****

Data Structure

Project Report

**Topic：Biosphere Simulator**

**Faculty 计算机科学与工程学院**

**Program Data Structure Course Design**

**Name 蔡xx 庄xx 卢xx 翁xx 姚xx**

**Student ID xxxxxxxxxxxx xxxxxxxxxxxx**

**xxxxxxxxxxxx xxxxxxxxxxxx**

**xxxxxxxxxxxx**

**Lecturer 陈百基**

**Course Code 045101682**

**Credit 1.0**

**Start/End Date 2022-2023**

|  |  |
| --- | --- |
| Teacher’s Comment | Signature：  Date： |
| Score |  |
| Remark |  |

**Topic**

1. **Background**

Environmental protection has always been a hot topic. There are many ways to explore the changes in the environment over a certain period of time, one of which is ecological environmental simulation. In order to contribute our own efforts to environmental protection, we decide to imitate the ecosphere in this project. There are five species in this tiny ecosphere, which are grass, cow, hare, tiger and fox. There exists predation relationship between these five species and we will discuss it in the second part of the report. We will also simulate some basic behaviors of these animals, such as growth, reproduction, movement, predation, excretion, aggregation, and death. All behaviors will follow the carefully designed logic that we have developed. We will also provide exquisite GUI to allow users to intuitively experience the changes in the position of each animal and the changes in species population.

In this project, we hope to simulate as many species as possible simultaneously. In order to achieve the goal, we use multiple data structures to improve the performance of our program. The data structures including heap, freelist, queue, red-black tree. Some of them are implemented completely by ourselves. We will discuss these data structures in detail in the third part of our project. We also apply BFS to our project to decrease the time complexity.

What’s more, we also consider the effect of environment in our project. We consider the effect of weather, season and terrain in our project, which makes the simulation more authenticity.

1. **Design Principle**

This section will state the requirement of each kind of animals and the functions of the program. The assumptions have already included in the requirement part and the function part.

**Requirements**

The requirements of five creatures are listed below.

Grass

* Grass can grow in the place that is not labeled as water and have the strong regenerative ability.
* Grass can be predated by cow.
* Grass can predate nothing.
* Grass can only provide a small amount of energy if they are predated.
* Grass has short-life span.
* Grass has parameters including growth time, time, and so on.
* The growth speed of the grass is affected by the weather and season

Cow

* Cow is strong and can run fast.
* Cow can be predated by tiger.
* Cow can predate grass.
* Cow can provide a great amount of energy if it is predated.
* Cow has a long-life span.
* Cow always tends to move together
* Cow has parameters including initial number, energy, age, speed, and so on.
* The moving speed and the eye sight of the cow can be affected by environment

Hare

* Hare is weak and can run fast.
* Hare can be predated by tiger and fox.
* Hare can predate grass.
* Hare can provide a great amount of energy if it is predated.
* Hare has a long-life span.
* Hare has parameters including initial number, energy, age, speed, and so on.
* The moving speed and the eye sight of the hare can be affected by environment.

Tiger

* Tiger is strong and can run very fast.
* Tiger can be predated by nothing.
* Tiger can predate cow and hare.
* Tiger has a long-life span.
* Tiger tends to move together when they are full
* Tiger has parameters including initial number, energy, age, speed, and so on.
* The moving speed and the eye sight of the tiger can be affected by environment.

Fox

* Fox is strong and can run very fast.
* Fox can be predated by nothing.
* Fox can predate cow and hare.
* Fox has a long-life span.
* Fox has parameters including initial number, energy, age, speed, and so on.
* The moving speed and the eye sight of the fox can be affected by environment.

**Functions**

For grass, the whole grassland is divided into thousands of small grass grids for individual. Grass in the same grid is assumed to have the same state and will be eaten by the sheep simultaneously.

The state of the grass is continuous, but we change the color of the grass only when the value of the function that describes the state of grass reaches particular thresholds. The changing rate of the grass will be influenced by other factors. The factors include weather, season and the excrement of other animals.

Each animal has an energy attribute, which mainly decide the behavior of the animal in a particular frame. The energy will change between two neighboring frames. Animals will try to seek food if their energy is lower than a particular bound and will have possibility to reproduce if their energy is higher than a particular bound. The loss and gain of energy are different between two neighboring frames due to their behavior. There are three circumstances in which an animal will die. The first is when the energy level drops below zero. The second is when the age exceed the age limit, and the third is when they are killed by other animals. One intriguing feature of animals is their ability to excrete waste. The excrement of animal will affect the growth rate of grass. All animals also have an eyesight attribute. If the animals cannot decide their behavior due to the information they get in their eyesight, they will just walk or run randomly. What should be notice is that we only list some of the attributes here. The behavior of animals in particular environment will be affected by some parameters. These parameters are shared among a particular kind of animal, which can lead to some interesting effect, such as aggregation.

In our project, we divide four kinds of animals, which are tiger, fox, hare and cow into two kinds, which are prey and predator. Two kinds of animal will have different functions, which are illustrated below.

For prey, they will get many information in a single frame. Different information may suggest them to go to different directions. In order to solve this conflict, each prey uses a heap to obtain the most important pieces of information, and the prey will decide its behaviors due to the analysis of these information. With the help of heaps, preys can handle certain complex situations, such as determining the optimal escape direction when being chased by multiple predators at the same time. The preys can also decide whether it is suitable to reproduce or excrete according to the information they obtain.

For predators, they can also get information in their eyesight. If the predators are hungry, they will search all the prey in their eyesight and try to trace the closest prey. If they found any prey, the predators also know whether it is possible for them to catch the animals in this frame and modify their behavior due to this important information. A predator will also share food with other predators if other predators also catch the same prey. Predators will move slowly if they are not hungry, otherwise they will move at a higher speed so that they can wither catch the prey with higher possibility or search a wider range at the same time. Of course, the predators can also decide whether it is suitable to reproduce or excrete based on their current activity.

Within the predator and prey, there are also variations in the behavior of different animals. After consulting relevant sources, we restrict that the cow can only be preyed by tiger, since the larger size of cows compared to foxes. Based on the reality, we have classified tigers and cows as social animals, while hares and foxes are considered solitary animals. Though the effect of the function that control aggregation is not obvious, it does affect the behavior of social animals.

1. **Model Discussion**

**Model**

This section will introduce our project’s logic flow.

In our project, we will firstly initialize the map and the initial number of animals according to the user’s input. Then update each frame’s map state and animal position. How to properly deal with the sequence of animals’ action is the main problem to solve. Our logic of updating each frame’s information will be shown below.

We firstly define the priority for animal’s different actions. For predator, if it has found a prey, the most urgent action of it should be chasing the prey until eat the prey. Otherwise, if the predator stops to do other thing when chasing a prey, the prey may run away from it. Similar to predator, the most urgent action of prey when chased by some predators is to run away. Then, the animal will consider to do other things like eating grass, walking around to look for a prey. Finally, they may excrete waste when they are not in urgent state, like eating the prey or randomly walking around.

To satisfy the requirement stated above, there are four loop in each frame’s update function to implement their action.

In the first loop, the program will traverse the prey list. Each prey will choose their action, and report the location they walk to. Our program will put these location into a map, waiting for being used in the following loop.

In the second loop, the program will traverse the predator list. Each predator will choose their action, report the location they walk to. The difference from the prey is that, when predator find that it can catch the prey in this frame, it will look for the prey’s location in the map, which is filled in the first loop, then go to there and report their need for extra information. The extra information decide how much energy the predator can get from the prey, which can settle down the conflict when two predators catch the prey in the same time. The predators that need extra information will be put into a stack, while other predator will update their states such as age, energy and position in the map.

In the third loop, the extra information will be calculated and sent to the predators since each prey caught has known the number of predators caught it. The predator in the stack will update their states.

In the fourth loop, it’s time for preys to update their position according to their location in map. Other state like age and energy will also be updated properly.

Besides the four loops illustrated above, the map states, such as season, weather, grass state, waste location, will also be updated in each frame. What’s more, the newly reproduced animal will be added into the map and animal list, waiting for updating state in the following frames.

