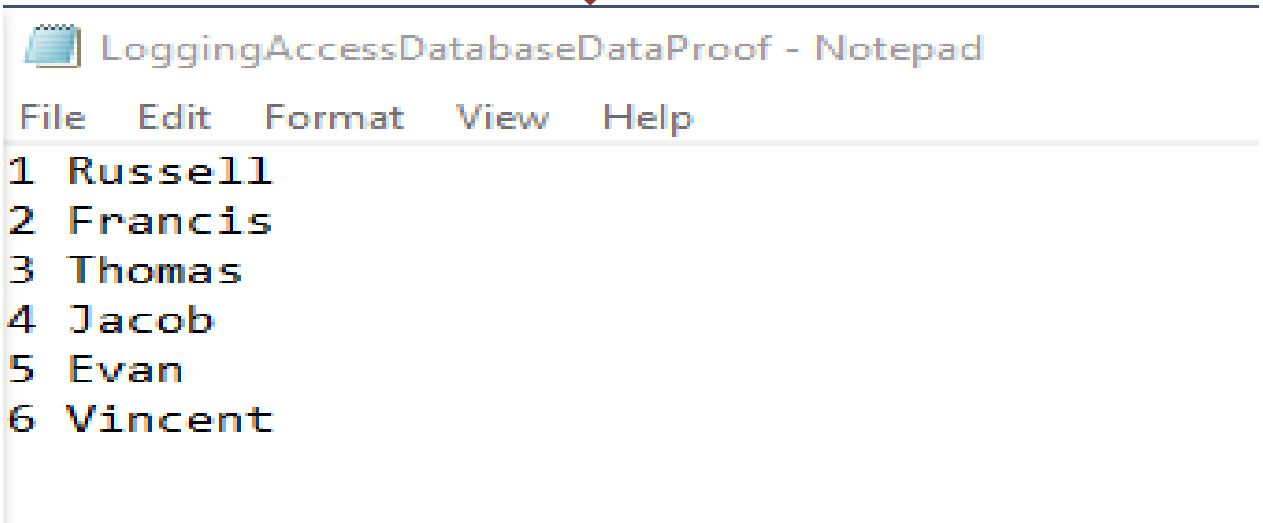
Code that selects all data from a Microsoft access database table. In order to work we need to first distinguish where the file is and then select what data you want retrieve, in this case all data from file "HowtoAccessDatabaseWith VB.accdb" and from table "Table1".

3. Proof of Concept:

To demonstrate how the code retrieves the data I made the database as described earlier and logged the information into a text file using a vb library called “streamwriter”. This vb library will also be used to log simulation data for my final program.



Retrieve data from access database and export as a “.txt” file.



Algorithms Represented by pseudocode

How Animals decide and move into a new position on the grid-world:

```
Function GetMove()

    Randomise()

    X = random integer between 1 and 10

    Select Case X

        Case 1 return "N"

        Case 2 return "NE"

        Case 3 return "E"

        Case 4 return "SE"

        Case 5 return "S"

        Case 6 return "SW"

        Case 7 return "W"

        Case 8 return "NW"

        Case Else return "No movement"

    End Select

End Function
```

Function GetMove()

The integer x is randomised each time the function is called, this means that every animal would make moves based on probability or randomness giving the animals a sense of autonomy.

```
Function ReturnCoordinate()

    Direction = Animal.GetMove()

    Select Case Direction

        Case "N"
            coordinate(0) = x
            coordinate(1) = y - 1
        Case "NE"
            coordinate(0) = x + 1
            coordinate(1) = y - 1
        Case "E"
            coordinate(0) = x + 1
            coordinate(1) = y
        Case "SE"
            coordinate(0) = x + 1
            coordinate(1) = y + 1
        Case "S"
            coordinate(0) = x
            coordinate(1) = y + 1
        Case "SW"
            coordinate(0) = x - 1
            coordinate(1) = y + 1


        Case "W"
            coordinate(0) = x - 1
            coordinate(1) = y
        Case "NW"
            coordinate(0) = x - 1
            coordinate(1) = y - 1
        Case "No movement"
            coordinate(0) = x
            coordinate(1) = y

        return coordinate(2)


    End Select

End Function
```

Function ReturnCoordinate ()

The returned value of function GetMove() will pass on to another select case which sorts new "x" "y" coordinates. You may have noticed that going north seems like going south and vice versa. This is because the origin of the grid-world is set at the top left of the grid and the x, y - axis increase towards the bottom right.

How an animals age and hunger level changes per simulation day:

```
Sub AnimalDayStatus()

        Age = age -1

        If age = 0 Then

                isDead = TRUE

        End If

        Hungerlevel = Hungerlevel -1

        If hungerlevel = 0 Then

                isDead = TRUE

        ELSE IF AnimalEatenFood = TRUE Then

                Hungerlevel = Hungerlevel + 30

        End If

End Sub
```

This pseudocode shows how the age and hunger level values decrease by 1 every day within the simulation. This is because the initial lifespan (at day 0) of every animal on the board is already at the maximum level. It then decreases from its set value to 0 where the animal dies. If a predator eats within this time the hunger level replenishes by 30, which represents that the predator can live on an extra 30 simulation days after eating.

Adding an animal to the queue linked list:

```
CLASS Node
        Piece as CLASS Animal

        X as INTEGER

        Y as INTEGER

        NextNode as CLASS Node

END CLASS
```

This class defines the properties of what every node within a linked list can possess. The contents of each node is the positions of animals on the grid-world.

```
Sub AddAnimalToQueue( NewAnimal as CLASS Animal, X, Y)
        Temp as new CLASS Node
        Temp.Piece = NewAnimal
        Temp.X = x
        Temp.Y = y
        If HeadNode = NOTHING then
            Current = Temp
            HeadNode = Temp
            TailNode = Temp
        ELSE
                TailNode.NextNode = Temp
                TailNode = Temp
        End If
        Populationnumber = PopulationNumber + 1
End Sub
```

For adding a new animal to the linked list the subroutine need the new animal object and coordinates that would be given to object from other functions. If the linked list has no nodes, the new node will be the current, head and tail node as it is first in the list. However if there is already content in the linked list it becomes the new tail as it came in last. The population number increases as the active number of animals on the board rises. This concept is important as it is the same process for how new borns are added from birth, and not only the initial population.

Deleting an animal from the linked list:

```
SUB DeleteAnimalFromQueue( x, y)
        Temp as CLASS Node
        Previous as CLASS Node
        isAnimalDeleted = False
        Temp = head
        Previous = head
```

This subroutine is needed to delete an animal off the simulation by first removing it off the linked list. The only inputs it needs are the “x” “y” coordinates of the animal that needs to be removed.

```
        IF Temp.X = x AND Temp.Y = y Then
                head = head.NextNode
                isAnimalDeleted = TRUE
        DO
                IF temp.NextNode = NOTHING Then
                        Previous.NextNode = NOTHING
                        IsAnimalDeleted = TRUE
                ELSE
                        IF   temp.NextNode.X = x AND temp.NextNode.Y = y Then
                                IF   temp.NextNode.NextNode != NOTHING Then
                                        temp.NextNode = temp.NextNode.NextNode
                                        isAnimalDeleted = TRUE
                                ELSE
                                        temp.NextNode = NOTHING
                                        isAnimalDeleted = TRUE
                                END IF
                        ELSE
                                Previous = temp
                                temp = temp.NextNode
                        END IF
                END IF
        LOOP UNTIL isAnimalDeleted = TRUE
        PopulationNumber = PopulationNumber – 1
END SUB
```

This if statement deletes the animal if the x and y coordinates match with the x and y coordinates of the head node. The node next to previous head becomes the new head node.

This if statement checks whether a node intended to be deleted is the tail by checking if the next node equals nothing. If so the intended node is the tail node in the list. This tail node is then removed.

This if statement deletes the animal if the x and y coordinates match with the x and y coordinates of the given parameters within the linked list. It is then removed. Otherwise the temp nodes content is deleted.

Most of the subroutine is placed within a loop so that it will keep searching till the intended node is found. When the node is removed; the population number decreases by 1.

Draw the Simulator Grid and Animals:

```
Rectangle as New RectangleF(0, 0, 500, 500)

Select Case Habitat

        Case "Desert"

                BrushColour = "LightYellow"

                FillRectangle(BrushColour, rectangle)

        Case "Greenland"

                BrushColour = "Green"

                FillRectangle(BrushColour, rectangle)

        Case "Arctic"

                BrushColour = "LightBlue"

                FillRectangle(BrushColour, rectangle)

        Case Else

                OUTPUT ("Error: Can't Find Habitat type.)

DO

        Counter = counter + 1

        IF linkedlist.current != Nothing Then

                Select Case LinkedList.current.AnimalName

                        Case AnimalONE

                                BrushColour = "Blue"

                        Case AnimalTWO
```

VB.Net has a drawing tool called brush that can make many shapes including the rectangle. The rectangle dimensions is first inputted for the habitat board size (500x500).

For each new simulation a different habitat can be asked for and depending on the type the colour of the grid world would change as shown to the left.

A loop contains the drawing of the animals as there is no constant population size for any simulation. It will run through the added animals of the linked list, drawing them one by one until the list is finished. On the other page, there are 5 populations as the options menu allows five different species in the simulation.

```
                    BrushColour = "Yellow"

                Case AnimalTHREE

                    BrushColour = "Red"

                Case AnimalFOUR

                    BrushColour = "DarkGreen"

                Case AnimalFIVE

                    BrushColour = "Purple"

            Rectangle = LinkedList.current.X * 10, LinkedList.current.Y * 10, 10, 10)

            FillRectangle(BrushColour, Rectangle)

        END IF

LOOP UNTIL LinkedList.NextNode = False
```

Testing

Testing will be used to ensure my program is at an appropriate standard to be used for students and teachers. I will be using black-box texting and white-box testing to do this. Black-box testing is testing the functionality of the program without looking into its code, so for my case testing the user experience. White-box testing is testing the internal structure of the code by giving certain inputs and expecting certain outputs to happen.

Validation

No.	Validation type	Explanation
1	Try and Catch	A try and catch is an error-handling technique that tries an outcome and if it fails; another outcome will occur to

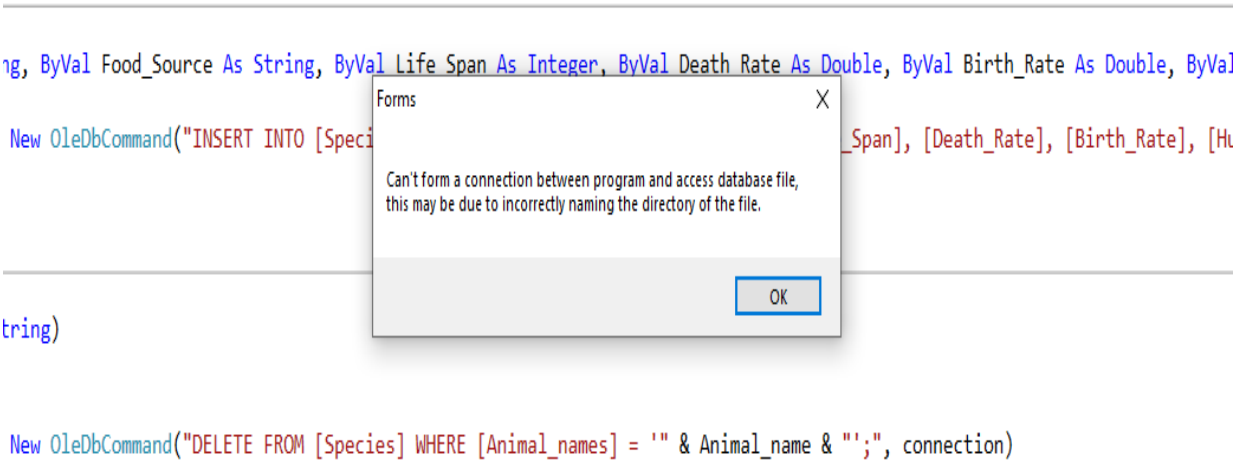
		get rid of the error.
2	Lookup	A lookup validation is selecting options that are given to the user, so it removes any other unknown input. E.g a drop-down menu.
3	Range Check	A range check is used to validate certain values in a given range that are allowed to be used as inputs.
4	Presence Check	Presence check is used to validate whether a value is entered and not left blank.

Try and Catch:

Here is an example of when I used try and catch to prevent the program crashing in case the directory of the database is not found.

```
Function selectAnimal() As DataTable
    Dim selectCommand As String = "Select * FROM [Species]"
    Dim reader As OleDbDataReader
    Dim command As New OleDbCommand(selectCommand, connection)
    Dim found As Boolean = False
    Dim dataTable As New DataTable("")

    Try 'See if its possible to open database file
        connection.Open()
        reader = command.ExecuteReader()
        dataTable.Load(reader)
        Return dataTable
    Catch 'Error handling when database file cannot be accessed.
        MsgBox("Can't form a connection between program and access database file, this may be due to incorrectly naming the directory of the file."
        Return Nothing
    End Try

End Function
```

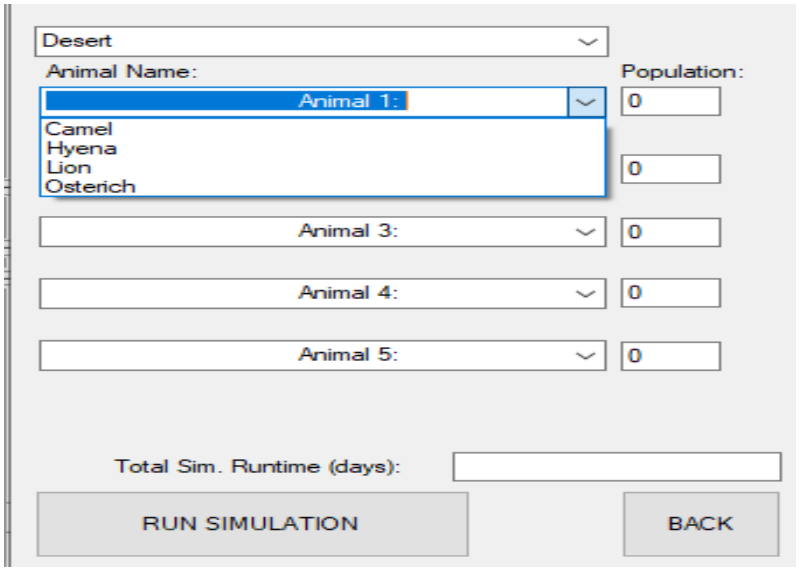
Result when the directory of the file cannot be found.



Lookup:

```
Sub ListAnimalNames(ByVal terrain As String)

    'Animal names listed under terrain type

    Dim ListofAnimal As New cl_Database
    Dim AnimalNamesdataTable As New DataTable

    If terrain = "" Then
        AnimalNamesdataTable = ListofAnimal.selectAnimal()
        If AnimalNamesdataTable.Rows.Count > 0 Then
            For x = 0 To AnimalNamesdataTable.Rows.Count - 1
                CBanimalONEpop.Items.Add(AnimalNamesdataTable.Rows(x).Item(0))
                CBanimalTWOpop.Items.Add(AnimalNamesdataTable.Rows(x).Item(0))
                CBanimalTHREEpop.Items.Add(AnimalNamesdataTable.Rows(x).Item(0))
                CBanimalFOURpop.Items.Add(AnimalNamesdataTable.Rows(x).Item(0))
                CBanimalFIVEpop.Items.Add(AnimalNamesdataTable.Rows(x).Item(0))

            Next
        End If

    Else
        AnimalNamesdataTable = ListofAnimal.selectAnimalTerrain(terrain)
        If AnimalNamesdataTable.Rows.Count > 0 Then
            For x = 0 To AnimalNamesdataTable.Rows.Count - 1
                CBanimalONEpop.Items.Add(AnimalNamesdataTable.Rows(x).Item(4))
                CBanimalTWOpop.Items.Add(AnimalNamesdataTable.Rows(x).Item(4))
                CBanimalTHREEpop.Items.Add(AnimalNamesdataTable.Rows(x).Item(4))
                CBanimalFOURpop.Items.Add(AnimalNamesdataTable.Rows(x).Item(4))
                CBanimalFIVEpop.Items.Add(AnimalNamesdataTable.Rows(x).Item(4))

            Next
        End If

    End If


End Sub
```

This is the code that arranges animals by habitat and produces a list for them to be selected by the user.



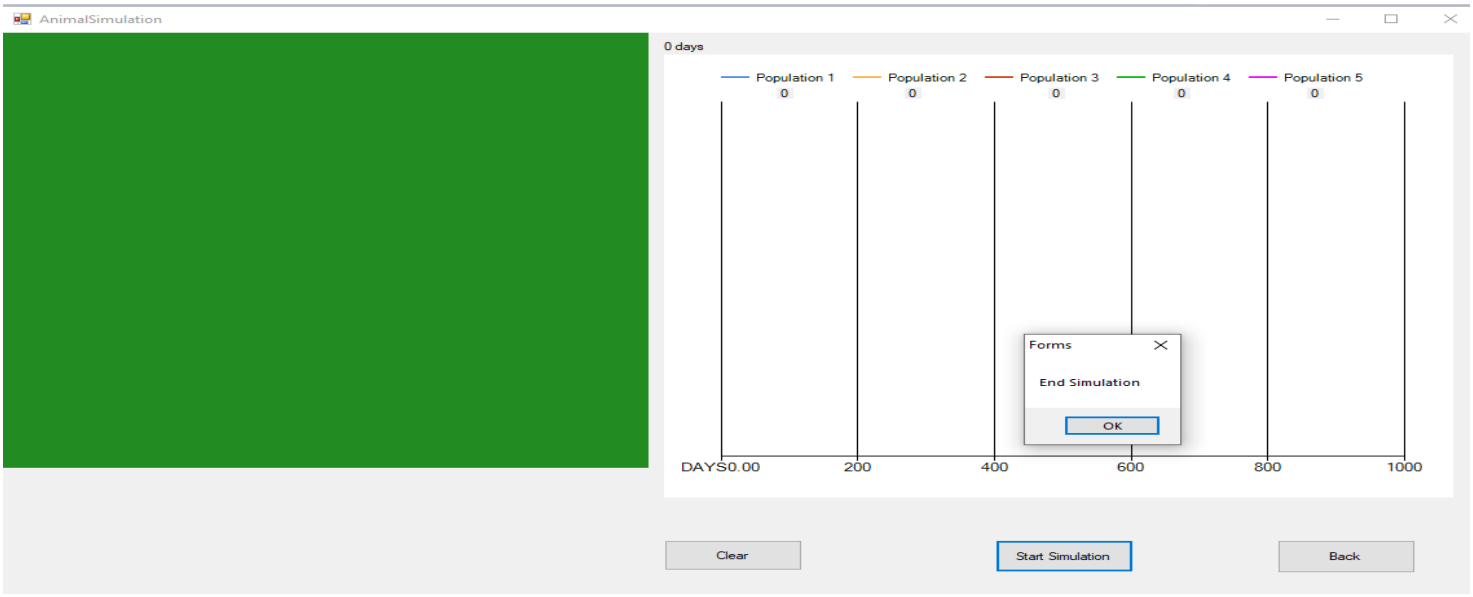
This is the result of the above code.

As you can see only the animals that are known to live in the desert are shown.

Range Check:

```
cl_GlobalSetting.Board = New cl_Board(CBterrainType.Text) 'Setup board from terraintyp
information in options

        'Setup population sizes and animal specie from options

        If TBpopulationONE.Text > 0 Then
            cl_GlobalSetting.Board.addAnimal(Animaldata(CBanimalONEpop.Text),
TBpopulationONE.Text)
            cl_GlobalSetting.AnimalONEpopulation = TBpopulationONE.Text
            cl_GlobalSetting.AnimalONE = CBanimalONEpop.Text
        End If

        If TBpopulationTWO.Text > 0 Then
            cl_GlobalSetting.Board.addAnimal(Animaldata(CBanimalTWOpop.Text),
TBpopulationTWO.Text)
            cl_GlobalSetting.AnimalTWOpopulation = TBpopulationTWO.Text
            cl_GlobalSetting.AnimalTWO = CBanimalTWOpop.Text
        End If

        If TBpopulationTHREE.Text > 0 Then
            cl_GlobalSetting.Board.addAnimal(Animaldata(CBanimalTHREEpop.Text),
TBpopulationTHREE.Text)
            cl_GlobalSetting.AnimalTHREEpopulation = TBpopulationTHREE.Text
            cl_GlobalSetting.AnimalTHREE = CBanimalTHREEpop.Text
        End If

        If TBpopulationFOUR.Text > 0 Then
            cl_GlobalSetting.Board.addAnimal(Animaldata(CBanimalFOURpop.Text),
TBpopulationFOUR.Text)
            cl_GlobalSetting.AnimalFOURpopulation = TBpopulationFOUR.Text
            cl_GlobalSetting.AnimalFOUR = CBanimalFOURpop.Text
        End If

        If TBpopulationFIVE.Text > 0 Then
            cl_GlobalSetting.Board.addAnimal(Animaldata(CBanimalFIVEpop.Text),
TBpopulationFIVE.Text)
            cl_GlobalSetting.AnimalFIVEpopulation = TBpopulationFIVE.Text
            cl_GlobalSetting.AnimalFIVE = CBanimalFIVEpop.Text
        End If

       End Sub
```

When inputting -1 as a population size to the program, the simulation would produce no animal on the board and end automatically.



Presence Check:

```
Private Sub FM_Options_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

    Call ListAnimalNames("")
    Call ListHabitatNames()

    'CBanimalONEpop.Text = "EMPTY"
    'CBanimalTWOpop.Text = "EMPTY"
    'CBanimalTHREEpop.Text = "EMPTY"
    'CBanimalFOURpop.Text = "EMPTY"
    'CBanimalFIVEpop.Text = "EMPTY"

    TBpopulationONE.Text = 0
    TBpopulationTWO.Text = 0
    TBpopulationTHREE.Text = 0
    TBpopulationFOUR.Text = 0
    TBpopulationFIVE.Text = 0

End Sub
```

By setting the population sizes to 0 initially, there is no need for the user to input a value in each text box. This essentially by-passes the presence check, and allows the simulation to work with less than 5 populations.

Data Structures

No.	Data Structure	Explanation
1	Arrays	A static data structure which is a collection of items that share the same data type. It can work on multiple or single dimensions.

2	Class Objects	Are instances of a class, meaning there is an “entity” which contains all the properties and methods that a class has.
3	Linked Lists	A dynamic data structure that contains a lists of items which has a determined ordering (stack or queue).
4	Text File	A file that contains human-readable characters.

Example of using an array:

```
Function returnCor(ByVal direction As String, ByVal x As Integer, ByVal y As Integer) As Integer()
    Dim coordinate(2) As Integer

    Select Case direction
        Case "N"
            coordinate(0) = x
            coordinate(1) = y - 1


        Case "NE"
            coordinate(0) = x + 1
            coordinate(1) = y - 1



        Case "E"
            coordinate(0) = x + 1
            coordinate(1) = y


        Case "SE"
            coordinate(0) = x + 1
            coordinate(1) = y + 1



        Case "S"
          coordinate(0) = x
            coordinate(1) = y + 1
```

Here I have used an array called coordinate of size 2 to store the new x and y coordinates for an animal to move too.

Example of using a class object:

```
Public Overrides Function GiveBirth() As cl_Animal
    'Dim baby As New cl_Carnivore
    Dim birthChance As Decimal
    Dim Baby

    Randomize()

    birthChance = Rnd() 'Within 0-1
    'MsgBox(birthChance)

    If birthChance <= Birth_Rate Then
        '   MsgBox("Got Baby- " & Me.AnimalName & " " & birthChance)
        Baby = New cl_Carnivore(Me.AnimalName, Me.LifeSpan, Me.DeathRate, Me.Birth_Rate, Me.HungerMeter, Me.AnimalSpeed, Me.FoodChainRank)



        Return Baby
    Else
        Return Nothing

    End If

End Function
```

Here I have instantiated the class cl_Carnivore under the name Baby. In this section of the code whenever a carnivore gives birth they essentially make a copy of themselves. Furthermore as cl_carnivore is a sub-class of cl_animal they obtain all the properties and methods like movement, hunger etc.

Example of using a linked list:

```
Sub addQueue(ByVal thisAnimal As cl_Animal, ByVal X As Integer, ByVal Y As Integer)
    Dim temp As New cl_Node
    temp.piece = thisAnimal
    temp.X = X
    temp.Y = Y
    If IsNothing(head) =
        True Then
        current = temp
        head = temp
        tail = temp

    Else
        tail.nextNode = temp
        tail = temp

    End If
    populationNum = populationNum + 1
End Sub
```

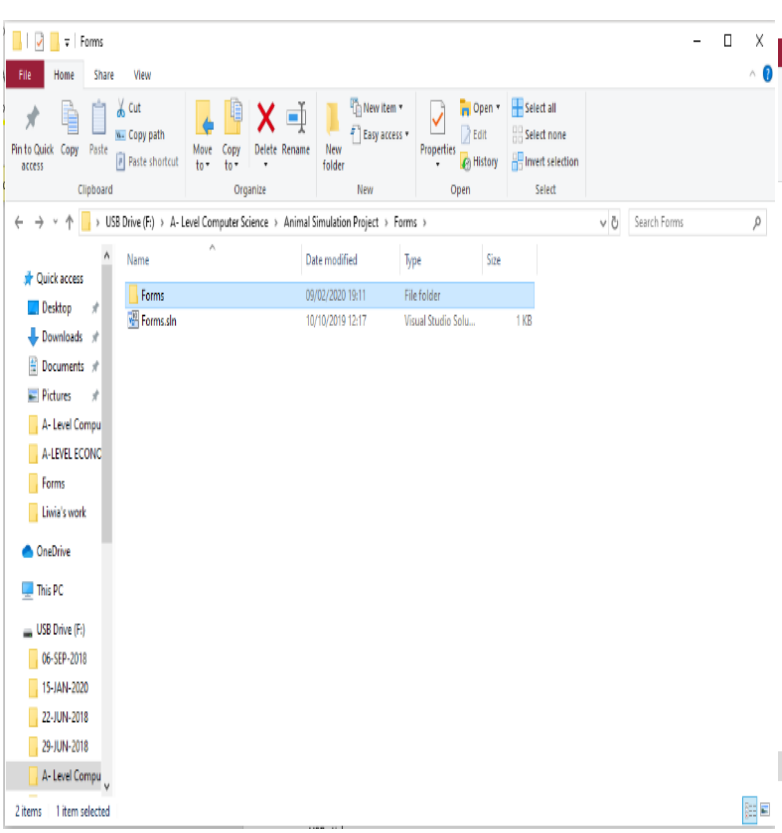
```
Sub delete(ByVal x As Integer, ByVal y As Integer) 'FLOWCHART FOR DOCUMENT ON HOW WE DELETED AN ANIMAL AFTER BEING EATEN
    Dim temp As cl_Node
    Dim isDeleted As Boolean
    Dim previous As cl_Node
    isDeleted = False
    temp = head
    previous = head

    If temp.X = x And temp.Y = y Then 'deletes head of node
        cl_GlobalSetting.log.write("ERROR 1.1")
        head = head.nextNode
        isDeleted = True
    End If

    Do
        If IsNothing(temp.nextNode) = True Then
            cl_GlobalSetting.log.write("ERROR 1.2")
            previous.nextNode = Nothing
            isDeleted = True
        Else
            If temp.nextNode.X = x And temp.nextNode.Y = y Then 'deletes a node within the body of the node
                cl_GlobalSetting.log.write("ERROR 1.3")
                If IsNothing(temp.nextNode.nextNode) = False Then
                    cl_GlobalSetting.log.write("ERROR 1.4")
                    temp.nextNode = temp.nextNode.nextNode
                    isDeleted = True
                Else
                    cl_GlobalSetting.log.write("ERROR 1.5")
                    temp.nextNode = Nothing
                    isDeleted = True
                End If
            Else
                cl_GlobalSetting.log.write("ERROR 1.6")
                previous = temp
                temp = temp.nextNode
            End If

        End If
        If isDeleted = True Then
            ' cl_GlobalSetting.log.write("Ends loop")
        End If
    Loop Until isDeleted = True
    populationNum = populationNum - 1
End Sub
```

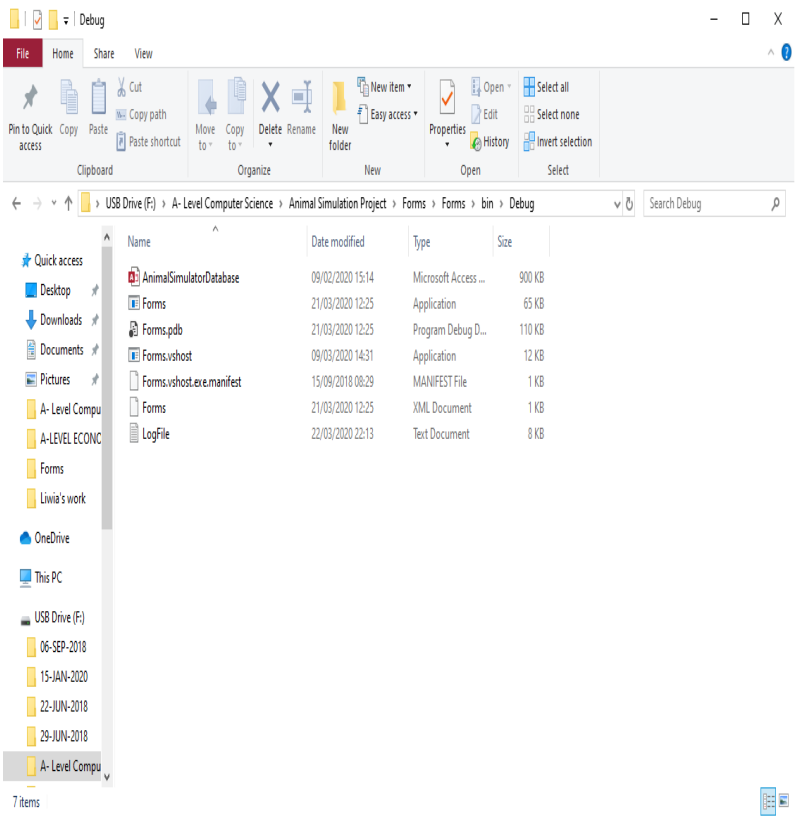
For my program I used a queue structure instead of a stack, due to its first in first out nature. This is important as it organises the turns for each animal to make a move in an organised rotation. Furthermore as it is a dynamic data structure, using the linked list was very efficient as dead animals were removed from the lists which reduced the number of calculations to be processed. The above code is used for adding animals to the simulation and adding new-borns. The code below is used for animals that have died, so they are removed from the linked list.

The last data structure, the text file will be mentioned in detail in the file structure section below.

File Structures

As I will be developing the program at my house and at school I currently save the source code in my USB stick. The source code is currently named Forms.vb, but for the final release it will be named something more appropriate for the students to easily identify. Furthermore the database file and simulation text file is in its debug folder called “AnimalSimulatorDatabase” and “LogFile” respectively.





For the final release I need to place the “Animal Simulation Project” folder in the Geography GCSE folder at Alec Reed Academy’s server. This is so that all students and teachers can access the

program as an executable file. In order to do this I would need the files to be authorised by either the geography department teachers or technicians who have authoritative rights over the school system.

In order for my program to work at home or school the location of the database changes, so I need to exchange between these lines of code depending on where its being developed. But for the final release the second line of code will be used as it is for the school server.

```
connectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source= F:\Current forms 3.1.2020\Forms\Forms\bin\Debug/AnimalSimulatorDatabase.accdb"
```

-Home

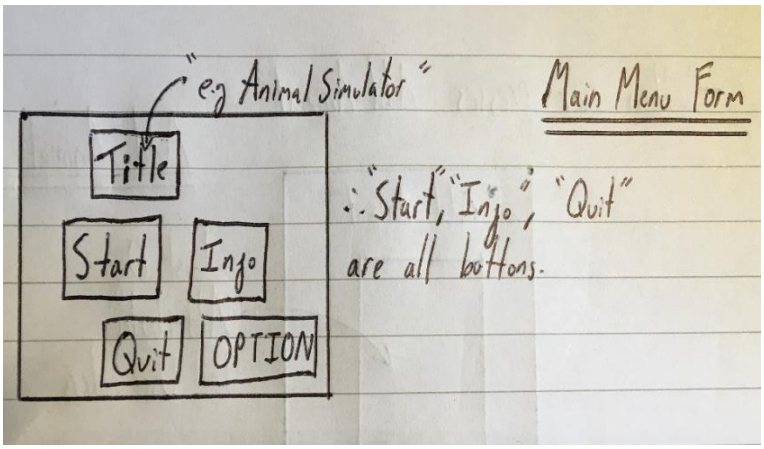
```
connectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source= N:/AnimalSimulatorDatabase.accdb"
```

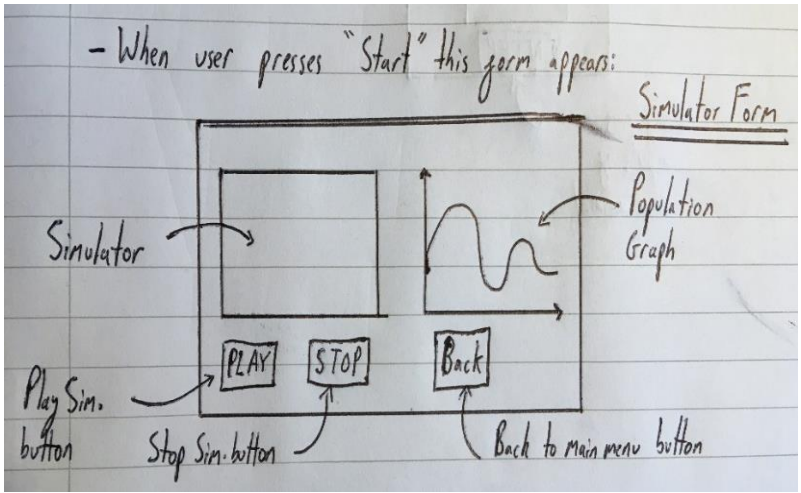
-School

Human-Computer Interactive Designs

HCI design of the Animal Simulator system draft 1:

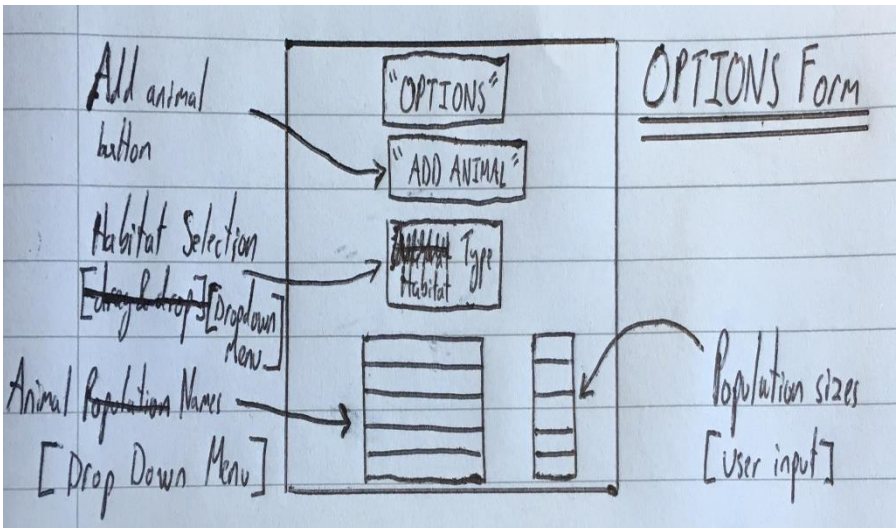
The program will begin from a main menu. It will contain buttons for the user to click on and transition too as shown to the right.

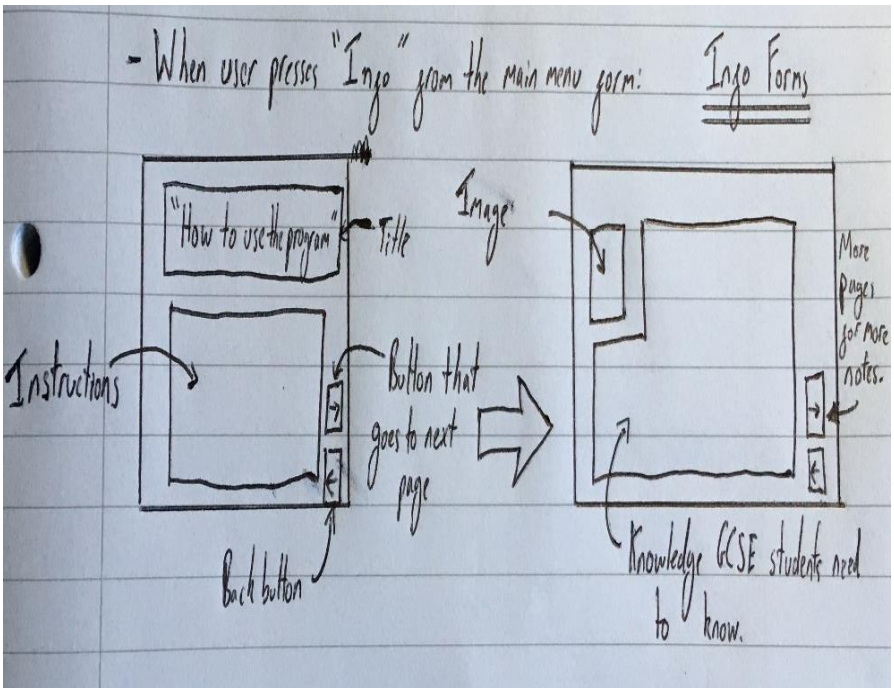




If the user clicks on the "start" button the user will see a new form which contains the simulation and line graph which records the populations.

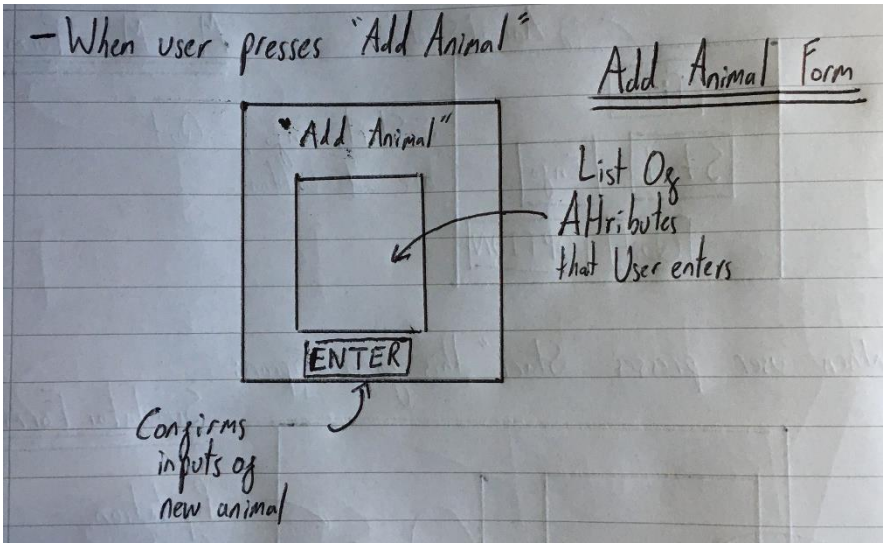
If the user selects the “options” button, they will transition to the Options form. Here the user would be able to transition to the Add Animal form via its button. In this form they are able to select the type of habitat, species and the population sizes of each for the simulation to run on later. The habitat and species are selected by the user via a drop-down menu, and the population sizes are inputted by the user.



