

Data Structure

Project Report

**Topic: Ecosphere**

|  |  |
| --- | --- |
| **Faculty** | **School of Computer Science and Engineering** |
| **Program** | **Data Structure Course Design** |
| **Name** | **杨浩斌 赵旻昆**  **张恭利 陈冠宇** |
| **Student ID** | **202030430240 202030430394**  **202030430271 202030430042** |
| **Lecturer** | **Dr. Patrick Chan** |
| **Course Code** | **045101682** |
| **Credit** | **1.0** |
| **Start/End Date** | **2021.11.30-2022.5.21** |

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

**Final Report: DS Couse Design**

**(Ecosphere)**

杨浩斌 202030430240 (group leader)

赵旻昆 202030430394

张恭利 202030430271

陈冠宇 202030430042

1. **Background**

The project we select is Ecosphere (Closed Ecosystem). In our project, we are going to simulate a miniature world. In this miniature world, we are going to create various kinds of species something like those we see in a real ecosystem. And we are going to simulate the habits and characteristics of the creatures included in our miniature world.

The ecosphere designed by us is a simple virtual ecosystem designed according to the structure and operation law of natural ecosystem. Among the living substances, we mainly introduce producers and consumers of different nutritional levels. Grass as the producer, horse, sheep and rabbit are the primary consumers, while tiger and wolf are the secondary consumers and also the highest nutritional level. At the same time, we are going to try our best to bring in inanimate substances from natural ecosystems, such as sunlight or rain, to change the living conditions of various creatures and make our ecosphere more realistic.

1. **Design Principle**
2. **Requirements and functions**

In this miniature world, there are many different species. The existence of species is affected by two factors: its preys and predators. Predators need to spend its energy to chase the prey. After predators catch the prey, predator can eat the preys and gain their energy. If they have enough energy, they will reproduce next generation after a period of time. If they have no energy or their age is larger than their max age, they will die. The animals/plants will also die if they are too old.

In the project, we are trying to use the functionality and features of data structure to simulate a real ecosystem. With the intention to correspond with our real life, we need to achieve some corresponding functions in our program, such as the design of aggregation, propagation, predation of the species, so that the consequence of our program is consistent with the characteristics of ecosystems in real world.

1. **Assumption**

**1. Tick**

In our program, we need to set an appropriate time interval tick, and in each tick, the animals will determine which strategy they should take based on their current state to maximize the benefits for themselves and their species. In our assumption, the time interval is about 500ms. With 20 ticks accumulated, the age of the entity will increase by 1.

**2. State**

Each entity needs a variable as a symbol to represent its situation in the life cycle. In our assumption, this variable is state. State consists of three parts: age, energy and tiredness. In our design, there are three trophic levels in the whole ecosystem. Creatures at different trophic levels have their maximum age. Entities at second trophic level and third trophic level have another two properties energy and tiredness. The relationship between state and these three properties is:

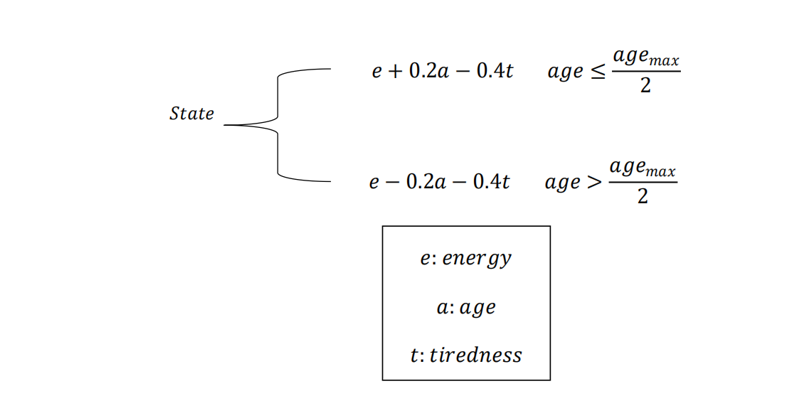


Figure 1 The velocity formula

**3. Observation**

In our assumption, every entity has its range vision. And only entities in this range will have an impact on its strategy. In each tick entity will send message to inform others to determine whether it is in the range vision of them.