The flowchart of our logic is shown below:

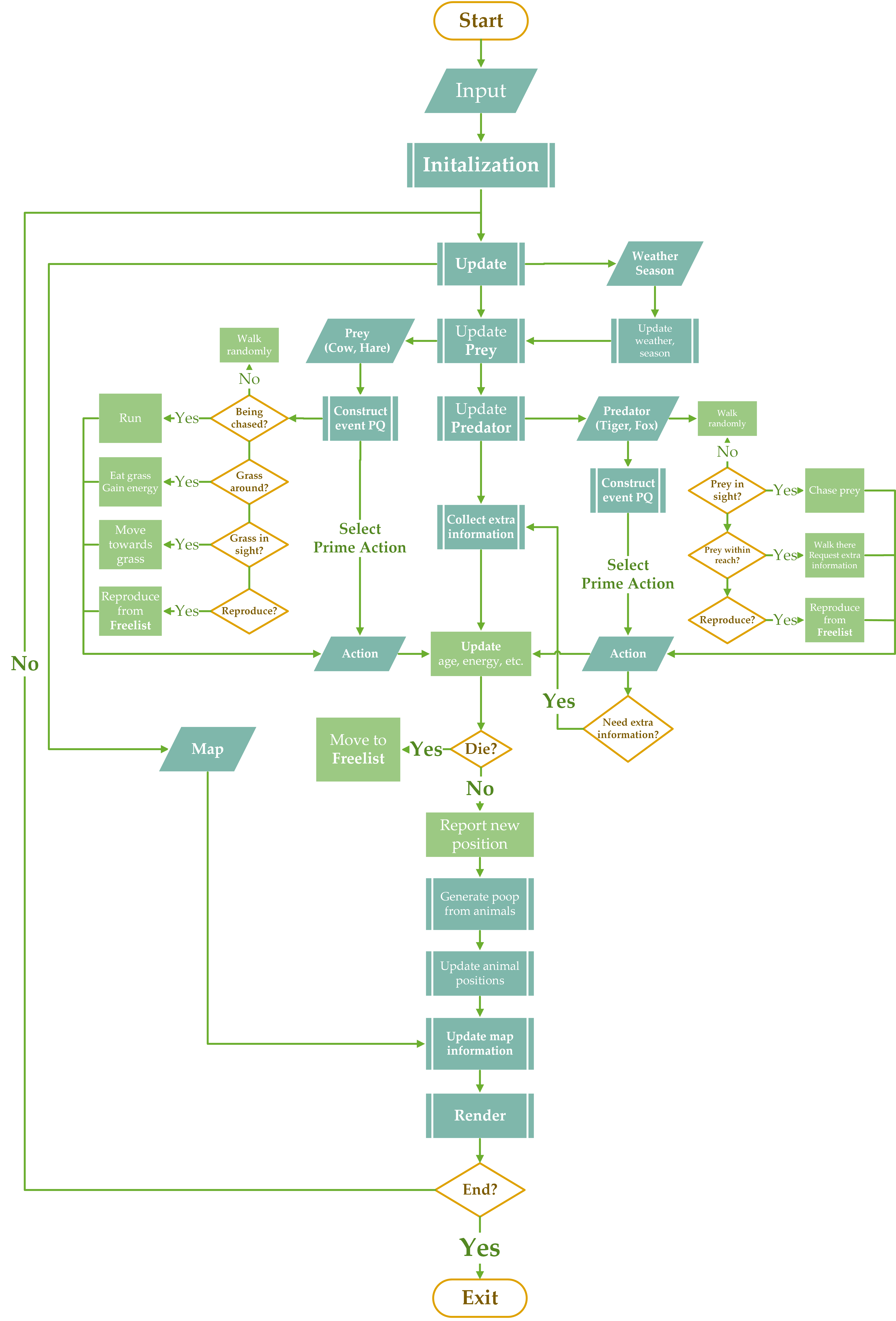


Figure 1

**Complexity Analysis**

Here are the symbols that we use during the analysis.

|  |  |
| --- | --- |
| Symbol name | Description |
|  | The number of grass grid |
|  | The number of cows |
|  | The number of hares |
|  | The number of tigers |
|  | The number of foxes |
|  | The number of grass grids in the eyesight of cows |
|  | The number of grass grids in the eyesight of hares |
|  | The number of grass grids in the eyesight of tigers |
|  | The number of grass grids in the eyesight of foxes |
|  | The number of events that is used to decide the prey’s behavior when there are numbers of events |
|  | The proportion of animals that excrete waste |
|  | The total number of animals |

In the project, the data that is needed in a single frame is calculated by the class about logic before transferring to the class which is responsible for rendering. Here we will analysis the time complexity in a single frame.

Step 1. Iterate through the link list that store all the prey for the first time. This loop will update the information of preys. For each prey, it will explore the grass grids in its eyesight from nearest to farthest. The moment it gets enough information to decide its behavior, it will quit the searching loop. For the worst situation, it will iterate all the grass grids in its eyesight. All information will be put into a heap and the most valuable pieces of information will be used to decide the behavior of cows. So, the time complexity for a single prey is:

Since is set to in our program and will be up to in the worst case, we simplify the time complexity in this stage for an animal to be . So, the time complexity for all the animal is:

What’s more, we also use a map to store the information about the prey and the position of it in this frame. The time complexity of inserting a value into a map, whose underlying implementation is a red-black tree is:

So, the time complexity of this step is:

Step 2. Iterate through the link list that store all the predators. This loop will update the information of predators. For each predator, there will be two situations.

Situation 1. The predator is hungry. In this situation, a tiger will try to iterate all the grass grids in its eyesight from nearest to farthest. The moments it sees the prey, it will quit the search process. In the worst case, it will iterate all the grids in its eyesight. The time complexity is:

In this situation, an animal needs to access the red-black tree if it can catch up with an animal. The time complexity of accessing a node in the red-black tree generated in step 1 is:

Situation 2: The predator is full. The predator will just walk around. The time complexity of direction decision is:

Situation 3: The predator is full and it is tiger, which means it try to find a tiger and follow it. The time complexity is:

Situation 4: The predator catch a prey and it is eating it. The time complexity in this situation is:

So, the time complexity in the worst situation, which means every predator is hungry and can catch up with a prey in this step is:

Step 3. Iterate through the link list that store the predators that catch a prey in step 2. In this loop, the logic will provide necessary for the predators that catches a prey. This loop is simply passing an integer value, and the time complexity in the worst case of the loop is:

Step 4. Iterate through the link list that store all the preys for the second time. This time, the logic will deal with the pointer of preys that is killed by the predators. The time complexity of the worst situation in this step is:

Step 5. Decide which animals will excrete by a queue. The time complexity of popping a particular number of animals is:

Step 6. Iterate through all the grass grid and deal with the energy of the grass. The time complexity is:

Step 7. Refresh the weather and season of the whole map. The time complexity is:

All the calculation that is needed to decide the state and position of the new animal is shown above. The total time complexity is:

In the program, when the number of animals becomes larger, the eyesight is relatively small compared with the number. The number of animals can even go to about ten thousand while the eyesight is fixed to less than one thousand. So, we think it is reasonable to simplify the total time complexity to be:

Now we use to denote the total number of animals at a particular time. Since is a fix number in the program, the final time complexity is:

We want to state here that we use Breadth First Search algorithm in step 2 and step 3, which means that each animal does not need to iterate all the grids in its eyesight. In our testing strategy, most of the animal can quit the searching loop after scanning only a few numbers of grids. This can significantly improve the performance of the program as the number of animals increases.

We also want to state that we use a red-black tree in step 2 and step 3. One may argue that we can also use an array or vector to pass the information between two iterations, since the map simply maps an integer to a STL pair. If we use the pair, we can decrease the time complexity of the whole program from to . But there are still some limitations. During the simulation of the biosphere, each animal be assigned a unique ID. After the animal died, their ID will not be recycled. This means that using an array to store the position information will lose lots of space, while keeping the time complexity of the program low. There is also a fatal problem of using vector rather than red-black tree. As the ID grows larger and larger, though the number of alive animals may not exist the boundary of the vector, the program may require a very large vector to store the information of animal, where most slots of the vector are null. The size of the vector may excess the boundary of vector and the allocation process may be stopped by the OS. This actually a tradeoff between the space complexity and time complexity. We decided to sacrifice the time complexity to keep the space complexity low. So, we use a red-black tree here instead of a vector or array with unacceptable size.

**Data Structures**

This section we will talk about how we apply the technique of data structure in our project.

1. Vector

We use vector which define in C++ STL Standard library to store the necessary data, such like the map’s information, the animal pointer, the position of animals etc. Vector is essentially a linked list, so we can quickly insert or delete elements at the end of the vector. The most important thing about vector is that it can achieve dynamic capacity growth. This is the most important reason we use vector, because the number of animals and other necessary data changes over time.

2. Map

We use map which is also define in C++ STL Standard library to map some data into other data to establish a connection, such like we map prey animal’s id to prey animal’s position, so we can easily update the position of prey animal according to the different id. The connection between prey animal’s id to prey animal’s position also can be used for the predator’s hunt, the predator can know the position of prey animal if it knows the prey animal’s id. The underlying implementation of map is red-black tree, red-black tree is an approximately balanced binary tree, and the Time complexity of searching is . The red-black tree has less rotation times, so it performs better in the process of addition and deletion elements.

3. Free list

We will implement the free list for animal instances. We use pointer to store the instance of animals. When an animal dies, we do not delete the pointer. Instead, we use the pointer to make the animal invisible and place it into the free list. When we need to create an animal instance, we firstly search in the free list. If there exist a pointer which is the type of the animal we want, we will retrieve the pointer form free list and reset all parameters to finish the task. Otherwise, we will create a new pointer to store a new instance of the animal. It reduces the times of using new and delete functions, resulting in a reduction in time complexity and space complexity.

4. Queue

We will use queues to store animal pointers that meet excretory conditions. When an animal excretes waste, it will pop out of the queue. When an animal meets the conditions for excreting waste, it will discharge to the end. Within a certain period of time, we will pop up a few animals from the queue and let them excrete waste.