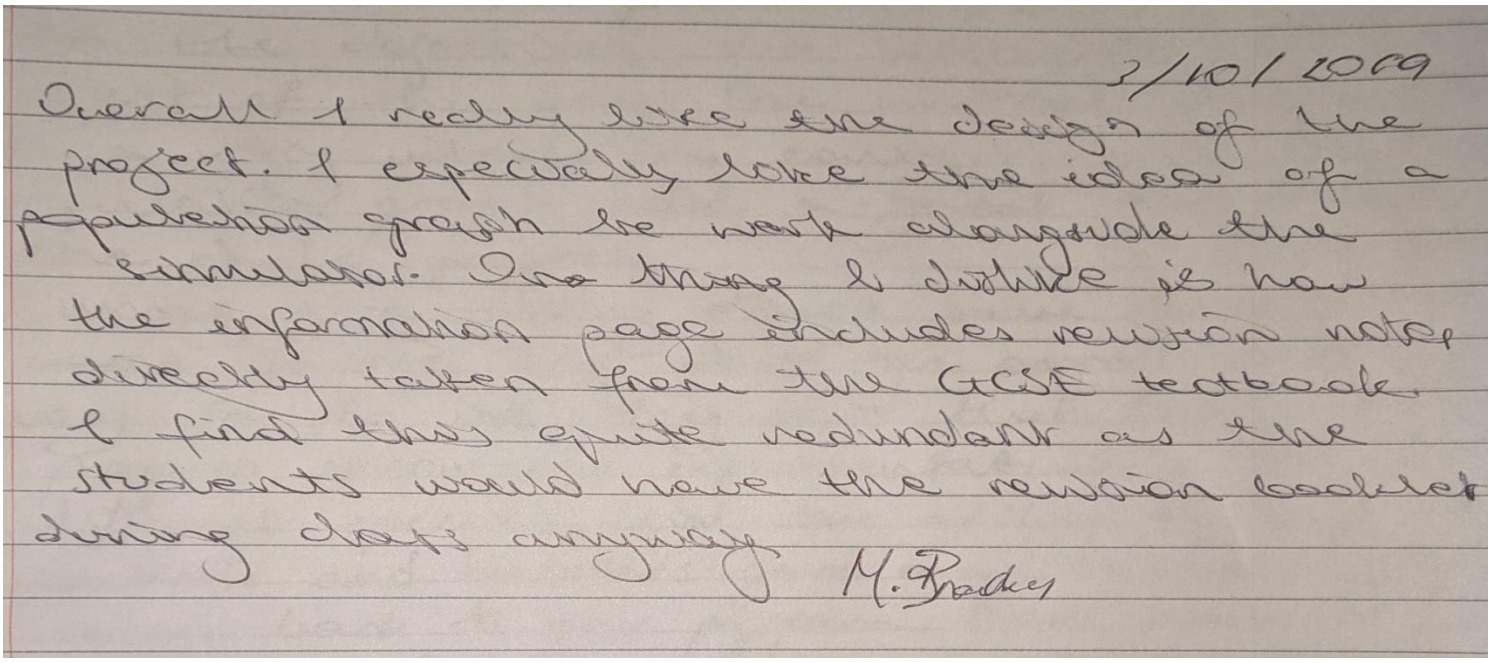


If the user selects the “Info” button, they would transition to the info form. This is an environment were the user understands how to work the simulation. Furthermore the info pages have revision notes from the GCSE geography booklet. This is for students to try out habitats and species in the simulation that they have learnt from the book, and possibly add new ones to the simulation.

Finally if the user clicks on the “Add Animal” button program transitions to the Add Animal Form. Here they can manually input all the given attributes for an animal, and then this data is sent to the animal database too be stored. This new animal can then be selected in options and be apart of a simulation.

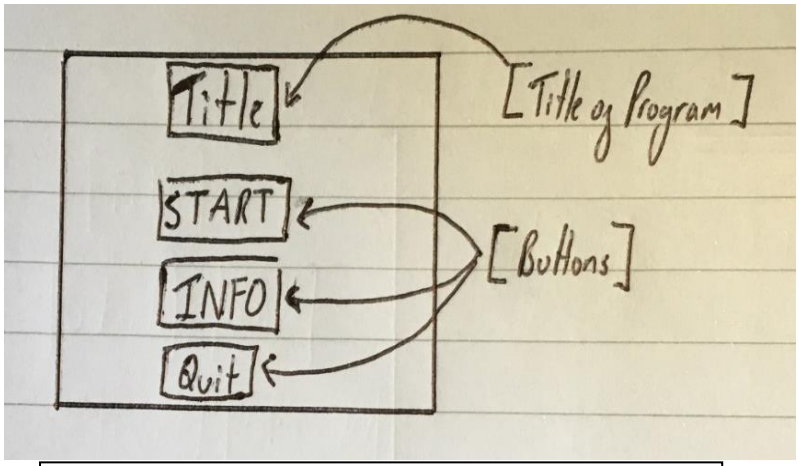


Mrs. Brookes comment on my first HCI draft.



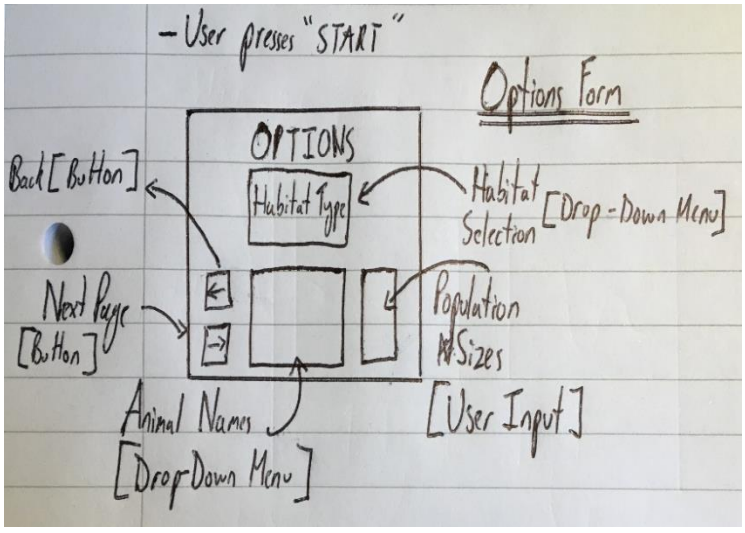
HCI design of the Animal Simulator system draft 2:

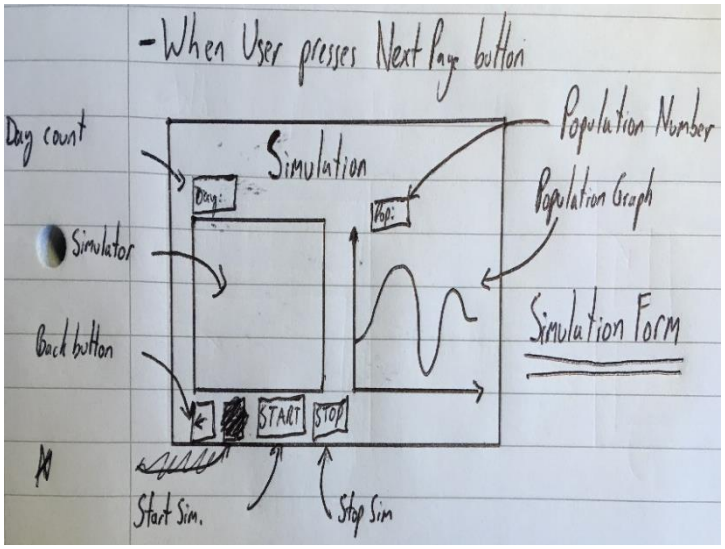
Majority of the system has been kept the same, but from this new draft I will be identifying the main changes to the users' experience of the program.



Removed the options button and placed the options form within the transition of the start button. This is because the simulation wouldn't run if there is no desired inputs for a simulation. So the options form would appear before the simulation form.

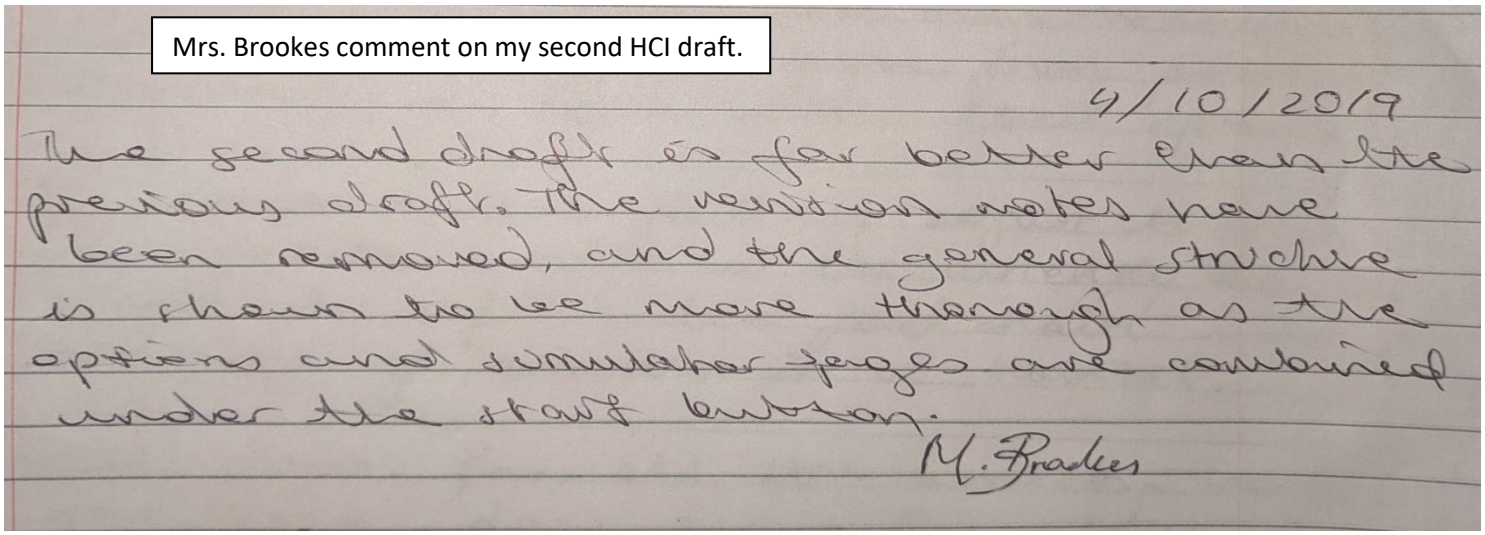
Entirely removed "Add Animal" as the user would be able to do this by accessing the database file which is given to them alongside the program. This is to save space for the program, but also allows the user to compare values of other animals for the simulation. This also allows the user to manipulate current animals attributes as well; therefore giving the users more control over the simulation.





This change was mentioned earlier, but the final transition of pressing the start button is moving to the simulation. Here the user can witness the simulation play out from their inputs of the options menu. Finally they are able to see the changes in the population via the line graph and alsobe given access to the simulation text file, which gives the user even more detail of the day-by-day activities of the simulation like eating and dying.

Mrs. Brookes comment on my second HCI draft.



Hardware Selection

These are the minimum requirements to run a visual basic application on a Windows 7 system. My program requires more space but that's no more than a MB.

Minimum:

- 1GHz or higher microprocessor
- VGA 640x480 or higher resolution screen supported by Windows
- Windows 7 or later
- 1GB RAM for Windows 7
- Disk space requirements: 100MB

As my project is being developed for use within Alec Reed Academy; the hardware required is the same as any school computer the students are given access to.

The specification of these computers are as follows:

The hardware of the systems the program is expected to run on:

- Intel i5 3.2GHz core
- Microsoft Windows 10
- 8GB RAM
- Minimum disk space requirements: 1 MB
- 1440*900 resolution monitor

My program requires more space but that's no more than a MB, which includes the Microsoft access file that will store animal data. However, as the user increases the database file size with more animal data, it can potentially be greater than a megabyte but it is unlikely.

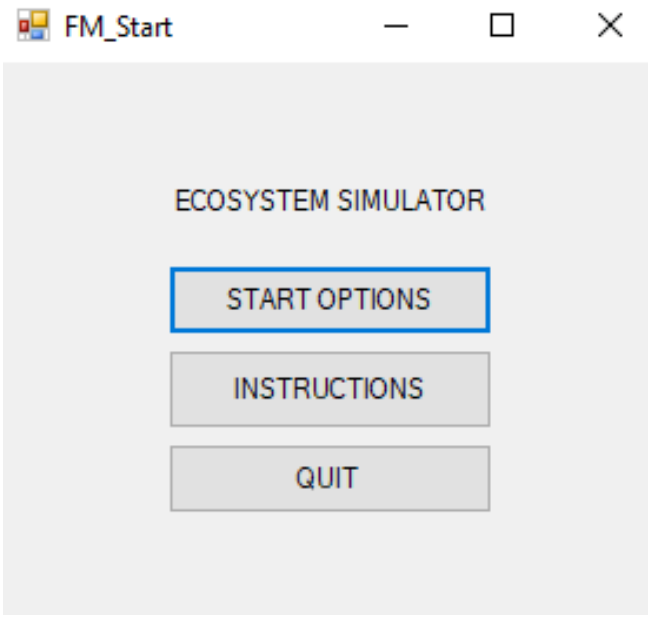
Implementation

Code with Comments

Found in the appendix section.

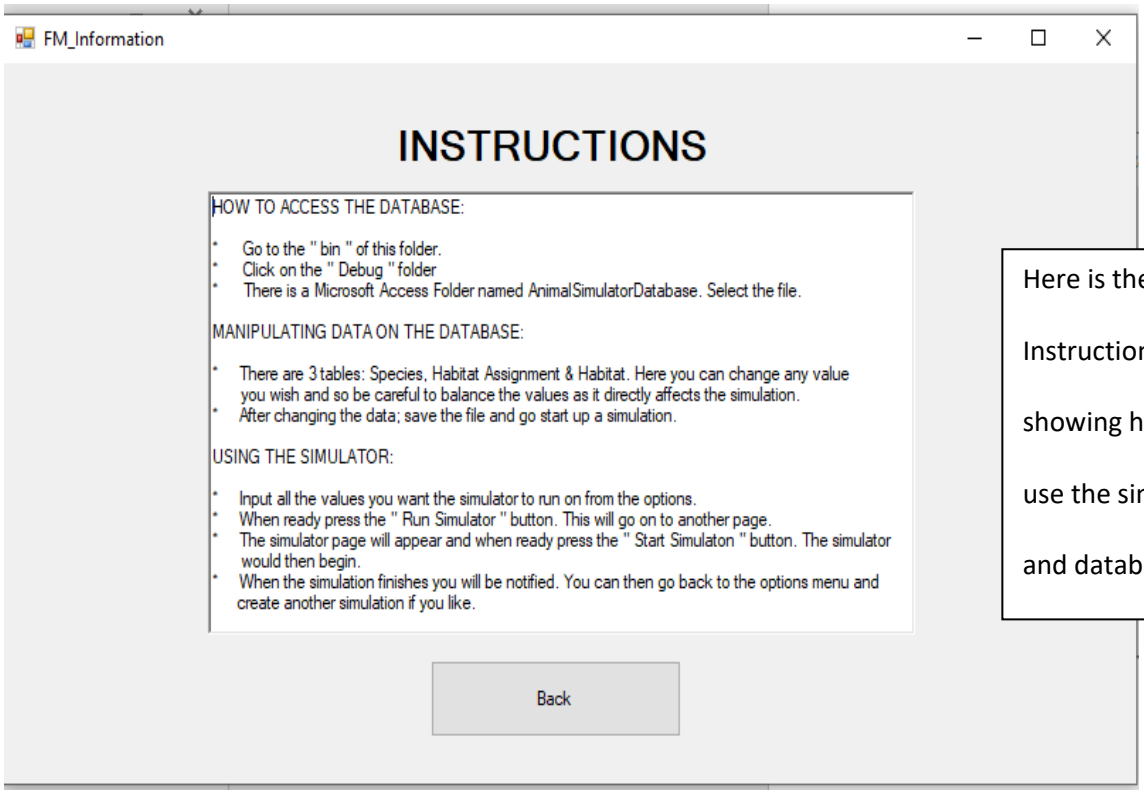
Technical Solution

My solution was built on the second draft of the Human-computer interactive designs I have made.



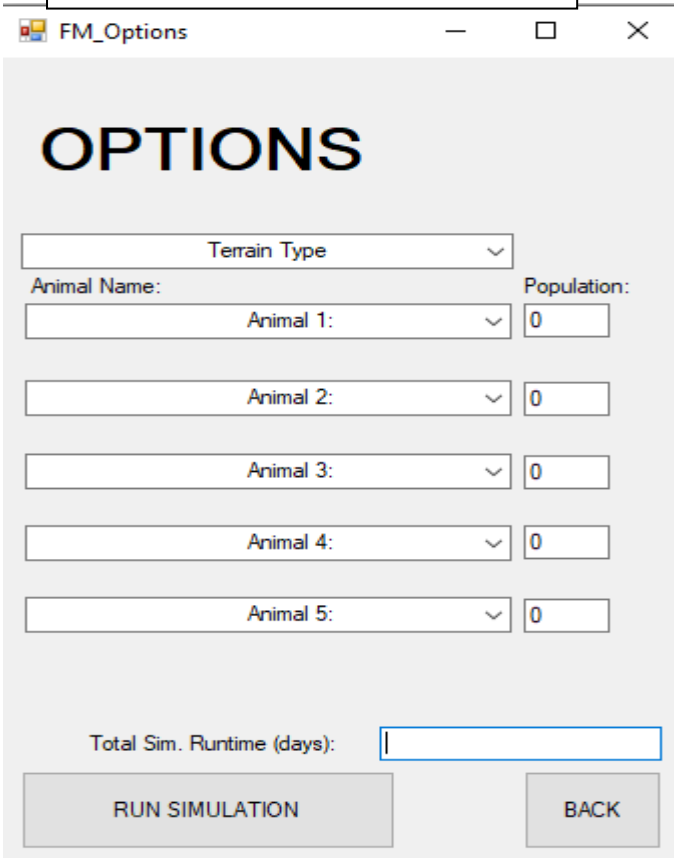
The program will begins with this main menu. It displays the title alongside contains 3 buttons for the user to go through.

NOTE: I changed the Info button to instructions as the content only contains instructions and not revision notes as mentioned in HCI.

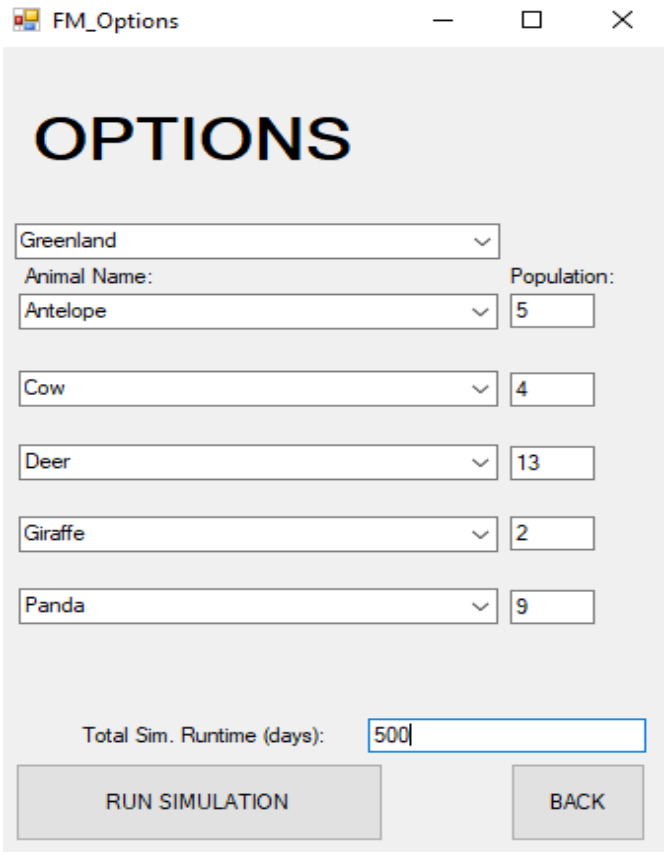


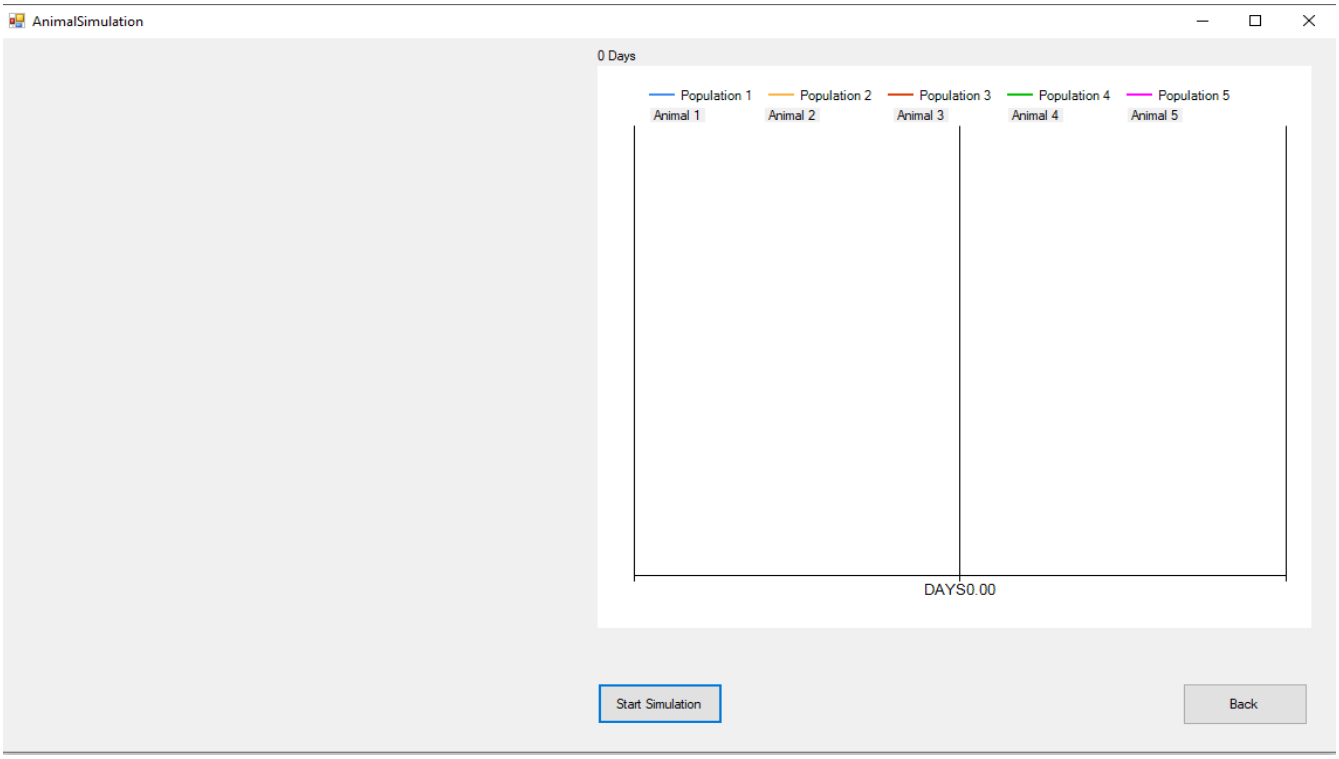
Here is the Instruction page showing how to use the simulator and database.

Here is the options menu when the user presses “start options”.

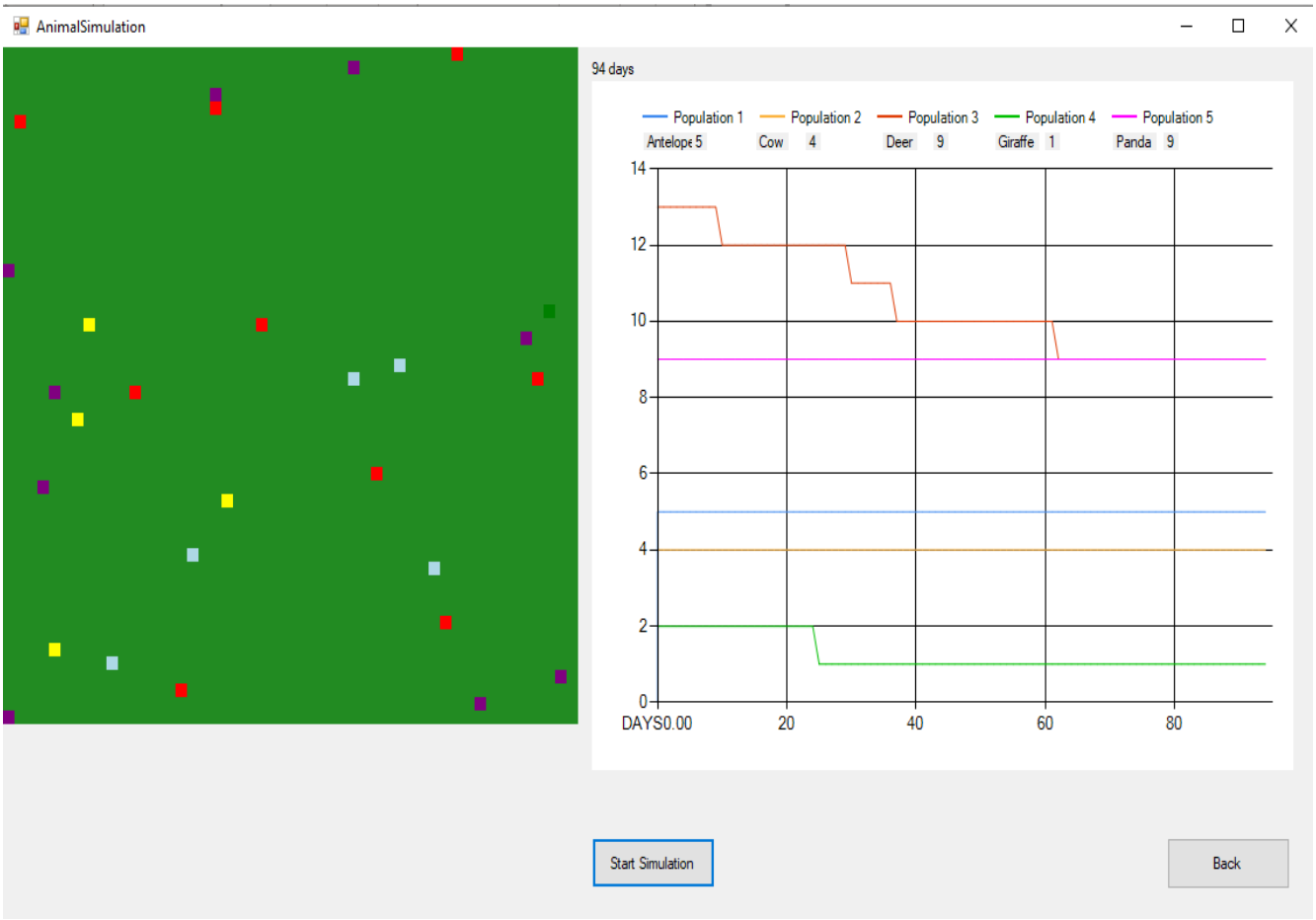


- Example of the user configuring a simulation.

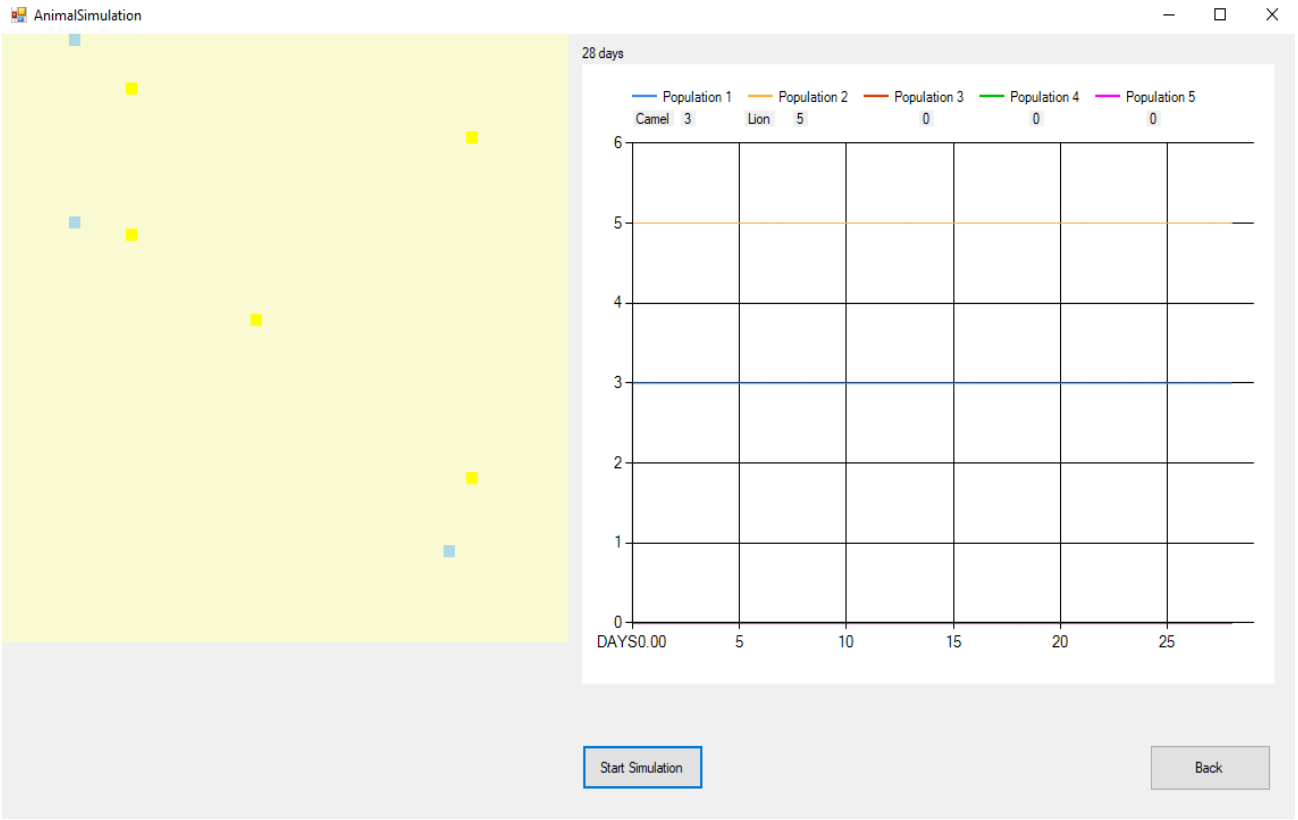




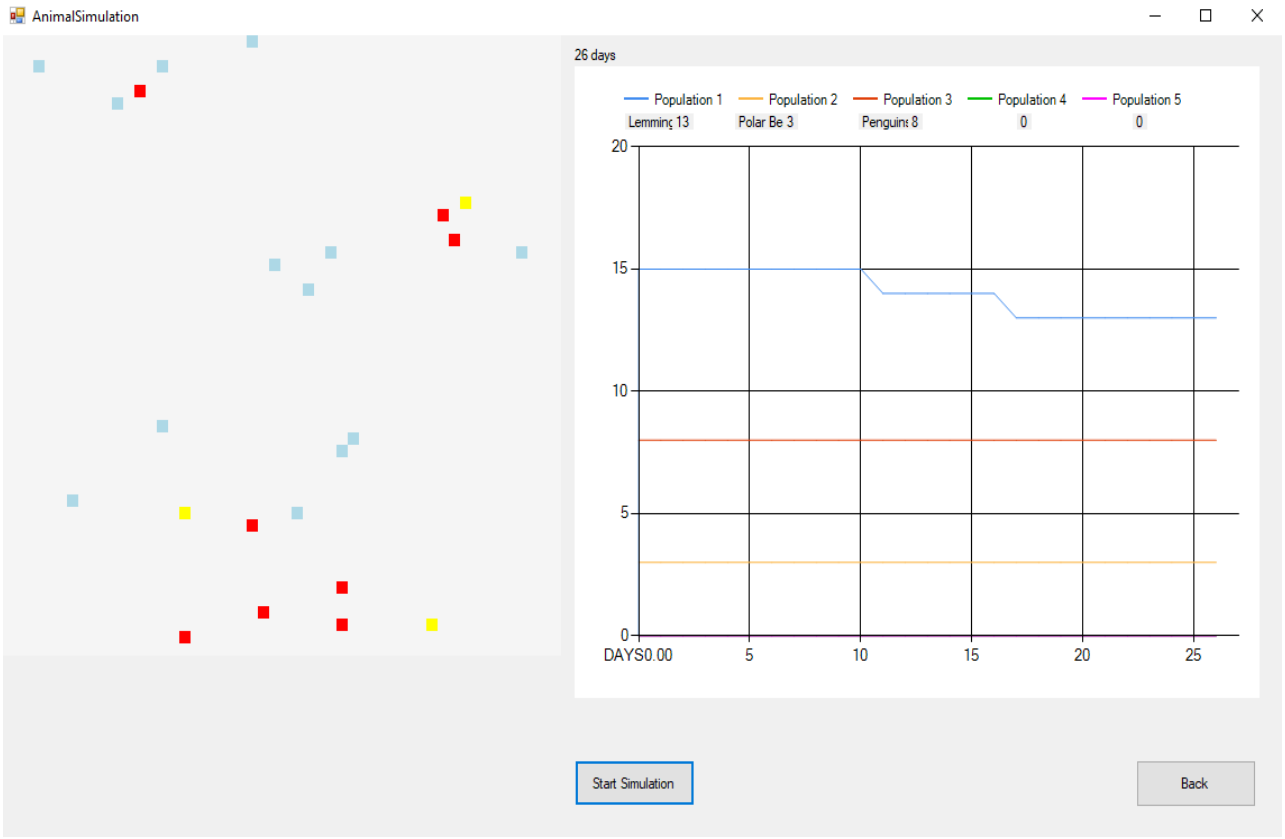
Above is the default position of the simulation form. Activity only begins when the user presses “Start Simulation”.



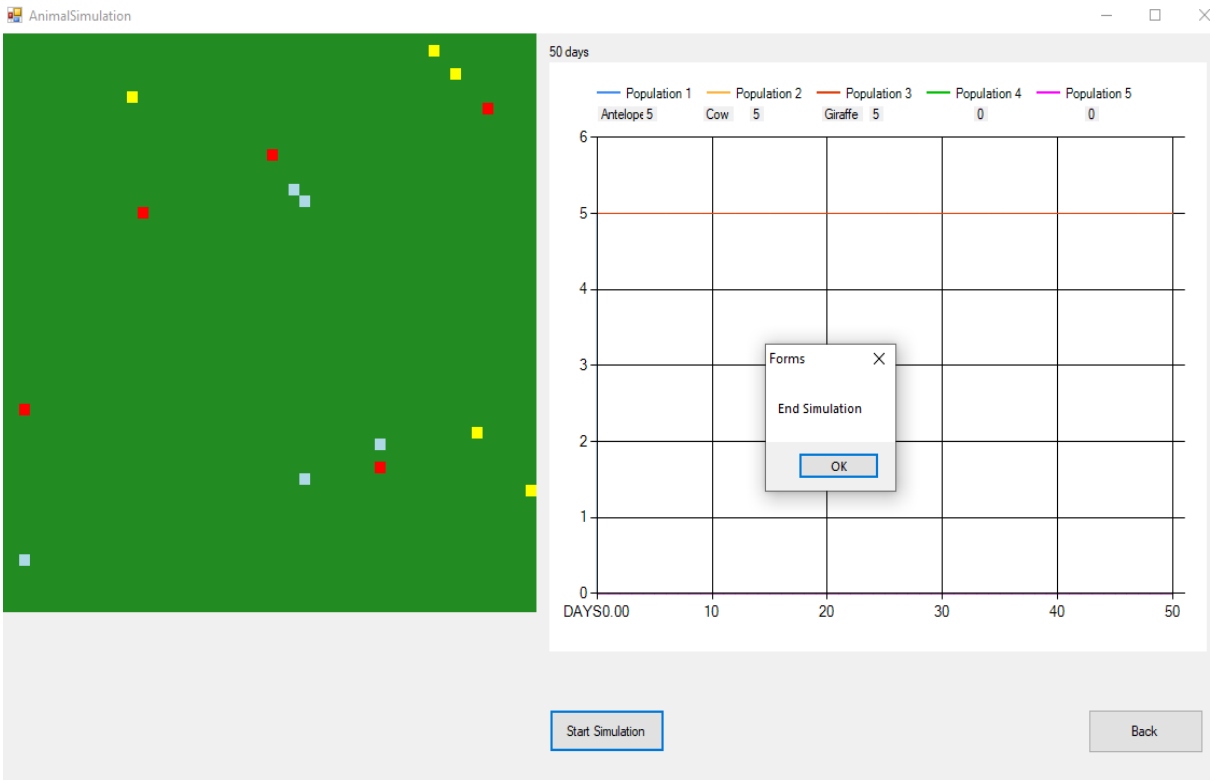
Example of a simulation set in the Greenlands. As you may have noticed, the activities in the simulation are reflected on by the population graph that tracks births and deaths over the simulation period. Furthermore, a live count of each population can be shown above the line graph, giving the user an easier interpretation of the graph and simulation.



Example of a simulation set in the Desert.



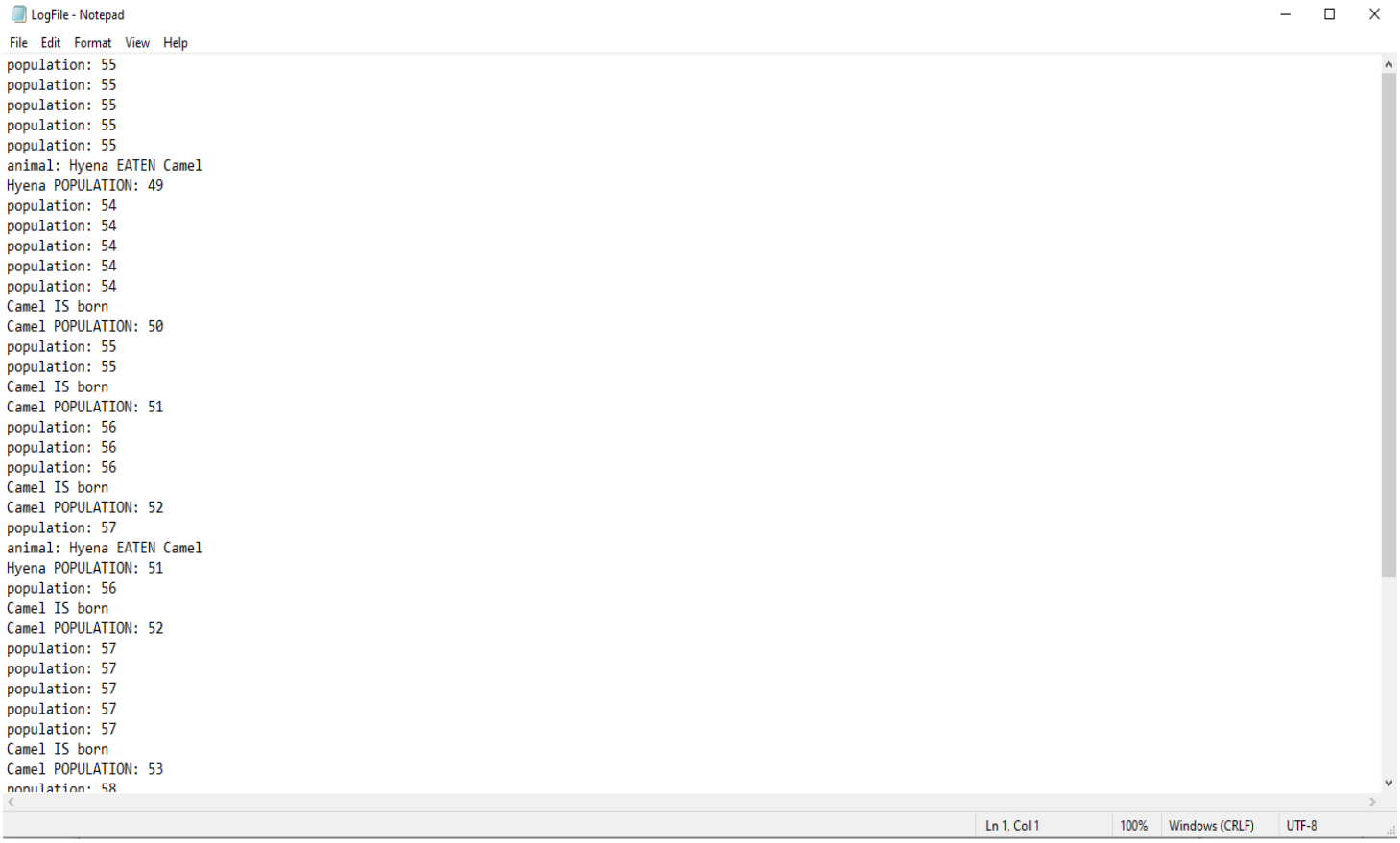
Example of a simulation set in the Arctic.



When a simulation has finished a message box appears to notify this to the user.