**4. Aggregation**

In a real ecosystem, sometimes animals tend to take collective actions. Therefore, our program takes this into consideration. We are required to design a reasonable algorithm to analyze when and how entity will take actions in group.

**5. Predation**

Predation is the basic functions of animals we need to achieve. In our assumption, the main part is that we try to make it reasonable for an entity to prey on targets to ensure it can maintain its state while achieving sustainable development.

**6. Propagation**

In this part, we ignore the sex of animals and consider this problem from a holistic perspective. We assume that we can use some algorithms to divide entities of the same species into groups based on their location. In each group, the number of new entities is related to the original population.

**7. Movement**

In our setting, the speed of an entity remained unchanged during its movement in each tick. For each entity, the value of speed is related with the value of state, and the value of state will be consumed during movement. The detailed relationship is

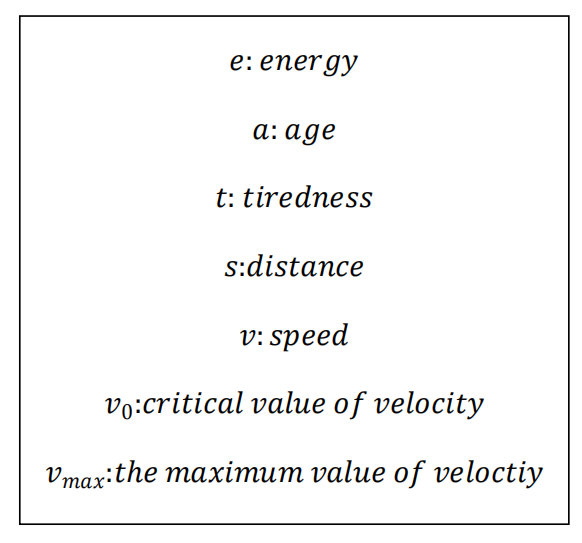


Figure 2 Some parameters of enities

1. **Model Discussion**

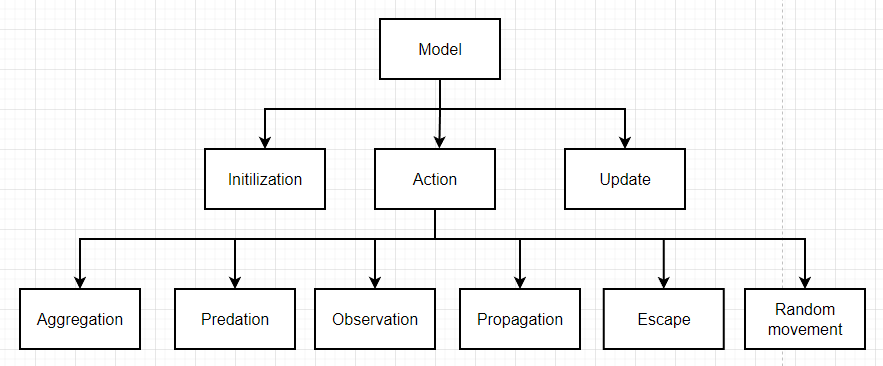


Figure 3 The model of the logical part

In our model, the logical part we mainly need to accomplish using the knowledge of algorithms and data structures can be divided into three aspects: initialization, state update, and the control of the entities’ action. We mainly describe our logical flow from these three perspectives.

* **Initialization:**

**Type 1: Generate randomly**

As the name shows, all the species’ locations are generated completely randomly.

**Type 2: Generate by group**

In this type, the plants’ locations are still generated randomly in the map. While the animals’ locations are randomly generated with the form of several groups.

Here are the main steps of this function:

**Step1:** Randomly determine the number of the groups and the number of animals in each group.

Precisely speaking, we regulated the scale of each animal group to be no less than 3 and no more than 15. That is to say, the scale of each group is a random number between 3 and 15.

**Step2:** For each group, randomly choose a coordinate in the map as the location of the 1st entity in the group. What’s more, this location also serves as the center of the current group and all the locations of other species in the group are to surround the center.

**Step3:** For each group, specify the remaining entities’ locations. Here, we use the normal distribution and let the coordinates of remaining entities obey the normal distribution.