5. Max Heap

We will build a max heap to store the possible actions of the prey animal. Due to prey animal have several possible behaviors, such as run away from predator, eat the highest priority grass, walk randomly, gather with other prey animals (only for cow), and so on. We firstly use the priority queue to define a min heap, then we push back the number of we select event into min heap. Next, we compare the priority between the event in the top of min heap and new event. If new event’s priority larger than the priority of the event in the top of min heap, we pop in min heap and then push back the new event into min heap, else we will do nothing. In the end, we build a max heap for the behaviors of prey animal. According to the behaviors store in max heap, we will decide what the prey animal should do next. For example, if there are two or more predators in the sight of prey animal, which direction should the prey animal run in.

6. Breadth first search

We use breath first search when animal search information. When animal need to update its information such as position, it will start search from the position of itself, then it will always expand the recent map position, if it considers it had gotten enough information, it will break from the search immediately, it will reduce the time consume for update information.

1. **Demonstration**

1. Start and Set

In the Biosphere Simulator start surface, as shown in Figure 2, there are four blanks designed for setting the initial numbers of the different species. Users input data into the corresponding blanks and complete the initial data setting. By the way, around the initial value setting blanks, there are PNG of four animals. When clicking on the images, users can replace the corresponding animal identification with other images they want, as shown in Figure 3. After setting, users click on the start button and the simulation functions start to run. The surface jumps to the next one.

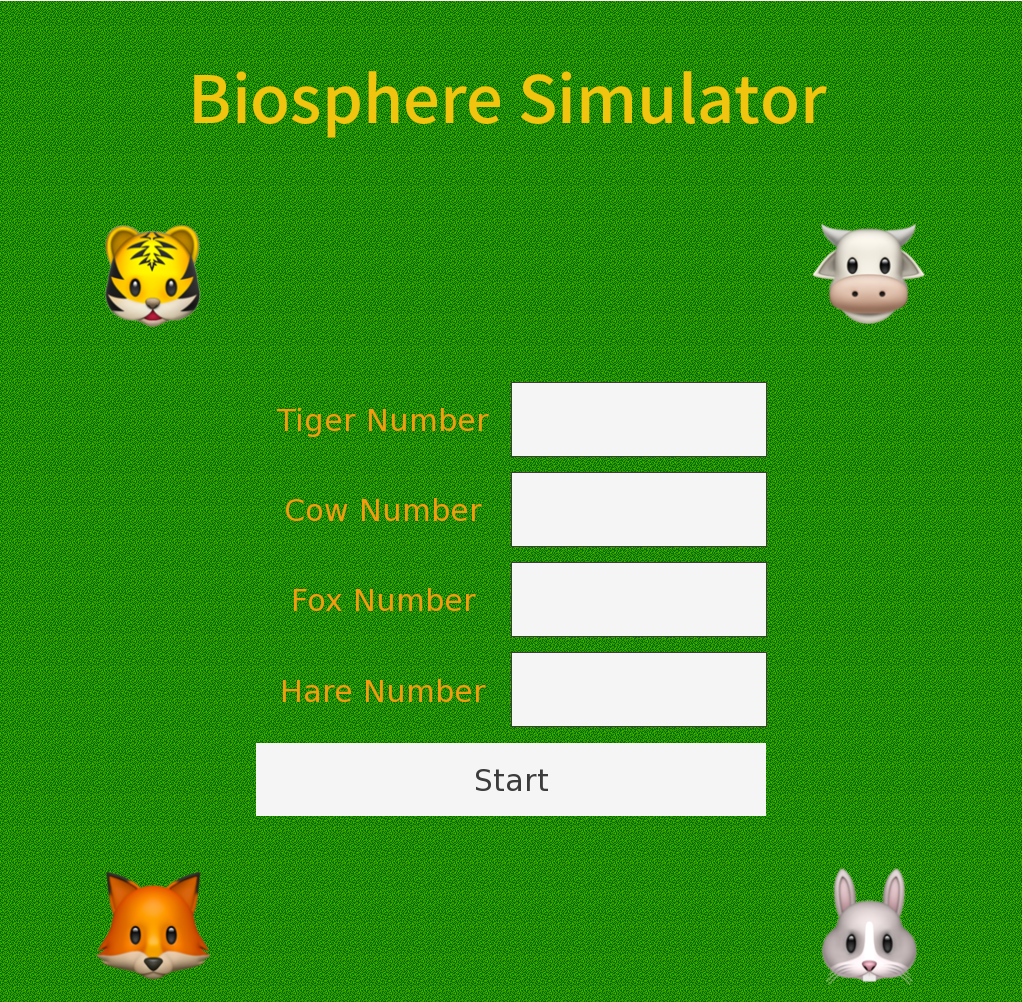
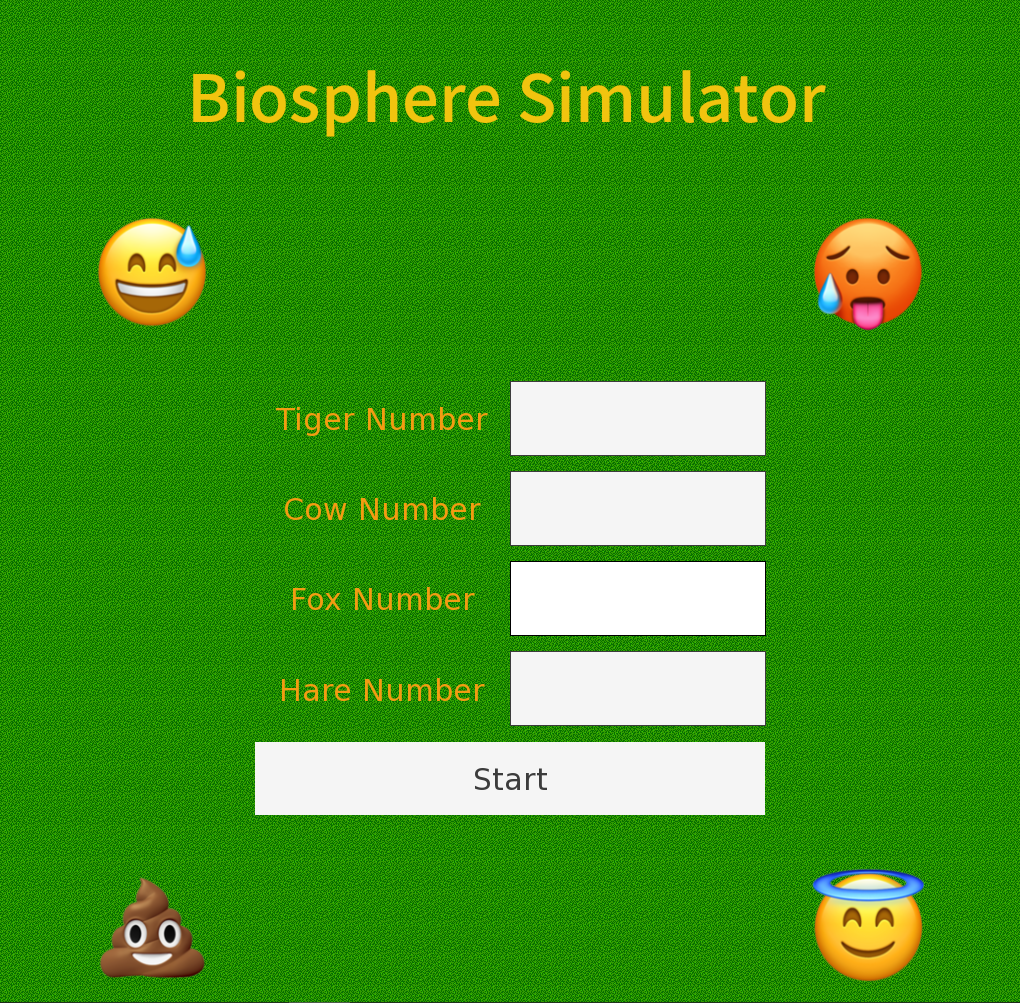
 

Figure 2 Figure 3

2.Simulate and display

In the second surface, it displays the behaviors of all species in the simulated biosphere, as shown in Figure 4. Users can get the information about the current season, weather, and numbers of species in the green blank on the right of this interface.

Explanation of the map: The map is composed of different blocks. The blue blocks represent puddles, where animals will slow down when passing through them. Other different colors of the blocks represent the states of grass growing in the corresponding blocks. The more grass, the darker the green of the block. Figure 4 and Figure 4 show the change of map in a few seconds.

As time goes by, the weather and seasons also change, affecting the growth of grass and the speed of animal movement.

Explanation of animal behavior: Every image represents an animal. To get energy, the cow and the hare may stop in the green block for eating grass; the tiger and the fox need to catch preys. When its energy is enough for the setting, there is a probability that the animal will give birth, a new image appearing on the block. When its energy is less than the setting, the animal will die with its image disappearing. When preys sense that there are predators nearby, they will accelerate their movement speed to escape. When predators see preys, they will run faster to catch them. Once caught, the prey will die and the image will disappear, remaining the predator’s image on the block. Figure 6 and figure 7 show the process of pursuit between a cow and a tiger. Every animal has its maximum lifespan. When the lifetime is over, the animal will die the image will disappear. Animals also randomly excrete waste. In that case, an image of waste will be generated on this block and exist for a period of time, accelerating the growth of the grass, as shown in Figure 8.