Animal_names	Food_Source_Type	Life_Span	Death_Rate	Birth_Rate	Hunger_Meter	Animal_Speed	Food_Chain	Click to Add
Antelope	Herbivore	5	0.5	0.2	6	1	6	
Artic Fox	Omnivore	3	0.5	0.2	6	1	3	
Camel	Herbivore	7	0.5	0.2	6	1	5	
Cow	Herbivore	7	0.5	0.2	6	1	5	
Deer	Herbivore	6	0.5	0.2	6	1	5	
Giraffe	Herbivore	8	0.5	0.2	6	1	8	
Hyena	Carnivore	4	0.5	0.2	6	1	6	
Lemmings	Herbivore	5	0.5	0.2	6	1	1	
Lion	Carnivore	6	0.5	0.2	6	1	7	
Osterich	Omnivore	6	0.5	0.2	6	1	6	
Panda	Omnivore	7	0.5	0.2	6	1	3	
Penguins	Carnivore	4	0.5	0.2	6	1	3	
Polar Bear	Carnivore	7	0.5	0.2	6	1	7	
Seal	Carnivore	5	0.5	0.2	6	1	4	
Snake	Carnivore	4	0.5	0.2	6	1	4	

The Species table found at the database file for user to change values and add new animals to the simulation by adding new records.



Here is the text file that contains smaller details of the simulation activity occurring in the simulation. Here we know the total populations at certain times of the simulation, which animal ate what animal and what animal was born at a time.

Testing

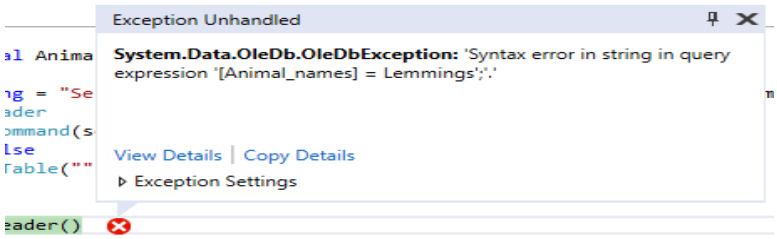
White Box Testing

For the test plan later on, I would be using white box testing to spot and evaluate whether each part of my code is working expectedly and report any issues to fix. In this section I will be telling you some of the issues I have dealt with whilst developing and testing the code.

ISSUE 1: Common syntax errors found in typing SQL commands.

```
Dim selectCommand As String = "Select * FROM [Species] WHERE [Animal_names] = " & AnimalName & "';"
```

From having to transfer data from the database to the simulator I had to make numerous SQL command calls. You probably haven't noticed but that line on the last page would result in a syntax error like this:



Here is an example were after selecting "Lemmings" and submitting the input for the simulation, a common syntax error would occur stating that I have missplaced a ' or " somewhere in the SQL command. Personally I find these two punctuation marks confusing and easily forgettable, especially when coding for long periods. To correct the last example I just had to pair the apostrophe's together like so:

```
Dim selectCommand As String = "Select * FROM [Species] WHERE [Animal_names] = '" & AnimalName & "';"
```

Here I highlighted over the missing apostrophe. Though the error can be easily fixed, it is slightly frustrating as sometimes I spend too long trying to figure out where I have missed the punctuation marks.

ISSUE 2: Inefficiency in loading initial populations onto the simulation.

Originally I used a nested for loops to load all the users inputted populations into the board. The for loop containing the second for loop was used to scan through the y - axis of the grid-world, and the second for loop was used to scan through the x - axis of the for loop. With the combinations of these coordinates can produce possible positions for animals in the simulation. It can be shown clearly here:

```
    b.addAnimal(h, 10)

    b.addAnimal(c, 20)
    b.addAnimal(o, 30)
    Dim ANimal As Pen
    ANimal = New Pen(Brushes.Black, 1)
    Brush = New SolidBrush(Color.Black)
    Dim r As New RectangleF(0.0, 0.0, 10, 10)
    myGraphics.FillRectangle(Brush, r)
    myGraphics.FillRectangle(Brush, r)


    For y = 0 To 49
        For x = 0 To 49
            counter = counter + 1
            If IsNothing(b.Board(x, y)) = False Then
                'MsgBox(b.Board(x, y).FoodType)

                Select Case b.Board(x, y).FoodType
                    Case 1
                        Brush = New SolidBrush(Color.Blue) 'herbivore

                    Case 2
                        Brush = New SolidBrush(Color.Yellow) 'omnivore

                    Case 3
                        Brush = New SolidBrush(Color.Crimson) 'carnivore

                    Case Else
                        Brush = New SolidBrush(Color.Black) 'backUp

                End Select

                Dim Animal As Pen 'Outline for each animal
                Animal = New Pen(Brushes.Black, 1)

                Dim r As New RectangleF(x * 10, y * 10, 10, 10) 'dimensions for each animal inside the board
                myGraphics.FillRectangle(Brush, r)
                myGraphics.FillRectangle(Brush, r)
            End If
        Next
    Next

    MsgBox(counter)

End Sub
```

Inefficiency is found in the nested for-loop.

I manually inputted animals into the system called h, c, o representing herbivore, carnivore and omnivore and gave each of them respective population size 10, 20, 30 (so 60 animals in total). Just to

note this solution was made early on in development so I manually inputted these values to test the system. They are then sent to addAnimal() where they are given random positions on the board and alongside turning them into objects of each respected animal class. Then they enter the nested for loop were they are drawn in order of each x,y component in b.board(x,y). This is were the inefficiency is found. This method is a brute force tactic that would go through the total amount of board(x,y) positions, even though I manually entered 60 animals into the simulation. Its inefficient as the method checks all (50^2) 2500 positions in board(x,y) due to the two for-loops repeating 50 times. This is a waste of processing as we only needed to repeat the process 60 times. This is where I began using linked lists to take advantage of its dynamic data structure.

New Linked-List Method:

```
Do
    counter = counter + 1
    If IsNothing(b.animalList.current) = False Then

        Select Case b.animalList.current.piece.AnimalName

            Case cl_GlobalSetting.AnimalONE
                Brush = New SolidBrush(Color.LightBlue) 'population 1

            Case cl_GlobalSetting.AnimalTWO
                Brush = New SolidBrush(Color.Yellow) 'population 2

            Case cl_GlobalSetting.AnimalTHREE
                Brush = New SolidBrush(Color.Red) 'population 3

            Case cl_GlobalSetting.AnimalFOUR
                Brush = New SolidBrush(Color.Green) 'population 4

            Case cl_GlobalSetting.AnimalFIVE
                Brush = New SolidBrush(Color.Purple) 'population 5


        End Select


        Dim Animaloutline As Pen 'Outline for each animal
        Animaloutline = New Pen(Brushes.Black, 1)

        Dim r As New RectangleF(b.animalList.current.X * 10, b.animalList.current.Y * 10, 10, 10) 'dimensions for each animal inside the board
        myGraphics.FillRectangle(Brush, r)
        myGraphics.FillRectangle(Brush, r)
    End If
    ' MsgBox("Pop " & b.animalList.populationNum)


Loop Until b.animalList.NextAnimalNode() = False
```

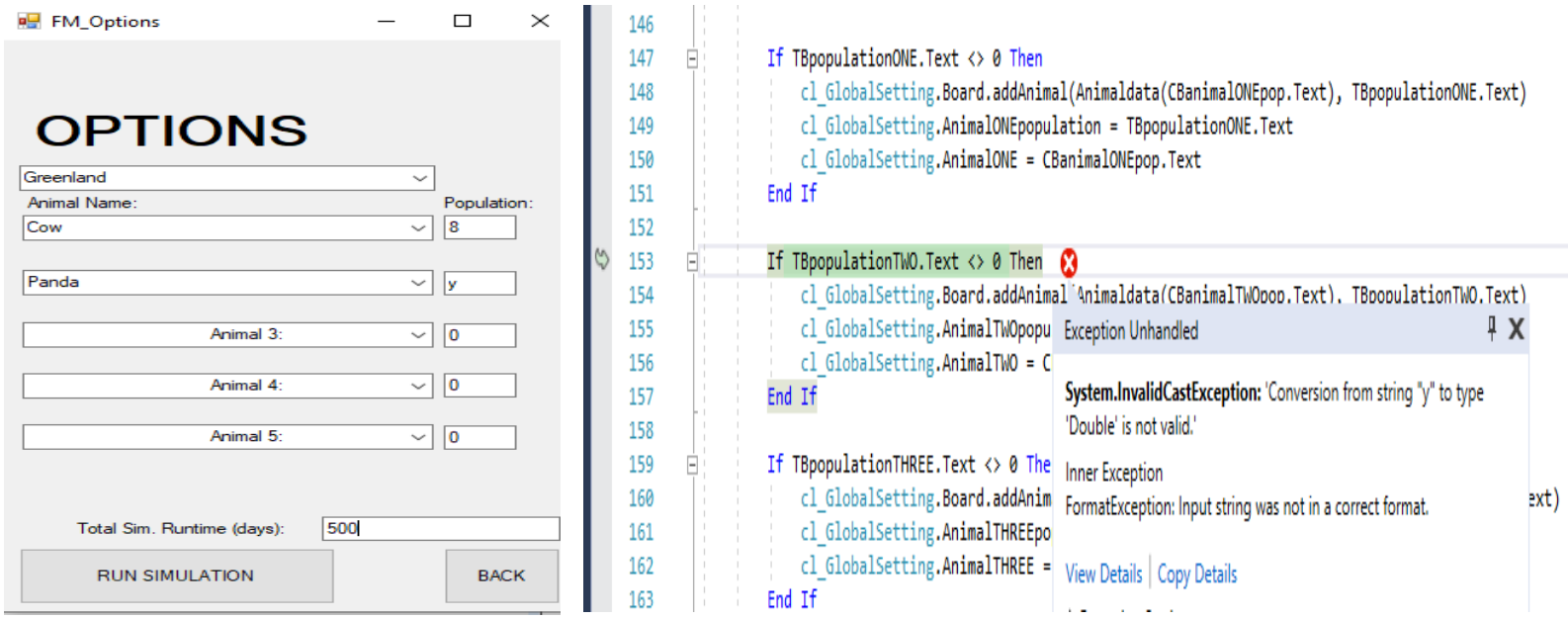
I named the linked list animalList and instead of having to go through each 2500 positions, I now can only repeat drawing each animal to the intended total population number. This total population is the exact same number of nodes in a given linked list. I contained this process in a do loop and so made the process finish only when we reached the tail node of the list. This last node is determined

if there is any content in the node next to a current node, in which case a tail node wouldn't have. This new method is very efficient compared to my original method, as it reduces the amount of processing power significantly as the number of processes match the number of inputted animals by the user.

Black Box Testing

I have asked some students from Mrs. Brookes year 10 Geography class to help test the program. The students were the best people to ask because at the start of the school year they seemed delighted to help me. This can be shown in the analysis section of this document, where I counted a vote for how many students were willing to be my testers. In addition they are the main users alongside the geography teachers. As Mrs Brookes has overseen my work; I felt more inclined for the students to finally see the program. I took the first four students who raised their hands up first to be my testers, and told them to report any errors they have found in the program. The students are kept anonymous for the sake of their own privacy. Below are the main issues they have spotted:

ISSUE 1: Placing a letter inside the initial population count of a specie.



As soon as testing began, students found out a run-time error would occur if the user places a non-numerical value into the population number textboxes. This wasn't a problematic issue to fix as the error would be solved by validating the inputs.

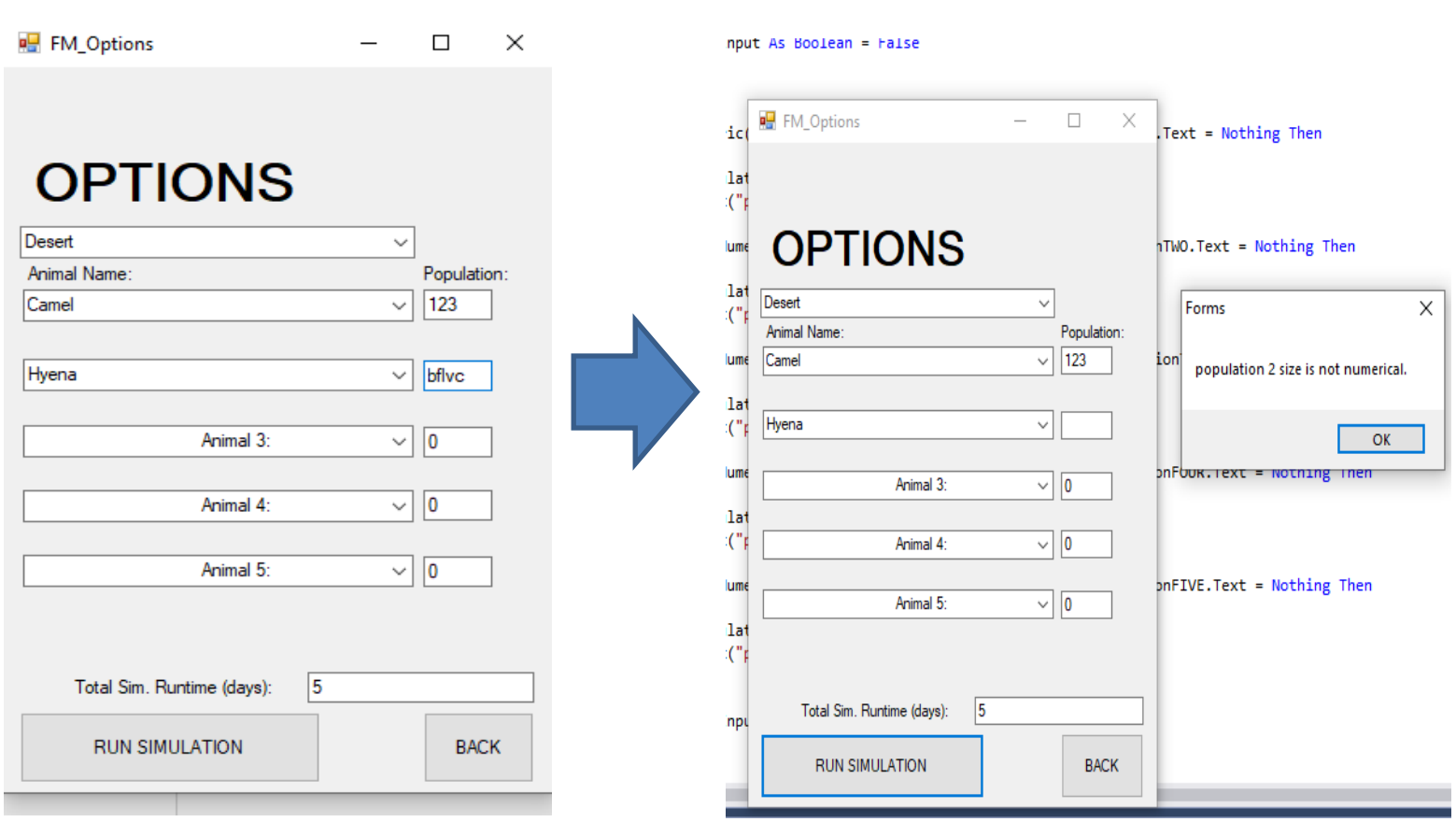
- Original code before testing

```
'button to move to simulation page
Private Sub BTRunSimulation_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BTRunSimulation.Click
    setup()
    cl_GlobalSetting.TotalSimulationTime = TBtotalRunduration.Text
    Me.Close()
    FM_AnimalSimulation.Show()
End Sub
```

- New code were the user inputs are validated before being entered into the simulation.

```
Private Sub BTRunSimulation_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BTRunSimulation.Click
    Dim validInput As Boolean = False



    If IsNumeric(TBpopulationONE.Text) = False Or TBpopulationONE.Text = Nothing Then

        TBpopulationONE.Text = Nothing
        MsgBox("population one size is not numerical.")

    ElseIf IsNumeric(TBpopulationTWO.Text) = False Or TBpopulationTWO.Text = Nothing Then

        TBpopulationTWO.Text = Nothing
        MsgBox("population 2 size is not numerical.")

    ElseIf IsNumeric(TBpopulationTHREE.Text) = False Or TBpopulationTHREE.Text = Nothing Then

        TBpopulationTHREE.Text = Nothing
        MsgBox("population 3 size is not numerical.")

    ElseIf IsNumeric(TBpopulationFOUR.Text) = False Or TBpopulationFOUR.Text = Nothing Then

        TBpopulationFOUR.Text = Nothing
        MsgBox("population 4 size is not numerical.")

    ElseIf IsNumeric(TBpopulationFIVE.Text) = False Or TBpopulationFIVE.Text = Nothing Then

        TBpopulationFIVE.Text = Nothing
        MsgBox("population 5 size is not numerical.")

    Else
        validInput = True
    End If

    If validInput = True Then
        setup()
        cl_GlobalSetting.TotalSimulationTime = TBtotalRunduration.Text
        Me.Close()
        FM_AnimalSimulation.Show()
    End If

End Sub
```

The validation would confirm only numerical values are entered and would deleted incorrect inputs for the user to re-type an appropriate input. Then the user would be notified if an input is incorrect. This means that the simulation would only carry out when all inputs are valid.



On the left is the initial inputs with the second population containing an incorrect population value. On the right is the result of submitting the inputs as mentioned on the last page. Notice how the input which had an incorrect value gets deleted automatically, this allows the user to re-submit a value after pressing the “OK” button on the message box.

ISSUE 2: Animal would give birth to itself if there is no movement between simulation days.

Sometimes when a simulation runs an animal would give birth by itself. This was spotted to happen by the end of the simulations were one animal was left alive. This occurrence happened to two of the testers as during testing they noticed that another animal sprite of the same specie was born. After the simulation finished, we checked the log file to confirm what happened and was surprised to find how a total population of 1 then resulted to 2 after a single simulation day. Unfortunately to how rare the occurrence is, we couldn’t provide any evidence of this to put in this document but over time I have realised the cause of how something as odd as this can occur.

After considering many possibilities even strange ones like undrawn animals having contact with visible animals. I have come to the conclusion that this occurrence happens when animals do not move within a simulation day. Each day every animal would be redrawn even if they do not move. If an animal decides to not move for a day, it essentially gains contact with itself in the next day. This triggers the animals birth function as they are both the same specie and then potentially cause a birth. From this it turns out that the **probability of a "self-birth" is the birth-rate multiplied by 1/9th.** This is true as 1/9th is the chance of an animal-movement to land on "No movement" by the function GetMove() in class cl_Animal. This is then multiplied by the value of birth-rate found in the species database file. These values are multiplied as the self-birth probability is the product of the event of a birth occurring and an event were it lands on "no movement." Similar to how the odds of flipping two heads of an unbiased coin consecutively is 0.25. This is because flipping heads is 0.5 chance of occurring, and flipping it again would half the probability again. This can be further studied in *Statistics and Mechanics Year 1/AS mathematics* under probabilities.

So by figuring this out and applying this idea across the entire simulation, the occurrence of self-births happen often depending if multiple animals in a simulation would choose not to move. This actually helps to understand how the issue of rapid growth spots (which will be explained next) occur too. From my analysis the problem is quite easy to solve, either reduce the birth rate probability, allow animal interactions to only occur for animals who move or the statement for births can only occur if they are different members of the same specie. In my case I did all three.

```
27  Sub RunDay()
28      Dim direction As String
29      Dim AnimalName As String
30
31
32      cl_GlobalSetting.log.write("population: " & animalList.populationNum)
33
34      If animalList.populationNum > 0 Then
35
36          animalList.reset()
37
38          Do
39              If IsNothing(animalList.current) <> True Then
40
41                  direction = animalList.current.piece.GetMove()
42
43                  If direction <> "No movement" Then 'Stops animal giving birth by itself if they contact themselves by not moving.
44
45
```

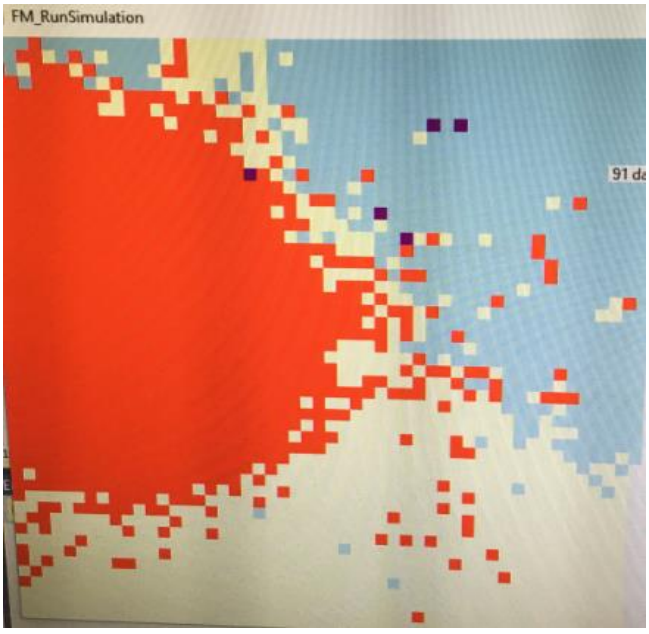
- Animal interactions only occur if the animal decides to move.

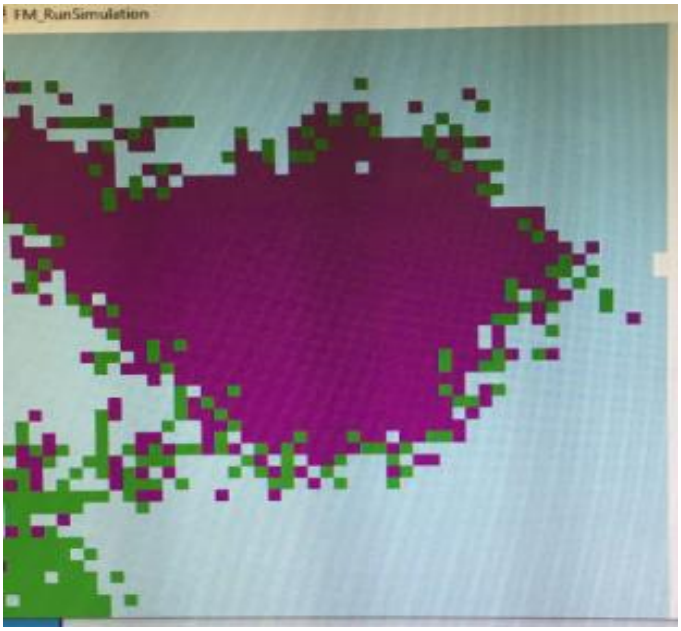
- Births can only occur now if they are different members of the same specie.

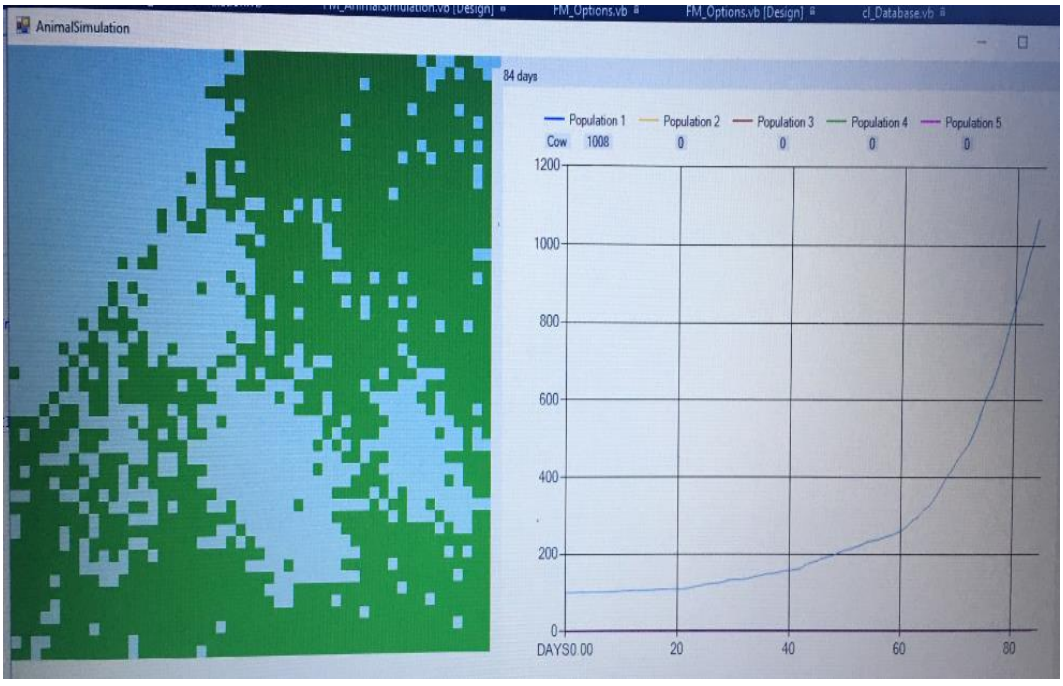
```
ElseIf Board(animalList.current.X, animalList.current.Y).AnimalName = Board(temp(0), temp(1)).AnimalName And Board(animalList.current.X, animalList.current.Y).age >= Board(temp(0), temp(1)).age Then
```

Lastly for when I run the simulation I chose to lower the birth rate from 0.25 to 0.15, but this doesn't matter so much as users are encouraged to experiment with all changeable values. But reducing the chance of self-birth wouldn't really be affected by reducing the birth-rate as due to these fixes mentioned above it doesn't matter what level of birth rate its on.

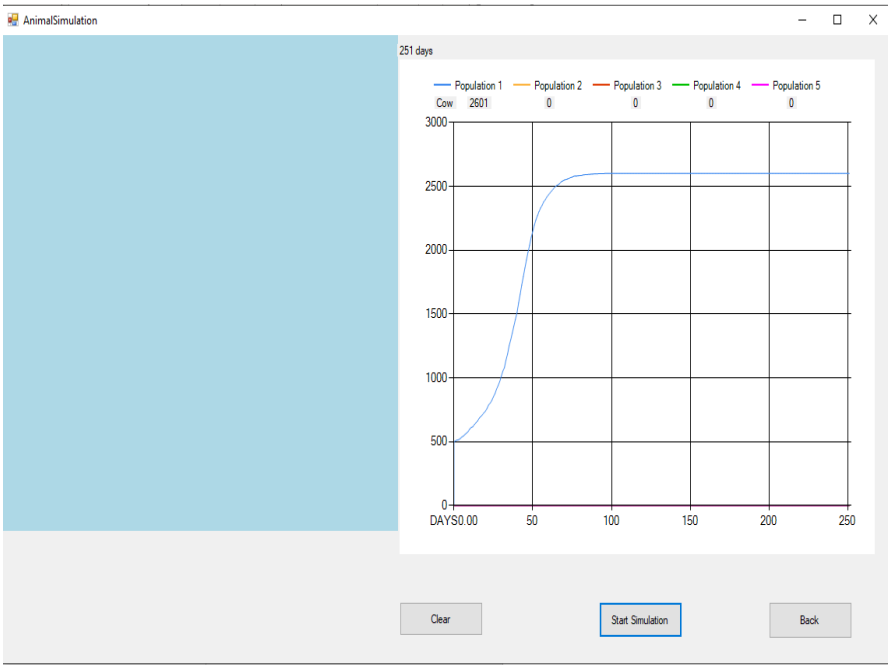
ISSUE 3: Unrealistic Rapid Growth Spots.







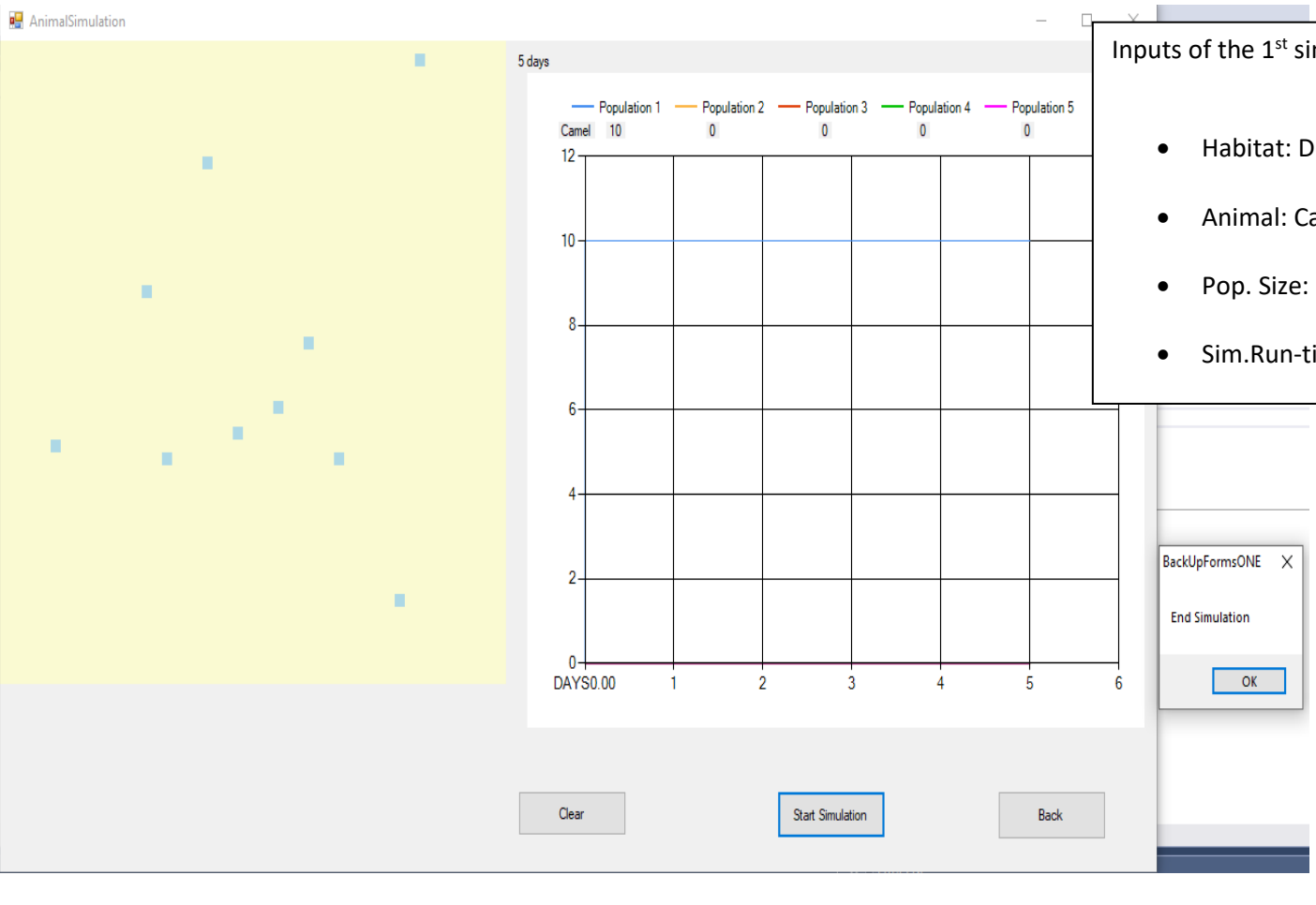
This issue was spotted by all testers, as they experimented with large initial population sizes. As you can see huge levels of growth can be seen forming patterns as shown in the last page. If you see the population graph on the third image you would notice that its very similar to an exponential graph. This goes to show how rapid the births occur when species overcrowd each other. Due to the quick growth, many calculations are needed to process the simulation and in result cause the simulation to lag drastically. Each simulation day slows down as populations exceed over the 1000's. Furthermore going beyond 2500 (the number of squares in the habitat) animals still grow in number but plateues around 2600 leaving no movements to be seen as the habitat gets filled with the assigned colour of an animal. This is shown in the image below. After testing with large populations myself I have noticed that the population graphs reflect the Big O notation's time complexity. The time it takes to process lengthens as more animals are added to the board, however when the population level flattens off the rate of each passing day seems to shorten to a approximately constant time. This may be due to the efficent working of the linked list, but I am not quite certain. Noted on page 175 of *AQA A-Level Computer Science by Bob Reeves*, the second most efficient algorithm is one with a big O notation similar to O (logN). The population graphs shape is similar to a 2-to-the-power of n graph and a logN graph put together, like as shown above. Though initially having a big O(2^n) is not efficient, still being able to calculate so many species is very impressive. I find it amazing that my population graph seems to monitor the time complexity of processing the algorithm, and I suspect that this is not sjust coincidence. But I am not fully sure if this is the case. This performance error could be solved if we cap the initial population sizes, introduce



carnivores to a simulation, position the animals more widely, increase the grid world or shorten the life spans. But I have decided not to change this in my program, as the main tools needed to fix this are in the users hands. Also its important to note that this sustained growth is only shown for herbivores as I have modelled them to have the longest lives and can live without food as they are producers (constantly fed by the environment). In addition though it's a model it does have a reflection on nature. If there are no predators around and a specie has acess to an abundant food source, naturally the specie would grow exponentially just like how the simulation reflects. However this is dependent if the food source doesn't run out which my simulation does not account for.

ISSUE 4: Try to go back to a new simulation to find out that the simulator doesn't work.

Another error was found when testers when back from a simulation and tried starting up a new simulation. Instead of the program showing the new simulation, they get the old simulation habitat without any new animals printed on the board. However the graph contains the new simulation data, which shows that the system did receive the new inputs from the user. Below is an example:



Inputs of the 1st simulation:

- Habitat: Desert
- Animal: Camel
- Pop. Size: 10
- Sim.Run-time: 5 days

Inputs of the 2nd simulation:

- Habitat: Greenland
- Animal: Panda
- Pop. Size: 5
- Sim.Run-time: 10 days

Notice how in the second simulation the intended animals, population sizes and run-time is updated on the population graph, but the simulator itself has no animals and doesn't display the Greenland habitat colour. This means that the simulator didn't get the new inputs from the user. This means a problem is needed to be fixed between the transition of the options form and simulator form. After testing I tried fixing the issue, to find myself back at the same logical error. So instead of trying to fix the issue directly I thought of just refreshing the entire program with the in-built `Application.Restart()` subroutine. This essentially restarts the entire program, removing any inputs being passed into the system so the whole program can refresh itself. But the only down-side to this solution is that instead of the program going back to the options menu; it goes back to the main menu page as this is where the user begins. Overall this solution is simple as its adds a single line to the back buttons code. At the time the solution was very suitable for me as my final deadline of showing the finished program to Mrs. Brookes was about two weeks away, so it saved a lot of valuable time for me to write this document.

Test Plan

The test plan will cover all possible normal and erroneous data found from white-box testing. In each test I will write what the type of input is, my expectation and the actual result. I will also mark whether any errors have been fixed.

Types of Test Input Data:

- **valid** - the most obvious or common data that should work.
- **valid extreme** - unusual, extreme or unexpected data, eg the highest and lowest (data that tests the limits but that should work).
- **invalid** - data that should definitely fail.
- **invalid extreme** - data that fails but is nearly acceptable.
- **erroneous** - data that is the wrong data type.