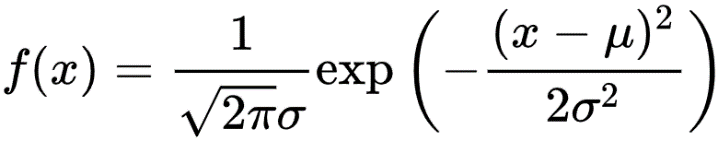


Figure 4 The formula of normal distribution

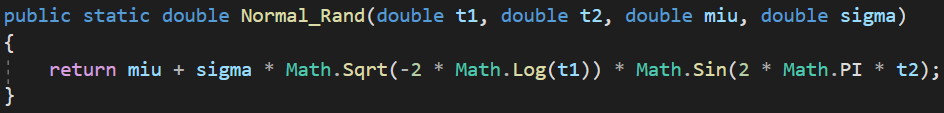


Figure 5 The code of the corresponding part

Here, the is the center’s coordinate and the is a parameter whose value is determined by the scale of the group.

Undoubtfully, using the normal distribution can let the distribution of the entities inside the groups be closer to reality.

**Other consideration**: We do not accept the case that a herbivores’ group is too close to a carnivores’ group, which is very detrimental for the survival of herbivores.

**Complexity Analysis:** Since we determine the initial positions of all the species one by one and every position is either generated randomly or determined by several parameters, the time complexity of this function is where n is the total number of all livings.

* **Update:**

In this part, we mainly need to change the weather and update the state of animals and plants in the environment after each tick. At the same time, the environment should be able to reclaim entities with low state values. And there are two functions used in this part: EnvironmentChange and UpdateState.

The logic of the EnvironmentChange if simple, we just use random numbers to randomly select different kinds of weather. Changes in weather will directly determine the state of the entities in first trophic level.

In the UpdateState function, the main task we do is to update the living state of each entity. In order to do this, we iterate through the list which stores all the current living entities, updating each entity's state and checking whether it is dead. The logic flow of the function:

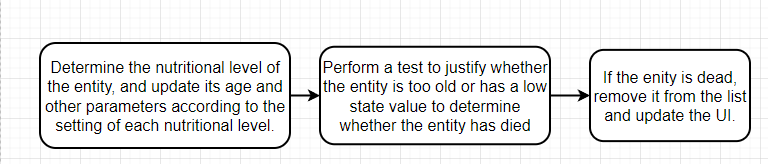


Figure 6 The logic flow of the UpdateState funtion

**Complexity Analysis:** In the EnvironmentChange function, we simply select the weather of the environment randomly, therefore the time complexity is O(1). In the UpdateState function, we need to traverse each entity in the list, thus the time complexity is O(n). Here n is the total number of living entities in the list.

* **Action:**

1. **Observation:**

In this part, we define and implement how the second trophic level animals and the third trophic level animals refresh the information of animals in their vision. If the distance between other animals and the current animal is less than the predefined maximal value of sight range, then these animals will be included in the vision of the current animal, which is straightforward and easy to understand.

**Complexity Analysis:** In this function Detect(), we scan the array containing all of the animals twice, which is double nested. Therefore, the time complexity is O().

1. **Aggregation:**

The intention of defining this action is to let the animals to propagate better and have a sustainable development.

This function is probably be called when the proportion of the current species is extremely less than that of the original case.

The only difficulty of this function is determining the moving direction of the animal. And determining the moving direction is equivalent to determining the coordinates of the moving target point. Here, we simplified the problem and let the average value of the coordinates of all its companions in it sight and itself be the center of the cluster. And then, we randomly choose a point in the circle whose center is the mentioned cluster’s center and radius relatively less than the radius of sight. In this case, we control the animals to form several clusters.

**Complexity Analysis:** Since the aggregation function requires traversing the companions in the animal’s sight in order to specify the coordinate of the cluster’s center point. So, its time complexity is where n is the number of the animal’s companions in its sight.

1. **Predation:**

This function will be called only when the animal is hungry. That is to say, the animal will go to predation only when it is hungry.