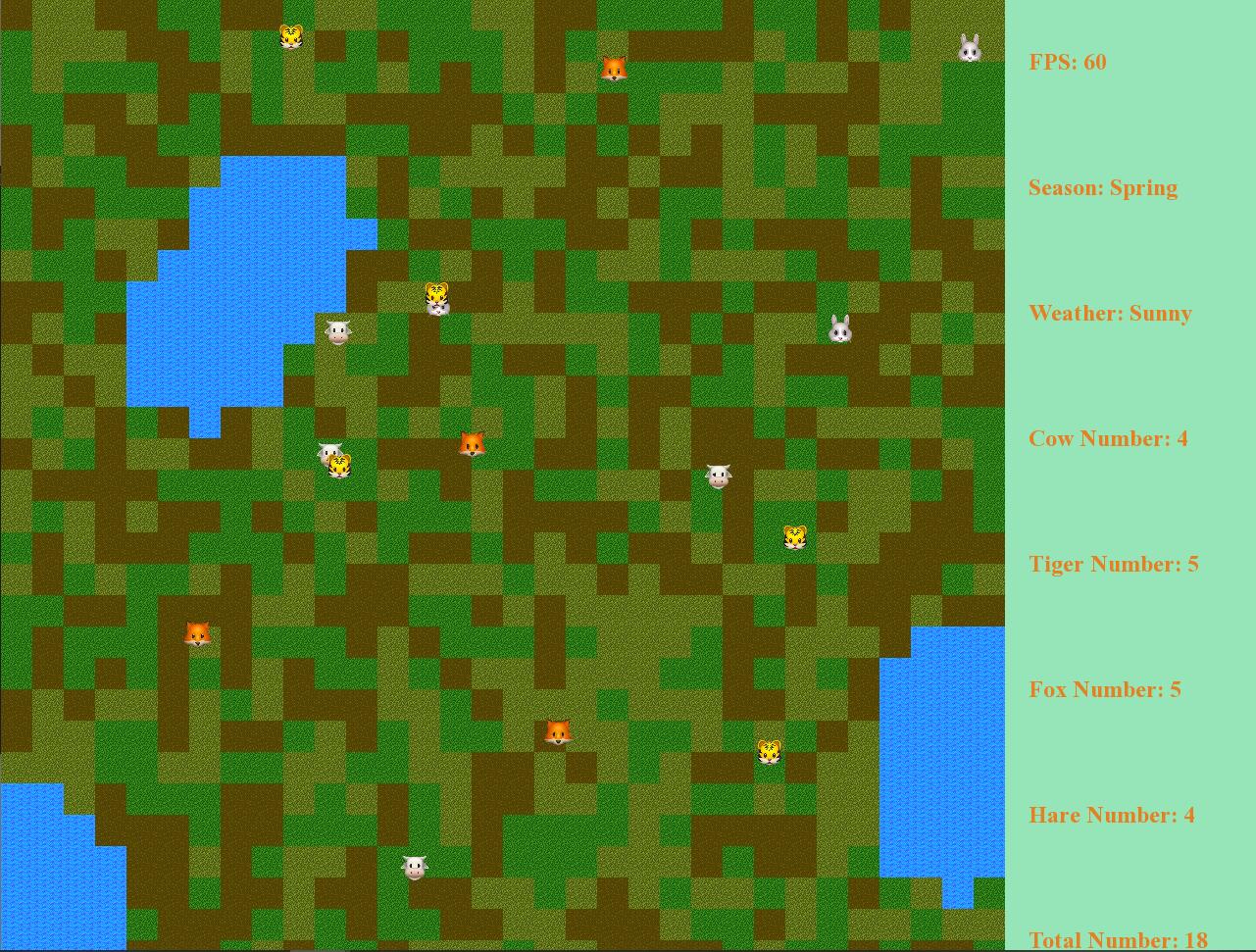
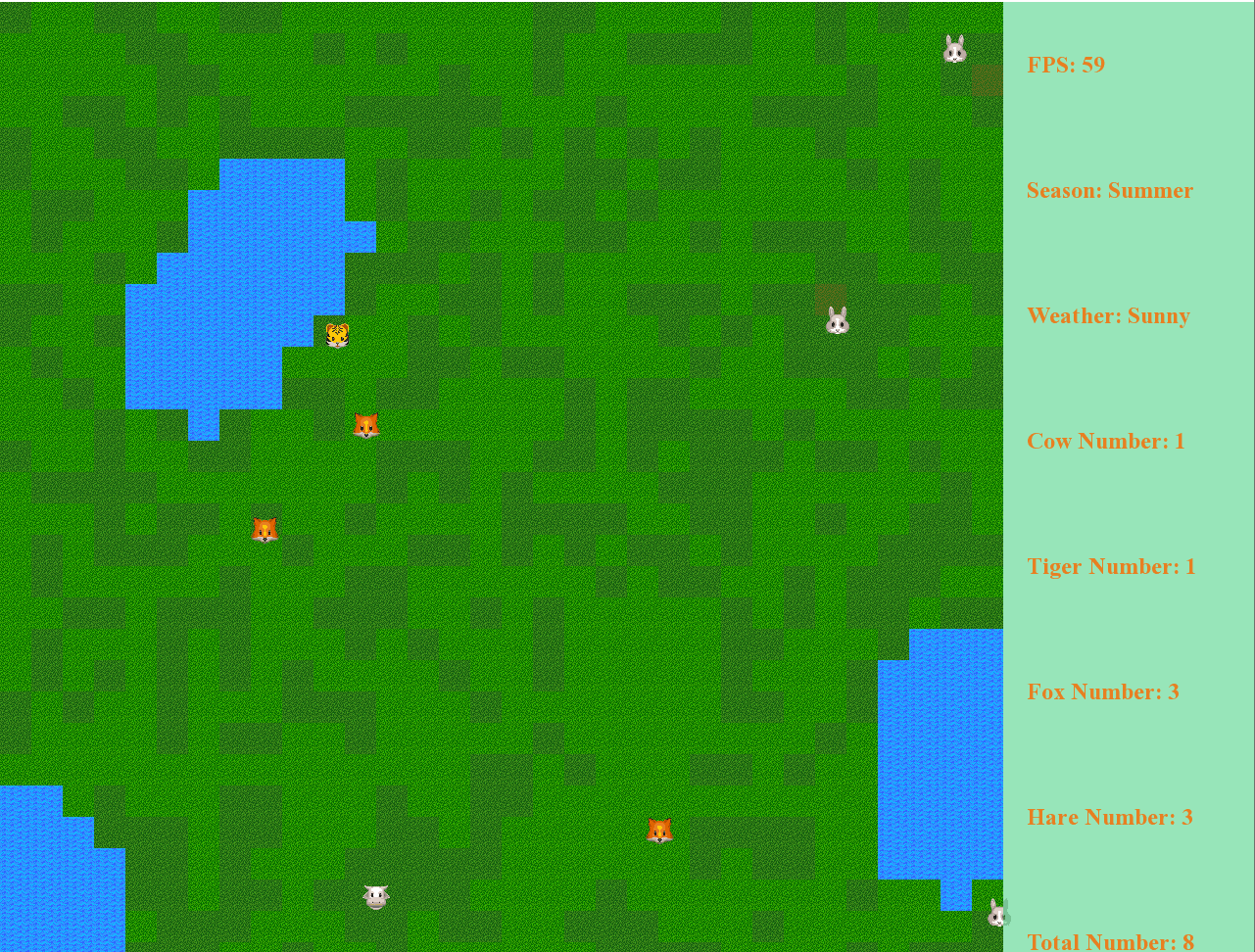
 

Figure 4 Figure 5

Figure 6 Figure 7 Figure 8

3.Report and analyze the results.

When the data of the animal number is reasonable, the result is that the animals spread uniformly and clearly, as shown in Figure 9. When the number is too large, the screen will become unsmooth and many details cannot be presented clearly, as shown in Figure 10. In the Figure 10, the initial numbers of all species are 5000, resulting that the screen is crowded and the frame per second is only 10.