- (Taken from BBC Bitesize)

Test No.	Input Data	Tested Method	Expectation	Result	Screenshot No.
1.	None	Display Main Menu and its buttons.	Main Menu form should pop open first with its contents.	Main Menu form should pop open first with its contents.	*1.*
2.	Valid: left-Click Button	Press “Quit” button.	The program should stop running.	The program stops running.	*2.*
3.	Invalid Extreme: Right-Click Button	Press “Quit” button.	The program should do nothing as its not a left-click press.	The program does not change.	
4.	Valid: Left-Click Button	Press “Instructions” Button.	The main menu form should close and open the instructions form.	The main menu form closes and opens the instructions form.	*3.*
5.	Invalid Extreme: Right-Click Button	Press “Instructions” Button.	The program should do nothing as its not a left-click press.	The program does not change.	
6.	Valid: Left-Click Button	Press the “back” button in the Instructions form.	Close instruction form and open the main menu form.	Closes instruction form and opens the main menu form.	*4.*
7.	Invalid extreme: Right-Click Button	Press the “back” button in the Instructions form.	The program should do nothing as its not a left-click press.	The program does not change.	
8.	Valid: Left-Click Button	Press “Start Options” Button.	The main menu form should close and open	The main menu form closes and opens the options form.	*5.*

			the options form.		
9.	Invalid Extreme: Right-Click Button	Press “Start Options” Button.	The program should do nothing as its not a left-click press.	The program does not change.	
10.	Valid: Left-Click Button	Clicks on the drop-down “Terrain Type” menu.	A list of Habitats from the Database should be displayed.	A list of Habitats from the Database are displayed.	*6.*
11.	Invalid Extreme: Right-Click Button	Clicks on the drop-down “Terrain Type” menu.	Nothing should happen.	Nothing happens.	
12.	Valid: Left-Click Button	Select “Greenland”.	Selecting “Greenland” should replace the “terrain type” text.	The “Greenland” text replaces the “terrain type” text.	*7.*
13.	Invalid Extreme: Right-Click Button	Select “Greenland”.	Nothing should happen.	Nothing happens.	
14.	Valid: Left-Click Button	Select “Desert”.	Selecting “Desert” should replace the “terrain type” text.	The “Desert” text replaces the “terrain type” text.	*8.*
15.	Invalid Extreme: Right-Click Button	Select “Desert”.	Nothing should happen.	Nothing happens.	
16.	Valid: Left-Click Button	Select “Arctic”.	Selecting “Arctic” should replace the “terrain type” text.	The “Arctic” text replaces the “terrain type” text.	*9.*
17.	Invalid Extreme: Right-Click Button	Select “Arctic”.	Nothing should happen.	Nothing happens.	
18.	Valid: Left-Click Button	Select a different terrain after an initial terrain has been selected.	The new selected terrains’ text should replace the old terrains’ text.	The new selected terrains’ text replaces the old terrains’ text for all terrain types.	*10.*

19.	Valid: Left-Click Button whilst “Greenland” is selected.	Click “Animal 1” drop-down menu.	A list of “Greenland” animals should be displayed after selecting “Animal 1”.	A list of “Greenland” animals are displayed.	*11.*
20.	Invalid Extreme: Right-Click Button whilst “Greenland” is selected.	Click “Animal 1” drop-down menu.	Nothing should happen.	Nothing happens.	
21.	Valid: Left-Click Button whilst “Greenland” is selected.	Click “Animal 2” drop-down menu.	A list of “Greenland” animals should be displayed after selecting “Animal 2”.	A list of “Greenland” animals are displayed.	*12.*
22.	Invalid Extreme: Right-Click Button whilst “Greenland” is selected.	Click “Animal 2” drop-down menu.	Nothing should happen.	Nothing happens.	
23.	Valid: Left-Click Button whilst “Greenland” is selected.	Click “Animal 3” drop-down menu.	A list of “Greenland” animals should be displayed after selecting “Animal 3”.	A list of “Greenland” animals are displayed.	*13.*
24.	Invalid Extreme: Right-Click Button whilst “Greenland” is selected.	Click “Animal 3” drop-down menu.	Nothing should happen.	Nothing happens.	
25.	Valid: Left-Click Button whilst “Greenland” is selected.	Click “Animal 4” drop-down menu.	A list of “Greenland” animals should be displayed after selecting “Animal 4”.	A list of “Greenland” animals are displayed.	*14.*
26.	Invalid Extreme: Right-Click Button whilst “Greenland” is selected.	Click “Animal 4” drop-down menu.	Nothing should happen.	Nothing happens.	
27.	Valid: Left-Click Button whilst	Click “Animal 5” drop-down menu.	A list of “Greenland” animals should	A list of “Greenland” animals are displayed.	*15.*

	"Greenland" is selected.		be displayed after selecting "Animal 5".		
28.	Invalid Extreme: Right-Click Button whilst "Greenland" is selected.	Click "Animal 5" drop-down menu.	Nothing should happen.	Nothing happens.	
29.	Valid: Left-Click on "cow" whilst "Greenland" is selected.	Selecting an animal (cow) in "Animal 1"	A selected animal name from the list should replace the "Animal 1" text. "Cow" should replace "Animal 1"	The selected animal name (cow) replaces the "Animal 1" text.	*16.*
30.	Invalid Extreme: Right-Click on "cow" whilst "Greenland" is selected.	Selecting an animal (cow) in "Animal 1"	Nothing should happen.	Nothing happens.	
31.	Valid: Left-Click on "antelope" whilst "Greenland" is selected.	Selecting an animal (Antelope) in "Animal 2"	A selected animal name from the list should replace the "Animal 2" text. "Antelope" should replace "Animal 2"	The selected animal name (antelope) replaces the "Animal 2" text.	*17.*
32.	Invalid Extreme: Right-Click on "antelope" whilst "Greenland" is selected.	Selecting an animal (antelope) in "Animal 2"	Nothing should happen.	Nothing happens.	
33.	Valid: Left-Click on "panda" whilst "Greenland" is selected.	Selecting an animal (Panda) in "Animal 3"	A selected animal name from the list should replace the "Animal 3" text.	The selected animal name (panda) replaces the "Animal 3" text.	*18.*

			"panda" should replace "Animal 3"		
34.	Invalid Extreme: Right-Click on "panda" whilst "Greenland" is selected.	Selecting an animal (Panda) in "Animal 3"	Nothing should happen.	Nothing happens.	
35.	Valid: Left-Click on "Giraffe" whilst "Greenland" is selected.	Selecting an animal (Giraffe) in "Animal 4"	A selected animal name from the list should replace the "Animal 4" text. "Giraffe" should replace "Animal 4".	The selected animal name (giraffe) replaces the "Animal 4" text.	*19.*
36.	Invalid Extreme: Right-Click on "Giraffe" whilst "Greenland" is selected.	Selecting an animal (Giraffe) in "Animal 4"	Nothing should happen.	Nothing happens.	
37.	Valid: Left-Click on "Snake" whilst "Greenland" is selected.	Selecting an animal (Snake) in "Animal 5"	A selected animal name from the list should replace the "Animal 5" text. "Snake" should replace "Animal 5"	The selected animal name (Snake) replaces the "Animal 5" text.	*20.*
38.	Invalid Extreme: Right-Click on "Snake" whilst "Greenland" is selected.	Selecting an animal (Snake) in "Animal 5"	Nothing should happen.	Nothing happens.	
39.	Valid: Left-Click on all Animal options	Change decision of animal to be selected.	The Selected Animals can be changed for another animal	Animals are replaced for another option in the list.	*21.*

			within the list.		
40.	Valid: Integer ~ 5	Place values into the simulation run-time textbox.	System should be able to hold the value. For testing a message box would appear to show how the textbox accepted 5 days.	System accepted 5 days as stated by the message box.	*22.*
41.	Valid Extreme: Float ~ 1000.5	Place values into the simulation run-time textbox.	System should be able to hold the value but as a rounded integer. For testing a message box would appear to show how the textbox accepted 1000.5 as 1001 .	Value was accepted into the simulation; noted by the message box. However the system rounded the simulation as 1000 days, so it rounded down instead of up.	*23.*
42.	Invalid: Integer ~ -5	Place values into the simulation run-time textbox.	System should accept the value but the simulation should finish immediately as its programmed to run from 0 onwards.	Simulator finishes immediately when played. <u>Improvement:</u> Could notify the user that inputs below 0 is not acceptable. ***Added Improvement – shown in screenshot**	*24.*
43.	Erroneous: String ~ “gda”	Place values into the simulation run-time textbox.	System should not accept the string as a value, and removes the value to leave empty box.	Run-time error: Couldn’t convert string to integer. I need to validate this. ***FIXED**	*25.*
44.	Valid: Integer~ 10	Place values into Animal 1’s Initial Population count.	System should accept the integer as a value. For testing a message box would appear	Message box appears confirming how the system accepts 10 as an initial population count for animal 1.	*26.*

			to confirm this.		
45.	Valid Extreme: Float ~ 1.5	Place values into Animal 1's Initial Population count.	System should accept this as a rounded integer value. Should produce 2 animal 1's. This will be confirmed on the population graph.	2 animal 1's are produced.	*27.*
46.	Invalid: Integer ~ -8	Place values into Animal 1's Initial Population count.	System should not accept this value. The user will be notified by a message box and the input is removed.	Program froze/crashed. I forgot to validate for numerical values below 0. ***Fixed**	*28.*
47.	Erroneous: String ~ "abc"	Place values into Animal 1's Initial Population count.	System should not accept this value. The user will be notified by a message box and the input would be removed.	Error: Can't convert string to integer. ***Fixed** System does not accept this value. A message box appears to confirm this and the input is set back to 0.	*29.*
48.	Valid: Integer ~ 15	Place values into Animal 2's Initial Population count.	System should accept the integer as a value. For testing a message box would appear to confirm this.	Message box appears confirming how the system accepts 15 as an initial population count for animal 2.	*30.*
49.	Valid Extreme: Float ~ 7.8	Place values into Animal 2's Initial Population count.	System should accept this as a rounded integer value. Should produce 8 animal 2's. This will be confirmed on the population graph.	8 animal 2's are produced.	*31.*

50.	Invalid: Integer ~ -2	Place values into Animal 2's Initial Population count.	System should not accept this value. The user will be notified by a message box and the input is removed.	Program crashed. I forgot to validate for numerical values below 0. ***Fixed**	*32.*
51.	Erroneous: String ~ "a!qv!?"	Place values into Animal 2's Initial Population count.	System should not accept this value. The user will be notified by a message box and the input would be removed.	Error: Can't convert string to integer. ***Fixed** System does not accept this value. A message box appears to confirm this and the input is set back to 0.	*33.*
52.	Valid: Integer ~ 4	Place values into Animal 3's Initial Population count.	System should accept the integer as a value. For testing a message box would appear to confirm this.	Message box appears confirming how the system accepts 4 as an initial population count for animal 3.	*34.*
53.	Valid Extreme: Float ~ 18.1	Place values into Animal 3's Initial Population count.	System should accept this as a rounded integer value. Should produce 18 animal 3's. This will be confirmed on the population graph.	18 animal 3's are produced.	*35.*
54.	Invalid: Integer ~ -9	Place values into Animal 3's Initial Population count.	System should not accept this value. The user will be notified by a message box and the input is removed.	Program crashed. I forgot to validate for numerical values below 0. ***Fixed**	*36.*
55.	Erroneous: String ~ "4gr"	Place values into Animal 3's Initial Population count.	System should not accept this value. The user will be notified by a message box and the	Error: Can't convert string to integer. ***Fixed** System does not accept this value. A message	*37.*

			input would be removed.	box appears to confirm this and the input is set back to 0.	
56.	Valid: Integer ~ 21	Place values into Animal 4's Initial Population count.	System should accept the integer as a value. For testing a message box would appear to confirm this.	Message box appears confirming how the system accepts 21 as an initial population count for animal 4.	*38.*
57.	Valid Extreme: integer ~ 1000	Place values into Animal 4's Initial Population count.	System should accept this as a rounded integer value. Should produce 1000 animal 4's. This will be confirmed on the population graph.	1000 animal 4's are produced. However causes rapid growth spots mentioned in black-box testing.	*39.*
58.	Invalid: Integer ~ -50	Place values into Animal 4's Initial Population count.	System should not accept this value. The user will be notified by a message box and the input is removed.	Program crashed. I forgot to validate for numerical values below 0. ***Fixed**	*40.*
59.	Erroneous: String ~ "Eleven"	Place values into Animal 4's Initial Population count.	System should not accept this value. The user will be notified by a message box and the input would be removed.	Error: Can't convert string to integer. ***Fixed** System does not accept this value. A message box appears to confirm this and the input is and is set back to 0.	*41.*
60.	Valid: Integer ~ 13	Place values into Animal 5's Initial Population count.	System should accept the integer as a value. For testing a message box would appear to confirm this.	Message box appears confirming how the system accepts 13 as an initial population count for animal 5.	*42.*
61.	Valid Extreme:	Place values into Animal	System should accept this as a	0 animal 5's are produced.	*43.*

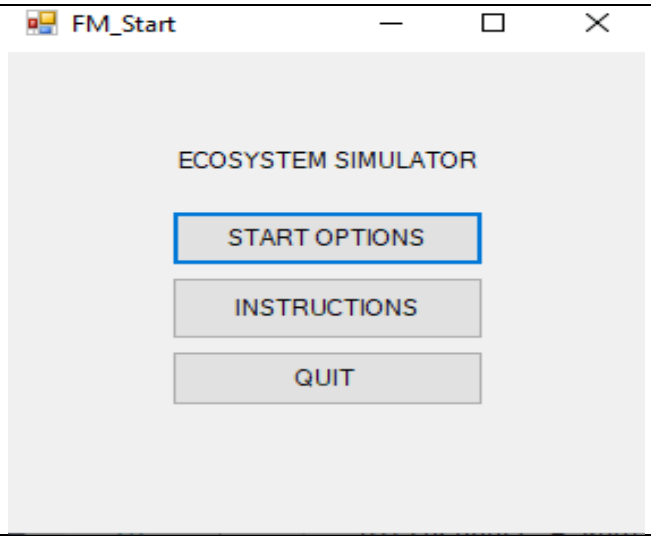
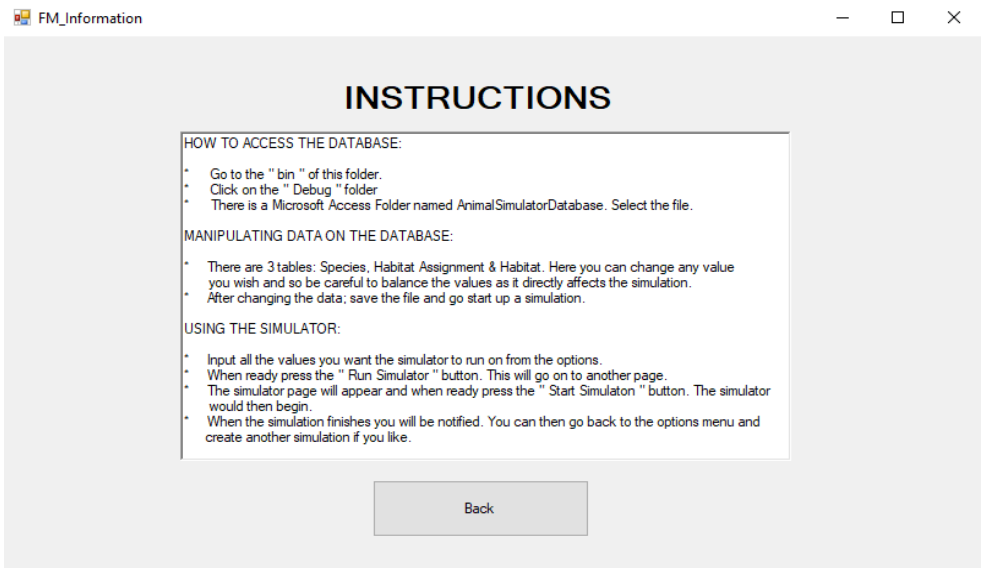
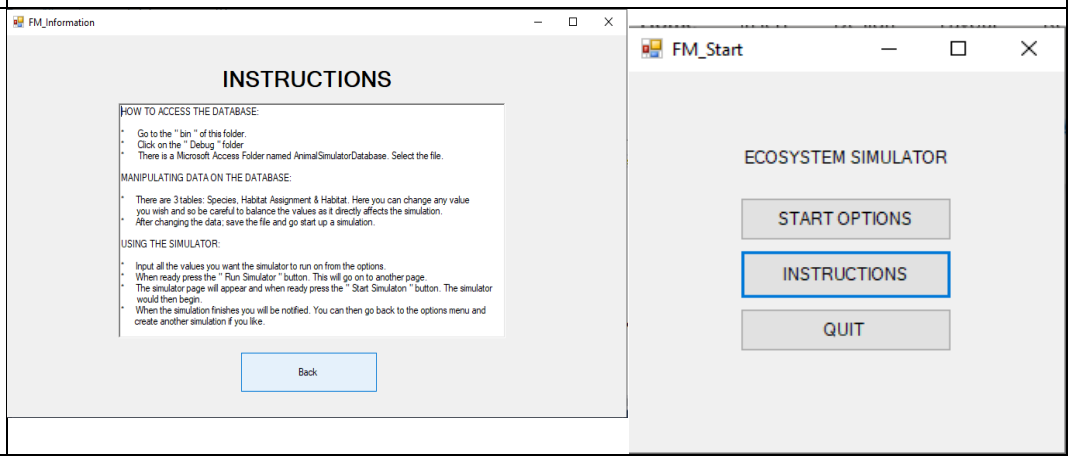
	integer ~ 0	5's Initial Population count.	rounded integer value. Should produce 0 animal 5's. This will be confirmed on the population graph.		
62.	Invalid: Integer ~ -99	Place values into Animal 5's Initial Population count.	System should not accept this value. The user will be notified by a message box and the input is removed.	Program crashed. I forgot to validate for numerical values below 0. ***Fixed**	*44.*
63.	Erroneous: String ~ "H3llo World"	Place values into Animal 5's Initial Population count.	System should not accept this value. The user will be notified by a message box and the input would be removed.	Error: Can't convert string to integer. ***Fixed** System does not accept this value. A message box appears to confirm this and the input is set back to 0.	*45.*
64.	Valid:	Be able to run without all the five animals being selected.	The system will accept the inputted data and disregard the ones which haven't been filled out. Animal 4 and Animal 5 will be missed out in this test.	System runs without the need to fill in all 5 animals.	*46.*
65.	Valid:	Be able to run without all the five initial animals population count being selected or used.	The system will accept the inputted data and set the ones not selected to 0. Animal 1 and Animal 3 population sizes will be missed out in this test.	System runs without the need to fill in all 5 animal population sizes.	*47.*

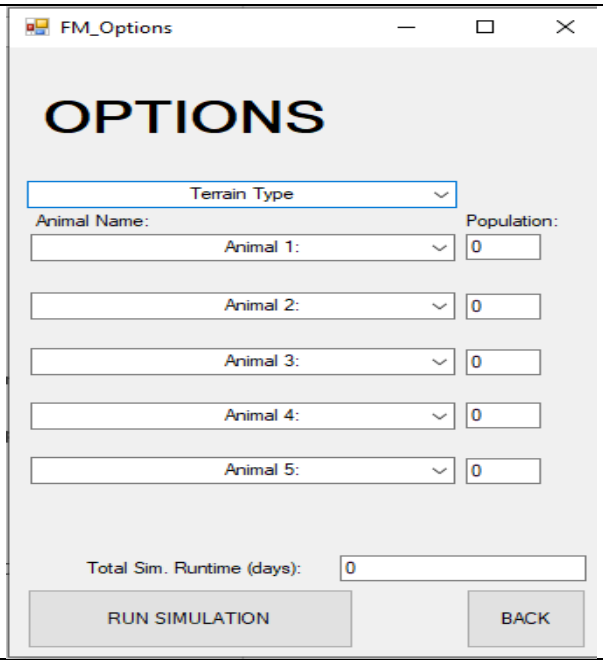
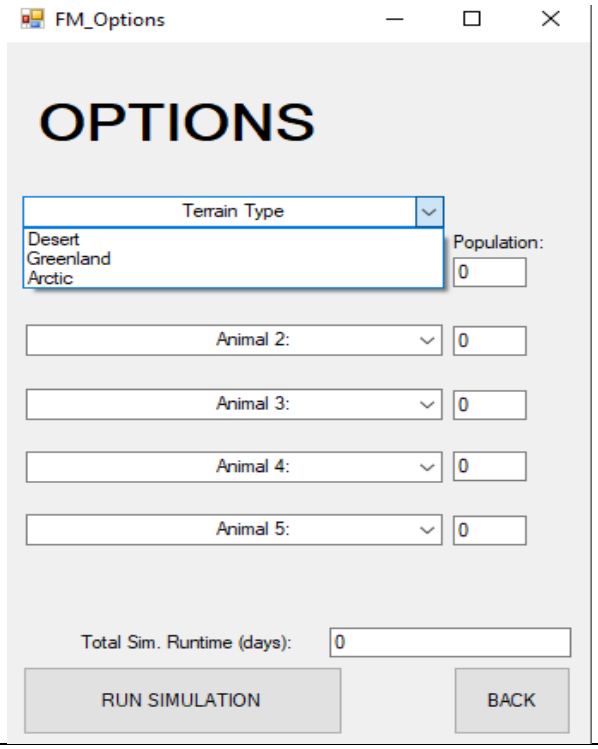
66.	Valid: Left-click press	Press the "Go to Simulation" Button.	The program should close the Options menu and transition to the simulator form.	Option menu closes and the Simulator form pops up.	*48.*
67.	Valid: Right-click press	Press the "Go to Simulation" Button.	Nothing should happen.	Nothing happens.	
68.	Valid: Left-click press	Press the "Start Simulation" Button	Simulator should run up to the same run-time length found in the options menu. Test: sim.run-time = 5	Simulator runs up to 5 days as intended.	*49.*
69.	Valid: Left-click press	Press the "Start Simulation" Button	Simulator should have the correct initial number of animals in the simulator as intended in the options menu. Test: ([animal] – [No.]) Animal 1 – 10 Animal 2 – 3 Animal 3 – 7 Animal 4 – 23 Animal 5 - 18	Exact results appears on the simulator.	*50.*
70.	Valid: Left-click press	Press the "Start Simulation" Button	Population graph should display a running line graph.	Population graph is displayed and runs alongside the simulator.	*51.*
71.	Valid: Left-click press	Press the "Start Simulation" Button	Animals should make one move everyday.	Animals complete a day when all new moves are made. But must set simulation run time to 0 as 1 allows two moves to be made.	*52.*
72.	Valid: Left-click press	Press the "Start Simulation" Button	Predator specie should kill prey when in contact.	Predator is shown to kill prey displayed in line graph and in text file.	*53.*

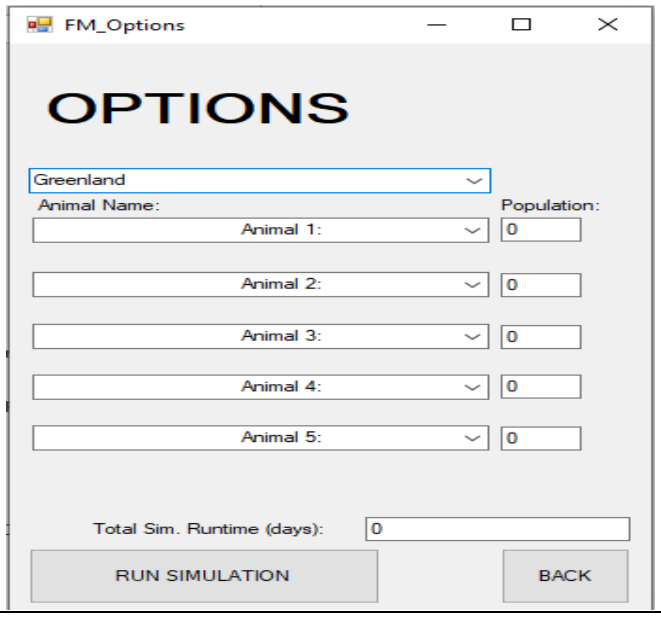
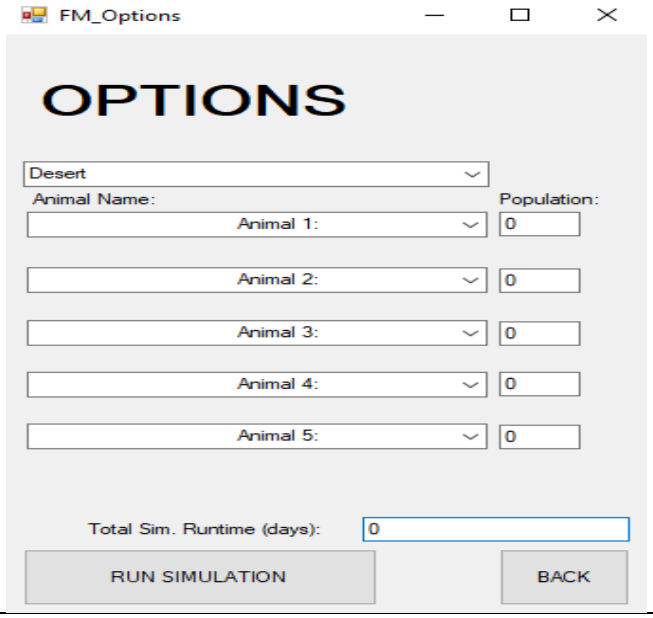
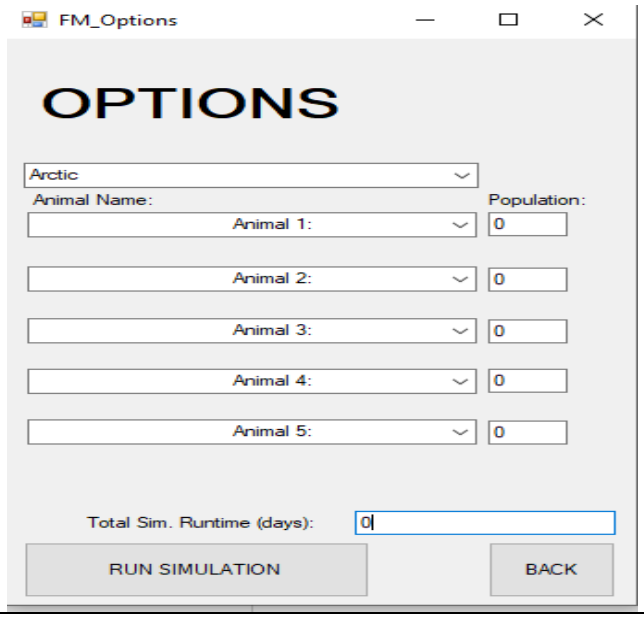
			Tested by using text file.		
73.	Valid: Left-click press	Press the "Start Simulation" Button	Same specie should give birth if birth probability is favourable. Tested by using text file.	Species are shown to give birth when probability is favourable.	*54.*
74.	Valid: Left-click press	Press the "Start Simulation" Button	Animals in the simulation can die from age or hunger.	Species can die without any animal interaction. Proven by log file.	*55.*
75.	Valid: Left-click press	Press the "Start Simulation" Button	Message box should appear when simulation ends.	When total simulation time = sim. Run-time or when all animals die, the user is notified by a message box that the simulation has ended.	*56.*
76.	Invalid Extreme: Right-click press	Press the "Start Simulation" Button	Simulator should not start.	Simulator does not start and no new activity occurs.	
78.	Valid: Left-click press	Press the "Back" button found in the simulator.	User should go back into the options menu from the simulator.	The simulator form closes and goes to the options form. The options form should still have the previous configuration for last simulation.	*57.*
79.	Valid: Left-click press	Go back and start a new simulation.	A new simulation should occur.	Simulation fails to load up and re uses terrain from previous simulation but without any animals. ***Fixed** Back button leads to main menu and user can follow-up on simulations without closing the program.	
80.	Valid: Left-click press	Press new "back" button.	User should go back to the main menu.	Simulator form closes and the program restarts itself at the main menu.	*58.*
81.	Invalid Extreme: Right-click press	Press new "back" button.	Nothing should happen.	Nothing happens.	

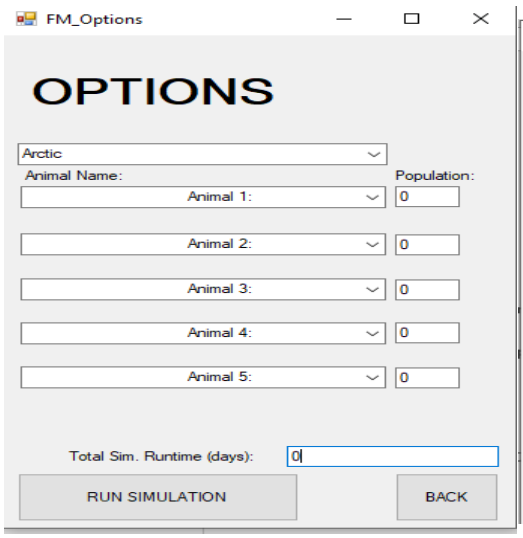
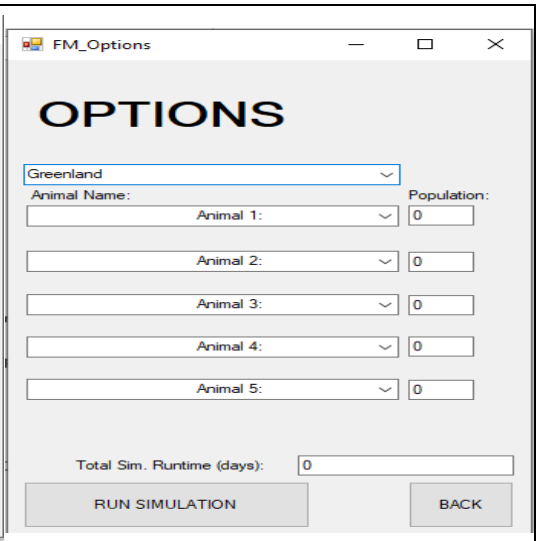
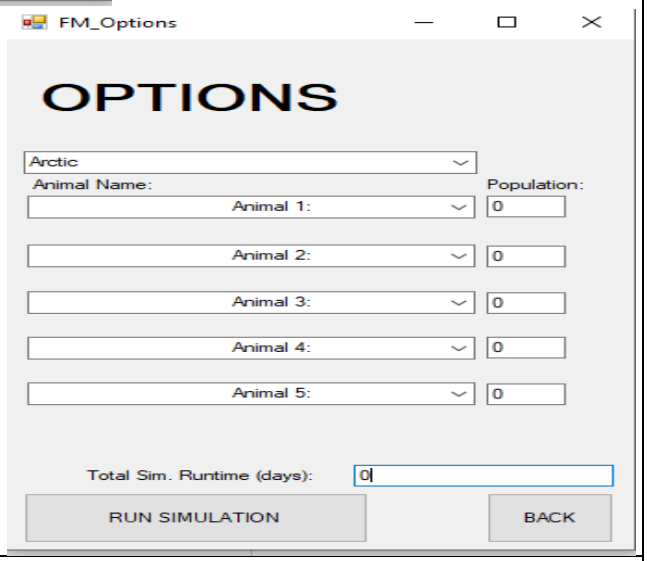
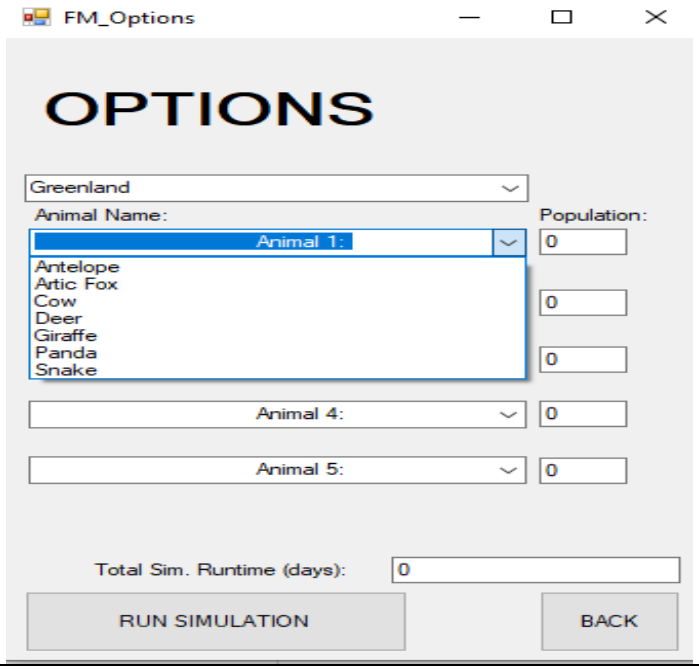
Screen-shot evidence of the test case are found next page.

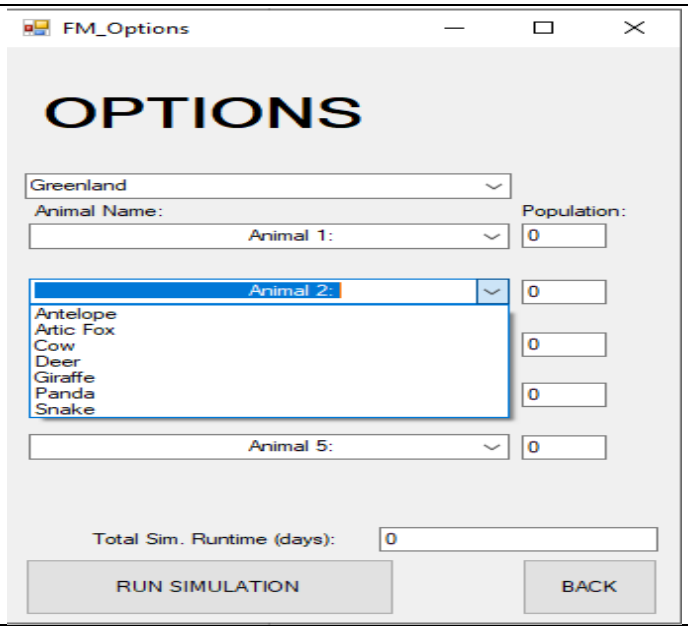
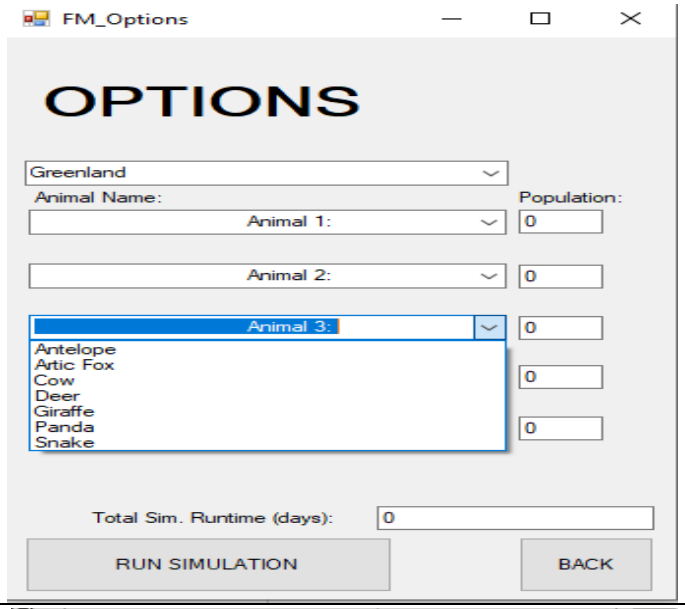
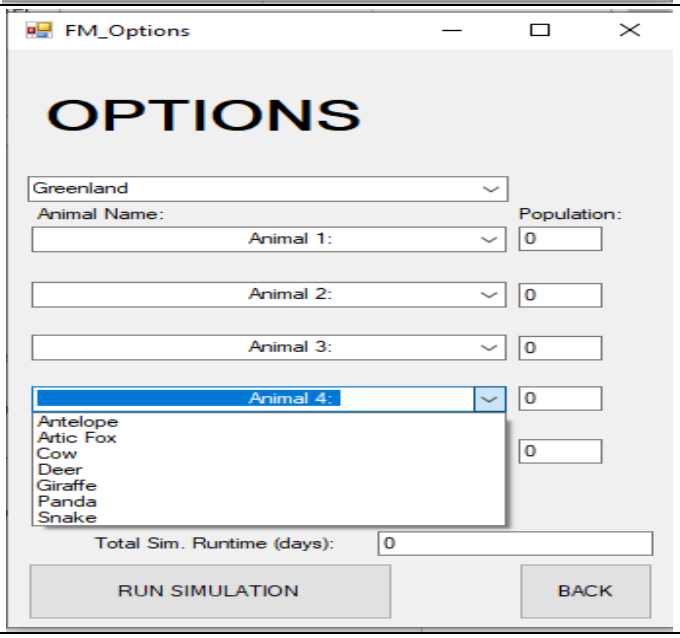
Screen-Shot Evidence of Test Case Results

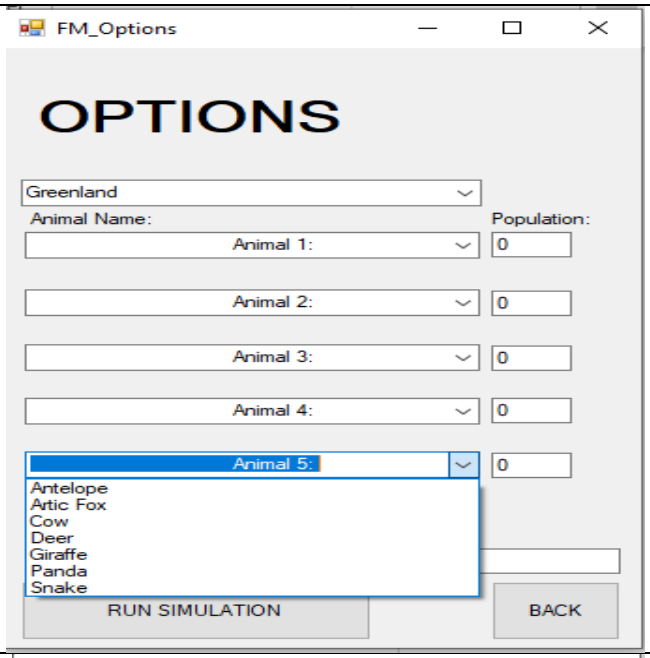
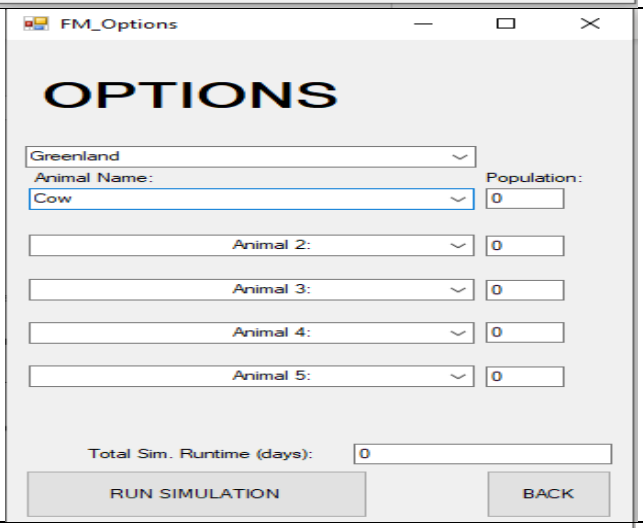
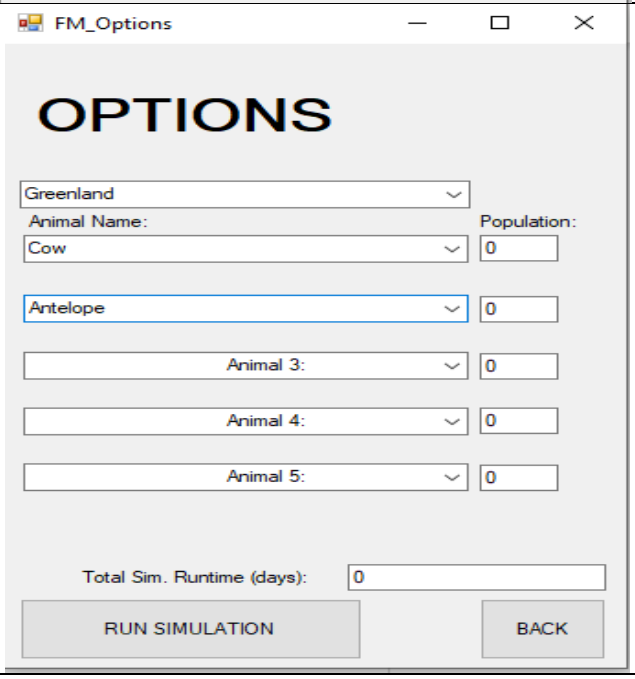
Screenshot _ No.	Screenshot Evidence
1.	
3.	
4.	