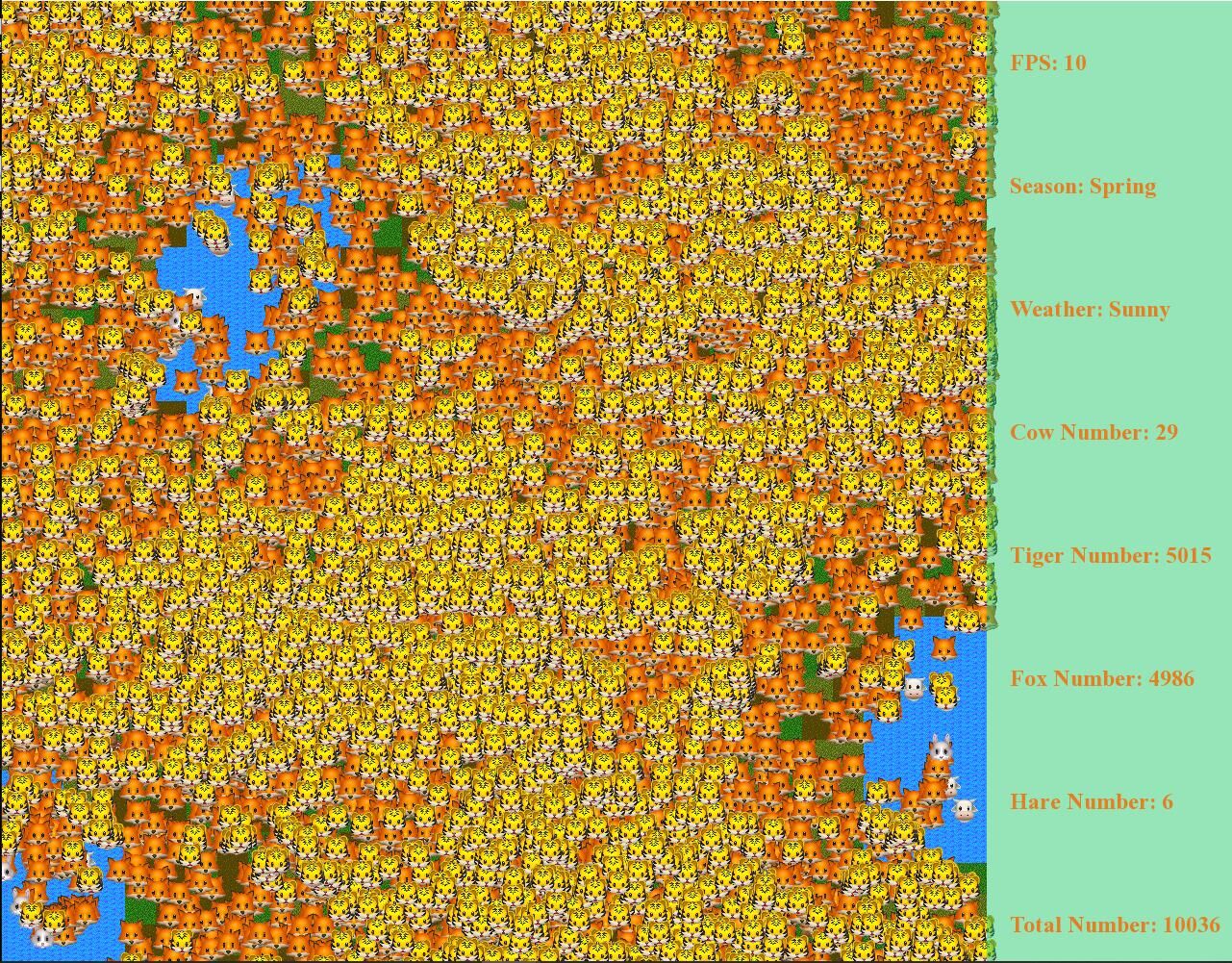
 

Figure 9 Figure 10

If predator is much more than prey, at the beginning of the process, prey will be caught easily and the number of prey will become zero immediately, as shown in Figure 11. After that, only tigers and foxes will be remained in the map, but they cannot find preys any more. Then they all starve to death, as shown in Figure 12.

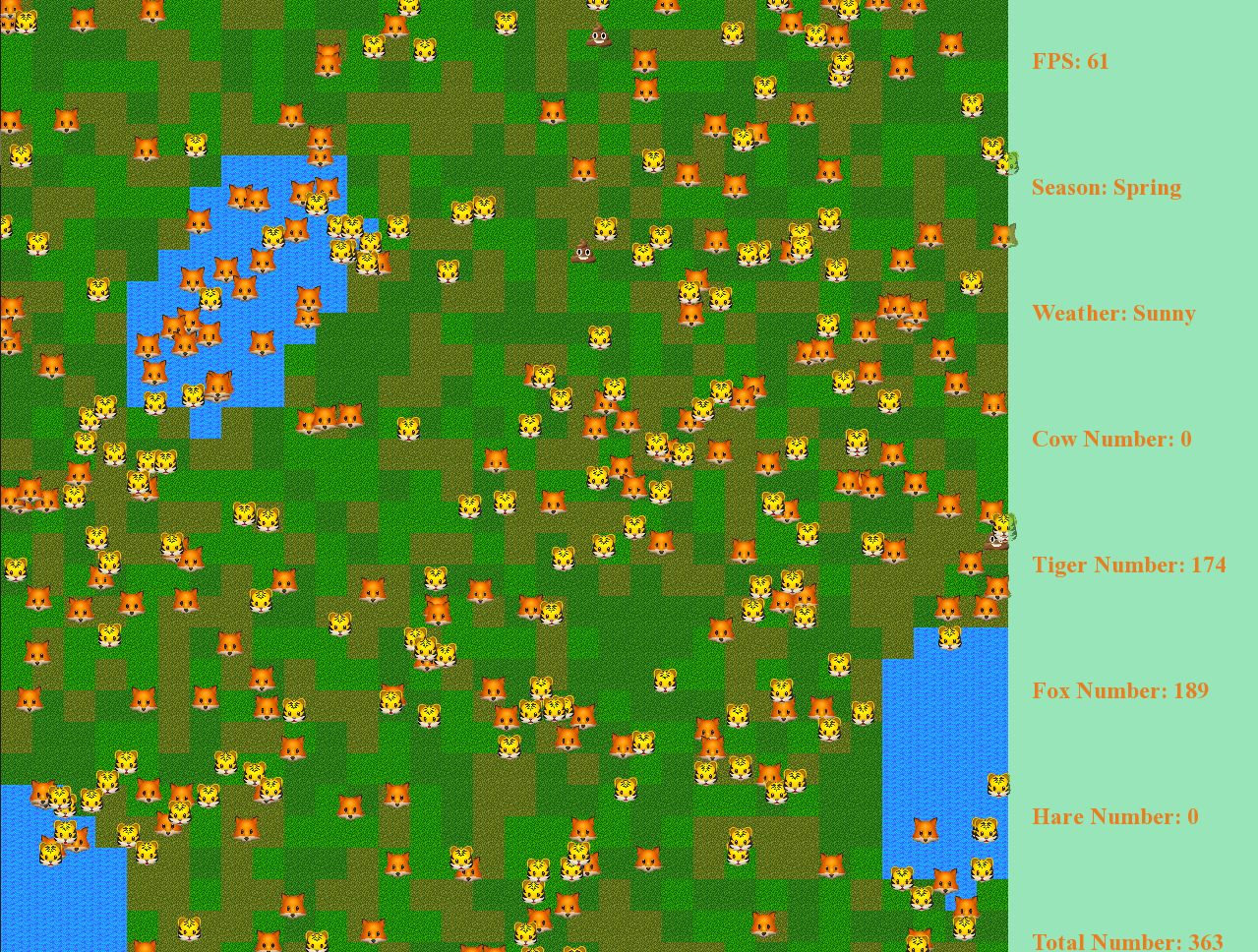
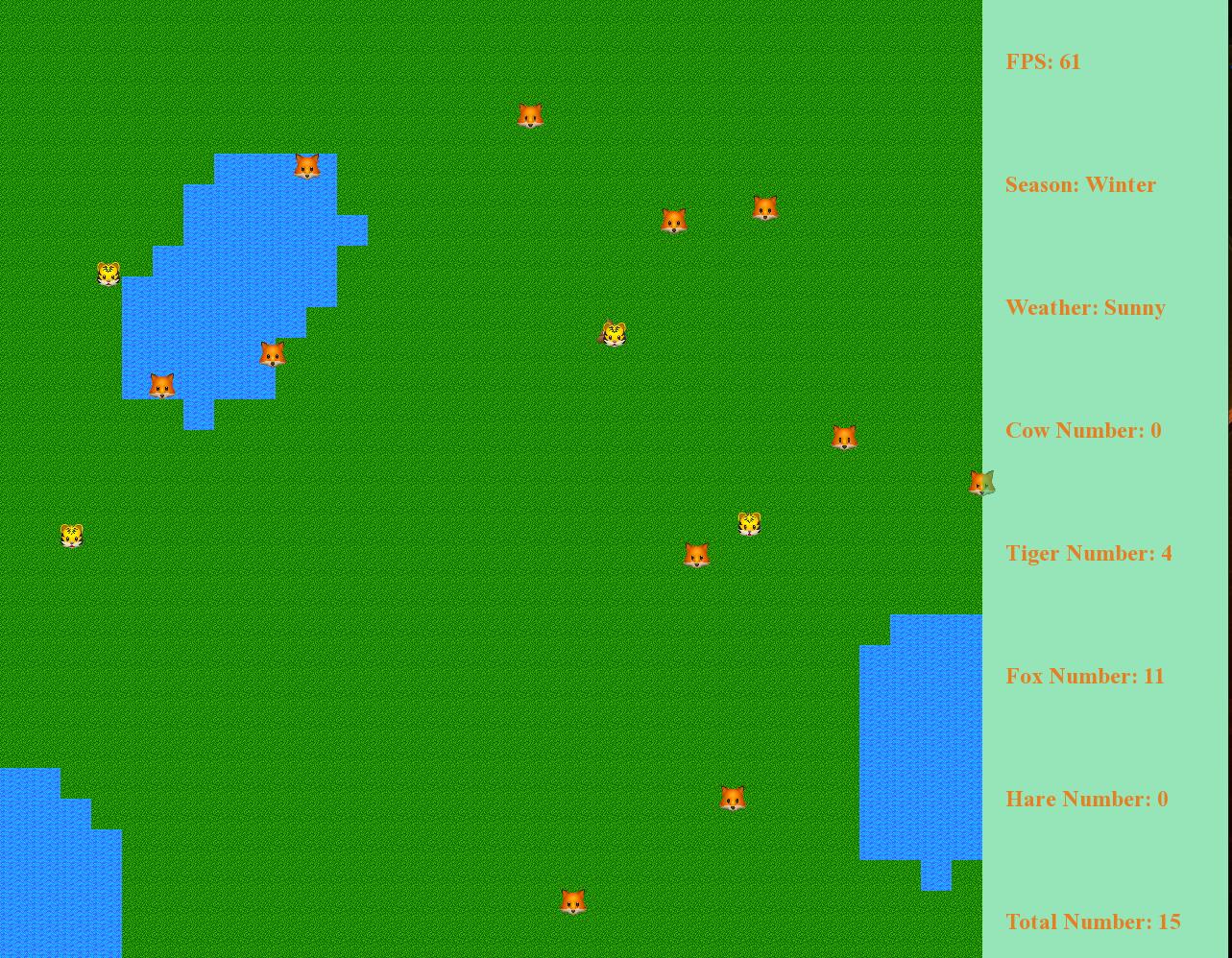
 

Figure 11 Figure 12

If the number of preys is much larger than predator’s, tigers and foxes will also eat all preys in a while, leaving only tigers and foxes on the map, as shown in Figure 13 and Figure 14. Later, without any preys, they all starve to death, as shown in Figure 15.

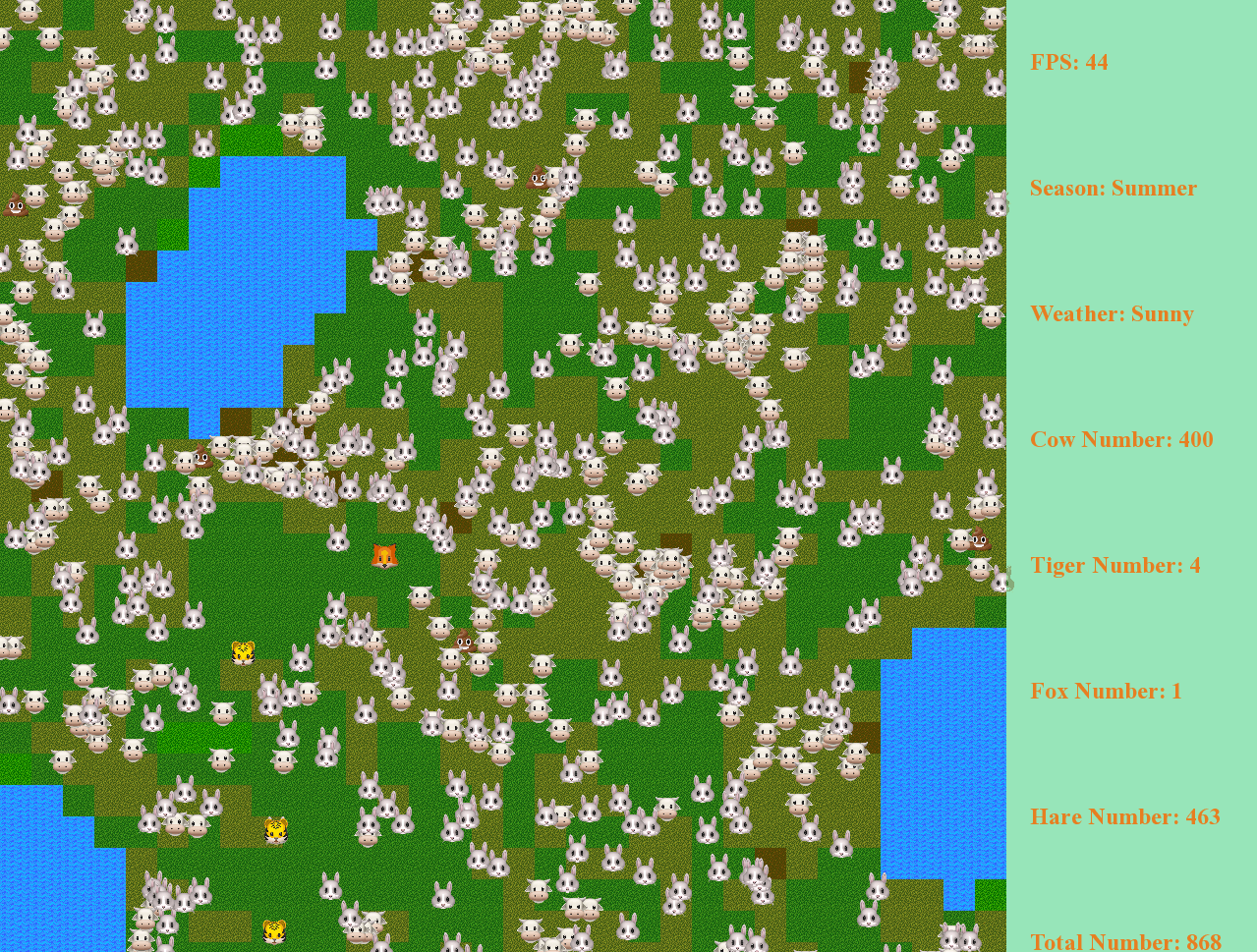
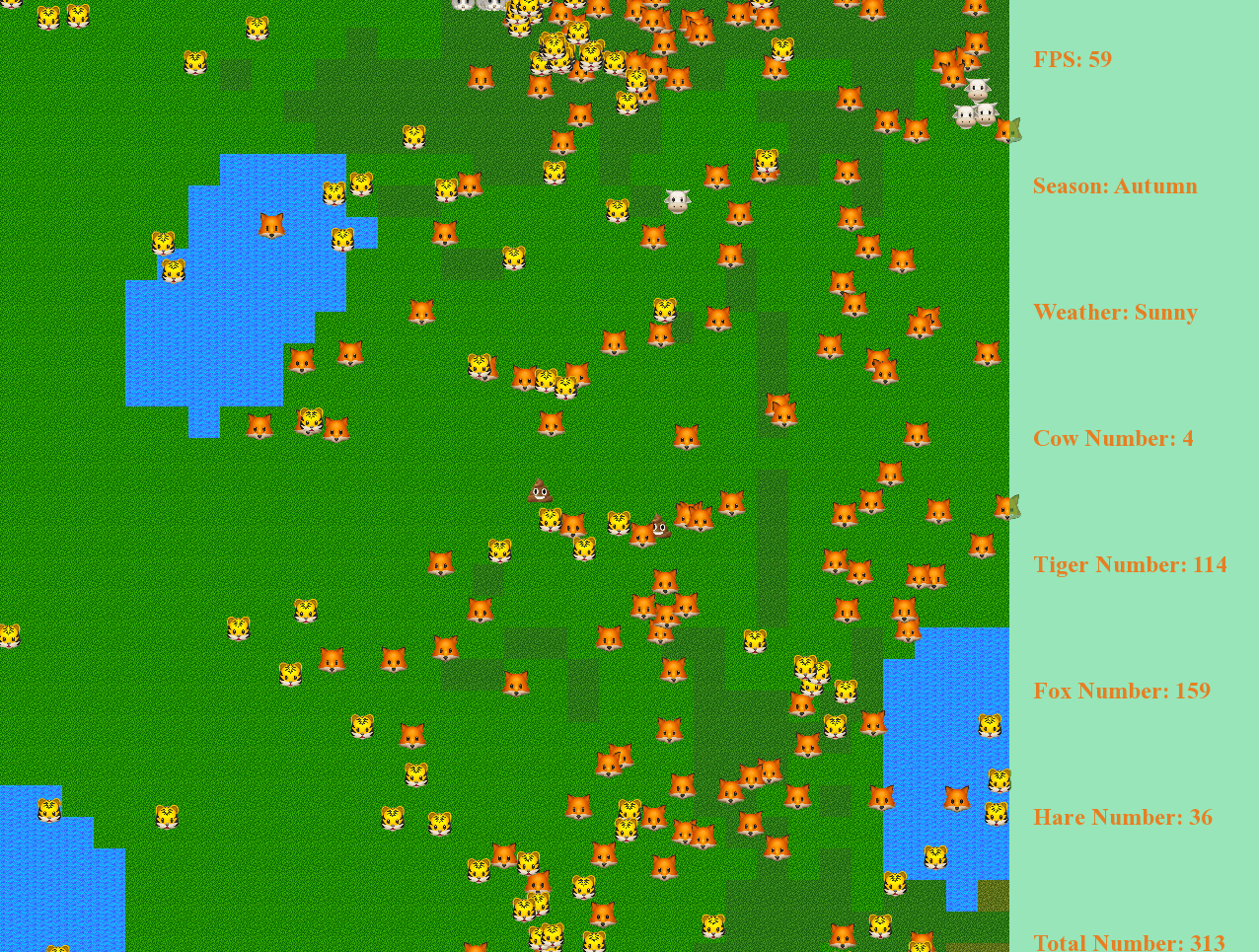
 