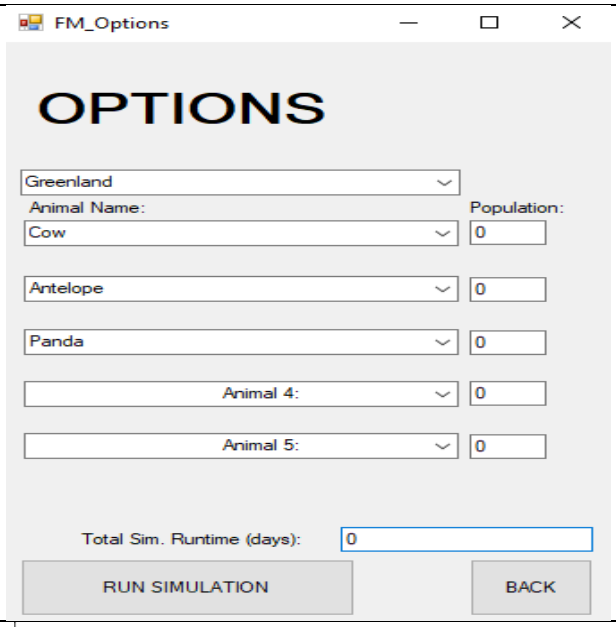
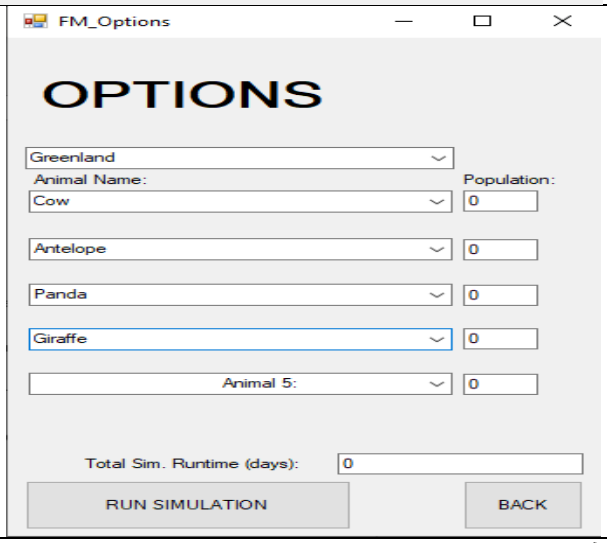
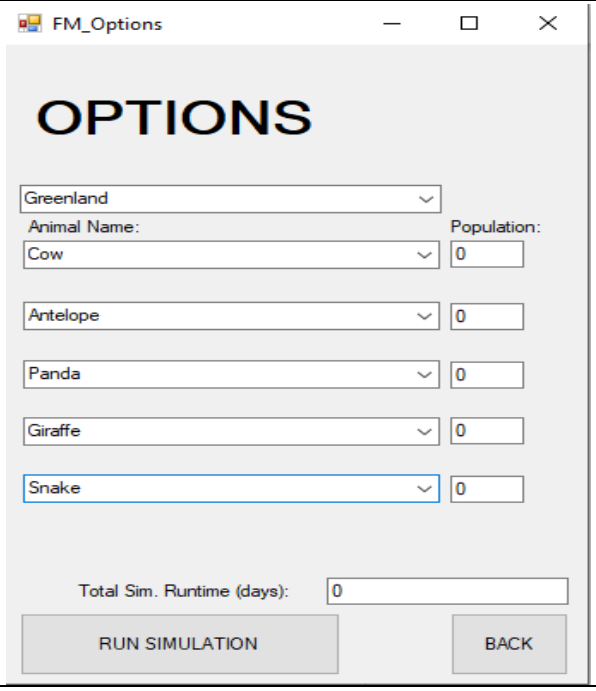
5.	 FM_Options OPTIONS Terrain Type Animal Name: Population: Animal 1: 0 Animal 2: 0 Animal 3: 0 Animal 4: 0 Animal 5: 0 Total Sim. Runtime (days): 0 RUN SIMULATION BACK	
6.	 FM_Options OPTIONS Terrain Type Desert Greenland Arctic Population: 0 Animal 2: 0 Animal 3: 0 Animal 4: 0 Animal 5: 0 Total Sim. Runtime (days): 0 RUN SIMULATION BACK	

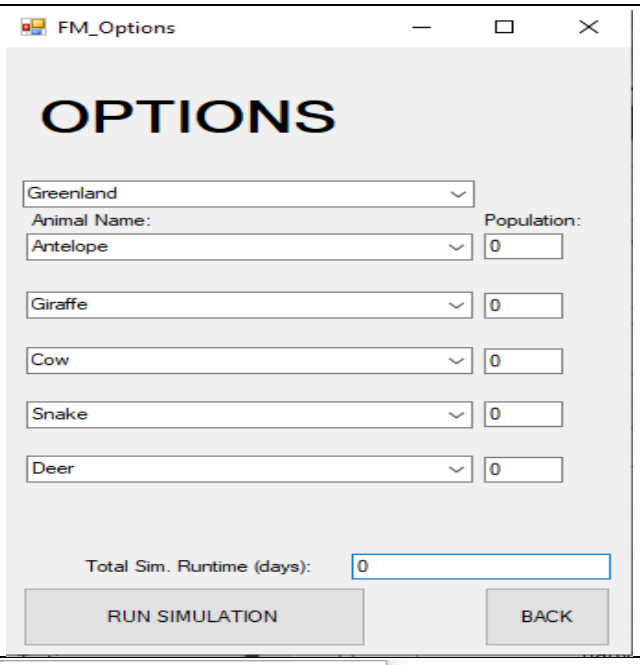
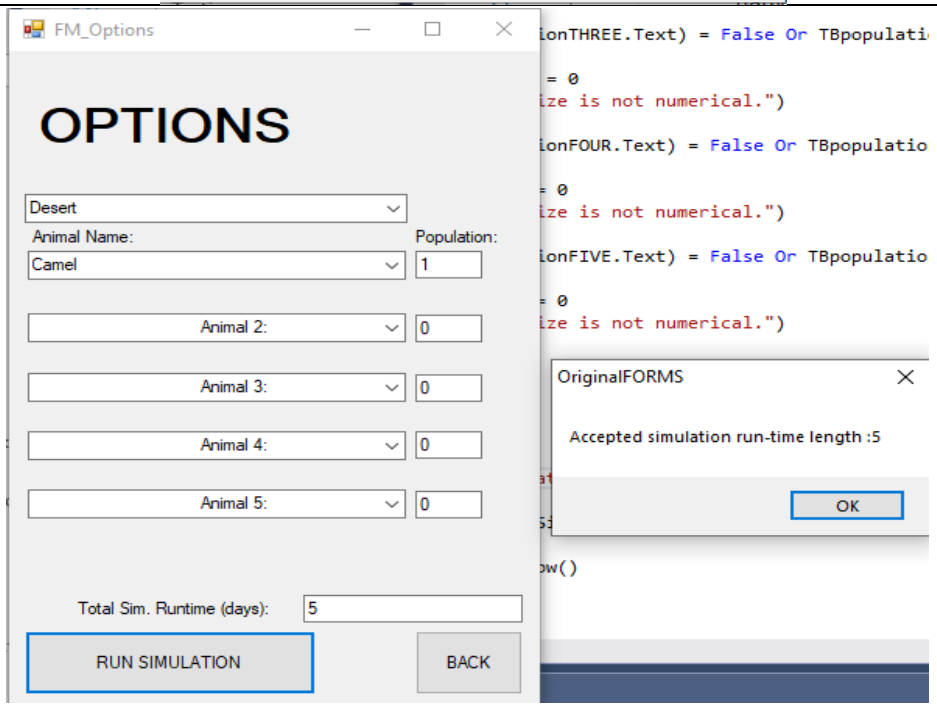
7.	 FM_Options OPTIONS Greenland Animal Name: Population: Animal 1: Animal 2: Animal 3: Animal 4: Animal 5: 0 Total Sim. Runtime (days): RUN SIMULATION BACK
8.	 FM_Options OPTIONS Desert Animal Name: Population: Animal 1: Animal 2: Animal 3: Animal 4: Animal 5: 0 Total Sim. Runtime (days): RUN SIMULATION BACK
9.	 FM_Options OPTIONS Arctic Animal Name: Population: Animal 1: Animal 2: Animal 3: Animal 4: Animal 5: 0 Total Sim. Runtime (days): RUN SIMULATION BACK

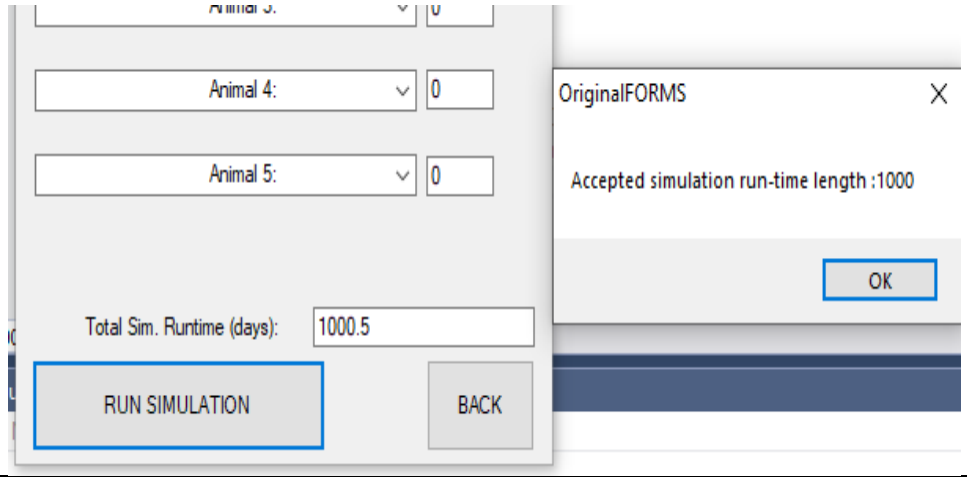
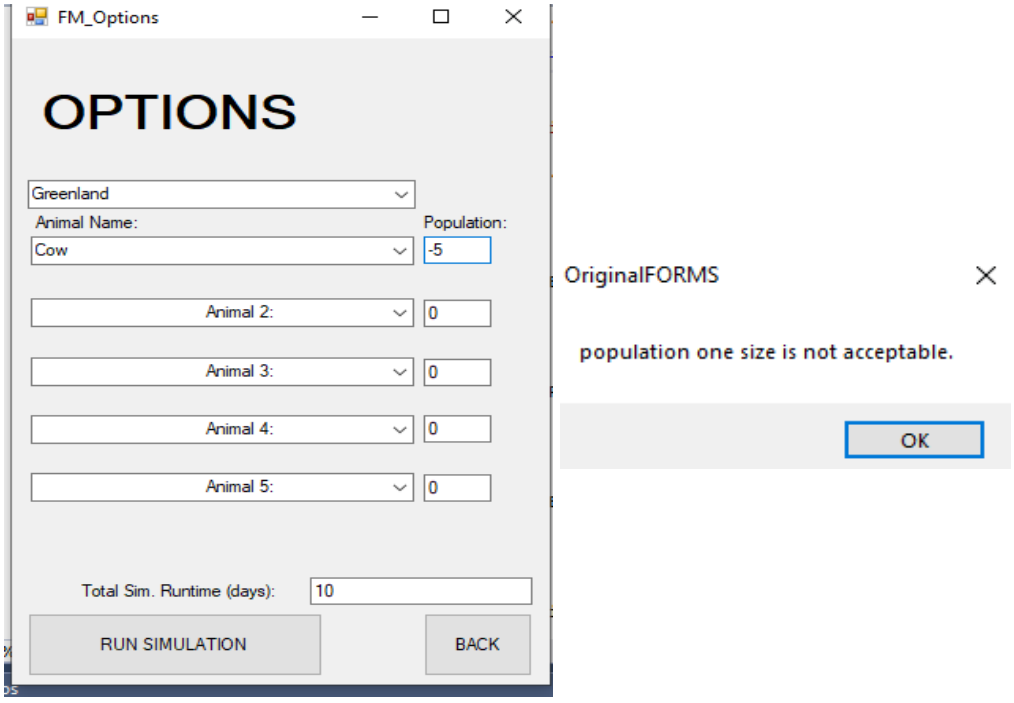
10.	  [TERRAIN TYPES ARE INTER-CHANGEABLE] 
11.	

12.	 FM_Options OPTIONS Greenland Animal Name: Population: Animal 1: 0 Animal 2: 0 Antelope Artic Fox Cow Deer Giraffe Panda Snake 0 0 Animal 5: 0 Total Sim. Runtime (days): 0 RUN SIMULATION BACK
13.	 FM_Options OPTIONS Greenland Animal Name: Population: Animal 1: 0 Animal 2: 0 Animal 3: 0 Antelope Artic Fox Cow Deer Giraffe Panda Snake 0 0 Total Sim. Runtime (days): 0 RUN SIMULATION BACK
14.	 FM_Options OPTIONS Greenland Animal Name: Population: Animal 1: 0 Animal 2: 0 Animal 3: 0 Animal 4: 0 Antelope Artic Fox Cow Deer Giraffe Panda Snake 0 Total Sim. Runtime (days): 0 RUN SIMULATION BACK

15.	 FM_Options OPTIONS Greenland Animal Name: Population: Animal 1: 0 Animal 2: 0 Animal 3: 0 Animal 4: 0 Animal 5: 0 Antelope Artic Fox Cow Deer Giraffe Panda Snake RUN SIMULATION BACK	
16.	 FM_Options OPTIONS Greenland Animal Name: Population: Cow 0 Animal 2: 0 Animal 3: 0 Animal 4: 0 Animal 5: 0 Total Sim. Runtime (days): 0 RUN SIMULATION BACK	
17.	 FM_Options OPTIONS Greenland Animal Name: Population: Cow 0 Antelope 0 Animal 3: 0 Animal 4: 0 Animal 5: 0 Total Sim. Runtime (days): 0 RUN SIMULATION BACK	

18.		 FM_Options OPTIONS Greenland Animal Name: Population: Cow 0 Antelope 0 Panda 0 Animal 4: 0 Animal 5: 0 Total Sim. Runtime (days): 0 RUN SIMULATION BACK
19.		 FM_Options OPTIONS Greenland Animal Name: Population: Cow 0 Antelope 0 Panda 0 Giraffe 0 Animal 5: 0 Total Sim. Runtime (days): 0 RUN SIMULATION BACK
20.		 FM_Options OPTIONS Greenland Animal Name: Population: Cow 0 Antelope 0 Panda 0 Giraffe 0 Snake 0 Total Sim. Runtime (days): 0 RUN SIMULATION BACK

21.	 FM_Options OPTIONS Greenland Animal Name: Population: Antelope 0 Giraffe 0 Cow 0 Snake 0 Deer 0 Total Sim. Runtime (days): 0 RUN SIMULATION BACK
22.	 FM_Options OPTIONS Desert Animal Name: Population: Camel 1 Animal 2: 0 Animal 3: 0 Animal 4: 0 Animal 5: 0 Total Sim. Runtime (days): 5 RUN SIMULATION BACK ionTHREE.Text) = False Or TBpopulati = 0 ize is not numerical.") ionFOUR.Text) = False Or TBpopulatio = 0 ize is not numerical.") ionFIVE.Text) = False Or TBpopulatio = 0 ize is not numerical.") OriginalFORMS Accepted simulation run-time length :5 OK ow()

23.	 Animal 4: 0 Animal 5: 0 Total Sim. Runtime (days): 1000.5 RUN SIMULATION BACK OriginalFORMS Accepted simulation run-time length :1000 OK
24.	 FM_Options OPTIONS Greenland Animal Name: Population: Cow -5 Animal 2: 0 Animal 3: 0 Animal 4: 0 Animal 5: 0 Total Sim. Runtime (days): 10 RUN SIMULATION BACK OriginalFORMS population one size is not acceptable. OK

25.

```
    MsgBox("population 5 size is not numerical.")

ElseIf TBtotalRunduration.Text < 0 Then
    TBtotalRunduration.Text = 0
    MsgBox("Simulation run-time length v
Else
    validInput = True
End If

If validInput = True Then

    cl_GlobalSetting.TotalSimulationTime
    MsgBox("Accepted simulation run-time
    setup()
    Me.Close()
```

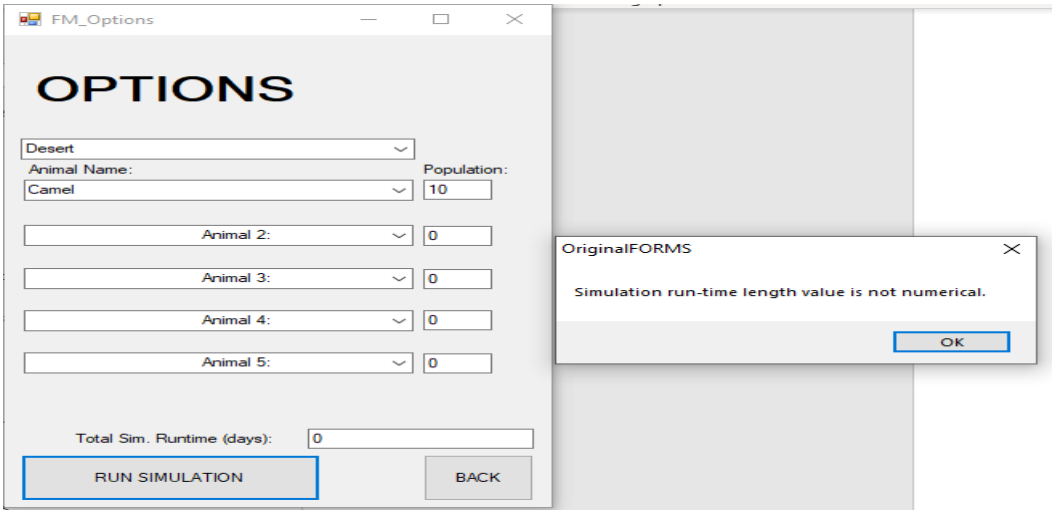
Exception Unhandled

System.InvalidCastException: 'Conversion from string "gda" to type 'Double' is not valid.'

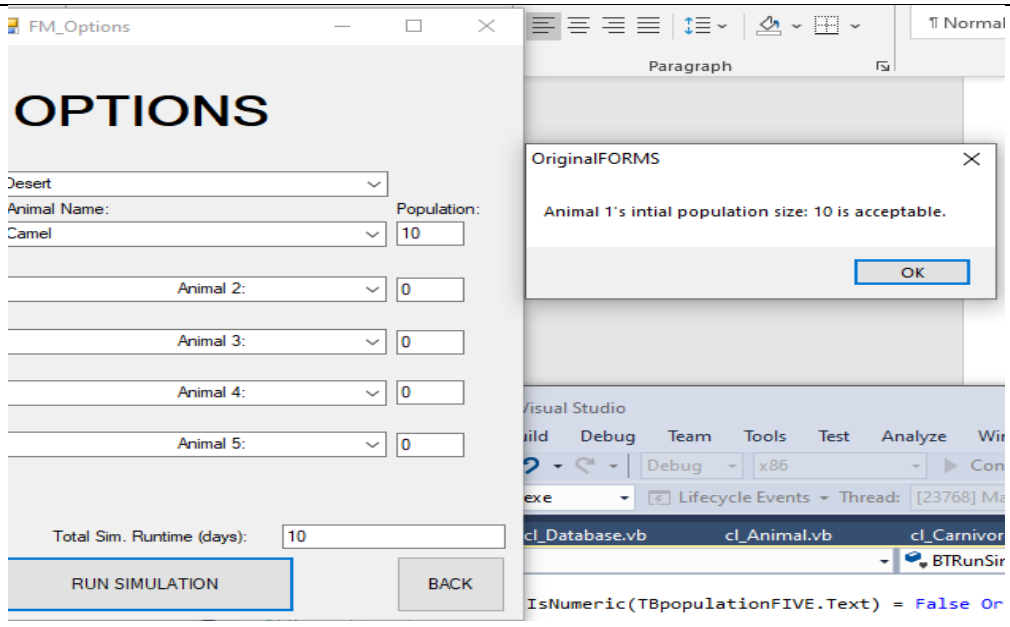
Inner Exception
FormatException: Input string was not in a correct format.

View Details | Copy Details

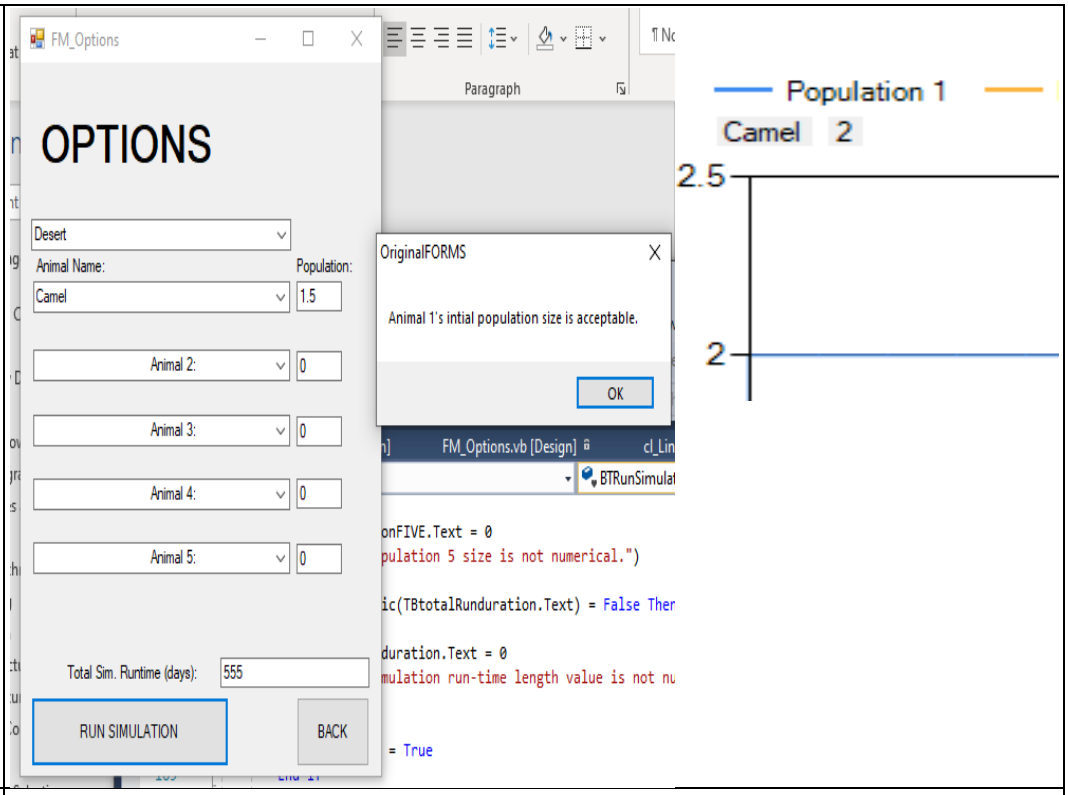
Exception Settings



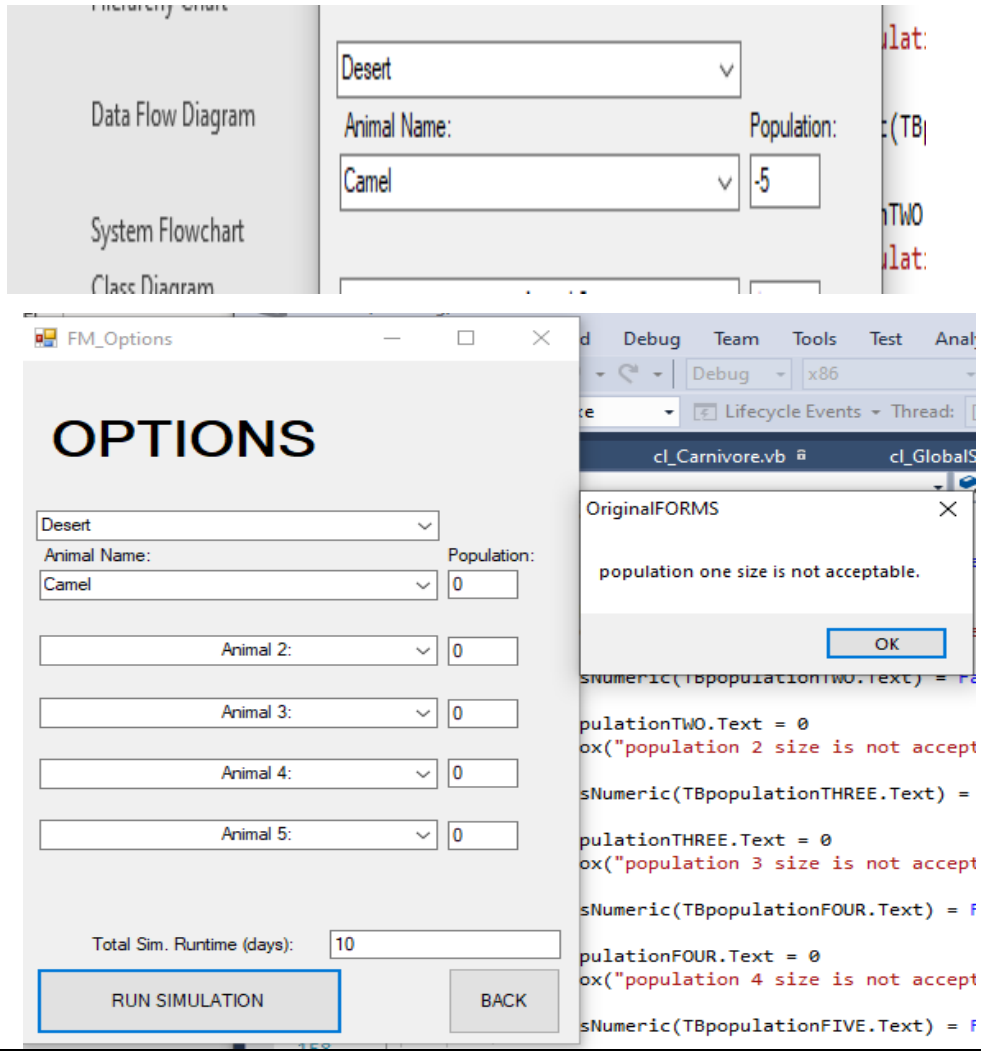
26.

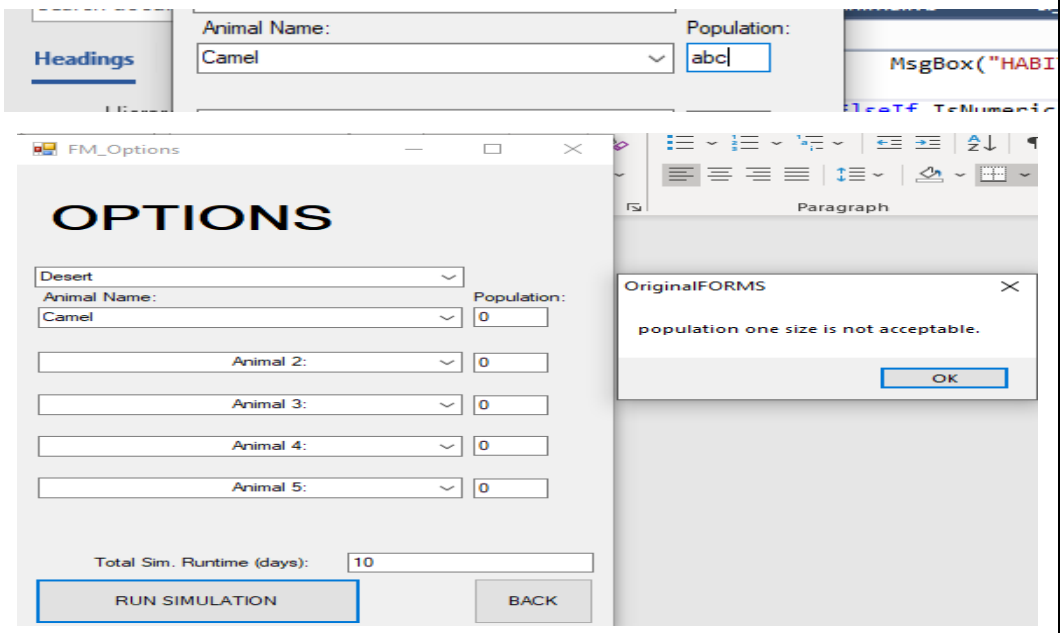
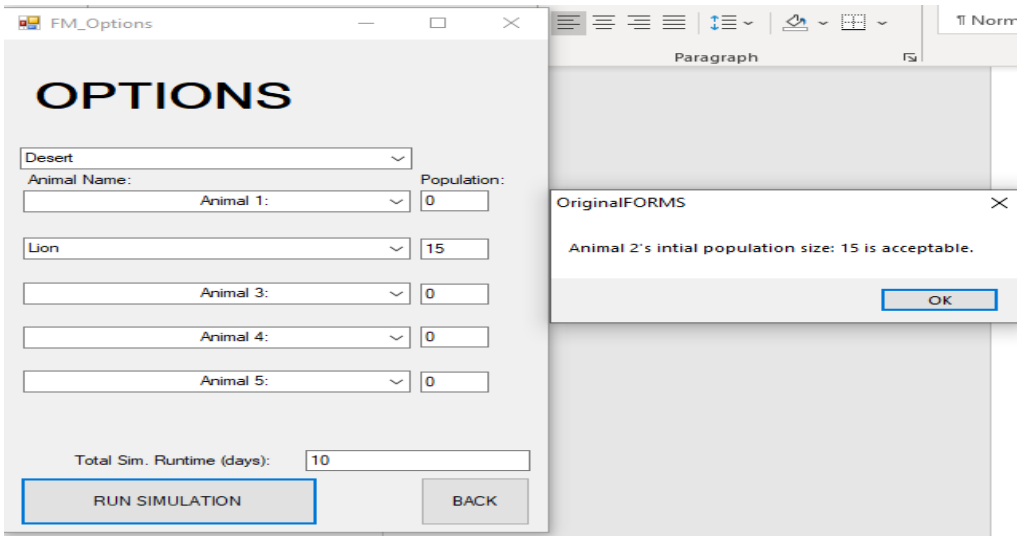


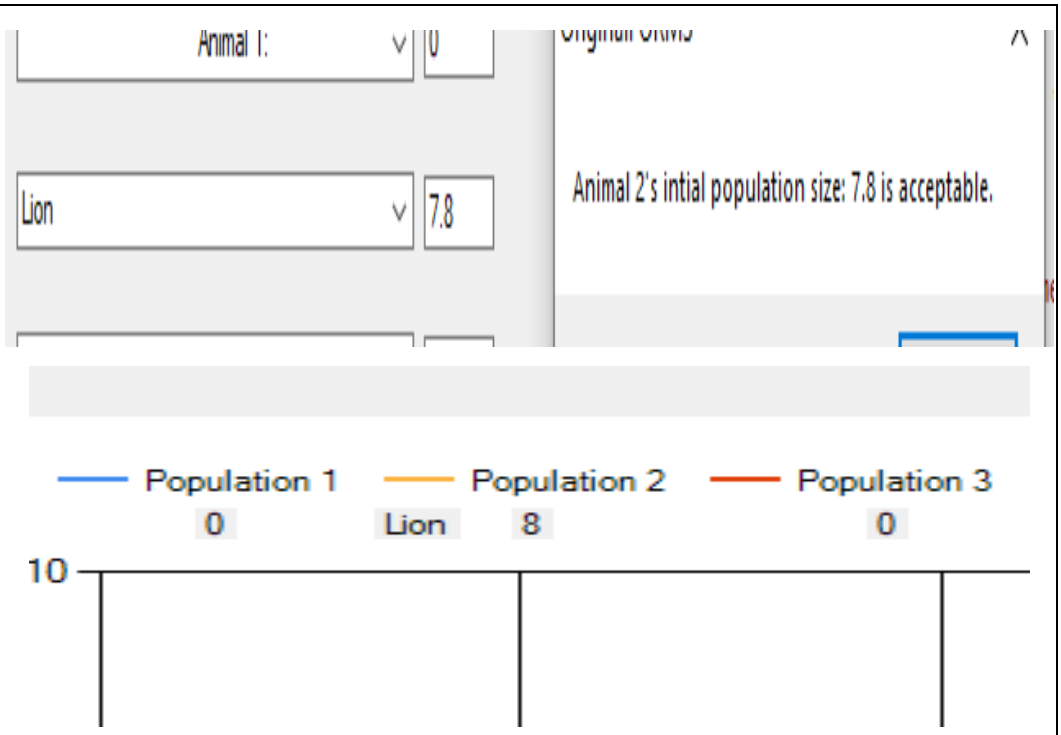
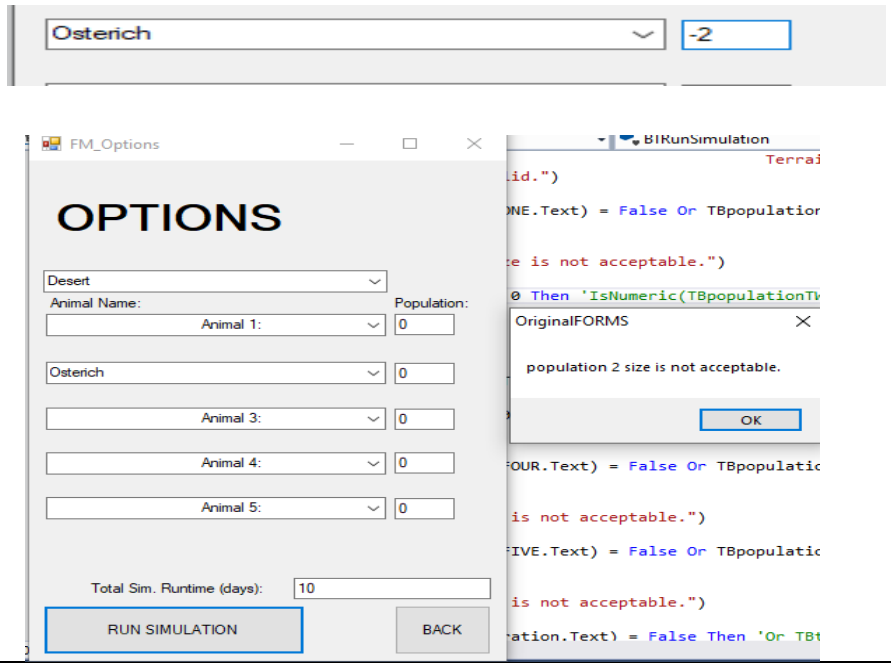
27.

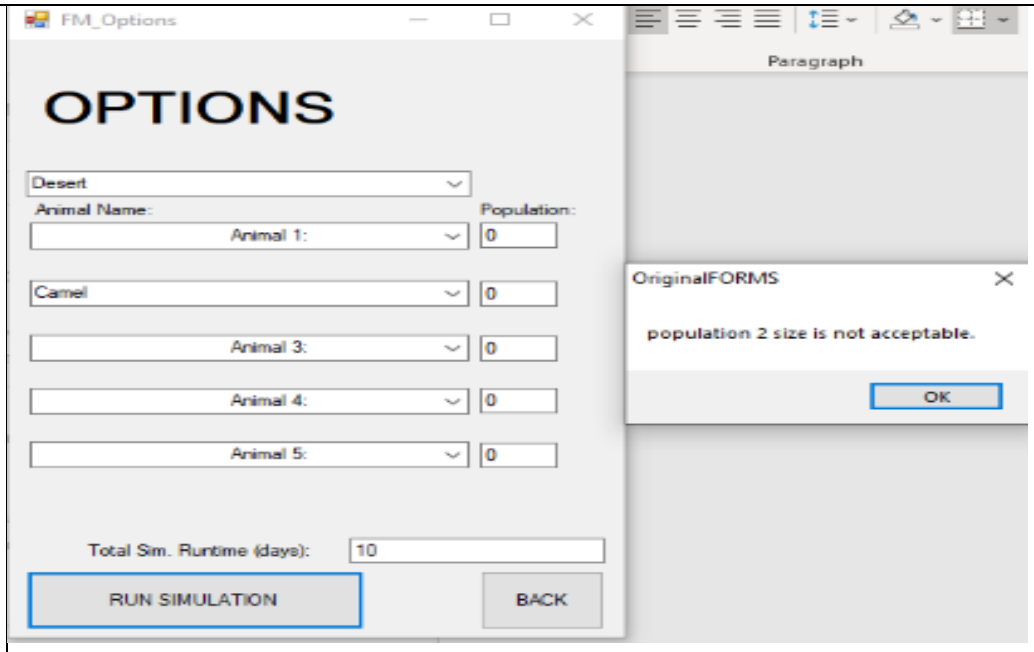

FM_Options
OPTIONS
Desert
Animal Name:
Population:
Camel
1.5
Animal 2:
0
Animal 3:
0
Animal 4:
0
Animal 5:
0
Total Sim. Runtime (days):
555
RUN SIMULATION
BACK
OriginalFORMS
Animal 1's intial population size is acceptable.
OK
Paragraph
FM_Options.vb [Design]
onFIVE.Text = 0
pulation 5 size is not numerical.")
ic(TBtotalRunduration.Text) = False Ther
duration.Text = 0
mulation run-time length value is not nu
= True
Population 1
Camel 2
2.5
2

28.

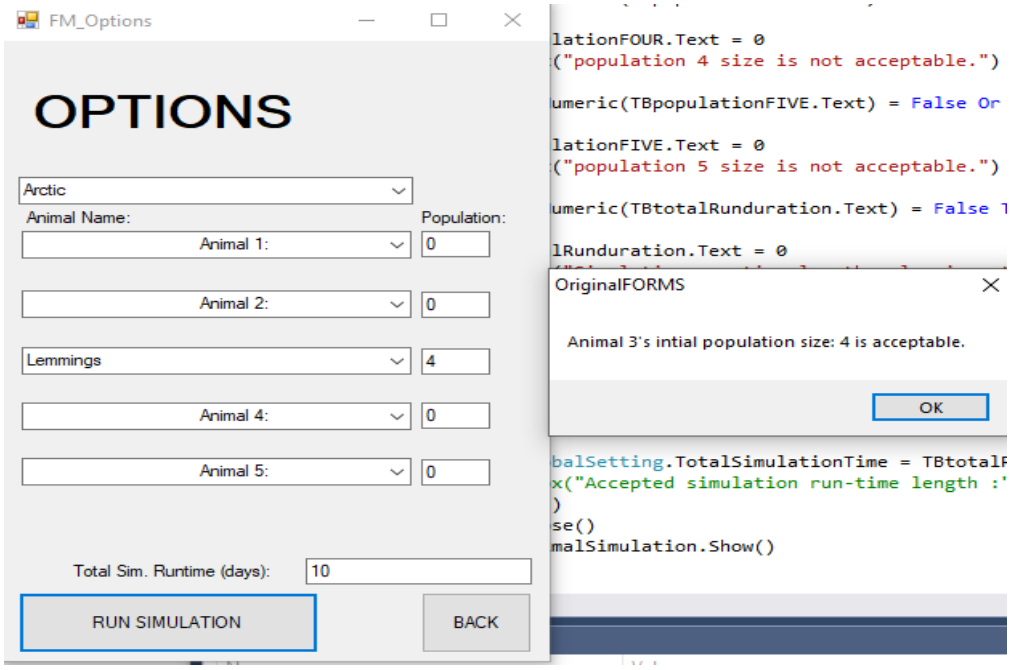

Data Flow Diagram
System Flowchart
Desert
Animal Name:
Population:
Camel
-5
FM_Options
OPTIONS
Desert
Animal Name:
Population:
Camel
0
Animal 2:
0
Animal 3:
0
Animal 4:
0
Animal 5:
0
Total Sim. Runtime (days):
10
RUN SIMULATION
BACK
Debug
Team
Tools
Test
Debug
x86
Lifecycle Events
Thread:
cl_Carnivore.vb
cl_GlobalS
OriginalFORMS
population one size is not acceptable.
OK
pulationTWO.Text = 0
ox("population 2 size is not accept
sNumeric(TBpopulationTHREE.Text) =
pulationTHREE.Text = 0
ox("population 3 size is not accept
sNumeric(TBpopulationFOUR.Text) =
pulationFOUR.Text = 0
ox("population 4 size is not accept
sNumeric(TBpopulationFIVE.Text) =

29.	 Headings Animal Name: Camel Population: abc MsgBox("HABI FM_Options OPTIONS Desert Animal Name: Camel Population: 0 Animal 2: 0 Animal 3: 0 Animal 4: 0 Animal 5: 0 Total Sim. Runtime (days): 10 RUN SIMULATION BACK Paragraph OriginalFORMS population one size is not acceptable. OK
30.	 FM_Options OPTIONS Desert Animal Name: Animal 1: Population: 0 Lion 15 Animal 3: 0 Animal 4: 0 Animal 5: 0 Total Sim. Runtime (days): 10 RUN SIMULATION BACK Paragraph OriginalFORMS Animal 2's intial population size: 15 is acceptable. OK

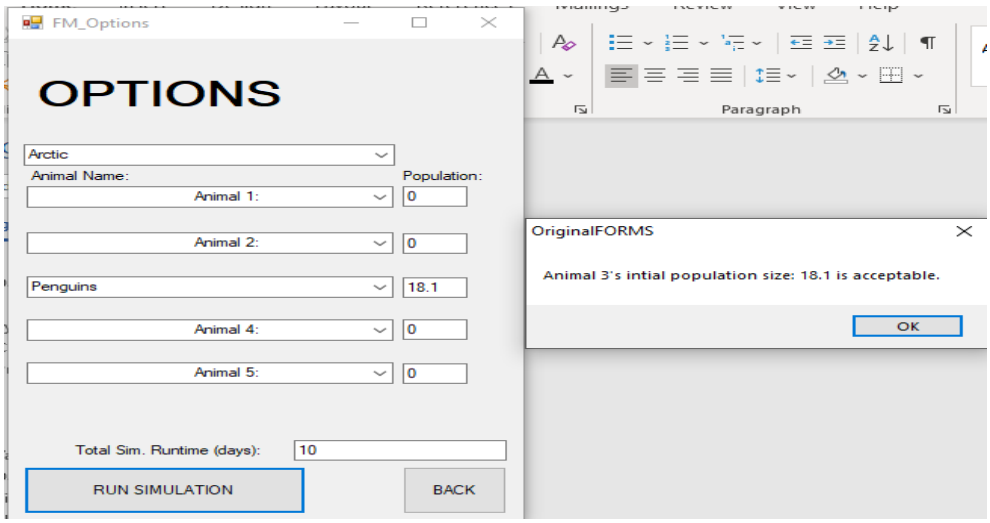
31.	 Animal 1: 0 Lion 7.8 Animal 2's intial population size: 7.8 is acceptable. Population 1 Population 2 Population 3 0 Lion 8 0 10
32.	 Osterich -2 FM_Options OPTIONS Desert Animal Name: Population: Animal 1: 0 Osterich 0 Animal 3: 0 Animal 4: 0 Animal 5: 0 Total Sim. Runtime (days): 10 RUN SIMULATION BACK BtRunSimulation OriginalFORMS population 2 size is not acceptable. OK
33.	 Camel a!qv!?


FM_Options
OPTIONS
Desert
Animal Name:
Population:
Animal 1:
0
Camel
0
Animal 3:
0
Animal 4:
0
Animal 5:
0
Total Sim. Runtime (days):
10
RUN SIMULATION
BACK
Paragraph
OriginalFORMS
population 2 size is not acceptable.
OK

34.

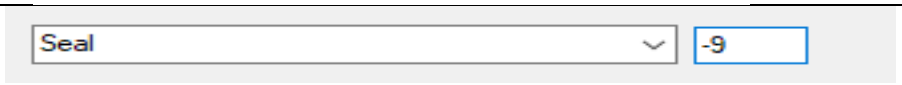

FM_Options
OPTIONS
Arctic
Animal Name:
Population:
Animal 1:
0
Animal 2:
0
Lemmings
4
Animal 4:
0
Animal 5:
0
Total Sim. Runtime (days):
10
RUN SIMULATION
BACK
lationFOUR.Text = 0
("population 4 size is not acceptable.")
umeric(TBpopulationFIVE.Text) = False Or
lationFIVE.Text = 0
("population 5 size is not acceptable.")
umeric(TBtotalRunduration.Text) = False
lRunduration.Text = 0
OriginalFORMS
Animal 3's intial population size: 4 is acceptable.
OK
balSetting.TotalSimulationTime = TBtotal
x("Accepted simulation run-time length :'
)
se()
malSimulation.Show()

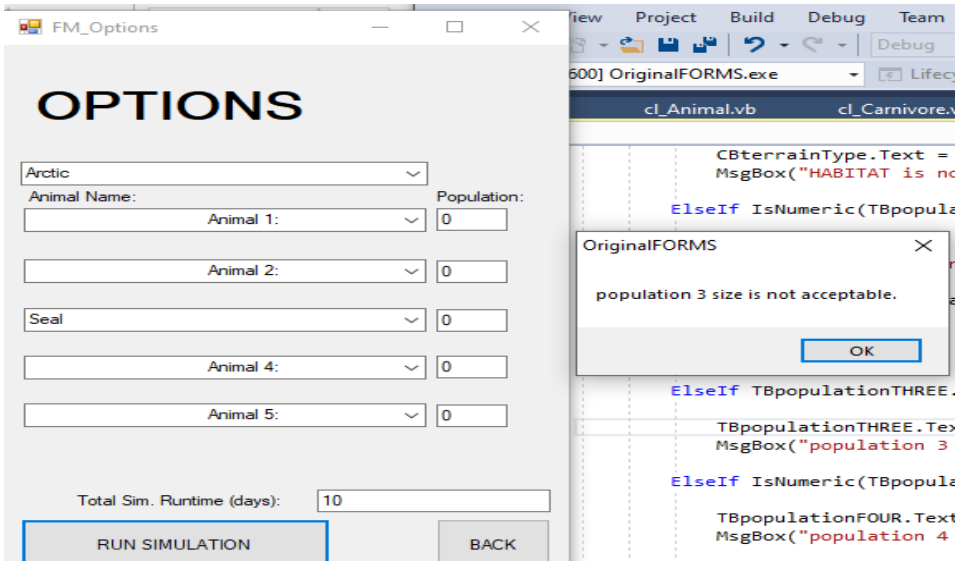
35.


FM_Options
OPTIONS
Arctic
Animal Name:
Population:
Animal 1:
0
Animal 2:
0
Penguins
18.1
Animal 4:
0
Animal 5:
0
Total Sim. Runtime (days):
10
RUN SIMULATION
BACK
Paragraph
OriginalFORMS
Animal 3's intial population size: 18.1 is acceptable.
OK

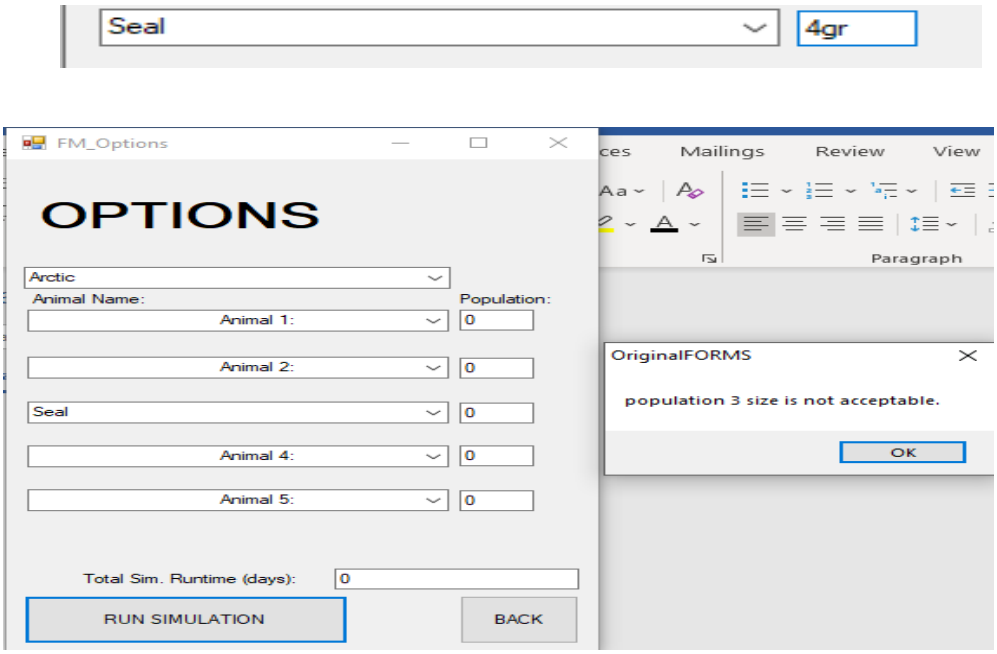

Population 3
Penguins: 18

36.

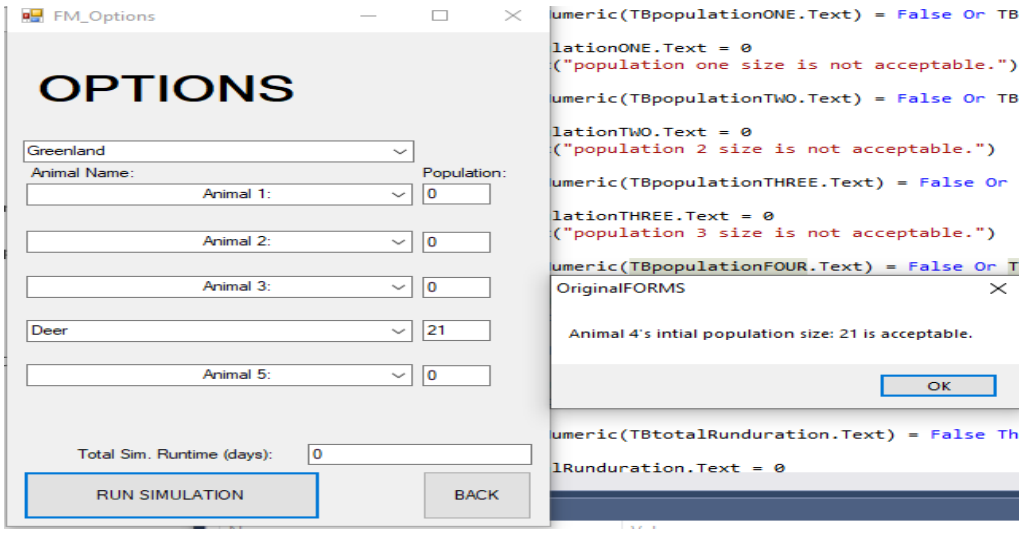

Seal
-9


FM_Options
OPTIONS
Arctic
Animal Name:
Population:
Animal 1:
0
Animal 2:
0
Seal
0
Animal 4:
0
Animal 5:
0
Total Sim. Runtime (days):
10
RUN SIMULATION
BACK
Project
Build
Debug
Team
Debug
cl_Animal.vb
cl_Carnivore.v
CBterrainType.Text =
MsgBox("HABITAT is nc
ElseIf IsNumeric(TBpopula
OriginalFORMS
population 3 size is not acceptable.
OK
ElseIf TBpopulationTHREE.
TBpopulationTHREE.Tex
MsgBox("population 3
ElseIf IsNumeric(TBpopula
TBpopulationFOUR.Text
MsgBox("population 4

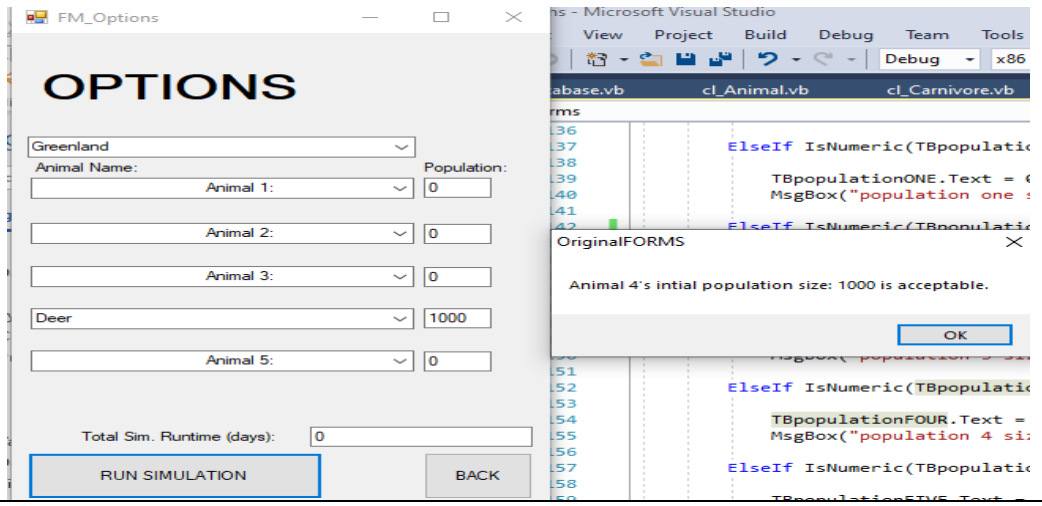
37.

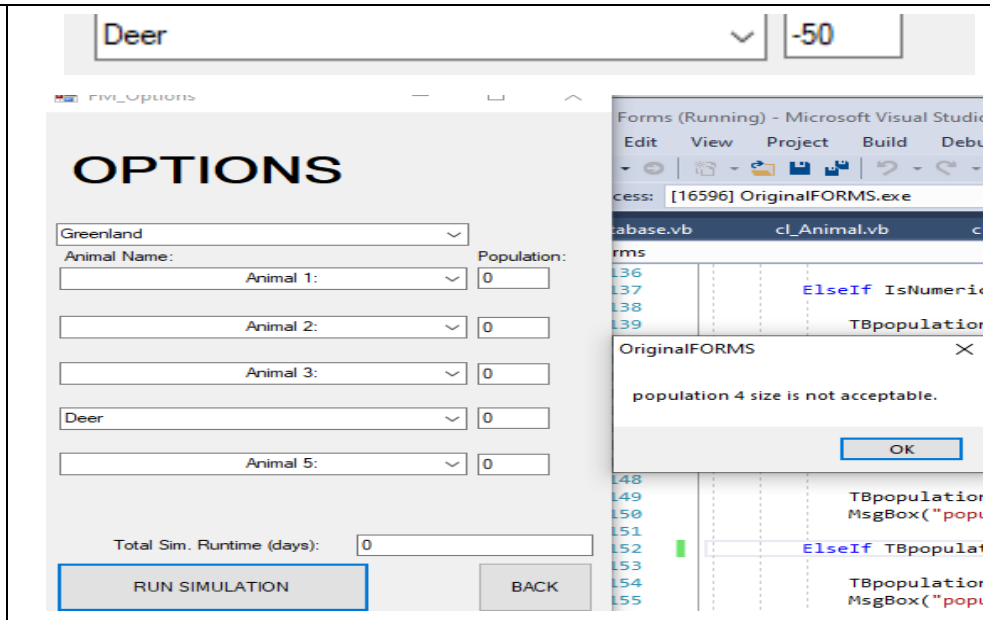
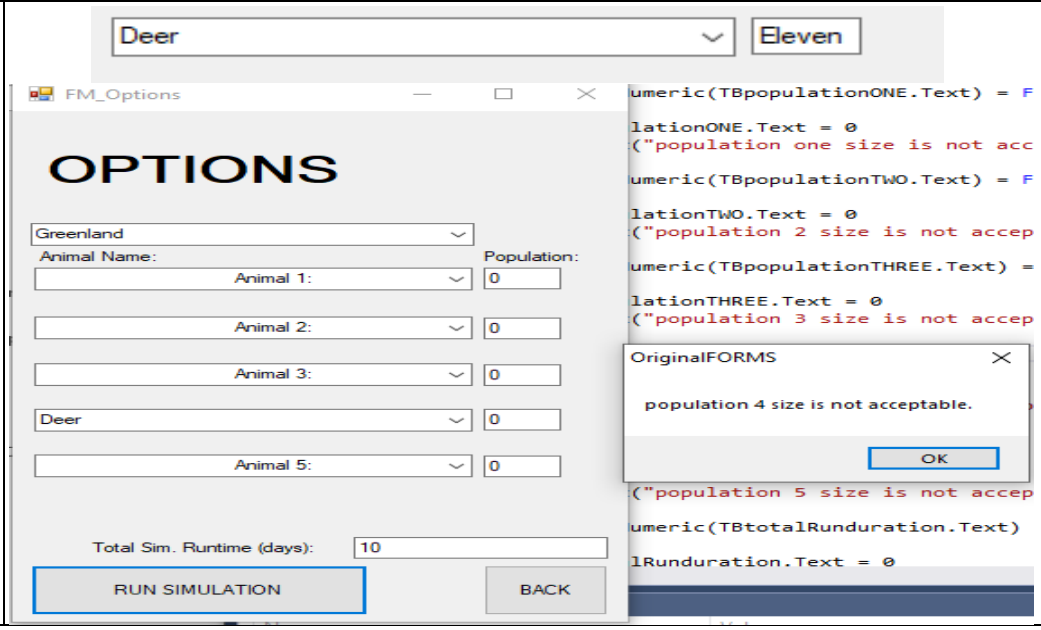

Seal
4gr
FM_Options
OPTIONS
Arctic
Animal Name:
Population:
Animal 1:
0
Animal 2:
0
Seal
0
Animal 4:
0
Animal 5:
0
Total Sim. Runtime (days):
0
RUN SIMULATION
BACK
Mailings
Review
View
Paragraph
OriginalFORMS
population 3 size is not acceptable.
OK

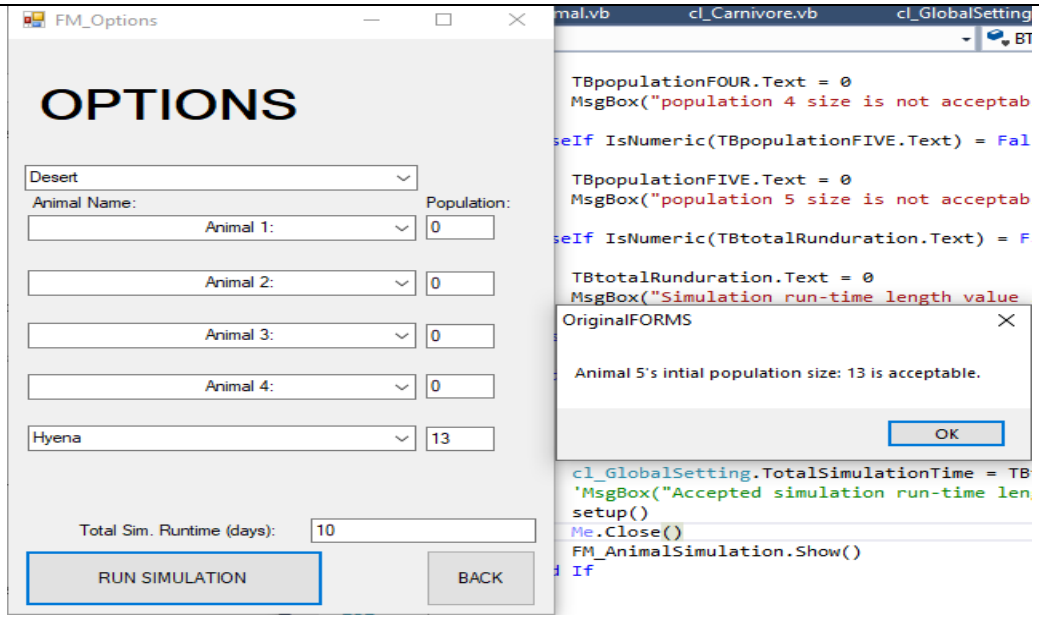
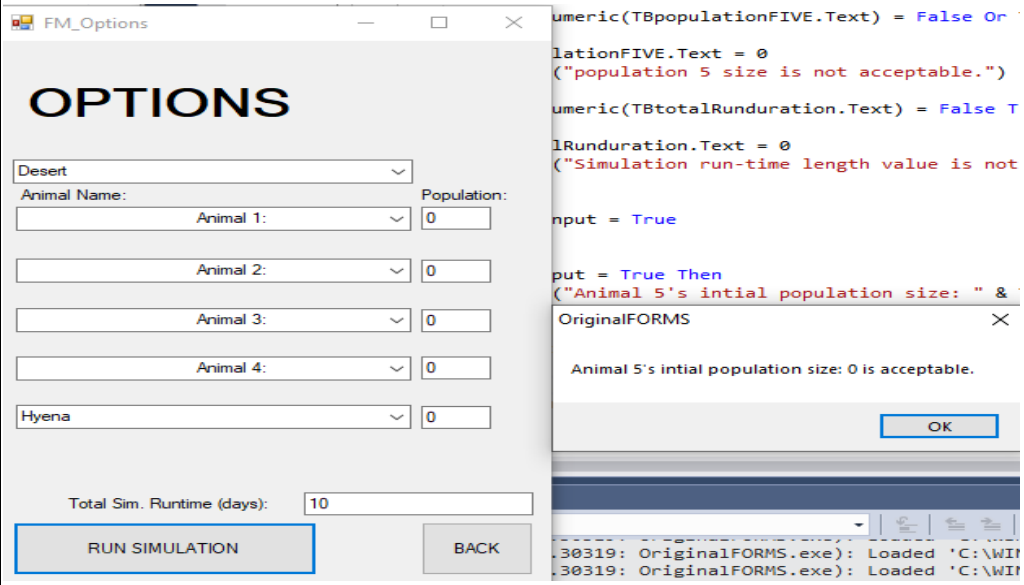
38.

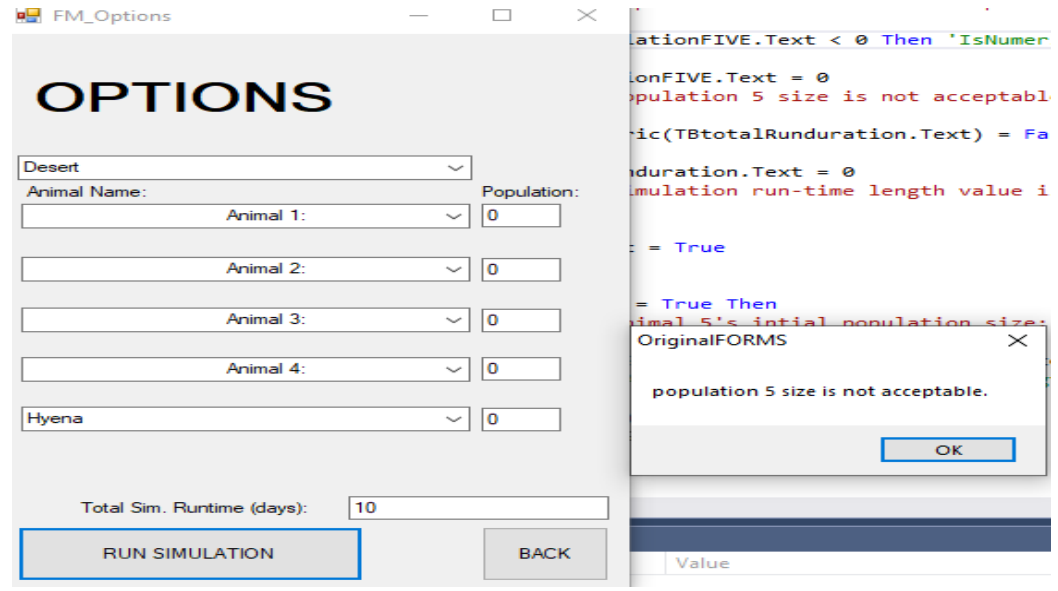
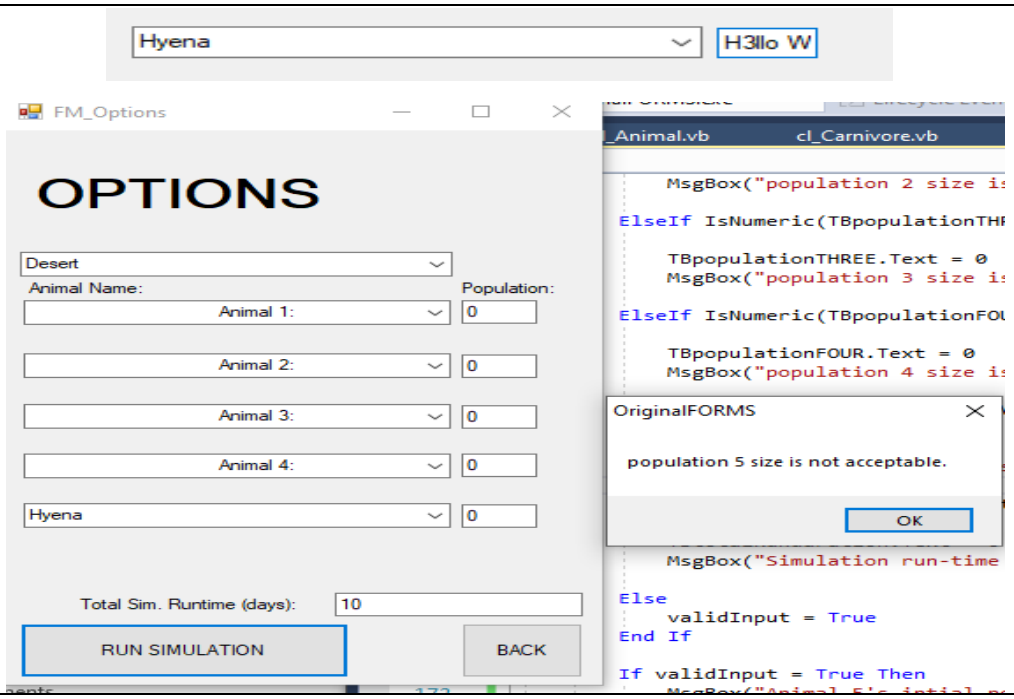
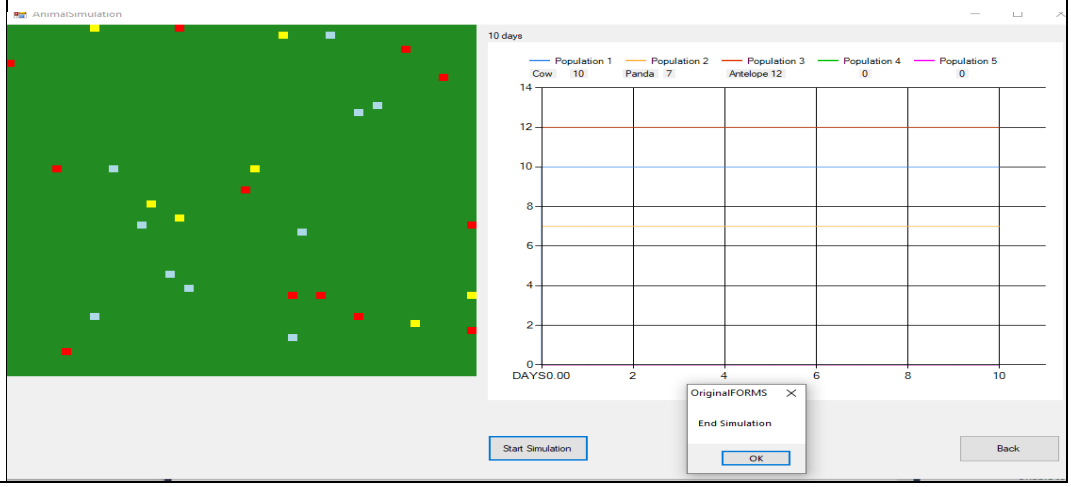

FM_Options
OPTIONS
Greenland
Animal Name:
Population:
Animal 1:
0
Animal 2:
0
Animal 3:
0
Deer
21
Animal 5:
0
Total Sim. Runtime (days):
0
RUN SIMULATION
BACK
umeric(TBpopulationONE.Text) = False Or TB
lationONE.Text = 0
("population one size is not acceptable.")
umeric(TBpopulationTWO.Text) = False Or TB
lationTWO.Text = 0
("population 2 size is not acceptable.")
umeric(TBpopulationTHREE.Text) = False Or
lationTHREE.Text = 0
("population 3 size is not acceptable.")
umeric(TBpopulationFOUR.Text) = False Or T
OriginalFORMS
Animal 4's intial population size: 21 is acceptable.
OK
umeric(TBtotalRunduration.Text) = False Th
lRunduration.Text = 0

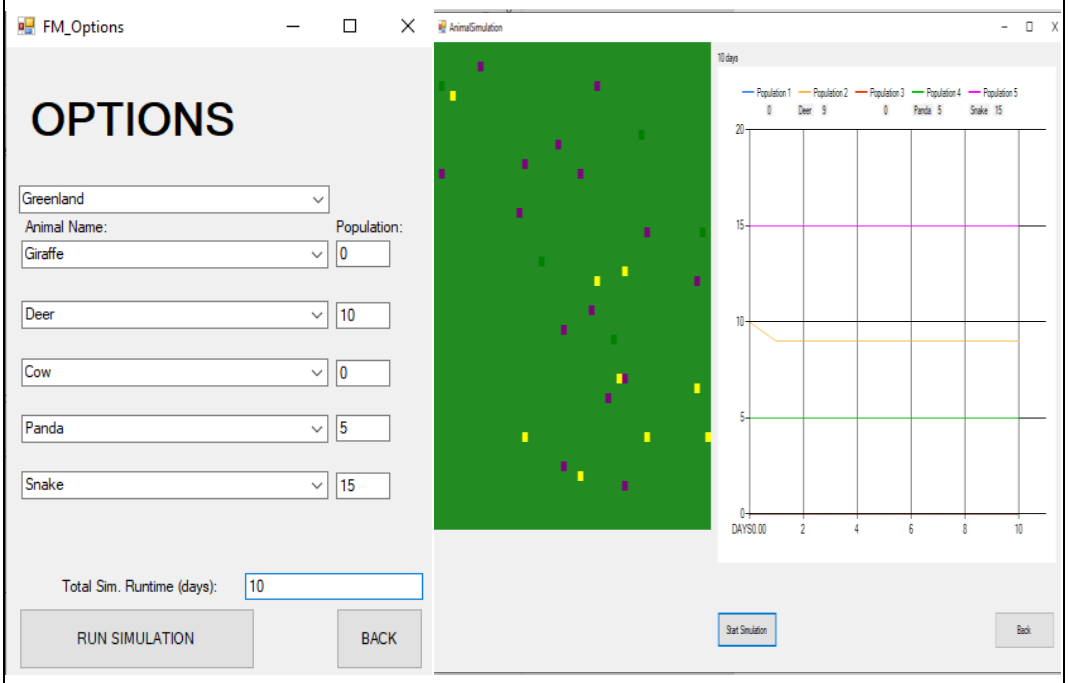
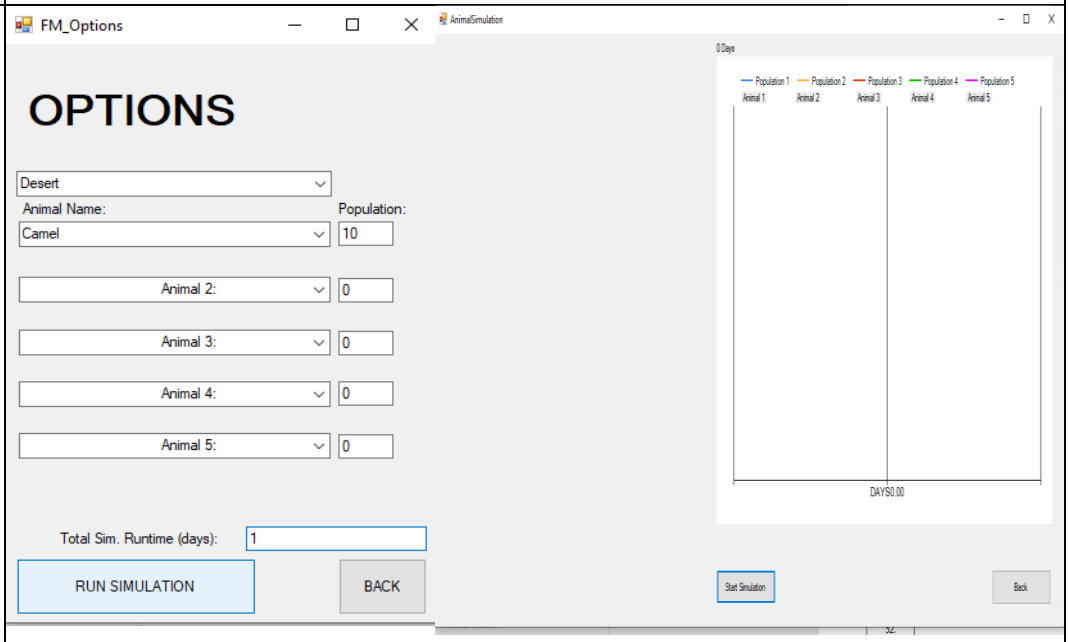
39.

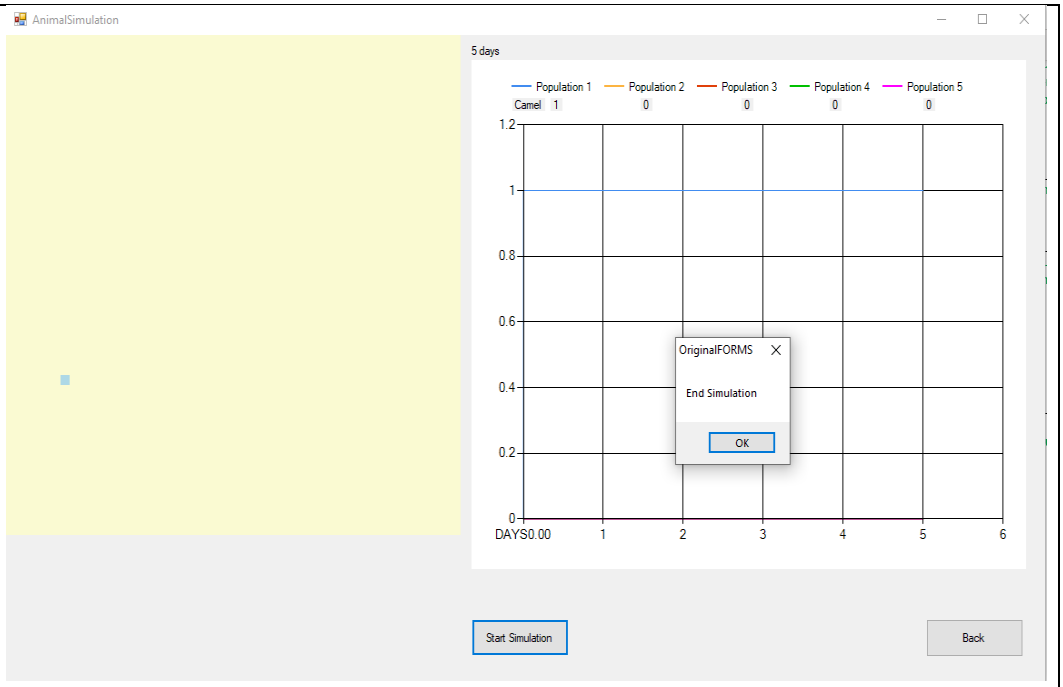

FM_Options
OPTIONS
Greenland
Animal Name:
Population:
Animal 1:
0
Animal 2:
0
Animal 3:
0
Deer
1000
Animal 5:
0
Total Sim. Runtime (days):
0
RUN SIMULATION
BACK
Microsoft Visual Studio
View
Project
Build
Debug
Team
Tools
Debug
x86
cl_Animal.vb
cl_Carnivore.vb
ElseIf IsNumeric(TBpopulatio
TBpopulationONE.Text =
MsgBox("population one
OriginalFORMS
Animal 4's intial population size: 1000 is acceptable.
OK
ElseIf IsNumeric(TBpopulatio
TBpopulationFOUR.Text =
MsgBox("population 4 si
ElseIf IsNumeric(TBpopulatio

40.	 Deer -50 OPTIONS Greenland Animal Name: Population: Animal 1: Animal 2: Animal 3: Deer Animal 5: Total Sim. Runtime (days): RUN SIMULATION BACK OriginalFORMS population 4 size is not acceptable. OK
41.	 Deer Eleven FM_Options OPTIONS Greenland Animal Name: Population: Animal 1: Animal 2: Animal 3: Deer Animal 5: Total Sim. Runtime (days): 10 RUN SIMULATION BACK OriginalFORMS population 4 size is not acceptable. OK

42.	 FM_Options OPTIONS Desert Animal Name: Population: Animal 1: 0 Animal 2: 0 Animal 3: 0 Animal 4: 0 Hyena 13 Total Sim. Runtime (days): 10 RUN SIMULATION BACK cl_Carnivore.vb cl_GlobalSetting TBpopulationFOUR.Text = 0 MsgBox("population 4 size is not acceptab seIf IsNumeric(TBpopulationFIVE.Text) = Fal TBpopulationFIVE.Text = 0 MsgBox("population 5 size is not acceptab seIf IsNumeric(TBtotalRunduration.Text) = F TBtotalRunduration.Text = 0 MsgBox("Simulation run-time length value OriginalFORMS Animal 5's intial population size: 13 is acceptable. OK cl_GlobalSetting.TotalSimulationTime = TB 'MsgBox("Accepted simulation run-time len setup() Me.Close() FM_AnimalSimulation.Show() If
43.	 FM_Options OPTIONS Desert Animal Name: Population: Animal 1: 0 Animal 2: 0 Animal 3: 0 Animal 4: 0 Hyena 0 Total Sim. Runtime (days): 10 RUN SIMULATION BACK umeric(TBpopulationFIVE.Text) = False Or lationFIVE.Text = 0 ("population 5 size is not acceptable.") umeric(TBtotalRunduration.Text) = False T lRunduration.Text = 0 ("Simulation run-time length value is not nput = True put = True Then ("Animal 5's intial population size: " & OriginalFORMS Animal 5's intial population size: 0 is acceptable. OK .30319: OriginalFORMS.exe): Loaded 'C:\WI .30319: OriginalFORMS.exe): Loaded 'C:\WI
44.	 Hyena -99

	 FM_Options OPTIONS Desert Animal Name: Population: Animal 1: Animal 2: Animal 3: Animal 4: Hyena 0 Total Sim. Runtime (days): 10 RUN SIMULATION BACK OriginalFORMS population 5 size is not acceptable. OK Value
45.	 Hyena H3llo W FM_Options OPTIONS Desert Animal Name: Population: Animal 1: Animal 2: Animal 3: Animal 4: Hyena 0 Total Sim. Runtime (days): 10 RUN SIMULATION BACK OriginalFORMS population 5 size is not acceptable. OK
46.	 10 days Population 1 Population 2 Population 3 Population 4 Population 5 Cow 10 Panda 7 Antelope 12 0 0 DAYS0.00 OriginalFORMS End Simulation OK Start Simulation Back

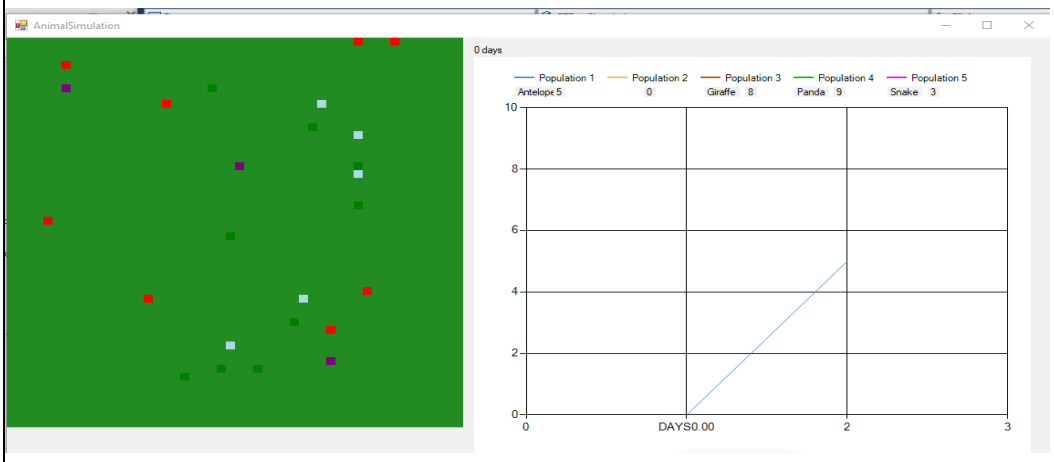
47.	 FM_Options OPTIONS Greenland Animal Name: Population: Giraffe 0 Deer 10 Cow 0 Panda 5 Snake 15 Total Sim. Runtime (days): 10 RUN SIMULATION BACK AnimalSimulation 10 days Population 1 Population 2 Population 3 Population 4 Population 5 0 Deer 9 0 Panda 5 Snake 15 20 15 10 5 0 DAYS0.00 2 4 6 8 10 Start Simulation Back
48.	 FM_Options OPTIONS Desert Animal Name: Population: Camel 10 Animal 2: 0 Animal 3: 0 Animal 4: 0 Animal 5: 0 Total Sim. Runtime (days): 1 RUN SIMULATION BACK AnimalSimulation 0 Days Population 1 Population 2 Population 3 Population 4 Population 5 Animal 1 Animal 2 Animal 3 Animal 4 Animal 5 DAYS0.00 Start Simulation Back
49.	

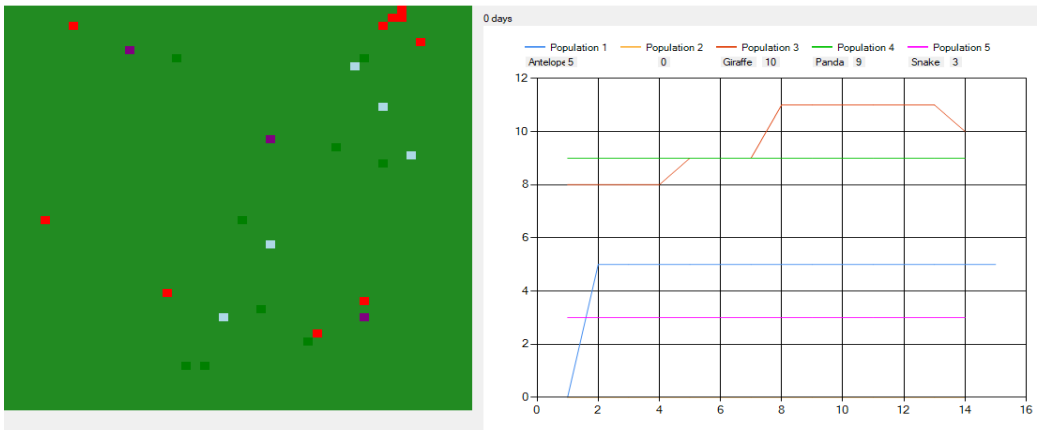

AnimalSimulation
5 days
Population 1
Population 2
Population 3
Population 4
Population 5
Camel 1
0
0
0
0
1.2
1
0.8
0.6
0.4
0.2
0
DAYS0.00
1
2
3
4
5
6
OriginalFORMS
End Simulation
OK
Start Simulation
Back

50.

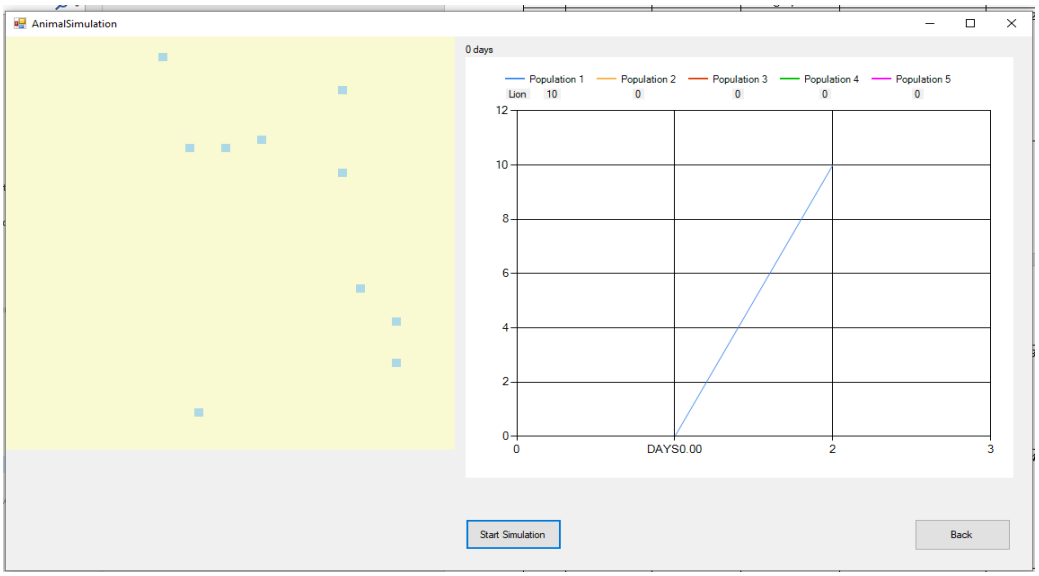

Population 1
Population 2
Population 3
Population 4
Population 5
Antelope 10
Cow 3
Deer 7
Panda 23
Snake 18

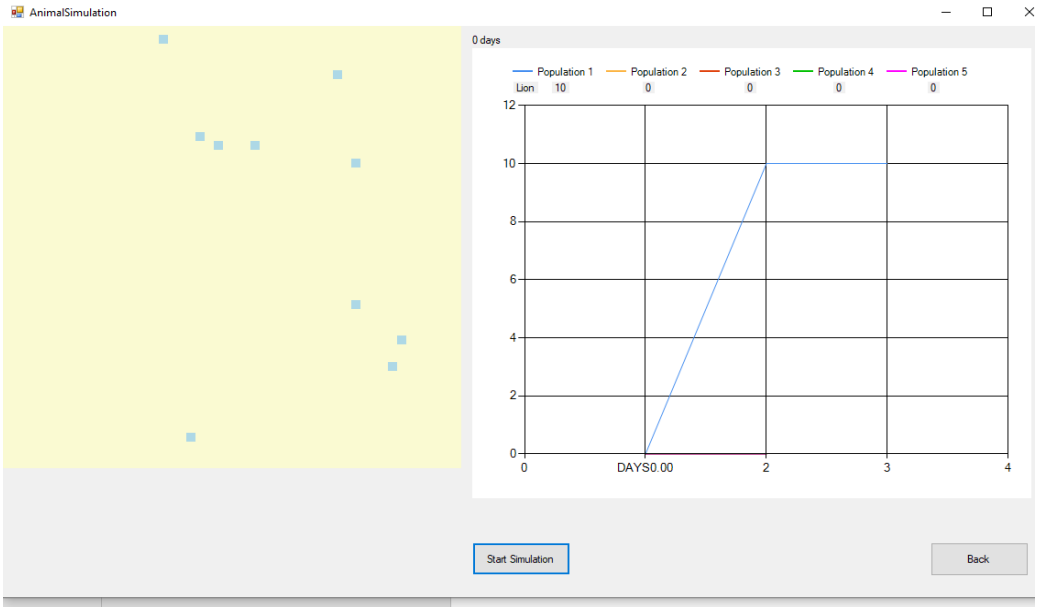
51.