Figure 13 Figure 14

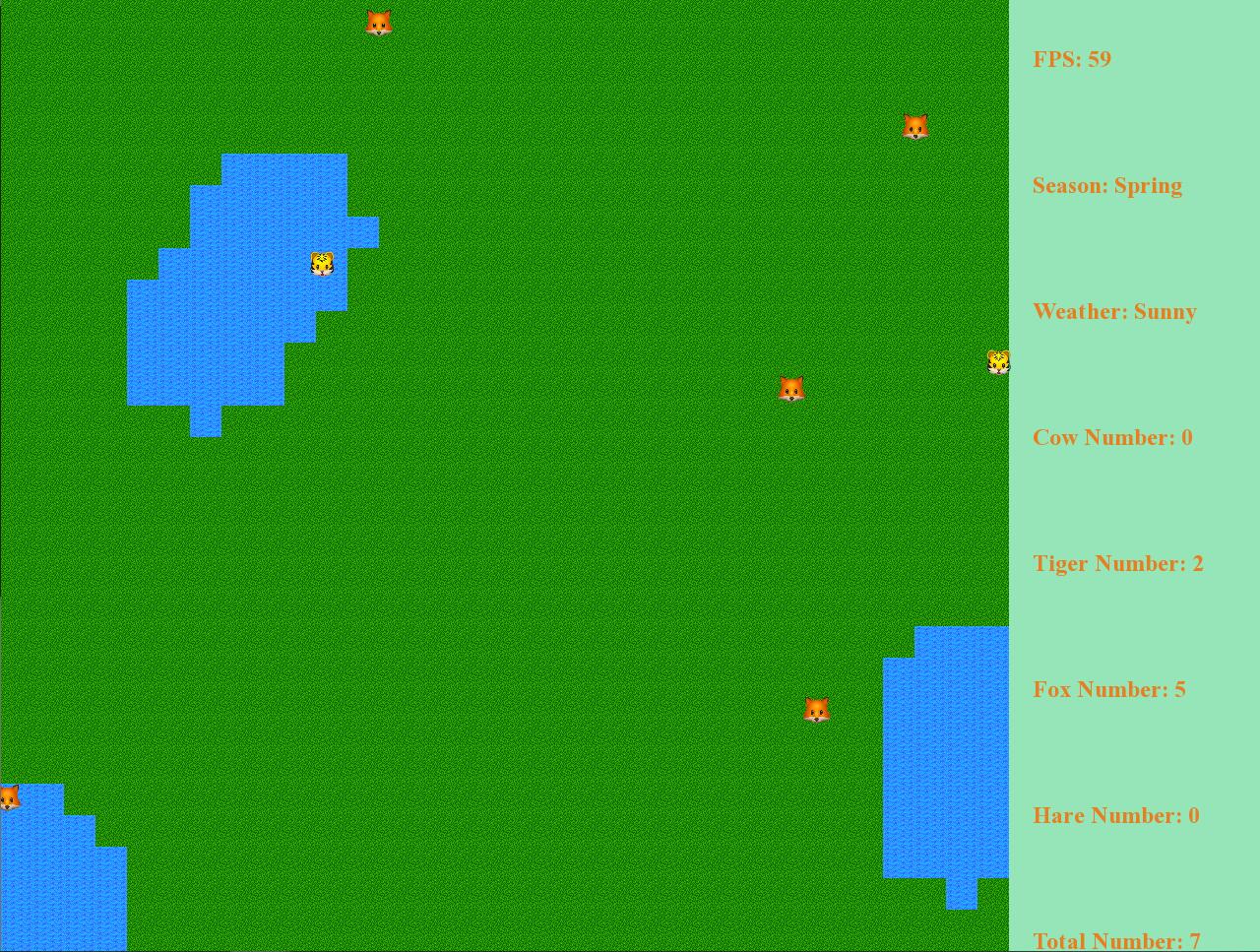


Figure 15

If the predator is zero, lack of nature enemy’s restriction, the cows and hares will grow freely and reach a balanced state finally, as shown in Figure 16. In the Figure 16, the initial number of cows and hares are 500. After a period of time, the numbers become stable. Cow’s number decreases a lot, only 25. Hare’s number is 1203, more than twice as much as before.



Figure 16

1. **Conclusion**

In this project, we use C++ to simulate a biosphere. There are totally five kinds of species, including grass, cow, hare, tiger and fox. Different species possess distinct attributes, and there are interactions among different species as well. We also consider the influence of environment to animal and the influence of excrement to environment.

For grass, the whole grassland is divided into thousands of small grass grids for individual. The changing rate of the grass can be affected by animals and environment.

In our project, we divide four kinds of animals, which are tiger, fox, hare and cow into two kinds, which are prey and predator. For prey, they will get many information in a single frame. Each prey uses a heap to obtain the most important pieces of information, and the prey will decide its behaviors due to the analysis of these information.

For predators, they can also get information in their eyesight. If the predators are hungry, they will search all the prey in their eyesight and try to trace the closest prey. If they found any prey, the predators also know whether it is possible for them to catch the animals in this frame and modify their behavior due to this important information. A predator will also share food with other predators if other predators also catch the same prey.

Within the predator and prey, there are also variations in the behavior of different animals. We restrict that the cow can only be preyed by tiger, since the larger size of cows compared to foxes. Based on the reality, we also classify tigers and cows as social animals, while hares and foxes are considered solitary animals.

We use a several link lists and a queue control the excrete behaviors and record all the animals. We also use two large 2D vector to record the information on each grass grid. This vector helps us to make the state of environment and the position of animals discrete, which greatly decrease the time complexity of the program.

At the end of the report, we want to show the highlights of out project.

1. High performance

Using to denote the total number of animals, in the worst case, the time complexity of our program is:

2. Rich species diversity

There are five kinds of species, including grass, cow, hare, tiger and fox. Different species possess distinct attributes, and there are interactions among different species as well.

3. Realistic environment

There two kinds of terrain, which are grass and water. We also add two kinds of weather, which are sunny and rainy, and four seasons into our project. Animals will have different behaviors in different environments.

4. Considering excrement

We take the effect of excrement of the animal into consideration, which makes the simulation more interesting.

1. **Appendix**

**Individual Report**

Name: 蔡xx Class: 计创 Student ID: xxxxxxxxxxxx

**Responsibility**

1. The design of the system. I participate in modeling the logic of our program, deciding which parts to implement data structurers.
2. The implementation of class. I am responsible for the code about animal and predator. Codes are included in the file whose name start with “animal”, “predator”, “fox” and “tiger”.
3. The implementation of BFS. I am responsible for using an array to enable the animals to search the world for nearest to farthest. Codes are included in the file whose name start with “distancenode”.
4. Report writing. Part A, part B, the analysis of time complexity in part C, part E of this report are written by me.

**Difficulty**

1. The reduce process of time complexity. After we put our codes together, we find out that the time complexity of the program is too high. I propose using BFS and store more information in a single map node. For the BFS, animal will search the nodes that are close to it, and steps out the searching loop as soon as it gets enough information to decide its behavior in this frame. We also store more information in a single graph node, so that the search time of an animal on a map node is independent of the number of animals on that node.
2. The determination of the behavior of predators. The predator cannot determine the moving direction of the prey, since it can only get the information of the prey in the last frame. We successfully found out a method to divide two information updating cycles, one for prey and one for predator, into four information updating cycles, two for prey and two for predator. Based on this method, a predator can get the position and direction information of any preys in this frame.
3. The method of controlling the aggregation behavior of tigers. Simply letting the tigers belong to the same population will not show the effect the aggregation obviously. Letting the tigers move to the leader of the population will let all the tigers move to the exactly same map node, and the image of these tigers will become overlapping. Finally, I solve this problem by letting tigers move to the closest tiger only if the closest tiger is at the edge of its eyesight.

**What I have learnt**

The most important things that I learn through this project is about how to change the knowledge that I learnt from the class to code. For example, the pseudocode of BFS is not only simple, but also easy to implement. However, the operations like push and pop of even a STL standard queue will cost lots of time. This taught me that there are many other ways to implement the data structure, besides the way that is taught in the course.

**Special challenges**

The biggest challenge is in the debug process. During the debug process, I can only see the strange behavior of some animals, but I cannot locate the animal, since the id of the animal is not output to the screen. I cannot simulate the flow of data just like what I do in former debug process, since the project is too complexity. Fortunately, I overcome all these problems by investing a significant amount of time.