AnimalSimulation
0 days
Population 1
Population 2
Population 3
Population 4
Population 5
Antelope 5
0
Giraffe 8
Panda 9
Snake 3
10
8
6
4
2
0
DAYS0.00
2
3

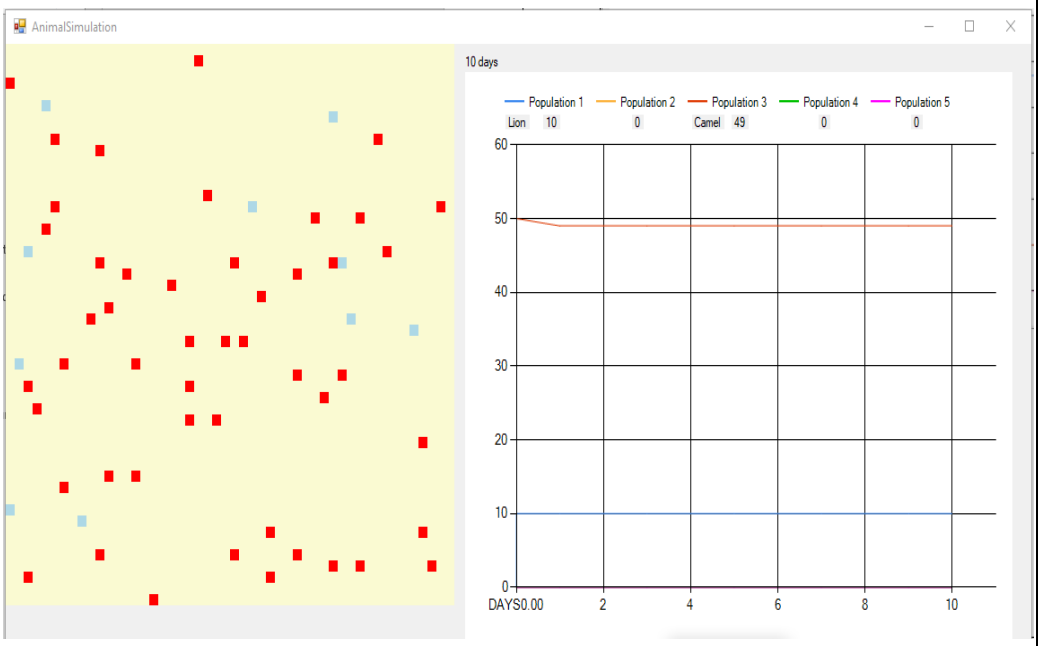

0 days
Population 1
Population 2
Population 3
Population 4
Population 5
Antelope 5
0
Giraffe 10
Panda 9
Snake 3
12
10
8
6
4
2
0
0
2
4
6
8
10
12
14
16

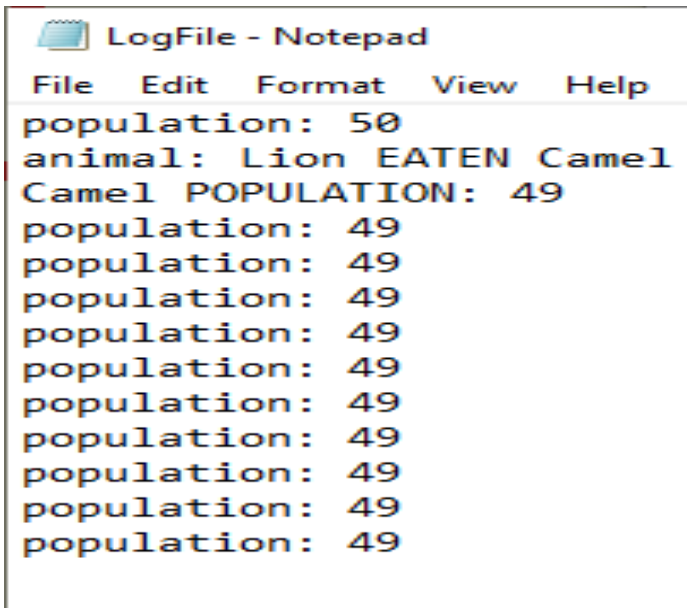
52.


AnimalSimulation
0 days
Population 1
Population 2
Population 3
Population 4
Population 5
Lion
10
0
0
0
0
12
10
8
6
4
2
0
0
DAYS0.00
2
3
Start Simulation
Back

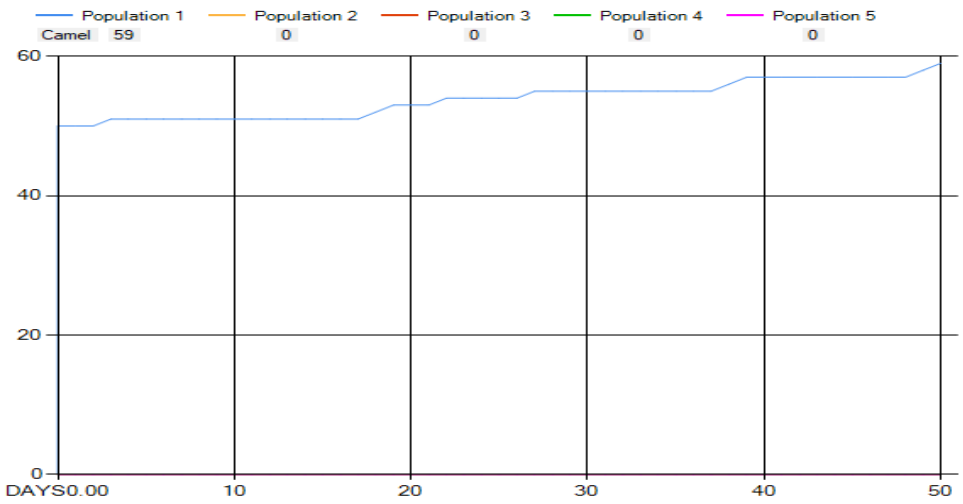

AnimalSimulation
0 days
Population 1
Population 2
Population 3
Population 4
Population 5
Lion
10
0
0
0
0
12
10
8
6
4
2
0
0
DAYS0.00
2
3
4
Start Simulation
Back

53.


AnimalSimulation
10 days
Population 1
Population 2
Population 3
Population 4
Population 5
Lion 10
0
Camel 49
0
0
60
50
40
30
20
10
0
DAYS0.00
2
4
6
8
10


LogFile - Notepad
File Edit Format View Help
population: 50
animal: Lion EATEN Camel
Camel POPULATION: 49
population: 49
population: 49
population: 49
population: 49
population: 49
population: 49
population: 49
population: 49
population: 49
population: 49

54.

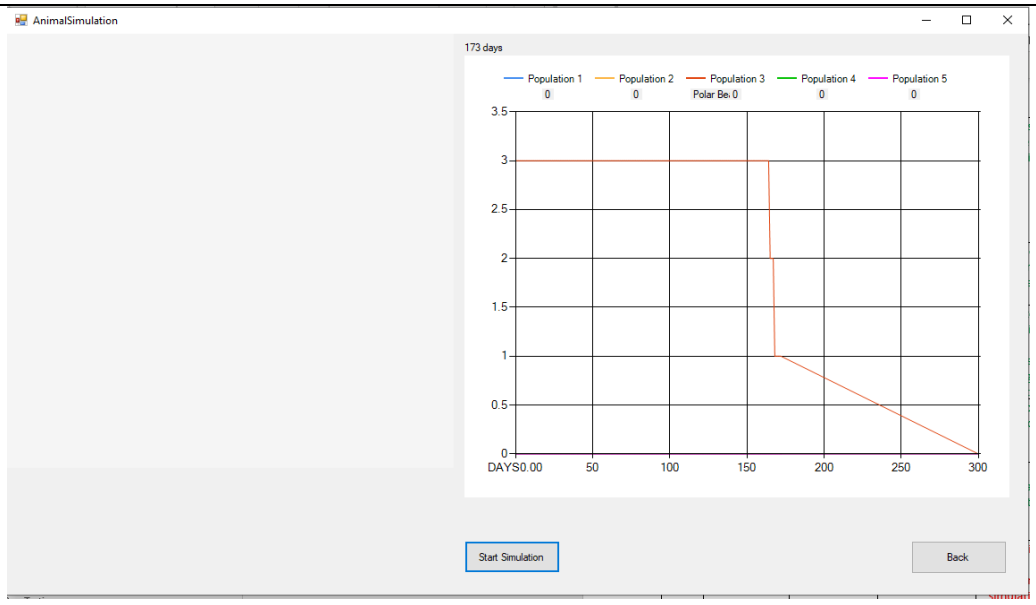


LogFile - Notepad

File Edit Format View Help

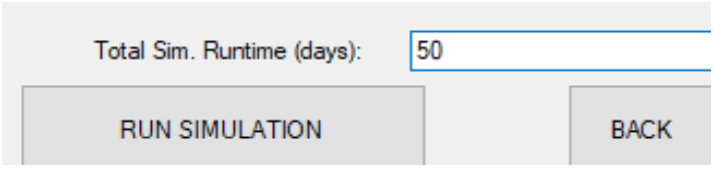
```
population:  50
population:  50
population:  50
Camel IS born
Camel POPULATION:  51
population:  51
population:  51
population:  51
population:  51
population:  51
population:  51
population:  51
population:  51
population:  51
population:  51
population:  51
population:  51
population:  51
population:  51
Camel IS born
Camel POPULATION:  52
population:  52
Camel IS born
Camel POPULATION:  53
population:  53
population:  53
population:  53
Camel IS born
Camel POPULATION:  54
population:  54
population:  54
population:  54
population:  54
population:  54
Camel IS born
```

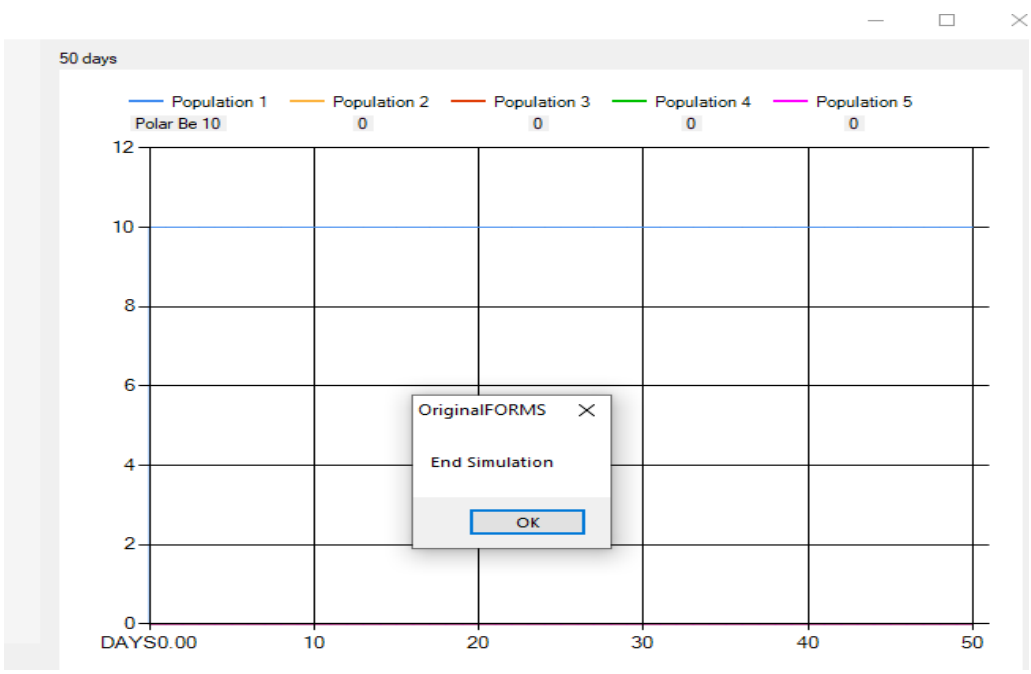
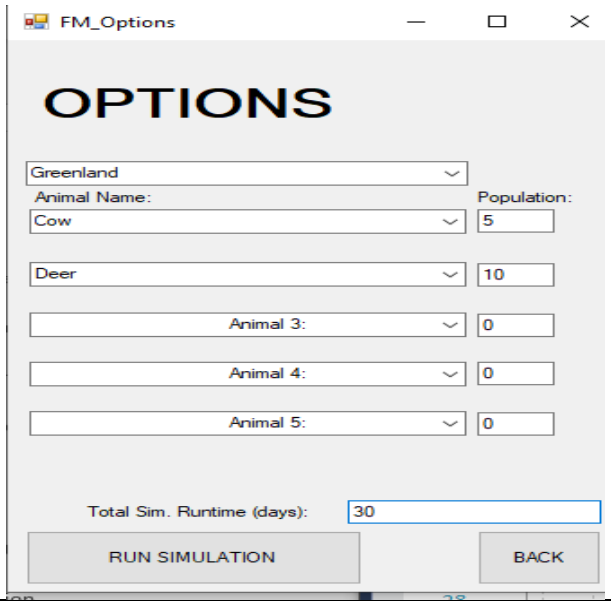
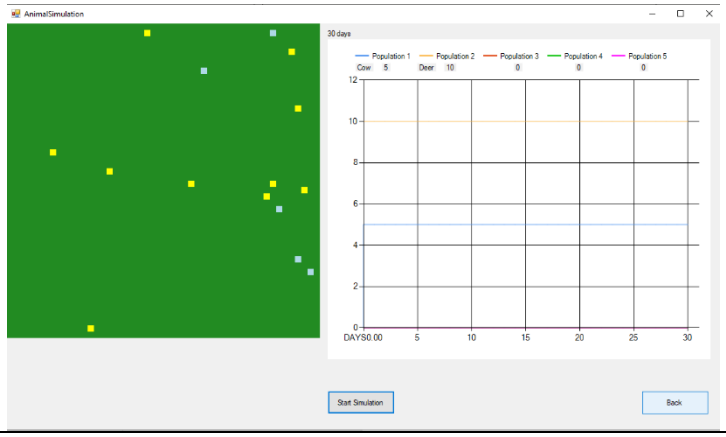
55.

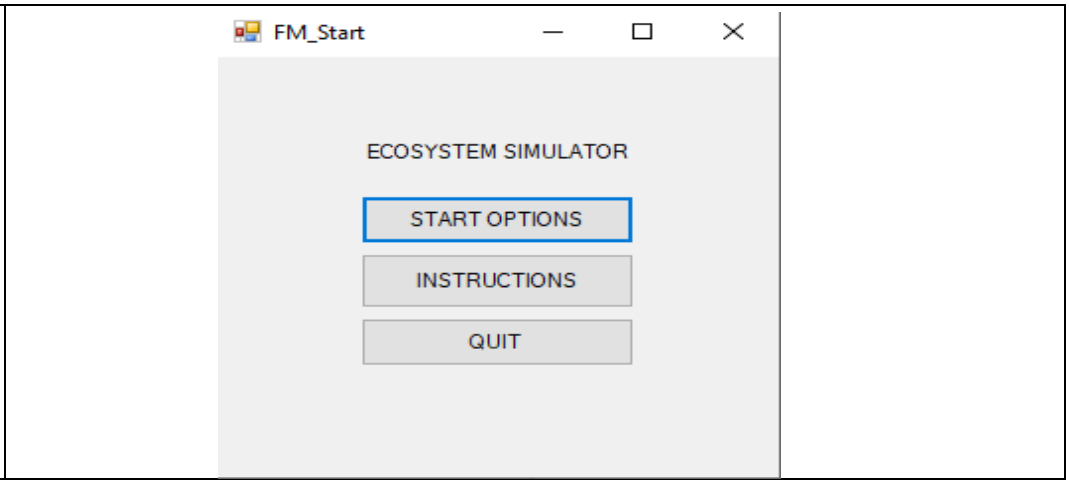



```
population: 3
population: 3
population: 3
population: 3
population: 3
population: 3
population: 3
population: 3
 name: Polar Bear by hunger or age.
Polar Bear POPULATION: 2
population: 2
population: 2
population: 2
 name: Polar Bear by hunger or age.
Polar Bear POPULATION: 1
population: 1
population: 1
population: 1
population: 1
population: 1
 name: Polar Bear by hunger or age.
Polar Bear POPULATION: 0
population: 0
```

56.



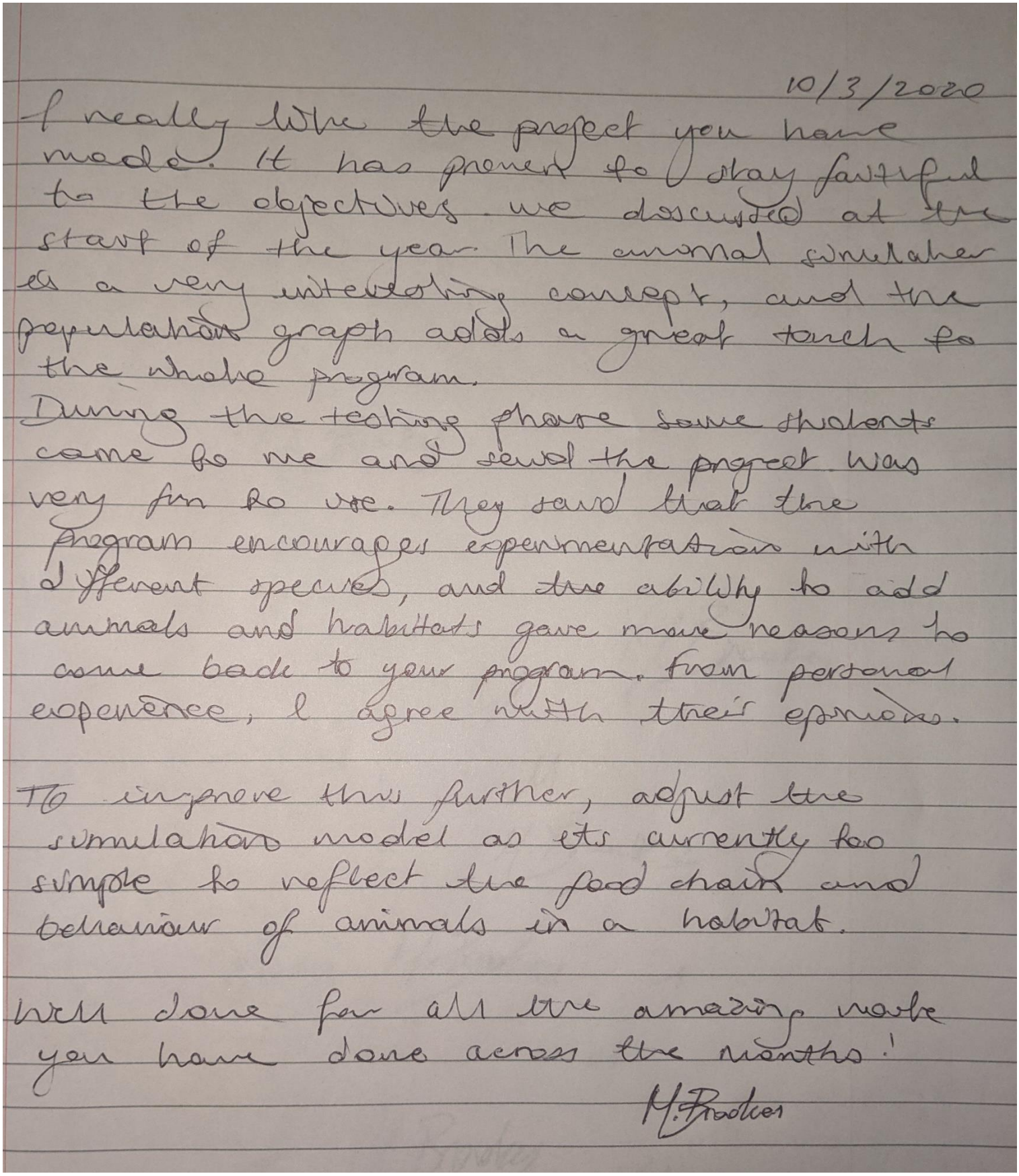
	 50 days Population 1 Population 2 Population 3 Population 4 Population 5 Polar Be 10 0 0 0 0 12 10 8 6 4 2 0 DAYS0.00 10 20 30 40 50 OriginalFORMS End Simulation OK
57.	 FM_Options OPTIONS Greenland Animal Name: Population: Cow 5 Deer 10 Animal 3: 0 Animal 4: 0 Animal 5: 0 Total Sim. Runtime (days): 30 RUN SIMULATION BACK
58.	 AnimalSimulation 30 days Start Simulation Back


FM_Start
ECOSYSTEM SIMULATOR
START OPTIONS
INSTRUCTIONS
QUIT

Evaluation

User Sign off & Client Feedback

Mrs. Brookes letter and sign-off on the finished animal simulator program.



10/3/2020

I really like the project you have made. It has proven to stay faithful to the objectives we discussed at the start of the year. The animal simulator is a very interesting concept, and the population graph adds a great touch to the whole program.

During the testing phase some students came to me and said that the program encourages experimentation with different species, and the ability to add animals and habitats gave more reasons to come back to your program. From personal experience, I agree with their opinions.

To improve this further, adjust the simulation model as it's currently too simple to reflect the food chain and behaviour of animals in a habitat.

Well done for all the amazing work you have done across the months!

M. Brookes

System Evaluation

The evaluation of my program is compared to the final objectives and user requirements found in the analysis section.

User Requirements:

No.	Requirement	Completed
1.	Needs to be user-friendly to the point year 10 students can manage it.	✓
2.	Provide information of the topic inside the revision guide to the user, or can easily be used alongside the revision guide.	✓
3.	Freely accessible for all students and teachers within the school to use.	–
4.	Have great visuals to give students a better look at how animals behave in an environment rather than just writing.	✓
5.	The solution needs to consider aspects of the ecosystem studied in the GCSE.	✓
6.	The solution needs to be robust ; being able to stay consistent in its utility.	✓

The first user requirement was achieved through the user of VB.net's graphical user interface. The user of buttons, textboxes and display was proven to be intuitive for students. During testing no one had an issue with the structure of the interface, and for students who might have had problems the instructions page would help them understand how to use the program.

The second user requirement has also been achieved. As the program is designed to be used alongside the GCSE revision guide as noted in the human-computer interactive design discussions with Mrs. Brookes. The simulator helps extend students knowledge as they are given tools to experiment with their findings in the book through customising the simulator.

At the time of writing the third user requirement is not completed as I need to gain authorization from other geography teachers to allow the programs files to be placed within the geography departments folder. This would take time as I would need to have individual meetings with the teachers to go through the program.

The fourth user requirement is completed as the simulator and real-time population line graph provide colourful representations for the simulation results. As the students are able to see the births and deaths of species across the simulation, it allows a more visual method of learning what they have learnt in class. It especially helps enforcing what they have learnt in class by customising the simulation to environments found in the GCSE revision guide.

The fifth user requirement is also completed for the same reason found in the fourth and second user requirements. Though incorporating more knowledge from the GCSE textbook would lead to different topics that don't relate with the food chain and ecosystem. If I did this it would be more beneficial for students but development would take too long, and topics like globalisation & trade would make no sense in relation to a visual simulator.

The last user requirement has also been completed as I have created the program to have as little faults as possible. I have dropped ideas like changing vegetation levels, having an "add animal" form and other processes that would make my program slow to process. I have made the simulator run more smoothly by swapping nested loops for linked list in order for processing less calculations. As I have iterated my ideas consistently; it led to better solutions that helped the program put less stress on the CPU which is ideal for school computers. The program itself is also robust from reducing errors found in the testing phase. I have validated text boxes that can receive wrong typos from students inputs. This means that the program would not crash from unexpected results making the program suitable for students and not disrupt the simulator.

Final Objectives:

No.	**Objectives**	**Completed**
1.	The program is easy to manage for students in Key Stage 3:	✓
2.	Options display allows easy configuration for simulations:	✓
3.	Animals reproduce, die and age:	✓
4.	Animal are able to move freely:	✓
5.	Ecosystem simulator components need to work efficiently:	✓

The first final objective is completed for the exact reasons why the first user requirements were completed.

The options display is very easy to use. The user can choose the type of species in the simulation, the number of each specie to be sent to the simulation, type of habitat and the length of the simulation. Inputs are validated to prevent silly errors and unwanted inputs to be used in the simulation. And by using drop down menu's it allows the options menu interface to be intuitive for the user, making it very easy to use.

Animals are able to reproduce in the exact same way as planned in the analysis. The birth idea was implemented to the final solution, and is activated when two members of the same species are in contact and have a favourable birth chance.

Animals die in numerous ways also. Firstly if a prey is in contact with a predator in the same grid space, a death probability activates and if its favourable to that chance the prey dies. The second is by age or hunger that work in a very similar way but with different rates. The hunger meter is a small

value that replenishes to its maximum value by eating prey. The higher the value, the better as each day the hunger meter decreases by a certain value and if it falls to 0 the animal dies. The age value is the maximum number of days an animal can live in the simulation, and it decreases every day that the animal lives, likewise when it reaches 0 it will die.

Animals are able to move in 8 direction and stand still. These 9 options are achieved via a random function that decides which way an animal would move in a day. If the animal is by a boundary and the move leads outside the simulation, the animal would be given another chance to move until it doesn't move past the boundary.

The last final objective has been completed as explained by the last user requirement as they are the same thing.

To note, there are similarities between user requirements and final objectives as the user requirements were used to develop the final objectives of my system.

Improvements

System improvements

- **Habitats can have a greater effect on species**. From my program the habitat is mainly used as an indicator of what animals can be selected to participate in a simulation. By extending the giving the habitats more attributes in the habitat database, they could contain different temperature, humidity, maximum vegetation levels etc. With these values it can make the simulation more complex as it directly effects the behaviour of animals in that environment. I would say that it would be very interesting to see how non-local animals would survive in other habitat locations.

- **Species birth, death, and other algorithms need to be more complex to represent the real world**. Leading on from my first improvement, to allow the habitats effects to impact animals, the habitat attributes need to be accounted for in the calculations of birth, death, age and hunger formulas. An example is like **New Hunger level = (hunger level – 1 day) – (hydration x Habitat temperature)**. This is an example were thehunger level is also reduced by hydration level (between 0 and 1) of an animal is multiplied by the value of the general temperature of a habitat. So animals would die faster if they are not used to a favourable temperature found in their local habitats. This would also bring the idea of animals being allowed to be experimented on in different habitats which my program does not allow.

- **Identify animals easier by assigning each specie with an image of their respective animal**, **e.g a cow is a image of a cow.** My program represents selected animals as coloured squares, which have been designated their colour by what animal position they have been added from (Animal 1, Animal 2, Animal 3 etc.) At times when the simulation runs its hard to distinguish which animal is which if you only look at the simulator itself.

- **Switch from Microsoft Access Database to MySQL**. Access databases are limited to 2GB storage. If many students access the program from shared resources and add many animals to the species table, it would greatly increase the file of the database. There may be a chance that their would be too much data on the file and so by switching Microsoft access to MySQL would stop the need to delete data values to not go beyond the storage limit. MySQL is advantageous in this aspect as it has an open-source solution to allow users to easily scale to their needs. This would benefit this project greatly as we do not know the full amount of data needed to be stored as it is continuously incremented.

Possible additional Features

- **Add the visual vegetation level changes as originally proposed in the analysis.** When starting this project I have wanted the herbivores eating behaviour to affect the surrounding habitat. I wanted the grid space of where the herbivore was to turn to a different colour to represent how the vegetation of that point in the environment has reduced. For example in a Greenland environment start dark green then light green if eaten, and finally brown when there is no vegetation left to represent soil. From this idea you can imagine how the look of the simulator would change drastically. This idea was meant to reflect how the eating habits of species impact the environment around it and over-eating would result in starvation in the future. But to add from this if they weren't eating a grid block, the vegetation would grow back incrementally as simulation days carried on. I couldn't add this aspect to the simulation as I was risking longer development times and slowing down the performance of the CPU from adding more calculations.

- **Users can access different graphs for added visuals and to study other aspects found in animals.** For example from using a pie chart and calculating the number of male : female of a species, we can see the population ratios of each gender in a simulation day. By monitoring this we can expect different rates of births to occur. By implementing more graphing tools to the simulation it would be allow the simulation to be more insightful and engaging for students.

- **Develop simulation features like a plague button or natural hazard button so the user can identify animal behaviours before and after an event.** My program does not include other factors like diseases or natural phenomena in the simulation. By adding buttons so the user knows exactly when they want to activate these events, it would be more engaging for the

user and may be more entertaining for students to see the impacts. As discussed in the analysis, young students tend to have short concentration spans and having features that keep the students engaged would be very beneficial for them to learn of how certain events can impact habitats and species.

Final Thoughts

I had fun making this project. I have learnt many programming ideas like the linked list and using classes to represent real-world entities. Even though this program was very difficult and exhausting to make; I know I will use what I have learnt to create new applications in the future. Working with Mrs. Brookes and her year 10 geography class was less stressful than I expected, and I appreciated her willingness to support me in doing this project. I am very thankful for her help throughout the development of the simulator as she allowed me to spend time with her students for my analysis and testing phases. Overall it was a rewarding experience that has tested my abilities in many ways, and I have vastly improved myself from it.

Appendix

Code:

```
Public Class cl_Animal
    Public AnimalName As String 'Name of animal.
    Public FoodType As Integer '[1] = herbivore, [2] = omnivore, [3] = carnivore
    Public LifeSpan As Integer  'Life span of animal.
    Public DeathRate As Double  'Probability of death for animal interactions.
    Public Birth_Rate As Double 'Probability of birth when in contact of the same
specie.
    Public HungerMeter As Integer 'Maximum day where animal can live without food.
    Public AnimalSpeed As Integer 'Speed at which an animal can move per day
    Public FoodChainRank As Integer 'How the animal ranks within the food chain
    Const AnimalSize As Integer = 10 'All animals are represented by a 10x10 square
    Public isDead As Boolean 'Checks if animal has died from hunger, killed or by age.
```

```
    Public age As Integer 'Maximum age an animal can live too.
    Public hungerlevel As Integer 'The current level of hunger derived from hunger
meter in a simulated day.


    'NOW TO ONLY ALLOW BIRTHS BETWEEN MODERATELY OLD MATE AND SAME SPECIE 14/1/2020


    Function getAnimalSize() As Integer 'Returns the size of all animals to then be
drawn.
        Return AnimalSize
    End Function

    Function GetMove() As String 'function that assigns an animal a move to be made
from a random function.
        Dim x As Integer

        Randomize()

        x = Int((9 * Rnd()) + 1)

        Select Case x
            Case 1
                Return "N"
            Case 2
                Return "NE"
            Case 3
                Return "E"
            Case 4
                Return "SE"
            Case 5
                Return "S"
            Case 6
                Return "SW"
            Case 7
                Return "W"
            Case 8
                Return "NW"
            Case Else
                Return "No movement"
        End Select
    End Function

    Overridable Function GiveBirth() As cl_Animal 'A birth function that is overriden
by the herbivore, carnivore and omnivore classes as each class gives birth
differently.


        Return Nothing
    End Function


    Overridable Function GiveDeath() As Boolean 'A death function that is overriden by
the herbivore, carnivore and omnivore classes as each class dies at different rates.




        Return Nothing
    End Function
```

```
    Overridable Function GetFood() As Boolean 'A get food function that is overriden
by the herbivore, carnivore and omnivore classes as each class gets food differently.


        Randomize()

        Return Nothing
    End Function

    Overridable Function GetFood(ByVal food As cl_Animal) As Boolean 'A get food
function that is overriden by the herbivore, carnivore and omnivore classes as each
class gets food differently.


        Randomize()

        Return Nothing
    End Function

    Public Overridable Sub DayStatus() 'Subroutine that changes the personal status of
all individual animals.


        age = age - 1 'formulae for aging
        If age = 0 Then
            isDead = True
        End If

        hungerlevel = hungerlevel - 1 'formulae for hunger level increasing (decreases
as 0 means died by hunger)
        If hungerlevel = 0 Then
            isDead = True
        ElseIf cl_GlobalSetting.Board.EatenFood = True Then
            hungerlevel = HungerMeter
            hungerlevel = hungerlevel + 30

        End If

    End Sub



End Class
```

```
Public Class cl_Board




   Const BOARDSIZEX As Integer = 500 'Width of the board or simulator


   Const BOARDSIZEY As Integer = 500 'Height of the board or simulator


   Public BoardType As String 'Type of habitat for a simulation
```

```
    Public Board(,) As cl_Animal 'Given class animal for all animals to be drawn in a given position.

    Public animalList As New cl_LinkedList 'linked list of all current animal positions in a simulation day

    Public EatenFood As Boolean 'check if an animal has been killed or has eaten prey




    Sub New(ByVal type As String) 'When starting up a new simulation a new board is given the wanted habitat from the user


        BoardType = type

        ReDim Board(BOARDSIZEX / 10, BOARDSIZEY / 10) 'board size 50X50




    End Sub



    Function returnBOARDSIZEX() As Integer 'Returns boards width

        Return BOARDSIZEX

    End Function
```

```
    Function returnBOARDSIZEY() As Integer 'returns boards height

        Return BOARDSIZEY

    End Function

    Sub RunDay() 'Simulates every day in the simulation by going through the linked list and
determining animal interaction results.



        Dim direction As String 'Determines where an animal will move too.

        Dim AnimalName As String 'Name of animal




        cl_GlobalSetting.log.write("population: " & animalList.populationNum) 'Log to textfile total
population no.



        If animalList.populationNum > 0 Then 'Allows simulation to begin if total population on the
board is > 0.



            animalList.reset() 'resets linked list to start from beginning
```

```
Do

    If IsNothing(animalList.current) <> True Then 'Checks if there is something at the head of the linked list


        direction = animalList.current.piece.GetMove() 'Recieves move from animal classes' getmove()


        If direction <> "No movement" Then 'Stops animal giving birth by itself if they contact themselves by not moving.


            'MsgBox(direction)

            ' MsgBox(animalList.current.X)

            'MsgBox(animalList.current.Y)

            Dim temp() As Integer 'Temporary move for an animal

            'Dim newBornTemp() As Integer

            Dim x, y As Integer

            temp = returnCor(direction, animalList.current.X, animalList.current.Y)
```

'newBornTemp = returnCor(direction, animalList.current.X, animalList.current.Y) 'Idea of finding newborn space

Board(animalList.current.X, animalList.current.Y).DayStatus()

If Board(animalList.current.X, animalList.current.Y).isDead = True Then 'check if animal dies from hunger or by age

cl_GlobalSetting.log.write(" name: " & animalList.current.piece.AnimalName & " by hunger or age.") 'Log to textfile of death

AnimalName = animalList.current.piece.AnimalName 'TRACK ANIMALS POPULATION when animals die

Select Case AnimalName

Case cl_GlobalSetting.AnimalONE

cl_GlobalSetting.AnimalONEpopulation -= 1

cl_GlobalSetting.log.write(animalList.current.piece.AnimalName & " POPULATION: " & cl_GlobalSetting.AnimalONEpopulation)