**Individual Report**

Name: 庄xx Class: 计创 Student ID: xxxxxxxxxxxx

**Responsibility**

1. Design and program the class of prey animal. It contains the following several files: prey.h, prey.cpp, cow.h, cow.cpp, hare.h, hare.cpp.
2. Design and realize the max heap of the behaviors of prey animal.

**Difficulty**

1. How to design the direction prey animal run in when it meet more than one predator? It is easy to design the direction when there exist only one predator, but it’s more difficult to solve when there exist more that one predator. Finally, I decide start from the best case, firstly calculate every direction angle for different predator, that plus all the angles and divide it by the number of predator. This method can solve this problem in the most cases, but for few special cases, it will lead to error direction.

2. How to reduce the update time? In our initial version, the update rime for each frame is too large that the program run so slow. We find the most time program consume is in the prey update function, more precisely in the build max heap function, because it sorted large number of events and was called many times. To deal with this problem, I rewrite the max heap function. It will use min heap to implement max heap rather than direct implement heap sort to build max heap. Also, the max heap will build in the search process rather than after the search process. After rewrite the max heap function, the time consume by prey animal update information greatly reduce.

3. How to realize the gather of cow? At first, I improve the priority of the map if there exist a cow. However, it will cause all cow gather in one small map, more precisely all cow overlap together. Although there exist many cow, we will just see a cow only. So, I improve the priority of the region if there exist a cow. After that, although cow don't overlap together, the display of cow living in groups is not meet our requirements, because all cow link to other cow then make it unable to distinguish group. Finally, I select some special cows, only for them the priority of the region around can be improved. After modify like that, the effect of presentation can meet our requirements.

**What I have learned**

1. I understand the power of data structure in reducing the time complexity. Before this project, the programs I write have little limit for time complexity, so I have not give too much attention to data structure. In this project, there have high requirements for time complexity, so it exercise my mastery and application of data structure.

2. The importance of program scalability. At first, we do not consider too much for future, when we finish our initial version, we want to add new things such like add new animal class, we find we need to modify a lot of things, which make us upset.

3. Plan the project structure: Before starting coding, plan the structure and architecture of the project. This can make the code easier to manage, improve code readability and maintainability.

4. Exception handling: When writing code, it is important to consider exceptions and handle errors.

**Individual Report**

Name: 卢xx Class: 计创 Student ID: xxxxxxxxxxxx

**Responsibility**

1. Project version control (git) setup and management. I set up a git service in a cloud server to synchronize the git repository of each team member, so that we can easily cooperate on development and rollback versions if something goes wrong.
2. UI design. I am responsible for the UI design of the program, including the widgets, event listening, render area and information display.
3. Rendering. I am responsible for the rendering of objects and map sent by the logic part of our game.
4. Program control flow. I construct a state machine to implement the control flow. There are 3 states in total: Main state for displaying the input prompt and widgets; Initialization state for accepting the input from main state and pass them to the logic component to initialize objects; Game state for running the logic component and render game view.
5. Help with modeling. I participate in modeling the logic of our program.

**Difficulty**

1. In UI design and rendering, the difficulty is to select a suitable library for displaying content of our game to the front end. The library should be light-weighted, easy to combine with logic backend, and able to achieve high rendering performance. I try major libraries, read the documentation of them and finally choose SFML as our rendering engine. It is well-documented, easy to use and can handle rendering of tens of thousands of sprites with small overhead. And I also use some pre-loading technique to increase rendering speed.
2. In order to use external library mentioned above, I have to link it with our project. The difficulty is to compile a suitable version of library from source code and configure in our CMake project. By reading the tutorial and learning the usage of Makefile, I was able to compile a working library and link to our project.
3. In program control flow, the difficulty is to design a state machine. Our program consists of three states: Main state, Initialization state and Game state, which are derived from base class State. They all implement their own update() and init() functions. Then a State Manager is used to decide which state is on the stage and call its update() function. For example, in Main state, input widgets are displayed. After user clicking the start button, the callback function calls the State Manager to switch to Initialization state and after initialization, it switches to Game state. The update() function is called in all time to display corresponding content.
4. In modeling, the difficulty is to design the structure of logic component of our game according to the rendering engine. The engine uses a continuous-update basis and therefore the logic part should implement an update() function to update the objects and to be called continuously.

**What I have learnt**

Version control, create different branch for new features added and testing. The ability of reading documents and link third-party library to the project. Team-work project configuration with external library. Rendering skills. Using state machine to design the control flow of a program.

**Individual Report**

Name: 翁xx Class: 计创 Student ID: xxxxxxxxxxxx

**Responsibility**

In this program, I am mainly responsible to the logic design and implement the related functions. These function contains: The initialization of map, initialization of animal, initialization and update of animal poop list, management of the deletion and reproduce of animals, the state change of map. The most important one is update() function, which combines other function together and determines the sequence of calling function.

I also participate in the function discussion and modeling the whole logic of our project. I am also responsible to the parameter setting.

**Difficulty**

The most difficult parts is the arrangement of the different function’s sequence to be called in the update() function. Not only need I determine the function to be called written by me, I also need to decide when to call the animal’s update\_information() function. So I need to communicate with my teammate to determine the content of some animal’s functions. Finally, to reduce the time complexity of update() function, which need to be called in each frames, the function involves four major loops.

We wish to make the poop function work more logically. That means, all animals queue up to excrete waste and when some animal cannot excrete waste in it’s turn because of being chased by predators, it should be arranged to excrete when it’s not in urgent situation. I choose to implement a queue, which is constructed by a manual linked list and it can retain the position of the front animals which is in urgent situation. What’s more, I check the animal’s id when it need to excrete waste in order to settle down the conflict that caused by the death and reproduce of animals.

**What I have learnt**

In this project, I have learnt many skills to implement the function and known some data structures in C++ I had not used before, such as map. The data structure’s knowledge learnt form the lectures is understood more deeply. This project also develops my ability to consider the time and space complexity of the program, I try to use the most effective data structure to reduce the time complexity. What’s more, I also try my best to make the code more readable and extendable by learning my teammates’ code writing habits. I also learn to cooperate and communicate with my teammate in a real project, which will benefit me a lot in the future projects. I also gain the chance to know many possible bugs in our codes since we spent a lot time in debug together.

**Special challenge**

There are too many challenges I encountered in this project. One of them is to settle down the conflict of animal actions in map. For each animal, the map state should be the same during a frame. But in our program’s initial version, when we update a animal’s position in map, the following animals’ information gained from the map will be a little different. This is not reasonable in the real word. And this conflict trapped me a long time. Finally, I decided to maintain another map. And the program has two map, current map and next map. In each frame, animals get information from current map, update their position in next map. By this way all animals get the same information from the map. Before the end of each frame, next map will be assigned to current map, which complete the update of current map.

**Individual Report**

Name: 姚xx Class: 计联 Student ID: xxxxxxxxxxxx

**Responsibility**

In the project, I participate in the modeling process of this project. I am responsible for setting the parameters in this project. I am responsible for writing functions for setting member states for animal and logic classes, as well as code for updating some maps. The simulation of this ecology requires a large amount of data as a reference. I consulted a lot of information to set up a lot of data in this project. The calculation and evaluation of data, the derivation of logical relationships between organisms, and the compliance with basic natural laws to make data meaningful are all considerations in setting basic parameters.

**Difficulty**

In the map-updating functions, time complexity of this block cannot be high. I have to try how to get another variable (assuming temp\_map). Its state changes with the current map. Then the pointer of each grass box is always empty, so I use an assignment sentence at the beginning of each frame to make next\_ Map=temp\_ Map. Abstract descriptions are difficult to understand. The description of data structures is relatively abstract, which may be difficult for people accustomed to natural language to understand. To achieve map scaling, how the parameters will change when the map size is changed to 512 256 128 64 32 should be considered. I create a variable called map zoom ratio. For example, if the zoom ratio is 2, the map size, eyesight, and running speed will be proportionally reduced to half of the original. The specific parameters are also needed to be scaled or doubled. Think about it and write it as a function. Directly operate on the parameters in the main file. The map has become smaller, the speed has changed, and the lifespan of the animals has correspondingly decreased. The field of view has decreased, the speed distance has decreased, and the chasing time remains the same. The energy consumption is based on seconds, and the chasing time of the animals has not changed. Therefore, the energy consumption has not changed, so the energy parameters do not change.

**What I have learnt**

Data structure is a basic course, for people who have programming experience, combined with their own programming experience to understand its ideas; In addition, I think with the rich programming experience, the more in-depth the experience of it, the initial contact is to some ideas may be just blunt memory, with the depth of learning gradually understand a lot. After reading the title of this course design, although the specific requirements are not clear, but in summary, it can be seen that we need to be able to reflect a specific case or a thing as a program to express, data structure is a bridge, through their own design, so that the application ability can be integrated, and the problem, with a preliminary analysis, and then the ability to solve it. I feel that it will be of great help to my future study. Learning is nothing more than practice.