Case cl_GlobalSetting.AnimalTWO

```
                    cl_GlobalSetting.AnimalTWOpopulation -= 1

                    cl_GlobalSetting.log.write(animalList.current.piece.AnimalName & "
POPULATION: " & cl_GlobalSetting.AnimalTWOpopulation)

                Case cl_GlobalSetting.AnimalTHREE

                    cl_GlobalSetting.AnimalTHREEpopulation -= 1

                    cl_GlobalSetting.log.write(animalList.current.piece.AnimalName & "
POPULATION: " & cl_GlobalSetting.AnimalTHREEpopulation)

                Case cl_GlobalSetting.AnimalFOUR

                    cl_GlobalSetting.AnimalFOURpopulation -= 1

                    cl_GlobalSetting.log.write(animalList.current.piece.AnimalName & "
POPULATION: " & cl_GlobalSetting.AnimalFOURpopulation)

                Case cl_GlobalSetting.AnimalFIVE

                    cl_GlobalSetting.AnimalFIVEpopulation -= 1

                    cl_GlobalSetting.log.write(animalList.current.piece.AnimalName & "
POPULATION: " & cl_GlobalSetting.AnimalFIVEpopulation)

            End Select

            x = animalList.current.X 'current nodes' x coordinate of an animal.

            y = animalList.current.Y 'current nodes' y coordinate of an animal.

            animalList.NextAnimalNode() 'Moves along linked list.
```

```
                animalList.delete(x, y) 'removes from linked list

                Board(x, y) = Nothing 'removes dead animal off board

            Else

                If checkBoundary(temp) = True Then 'Checking if temp coordinate is within board
boundary

                    If checkEmpty(temp) = True Then 'checking if position is empty

                        Board(temp(0), temp(1)) = Board(animalList.current.X, animalList.current.Y)
'Assigning specie with new coordinates

                        Board(animalList.current.X, animalList.current.Y) = Nothing
```

```
                    animalList.current.X = temp(0) ' gives new x coordinate

                    animalList.current.Y = temp(1) 'Gives new y coordinate



                Else 'Conditions were they are on the same position

                    If Board(animalList.current.X, animalList.current.Y).AnimalName <>
Board(temp(0), temp(1)).AnimalName Then ' Check different specie meet



                        'MsgBox(Board(animalList.current.X, animalList.current.Y).AnimalName & "
met with " & Board(temp(0), temp(1)).AnimalName)



                        'Remove Animal off board as it is killed by predator

                        If Board(animalList.current.X, animalList.current.Y).FoodType >
Board(temp(0), temp(1)).FoodType Then

                            cl_GlobalSetting.log.write("animal: " &
animalList.current.piece.AnimalName & " EATEN " & Board(temp(0), temp(1)).AnimalName) 'log
death to textfile



                            AnimalName = Board(temp(0), temp(1)).AnimalName 'TRACK ANIMAL
POPULATION when animal dies
```

```
                    Select Case AnimalName


                        Case cl_GlobalSetting.AnimalONE

                            cl_GlobalSetting.AnimalONEpopulation -= 1

                            cl_GlobalSetting.log.write(animalList.current.piece.AnimalName & "
POPULATION: " & cl_GlobalSetting.AnimalONEpopulation)

                        Case cl_GlobalSetting.AnimalTWO

                            cl_GlobalSetting.AnimalTWOpopulation -= 1

                            cl_GlobalSetting.log.write(animalList.current.piece.AnimalName & "
POPULATION: " & cl_GlobalSetting.AnimalTWOpopulation)

                        Case cl_GlobalSetting.AnimalTHREE

                            cl_GlobalSetting.AnimalTHREEpopulation -= 1

                            cl_GlobalSetting.log.write(animalList.current.piece.AnimalName & "
POPULATION: " & cl_GlobalSetting.AnimalTHREEpopulation)

                        Case cl_GlobalSetting.AnimalFOUR

                            cl_GlobalSetting.AnimalFOURpopulation -= 1

                            cl_GlobalSetting.log.write(animalList.current.piece.AnimalName & "
POPULATION: " & cl_GlobalSetting.AnimalFOURpopulation)

                        Case cl_GlobalSetting.AnimalFIVE
```

```
                cl_GlobalSetting.AnimalFIVEpopulation -= 1

                cl_GlobalSetting.log.write(animalList.current.piece.AnimalName & "
POPULATION: " & cl_GlobalSetting.AnimalFIVEpopulation)

            End Select

            Board(temp(0), temp(1)) = Board(animalList.current.X, animalList.current.Y)

            animalList.delete(temp(0), temp(1)) 'removes from linked list



            Board(animalList.current.X, animalList.current.Y) = Nothing 'removes off
board



            animalList.current.X = temp(0) ' gives new x coordinate



            animalList.current.Y = temp(1) 'Gives new y coordinate



            EatenFood = True
```

```
                    'MsgBox("EATEN")
                Else
                    EatenFood = False



                End If
                'Species are the same or older, chance to activate BIRTH more regularly!
            ElseIf Board(animalList.current.X, animalList.current.Y).AnimalName =
Board(temp(0), temp(1)).AnimalName And Board(animalList.current.X, animalList.current.Y).age >=
Board(temp(0), temp(1)).age Then 'Species are the same or older, chance to activate BIRTH more
regularly!


                '16/01/2020 updated conditions to mate so age of species and animal name
are good for mating
                'Also age is equal to minimise growth (I know naming is weird :P)
                'Updated birth rate to make the chance of births to be less
```

```
'Need a way to make them die faster from death as program keeps crashing and seems like they dont die

'Program seems to break suddenly even when i lower birth rate to 0.05



Dim newBorn As cl_Animal 'Object for potential new born.

Dim newCoor() As Integer 'coordinate for ne born to be placed to the simulation.

If checkEmpty(temp) = False Then



newBorn = Board(animalList.current.X, animalList.current.Y).GiveBirth() 'Give Birth gives probability and returns copy of parent as new born.



If IsNothing(newBorn) = False Then



newCoor = checkAvailableSpace(animalList.current.X, animalList.current.Y) 'check where to place new born
```

```
                    If newCoor(0) <> -1 Then 'validate new coordinate position for new born
                        cl_GlobalSetting.log.write(newBorn.AnimalName & " IS born")

                        Board(newCoor(0), newCoor(1)) = newBorn 'Births new animal!
                        animalList.addQueue(newBorn, newCoor(0), newCoor(1)) 'Adds new
born to linked list

                        AnimalName = newBorn.AnimalName 'Shares same name as parent
                        Select Case AnimalName          'TRACK ANIMAL POPULATION
when new births occur.

                            Case cl_GlobalSetting.AnimalONE
                                cl_GlobalSetting.AnimalONEpopulation += 1
                                cl_GlobalSetting.log.write(animalList.current.piece.AnimalName &
" POPULATION: " & cl_GlobalSetting.AnimalONEpopulation)
                            Case cl_GlobalSetting.AnimalTWO
                                cl_GlobalSetting.AnimalTWOpopulation += 1
                                cl_GlobalSetting.log.write(animalList.current.piece.AnimalName &
" POPULATION: " & cl_GlobalSetting.AnimalTWOpopulation)
```

```
                Case cl_GlobalSetting.AnimalTHREE

                    cl_GlobalSetting.AnimalTHREEpopulation += 1

                    cl_GlobalSetting.log.write(animalList.current.piece.AnimalName &
" POPULATION: " & cl_GlobalSetting.AnimalTHREEpopulation)

                Case cl_GlobalSetting.AnimalFOUR

                    cl_GlobalSetting.AnimalFOURpopulation += 1

                    cl_GlobalSetting.log.write(animalList.current.piece.AnimalName &
" POPULATION: " & cl_GlobalSetting.AnimalFOURpopulation)

                Case cl_GlobalSetting.AnimalFIVE

                    cl_GlobalSetting.AnimalFIVEpopulation += 1

                    cl_GlobalSetting.log.write(animalList.current.piece.AnimalName &
" POPULATION: " & cl_GlobalSetting.AnimalFIVEpopulation)

            End Select



        End If



    End If
```

```
                End If

            End If

        End If

    End If

End If

End If

    Loop Until animalList.NextAnimalNode() = False Or animalList.populationNum <= 0

End If 'Population = 0

End Sub

Function returnCor(ByVal direction As String, ByVal x As Integer, ByVal y As Integer) As Integer()
'Returns potential new coordinates for animal to move too.

Dim coordinate(2) As Integer

Select Case direction
```

```
Case "N"
    coordinate(0) = x
    coordinate(1) = y - 1

Case "NE"
    coordinate(0) = x + 1
    coordinate(1) = y - 1

Case "E"
    coordinate(0) = x + 1
    coordinate(1) = y

Case "SE"
    coordinate(0) = x + 1
```

```
    coordinate(1) = y + 1

Case "S"
    coordinate(0) = x
    coordinate(1) = y + 1

Case "SW"
    coordinate(0) = x - 1
    coordinate(1) = y + 1

Case "W"
    coordinate(0) = x - 1
    coordinate(1) = y
```

```
        Case "NW"
            coordinate(0) = x - 1
            coordinate(1) = y - 1

        Case Else
            coordinate(0) = x
            coordinate(1) = y

    End Select

    Return coordinate
End Function

Function checkBoundary(ByVal coordinate() As Integer) As Boolean 'Check if potential new move is within the simulation grid.
```

```
        If coordinate(0) < 0 Or coordinate(0) > 49 Or coordinate(1) < 0 Or coordinate(1) > 49 Then
'Solved this via boolean algebra for documentation a.b.c.d & a+b+c+d
            Return False
        Else
            Return True
        End If

    End Function

    Function checkEmpty(ByVal coordinate() As Integer) As Boolean 'Check new move is not already
occupied.

        If coordinate(0) >= 0 And coordinate(1) >= 0 And coordinate(0) <= 50 And coordinate(1) <= 50
Then

            If IsNothing(Board(coordinate(0), coordinate(1))) = True Then
                Return True
            Else
                Return False
```

```
        End If


    Else

        Return False


    End If


End Function


'Function PredatorToPrey() As Boolean 'Foodchainrank is used            NOT NEEDED


'    Return Nothing

'End Function


'Function SameSpecie() As Boolean   'Animal Name is compared


'    Return Nothing
```

```
    'End Function



    'Function PreyToPredator() As Boolean ' Might need this one



    '    Return Nothing

    'End Function









    Sub addAnimal(ByVal animal As cl_Animal, ByVal amount As Integer) 'Adds a specific amount of a specie into the simulation

        Dim X As Integer

        Dim Y As Integer



        Dim count As Integer = 0
```

```
Do

    Randomize() 'Randomise inital coordinates of animals onto the board.

    X = Int(49 * Rnd())

    Y = Int(49 * Rnd())



    If IsNothing(Board(X, Y)) Then 'Places if inital board positions are not occupied



        Select Case animal.FoodType 'Gives correct data of every type of animal from database

            Case 1

                animal = New cl_Herbivore(animal.AnimalName, animal.LifeSpan, animal.DeathRate,
animal.Birth_Rate, animal.HungerMeter, animal.AnimalSpeed, animal.FoodChainRank)

            Case 2

                animal = New cl_Omnivore(animal.AnimalName, animal.LifeSpan, animal.DeathRate,
animal.Birth_Rate, animal.HungerMeter, animal.AnimalSpeed, animal.FoodChainRank)

            Case 3

                animal = New cl_Carnivore(animal.AnimalName, animal.LifeSpan, animal.DeathRate,
animal.Birth_Rate, animal.HungerMeter, animal.AnimalSpeed, animal.FoodChainRank)

            Case Else

                MsgBox("Animal type is unrecognizable")
```

```
            End Select

            animalList.addQueue(animal, X, Y) 'adds each new animal to linked list

            Board(X, Y) = animal 'animal will be drawn to board at positions x y

            count = count + 1

            ' MsgBox("Counter " & count & " Type:" & animal.FoodType & " x:" & X & "y:" & Y)

        End If



    Loop Until count = amount 'Gives outputting animals to simulation until inital quantity is equal to no.of animals



End Sub



Function checkAvailableSpace(ByVal x As Integer, ByVal y As Integer) As Integer() 'Used for new borns to be placed on board.

    'Checks surrounding space of parent who gave birth.

    Dim EmptySpace(2) As Integer

    Dim newCor(2) As Integer

    Dim currentSpace(2) As Integer
```

```
currentSpace(0) = x

currentSpace(1) = y



newCor(0) = currentSpace(0) - 1

newCor(1) = currentSpace(1) - 1

'validation is needed

If checkEmpty(newCor) = False Then

    newCor(0) = currentSpace(0)

    newCor(1) = currentSpace(1) - 1



    If checkEmpty(newCor) = False Then

        newCor(0) = currentSpace(0) + 1

        newCor(1) = currentSpace(1) - 1



        If checkEmpty(currentSpace) = False Then

            newCor(0) = currentSpace(0) + 1

            newCor(1) = currentSpace(1)
```

```
If checkEmpty(newCor) = False Then
    newCor(0) = currentSpace(0) + 1
    newCor(1) = currentSpace(1) + 1

    If checkEmpty(newCor) = False Then
        newCor(0) = currentSpace(0)
        newCor(1) = currentSpace(1) + 1

        If checkEmpty(newCor) = False Then
            newCor(0) = currentSpace(0) - 1
            newCor(1) = currentSpace(1) + 1

            If checkEmpty(newCor) = False Then
                newCor(0) = currentSpace(0) - 1
                newCor(1) = currentSpace(1)

                If checkEmpty(newCor) = False Then
                    newCor(0) = -1
```

```
                    newCor(1) = -1

                End If

            End If

        End If

    End If

End If

End If

End If

End If

End If

EmptySpace = newCor

Return EmptySpace

End Function
```

End Class

```
Public Class cl_Carnivore 'Class containing all properties and methods of every
carnivore.
    Inherits cl_Animal


    'Data give from database of an animal carnivore
    Sub New(ByVal animalN As String, ByVal animalLS As Integer, ByVal animalDR As
Double, ByVal animalBR As Double, ByVal animalHM As Integer, ByVal animalAS As
Integer, ByVal animalFCR As Integer)
        FoodType = 3
        Birth_Rate = animalBR
        AnimalName = animalN
        LifeSpan = animalLS
        DeathRate = animalDR
        HungerMeter = animalHM
        AnimalSpeed = animalAS
        FoodChainRank = animalFCR

        age = LifeSpan * 50 'Start off at highest possible life number then decreases
        hungerlevel = HungerMeter * 25

    End Sub
    '
    Public Overrides Function GiveBirth() As cl_Animal 'gives new born carnivore the
same features as parent
        'Dim baby As New cl_Carnivore
        Dim birthChance As Decimal
        Dim Baby

        Randomize()

        birthChance = Rnd() 'Within 0-1
        'MsgBox(birthChance)

        If birthChance <= Birth_Rate Then
            '   MsgBox("Got Baby- " & Me.AnimalName & " " & birthChance)
            Baby = New cl_Carnivore(Me.AnimalName, Me.LifeSpan, Me.DeathRate,
Me.Birth_Rate, Me.HungerMeter, Me.AnimalSpeed, Me.FoodChainRank)




            Return Baby
        Else
            Return Nothing

        End If


    End Function

    Overrides Function GetFood(ByVal food As cl_Animal) As Boolean 'Carnivore getting
food determined by death rate and food chain rank.
        Dim FoodChance As Decimal

        Randomize()

        FoodChance = Rnd()
```

```
        If FoodChance <= DeathRate And food.FoodChainRank < Me.FoodChainRank Then


        End If


        Randomize()

        Return Nothing
    End Function




End Class


Imports System.Data.OleDb
Public Class cl_Database 'Handles everything associated with the database

    Public Shared connectionString As String
    Private connection As OleDbConnection
    Sub New() 'determines the location of the database file.
        connectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source= F:\Current
forms 3.1.2020\Forms\Forms\bin\Debug/AnimalSimulatorDatabase.accdb"
'N:/AnimalSimulatorDatabase.accdb" ' F:\Current forms
3.1.2020\Forms\Forms\bin\Debug/AnimalSimulatorDatabase.accdb" 'Searches for desired
database in directory
        connection = New OleDbConnection(connectionString)
    End Sub


    'Selects animals to be used in simulation in options.
    Function selectAnimal() As DataTable '(ByVal table As String, ByVal fieldname As
String) As DataTable
        Dim selectCommand As String = "Select * FROM [Species]"
        Dim reader As OleDbDataReader
        Dim command As New OleDbCommand(selectCommand, connection)
        Dim found As Boolean = False
        Dim dataTable As New DataTable("")

        connection.Open()
        reader = command.ExecuteReader()
        dataTable.Load(reader)

        Return dataTable
    End Function
    Function selectAnimalTerrain(ByVal HabitatName As String) As DataTable 'Selects a
list of animals that relate to a given habitat type.
        Dim selectCommand As String = "SELECT * FROM [Habitat]" & " INNER JOIN
[Habitat_Assignment] ON  Habitat.Habitat  = Habitat_Assignment.Habitat WHERE
Habitat.Habitat_Name = '" & HabitatName & "';"
        Dim reader As OleDbDataReader
        Dim command As New OleDbCommand(selectCommand, connection)
        Dim dataTable As New DataTable("")

        connection.Open()
        reader = command.ExecuteReader()
        dataTable.Load(reader)

        reader.Close()
```

```
        connection.Close()


        Return dataTable
    End Function

    Function selectTerrain() As DataTable 'gives the user all the options of habitats
to be used in the simulation
        Dim selectCommand As String = "SELECT * FROM [Habitat] "
        Dim reader As OleDbDataReader
        Dim command As New OleDbCommand(selectCommand, connection)
        Dim found As Boolean = False
        Dim dataTable As New DataTable("")

        connection.Open()
        reader = command.ExecuteReader()
        dataTable.Load(reader)

        Return dataTable
    End Function

    Function selectAnimalData(ByVal AnimalName As String) As DataTable 'Provides all
attributes an animal has in the database.

        Dim selectCommand As String = "Select * FROM [Species] WHERE [Animal_names] =
'" & AnimalName & "';"
        Dim reader As OleDbDataReader
        Dim command As New OleDbCommand(selectCommand, connection)
        Dim found As Boolean = False
        Dim dataTable As New DataTable("")

        connection.Open()
        reader = command.ExecuteReader()
        dataTable.Load(reader)

        Return dataTable
    End Function

End Class


'Provides the whole solution access to specie name, population sizes, runtime and
ability to access the board and log to textfile.
Public Class cl_GlobalSetting
    Public Shared Terrain As String

    Public Shared AnimalONE As String
    Public Shared AnimalTWO As String
    Public Shared AnimalTHREE As String
    Public Shared AnimalFOUR As String
    Public Shared AnimalFIVE As String

    Public Shared AnimalONEpopulation As Integer
    Public Shared AnimalTWOpopulation As Integer
    Public Shared AnimalTHREEpopulation As Integer
    Public Shared AnimalFOURpopulation As Integer
    Public Shared AnimalFIVEpopulation As Integer

    Public Shared TotalSimulationTime As Integer

    'Public Shared TerrainType As String
```

```
    Public Shared Board As cl_Board

    Public Shared log As cl_log

End Class


Public Class cl_Herbivore 'Class containing all properties and methods of every
herbivore.
    Inherits cl_Animal

    'Data give from database of an animal herbivore.
    Sub New(ByVal animalN As String, ByVal animalLS As Integer, ByVal animalDR As
Double, ByVal animalBR As Double, ByVal animalHM As Integer, ByVal animalAS As
Integer, ByVal animalFCR As Integer)
        AnimalName = animalN
        FoodType = 1
        LifeSpan = animalLS
        DeathRate = animalDR
        Birth_Rate = animalBR
        HungerMeter = animalHM
        AnimalSpeed = animalAS
        FoodChainRank = animalFCR

        age = LifeSpan * 100
        hungerlevel = HungerMeter * 25


    End Sub
    '
    Overrides Function GetFood() As Boolean
        'In this simualtion, herbivores eat constantly as they are producers so the
habitat provides the food.

        Randomize()

        Return Nothing
    End Function

    Public Overrides Function GiveBirth() As cl_Animal 'gives new born herbivore the
same features as parent
        'Dim baby As New cl_Carnivore
        Dim birthChance As Decimal
        Dim Baby

        Randomize()

        birthChance = Rnd() 'Within 0-1
        ' MsgBox(birthChance)

        If birthChance <= Birth_Rate Then
            'MsgBox("Got Baby- " & Me.AnimalName & " " & birthChance)
            Baby = New cl_Herbivore(Me.AnimalName, Me.LifeSpan, Me.DeathRate,
Me.Birth_Rate, Me.HungerMeter, Me.AnimalSpeed, Me.FoodChainRank)
```

```
            Return Baby
        Else
            Return Nothing

        End If


    End Function


    Public Overrides Sub DayStatus() 'Hunger does not apply to herbivore as they are
producers.


        age = age - 1 'formulae for life (2/2)
        If age = 0 Then
            isDead = True
        End If



    End Sub

End Class


Public Class cl_LinkedList 'Class containing the structure of the linked list
    Public head As cl_Node 'First node of the linked list
    Public tail As cl_Node 'Last node of the linked list
    Public current As cl_Node 'current node being used in the linked list
    Private previous As cl_Node 'previous node of the current in the linked list
    Public populationNum As Integer 'The no. of nodes in the linekd list / active
animals on board.

    Sub New() 'New linked list has no nodes / animals
        populationNum = 0
    End Sub
    'adds animal in the queue form of a linked list.
    Sub addQueue(ByVal thisAnimal As cl_Animal, ByVal X As Integer, ByVal Y As
Integer)
        Dim temp As New cl_Node
        temp.piece = thisAnimal
        temp.X = X
        temp.Y = Y
        If IsNothing(head) =
            True Then
            current = temp
            head = temp
            tail = temp

        Else
            tail.nextNode = temp
            tail = temp

        End If
        populationNum = populationNum + 1
    End Sub


    Function NextAnimalNode() As Boolean 'passes to next node in linked list

        If IsNothing(current) = False Then
```

```
            If IsNothing(current.nextNode) = False Then
                previous = current

                current = current.nextNode
                Return True

            Else



                Return False

            End If
        Else
            Return False
        End If
    End Function

    Sub reset() 'restarts linked list from the beginning when sim. day finishes
        current = head
        previous = Nothing

    End Sub


    'Deletes nodes after simulation activities occur
    Sub delete(ByVal x As Integer, ByVal y As Integer) 'FLOWCHART FOR DOCUMENT ON HOW
WE DELETED AN ANIMAL AFTER BEING EATEN
        Dim temp As cl_Node
        Dim isDeleted As Boolean
        Dim previous As cl_Node



        isDeleted = False
        temp = head
        previous = head

        If temp.X = x And temp.Y = y Then 'deletes head node
            'cl_GlobalSetting.log.write("ERROR 1.1")
            head = head.nextNode
            isDeleted = True
        End If

        Do
            If IsNothing(temp.nextNode) = True Then 'deletes tail node
                'cl_GlobalSetting.log.write("ERROR 1.2")
                previous.nextNode = Nothing
                isDeleted = True
            Else
                If temp.nextNode.X = x And temp.nextNode.Y = y Then 'deletes a node
within the body of the node
                    'cl_GlobalSetting.log.write("ERROR 1.3")
                    If IsNothing(temp.nextNode.nextNode) = False Then
                        'cl_GlobalSetting.log.write("ERROR 1.4")
                        temp.nextNode = temp.nextNode.nextNode
                        isDeleted = True
                    Else
                        'cl_GlobalSetting.log.write("ERROR 1.5")
                        temp.nextNode = Nothing
                        isDeleted = True
                    End If
```

```
                Else
                    'cl_GlobalSetting.log.write("ERROR 1.6")
                    previous = temp
                    temp = temp.nextNode
                End If

            End If
            If isDeleted = True Then
                ' cl_GlobalSetting.log.write("Ends loop")
            End If
        Loop Until isDeleted = True
        populationNum = populationNum - 1

    End Sub
End Class


'Allows simulation activities to be logged to a textfile
Imports System.IO
Public Class cl_log
    Dim wfile As StreamWriter
    Public Sub New() 'Deletes old simulation data in preparation for new one.
        wfile = New StreamWriter("LogFile.txt", False)
        wfile.Close()


    End Sub

    Public Sub write(ByVal text As String) 'Writes new inormation when called.
        wfile = New StreamWriter("LogFile.txt", True)
        wfile.WriteLine(text)
        wfile.Close()

    End Sub


End Class


'Contains properties of all nodes within a linked list.
Public Class cl_Node
    Public piece As cl_Animal
    Public X As Integer
    Public Y As Integer
    Public nextNode As cl_Node

End Class


Public Class cl_Omnivore 'Class containing all properties and methods of every
omnivore.
    Inherits cl_Animal

    'Data give from database of an animal omnivore.
    Sub New(ByVal animalN As String, ByVal animalLS As Integer, ByVal animalDR As
Double, ByVal animalBR As Double, ByVal animalHM As Integer, ByVal animalAS As
Integer, ByVal animalFCR As Integer) 'Runs first
        FoodType = 2

        AnimalName = animalN

        LifeSpan = animalLS
```

```
        DeathRate = animalDR
        Birth_Rate = animalBR
        HungerMeter = animalHM
        AnimalSpeed = animalAS
        FoodChainRank = animalFCR

        age = LifeSpan * 75
        hungerlevel = HungerMeter * 25

    End Sub


    Public Overrides Function GiveBirth() As cl_Animal 'gives new born omnivore the
same features as parent
        'Dim baby As New cl_Carnivore
        Dim birthChance As Decimal
        Dim Baby

        Randomize()

        birthChance = Rnd() 'Within 0-1
        ' MsgBox(birthChance)

        If birthChance <= Birth_Rate Then
            '  MsgBox("Got Baby- " & Me.AnimalName & " " & birthChance)
            Baby = New cl_Omnivore(Me.AnimalName, Me.LifeSpan, Me.DeathRate,
Me.Birth_Rate, Me.HungerMeter, Me.AnimalSpeed, Me.FoodChainRank)




            Return Baby
        Else
            Return Nothing

        End If


    End Function



End Class


Public Class FM_AnimalSimulation 'Forms containing the design of the simulation
environment
    Dim b As cl_Board
    Private Sub AnimalSimulation_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        'MsgBox("HI")
        b = cl_GlobalSetting.Board


    End Sub
    'Button for user to go back to the main menu
    Private Sub BTBack_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BTBack.Click

        Me.Chart1.Series("Population 1").Points.Clear()
        Me.Chart1.Series("Population 2").Points.Clear()
        Me.Chart1.Series("Population 3").Points.Clear()
```

```
        Me.Chart1.Series("Population 4").Points.Clear()
        Me.Chart1.Series("Population 5").Points.Clear()
        Application.Restart()
        Me.Hide()

        FM_Start.Show()
    End Sub
    'Draws simulation and population graph and initiates simulation.
    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click

        Dim counter As Integer = 0
        Dim log As New cl_log
        'Assigns animal names for labels in the population graph
        lblanimalONE.Text = cl_GlobalSetting.AnimalONE
        lblanimalTWO.Text = cl_GlobalSetting.AnimalTWO
        lblanimalTHREE.Text = cl_GlobalSetting.AnimalTHREE
        lblanimalFOUR.Text = cl_GlobalSetting.AnimalFOUR
        lblanimalFIVE.Text = cl_GlobalSetting.AnimalFIVE



        cl_GlobalSetting.log = log

        For day = 0 To cl_GlobalSetting.TotalSimulationTime
            'Assigns population sizes for labels in the population graph
            lblAnimalOnePOP.Text = cl_GlobalSetting.AnimalONEpopulation
            lblAnimalTwoPOP.Text = cl_GlobalSetting.AnimalTWOpopulation
            lblAnimalTHREEPOP.Text = cl_GlobalSetting.AnimalTHREEpopulation
            lblAnimalFOURPOP.Text = cl_GlobalSetting.AnimalFOURpopulation
            lblAnimalFIVEPOP.Text = cl_GlobalSetting.AnimalFIVEpopulation



            'Display or creates board
            lbl_day.Text = day & " days"    ' counts number of days
            Dim myGraphics As Graphics = Me.CreateGraphics
            Dim myPen As Pen
            myPen = New Pen(Brushes.Black, 1)
            Dim Brush As SolidBrush
            Dim rect As New RectangleF(0.0, 0.0, b.returnBOARDSIZEX,
b.returnBOARDSIZEY)

            ' Create rectangle.  ' Fill rectangle to screen.
            Select Case b.BoardType
                Case "Desert"
                    Brush = New SolidBrush(Color.LightGoldenrodYellow)
                    myGraphics.FillRectangle(Brush, rect)
                    myGraphics.FillRectangle(Brush, rect)
                Case "Greenland"
                    Brush = New SolidBrush(Color.ForestGreen)
                    myGraphics.FillRectangle(Brush, rect)
                    myGraphics.FillRectangle(Brush, rect)
                Case "Arctic"
                    Brush = New SolidBrush(Color.WhiteSmoke)
                    myGraphics.FillRectangle(Brush, rect)
                    myGraphics.FillRectangle(Brush, rect)
                Case Else
                    MsgBox(":( Type error for boardtype")

            End Select
```

```
            b.animalList.reset() 'sets linked list from the beginning


            'Create animals
            Do
                counter = counter + 1
                If IsNothing(b.animalList.current) = False Then

                    Select Case b.animalList.current.piece.AnimalName

                        Case cl_GlobalSetting.AnimalONE
                            Brush = New SolidBrush(Color.LightBlue) 'population 1

                        Case cl_GlobalSetting.AnimalTWO
                            Brush = New SolidBrush(Color.Yellow) 'population 2

                        Case cl_GlobalSetting.AnimalTHREE
                            Brush = New SolidBrush(Color.Red) 'population 3

                        Case cl_GlobalSetting.AnimalFOUR
                            Brush = New SolidBrush(Color.Green) 'population 4

                        Case cl_GlobalSetting.AnimalFIVE
                            Brush = New SolidBrush(Color.Purple) 'population 5


                    End Select


                    Dim Animaloutline As Pen 'Outline for each animal
                    Animaloutline = New Pen(Brushes.Black, 1)

                    Dim r As New RectangleF(b.animalList.current.X * 10,
b.animalList.current.Y * 10, 10, 10) 'dimensions for each animal inside the board
                    myGraphics.FillRectangle(Brush, r)
                    myGraphics.FillRectangle(Brush, r)
                End If


            Loop Until b.animalList.NextAnimalNode() = False

            If b.animalList.populationNum <= 0 Then 'end simulation when total
population numer = 0
                day = cl_GlobalSetting.TotalSimulationTime

            End If
            'Sets up population graph with day as x-axis and population size as y-axis
            Me.Chart1.Series("Population 1").Points.AddXY(day,
cl_GlobalSetting.AnimalONEpopulation)
            Me.Chart1.Series("Population 2").Points.AddXY(day,
cl_GlobalSetting.AnimalTWOpopulation)
            Me.Chart1.Series("Population 3").Points.AddXY(day,
cl_GlobalSetting.AnimalTHREEpopulation)
            Me.Chart1.Series("Population 4").Points.AddXY(day,
cl_GlobalSetting.AnimalFOURpopulation)
            Me.Chart1.Series("Population 5").Points.AddXY(day,
cl_GlobalSetting.AnimalFIVEpopulation)


            b.RunDay() 'run simulation day
```

```
            '20/1/2020 need to end simulation when population = 0



            System.Threading.Thread.Sleep(75) ' Speed at which the simulation runs
            Application.DoEvents()



        Next

        MsgBox("End Simulation")
    End Sub
    Public Sub AddNewBornToBoard() 'Draw new born animals to board



        Dim myGraphics As Graphics = Me.CreateGraphics
        Dim myPen As Pen
        myPen = New Pen(Brushes.Black, 1)
        Dim Brush As SolidBrush
        Dim rect As New RectangleF(0.0, 0.0, b.returnBOARDSIZEX, b.returnBOARDSIZEY)

        Select Case b.animalList.current.piece.AnimalName

            Case cl_GlobalSetting.AnimalONE
                Brush = New SolidBrush(Color.LightBlue) 'population 1

            Case cl_GlobalSetting.AnimalTWO
                Brush = New SolidBrush(Color.Yellow) 'population 2

            Case cl_GlobalSetting.AnimalTHREE
                Brush = New SolidBrush(Color.Red) 'population 3

            Case cl_GlobalSetting.AnimalFOUR
                Brush = New SolidBrush(Color.Green) 'population 4

            Case cl_GlobalSetting.AnimalFIVE
                Brush = New SolidBrush(Color.Purple) 'population 5

        End Select


        Dim Animaloutline As Pen 'Outline for each animal
        Animaloutline = New Pen(Brushes.Black, 1)

        Dim addBaby As New RectangleF(b.animalList.current.X * 10,
b.animalList.current.Y * 10, 10, 10) 'dimensions for each animal inside the board
        myGraphics.FillRectangle(Brush, addBaby)
        myGraphics.FillRectangle(Brush, addBaby)







    End Sub

End Class
```

```
'form containing the design of the instructions page
Public Class FM_Information
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
'back button
        Me.Hide()
        FM_Start.Show()
    End Sub

End Class
```

'form containing the options page where the user can configure a simulation.

Public Class FM_Options

Private Sub FM_Options_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)

Handles MyBase.Load 'Calls database to give information and sets all inputs to 0

Call ListAnimalNames("")

Call ListHabitatNames()

TBpopulationONE.Text = 0

TBpopulationTWO.Text = 0

TBpopulationTHREE.Text = 0

TBpopulationFOUR.Text = 0

TBpopulationFIVE.Text = 0

```
        TBtotalRunduration.Text = 0
    End Sub

    Sub ListAnimalNames(ByVal terrain As String) 'calls list of animals from database and displays them to user.

        'Animal names listed under terrain type

        Dim ListofAnimal As New cl_Database
        Dim AnimalNamesdataTable As New DataTable

        If terrain = "" Then
            AnimalNamesdataTable = ListofAnimal.selectAnimal()
            If AnimalNamesdataTable.Rows.Count > 0 Then
                For x = 0 To AnimalNamesdataTable.Rows.Count - 1
                    CBanimalONEpop.Items.Add(AnimalNamesdataTable.Rows(x).Item(0))
                    CBanimalTWOpop.Items.Add(AnimalNamesdataTable.Rows(x).Item(0))
                    CBanimalTHREEpop.Items.Add(AnimalNamesdataTable.Rows(x).Item(0))
```

```
            CBanimalFOURpop.Items.Add(AnimalNamesdataTable.Rows(x).Item(0))

            CBanimalFIVEpop.Items.Add(AnimalNamesdataTable.Rows(x).Item(0))



        Next

    End If



Else

    AnimalNamesdataTable = ListofAnimal.selectAnimalTerrain(terrain)

    If AnimalNamesdataTable.Rows.Count > 0 Then

        For x = 0 To AnimalNamesdataTable.Rows.Count - 1

            CBanimalONEpop.Items.Add(AnimalNamesdataTable.Rows(x).Item(4))

            CBanimalTWOpop.Items.Add(AnimalNamesdataTable.Rows(x).Item(4))

            CBanimalTHREEpop.Items.Add(AnimalNamesdataTable.Rows(x).Item(4))

            CBanimalFOURpop.Items.Add(AnimalNamesdataTable.Rows(x).Item(4))

            CBanimalFIVEpop.Items.Add(AnimalNamesdataTable.Rows(x).Item(4))



        Next

    End If
```

```
    End If

End Sub

Sub ListHabitatNames() 'Calls list of habitats from database and displays to user.

    'Habitat Names are listed under the terrain type

    Dim HabitatNamesdataTable As New DataTable
    Dim ListofHabitat As New cl_Database

    HabitatNamesdataTable = ListofHabitat.selectTerrain()

    If HabitatNamesdataTable.Rows.Count > 0 Then
        For x = 0 To HabitatNamesdataTable.Rows.Count - 1
```

```
            CBterrainType.Items.Add(HabitatNamesdataTable.Rows(x).Item(1))



        Next

    End If




End Sub

Function Animaldata(ByVal AnimalName As String) As cl_Animal 'Gets selected animals' data from database and sends to simulation.

    'Get animaldata

    Dim AnimaldataTable As New DataTable

    Dim AnimalDb As New cl_Database

    AnimaldataTable = AnimalDb.selectAnimalData(AnimalName)
```

```
        If AnimaldataTable.Rows.Count > 0 Then


            Select Case AnimaldataTable.Rows(0).Item(1)

                Case "Herbivore"

                    Dim a As New cl_Herbivore(AnimaldataTable.Rows(0).Item(0),
AnimaldataTable.Rows(0).Item(2), AnimaldataTable.Rows(0).Item(3),
AnimaldataTable.Rows(0).Item(4), AnimaldataTable.Rows(0).Item(5),
AnimaldataTable.Rows(0).Item(6), AnimaldataTable.Rows(0).Item(7))


                    Return a

                Case "Omnivore"

                    Dim a As New cl_Omnivore(AnimaldataTable.Rows(0).Item(0),
AnimaldataTable.Rows(0).Item(2), AnimaldataTable.Rows(0).Item(3),
AnimaldataTable.Rows(0).Item(4), AnimaldataTable.Rows(0).Item(5),
AnimaldataTable.Rows(0).Item(6), AnimaldataTable.Rows(0).Item(7))

                    Return a

                Case "Carnivore"

                    Dim a As New cl_Carnivore(AnimaldataTable.Rows(0).Item(0),
AnimaldataTable.Rows(0).Item(2), AnimaldataTable.Rows(0).Item(3),
AnimaldataTable.Rows(0).Item(4), AnimaldataTable.Rows(0).Item(5),
AnimaldataTable.Rows(0).Item(6), AnimaldataTable.Rows(0).Item(7))
```

```
                Return a

            End Select

        Else

            Return Nothing

        End If




        Return Nothing



    End Function



    Private Sub BTback_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
BTback.Click 'back button to main menu

        Me.Hide()

        FM_Start.Show()

    End Sub
```

```
    'button to move to simulation page

    Private Sub BTRunSimulation_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles BTRunSimulation.Click

        Dim validInput As Boolean = False



        If CBterrainType.Text = Nothing Or CBterrainType.Text = "                    Terrain Type" Then

            CBterrainType.Text = "                    Terrain Type"

            MsgBox("HABITAT is not valid.")



        ElseIf TBpopulationONE.Text < 0 Then 'Or IsNumeric(TBpopulationONE.Text) = False Or
TBpopulationONE.Text = Nothing



            TBpopulationONE.Text = 0

            MsgBox("population one size is not acceptable.")



        ElseIf IsNumeric(TBpopulationTWO.Text) = False Or TBpopulationTWO.Text = Nothing Or
TBpopulationTWO.Text < 0 Then '



            TBpopulationTWO.Text = 0
```

```
        MsgBox("population 2 size is not acceptable.")


    ElseIf IsNumeric(TBpopulationTHREE.Text) = False Or TBpopulationTHREE.Text = Nothing Or
TBpopulationTHREE.Text < 0 Then '


        TBpopulationTHREE.Text = 0

        MsgBox("population 3 size is not acceptable.")


    ElseIf IsNumeric(TBpopulationFOUR.Text) = False Or TBpopulationFOUR.Text = Nothing Or
TBpopulationFOUR.Text < 0 Then '


        TBpopulationFOUR.Text = 0

        MsgBox("population 4 size is not acceptable.")


    ElseIf IsNumeric(TBpopulationFIVE.Text) = False Or TBpopulationFIVE.Text = Nothing Or
TBpopulationFIVE.Text < 0 Then '


        TBpopulationFIVE.Text = 0

        MsgBox("population 5 size is not acceptable.")
```

```
ElseIf IsNumeric(TBtotalRunduration.Text) = False Then 'Or TBtotalRunduration.Text < 0



    TBtotalRunduration.Text = 0

    MsgBox("Simulation run-time length value is not numerical.")



Else

    validInput = True

End If



If validInput = True Then



    cl_GlobalSetting.TotalSimulationTime = TBtotalRunduration.Text



    setup()

    Me.Close()

    FM_AnimalSimulation.Show()
```

```
        End If



    End Sub


    Private Sub setup()


        cl_GlobalSetting.Board = New cl_Board(CBterrainType.Text) 'Setup board from terraintype
information in options


        'Setup population sizes and animal specie from options


        If TBpopulationONE.Text <> 0 Then

            cl_GlobalSetting.Board.addAnimal(Animaldata(CBanimalONEpop.Text),
TBpopulationONE.Text)

            cl_GlobalSetting.AnimalONEpopulation = TBpopulationONE.Text

            cl_GlobalSetting.AnimalONE = CBanimalONEpop.Text

        End If
```

```
        If TBpopulationTWO.Text <> 0 Then

            cl_GlobalSetting.Board.addAnimal(Animaldata(CBanimalTWOpop.Text),
TBpopulationTWO.Text)

            cl_GlobalSetting.AnimalTWOpopulation = TBpopulationTWO.Text

            cl_GlobalSetting.AnimalTWO = CBanimalTWOpop.Text

        End If



        If TBpopulationTHREE.Text <> 0 Then

            cl_GlobalSetting.Board.addAnimal(Animaldata(CBanimalTHREEpop.Text),
TBpopulationTHREE.Text)

            cl_GlobalSetting.AnimalTHREEpopulation = TBpopulationTHREE.Text

            cl_GlobalSetting.AnimalTHREE = CBanimalTHREEpop.Text

        End If



        If TBpopulationFOUR.Text <> 0 Then

            cl_GlobalSetting.Board.addAnimal(Animaldata(CBanimalFOURpop.Text),
TBpopulationFOUR.Text)

            cl_GlobalSetting.AnimalFOURpopulation = TBpopulationFOUR.Text

            cl_GlobalSetting.AnimalFOUR = CBanimalFOURpop.Text
```

```
        End If



        If TBpopulationFIVE.Text <> 0 Then

            cl_GlobalSetting.Board.addAnimal(Animaldata(CBanimalFIVEpop.Text),
TBpopulationFIVE.Text)

            cl_GlobalSetting.AnimalFIVEpopulation = TBpopulationFIVE.Text

            cl_GlobalSetting.AnimalFIVE = CBanimalFIVEpop.Text



        End If



    End Sub







    'Creates a list of animals for a specific habitat

    Private Sub CBterrainType_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles CBterrainType.SelectedIndexChanged

        Dim terrain As String = CBterrainType.Text

        CBanimalONEpop.Items.Clear()
```

```
        CBanimalTWOpop.Items.Clear()

        CBanimalTHREEpop.Items.Clear()

        CBanimalFOURpop.Items.Clear()

        CBanimalFIVEpop.Items.Clear()




        Call ListAnimalNames(terrain)




    End Sub



    'STOP USER FROM SELECTING A SELECTED ANIMAL

    Private Sub CBanimalONEpop_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles CBanimalONEpop.SelectedIndexChanged

        CBanimalTWOpop.Items.Remove(CBanimalONEpop.SelectedItem)

        CBanimalTHREEpop.Items.Remove(CBanimalONEpop.SelectedItem)
```

```
        CBanimalFOURpop.Items.Remove(CBanimalONEpop.SelectedItem)

        CBanimalFIVEpop.Items.Remove(CBanimalONEpop.SelectedItem)




    End Sub

    'STOP USER FROM SELECTING A SELECTED ANIMAL

    Private Sub CBanimalTWOpop_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles CBanimalTWOpop.SelectedIndexChanged

        CBanimalTHREEpop.Items.Remove(CBanimalTWOpop.SelectedItem)

        CBanimalFOURpop.Items.Remove(CBanimalTWOpop.SelectedItem)

        CBanimalONEpop.Items.Remove(CBanimalTWOpop.SelectedItem)

        CBanimalFIVEpop.Items.Remove(CBanimalTWOpop.SelectedItem)



    End Sub

    'STOP USER FROM SELECTING A SELECTED ANIMAL

    Private Sub CBanimalTHREEpop_SelectedIndexChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles CBanimalTHREEpop.SelectedIndexChanged

        CBanimalFOURpop.Items.Remove(CBanimalTHREEpop.SelectedItem)

        CBanimalFIVEpop.Items.Remove(CBanimalTHREEpop.SelectedItem)
```

```
        CBanimalTWOpop.Items.Remove(CBanimalTHREEpop.SelectedItem)

        CBanimalONEpop.Items.Remove(CBanimalTHREEpop.SelectedItem)

    End Sub

    'STOP USER FROM SELECTING A SELECTED ANIMAL

    Private Sub CBanimalFOURpop_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles CBanimalFOURpop.SelectedIndexChanged

        CBanimalFIVEpop.Items.Remove(CBanimalFOURpop.SelectedItem)

        CBanimalTHREEpop.Items.Remove(CBanimalFOURpop.SelectedItem)

        CBanimalTWOpop.Items.Remove(CBanimalFOURpop.SelectedItem)

        CBanimalONEpop.Items.Remove(CBanimalFOURpop.SelectedItem)

    End Sub

    'STOP USER FROM SELECTING A SELECTED ANIMAL

    Private Sub CBanimalFIVEpop_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles CBanimalFIVEpop.SelectedIndexChanged

        CBanimalONEpop.Items.Remove(CBanimalFIVEpop.SelectedItem)

        CBanimalFOURpop.Items.Remove(CBanimalFIVEpop.SelectedItem)

        CBanimalTHREEpop.Items.Remove(CBanimalFIVEpop.SelectedItem)

        CBanimalTWOpop.Items.Remove(CBanimalFIVEpop.SelectedItem)
```

End Sub

End Class

```
Public Class FM_Start 'Form used as the main menu for the program.

    Private Sub FM_Start_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load 'loads main menu
    End Sub

    'button that moves user to options
    Private Sub btOption_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btOption.Click

        Me.Hide()
        FM_Options.Show()
    End Sub


    Private Sub btQuit_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btQuit.Click 'close entire program
        Close()

    End Sub
    'button that moves user to information page
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Me.Hide()
        FM_Information.Show()
    End Sub


End Class
```

Reference