The logic of this function is as following:

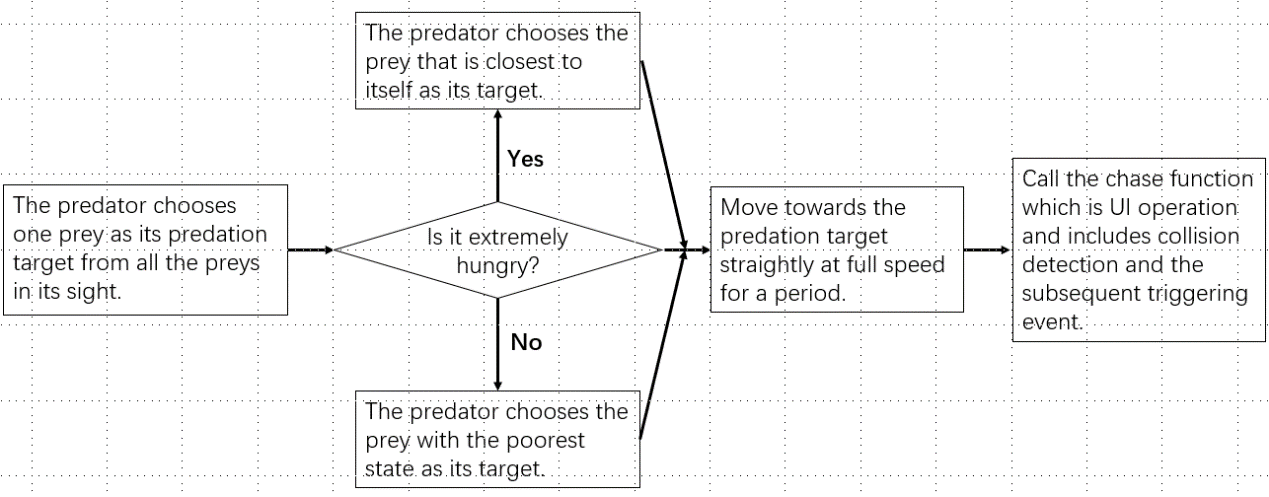


Figure 7 The logic flow of the action decision

Explanation: When the animal is extremely hungry, it requires food urgently and has no other consideration. So, it chooses the prey that is closest to itself directly. Nevertheless, when the animal is not too hungry, it is more strategic and chooses the prey with the poorest state as its target. And obviously, it is beneficial for both the predators and preys to have a sustainable development, which is good for the whole ecological environment.

**Complexity Analysis:** For choosing one prey as its predation target, transverse the preys in its sight is required. So, the time complexity is where n is the number of its preys in its sight. While the time complexity of the remaining part of predation function is . Because controlling the predator to move is just for one entity and the time complexity of the called chase function is also .

1. **Stay:**

Only when the animal is tired and there is no natural enemy in its sight will this function be called and the animal chooses to stay in situ with the aim of refection.

And this function is simply deducing its tiredness.

**Complexity Analysis:**  obviously.

1. **Propagation:**

**The function execution flowchart of the propagation mechanism:**

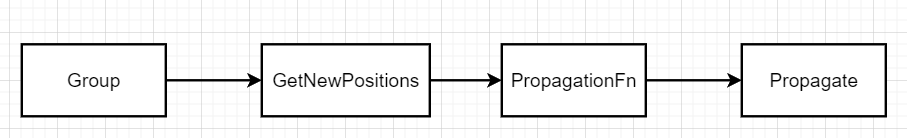


Figure 8 The logic flow of the propagation mechanism

**4.1) Group:** The main purpose of this function is to divide the entities of the same species into different groups according to their locations. This function mainly involves the application of two data structures, Min Heap and General Tree. The logic of the function can be described like this:

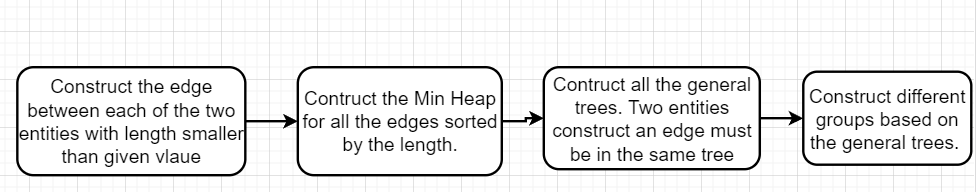


Figure 9 The logic flow of the Group function

**4.2) GetNewPositions:** The main purpose of this function is to determine the number and position of newly generated entities for different groups. For different groups, the position of the newly generated entity is the midpoint of the entities which construct the edge, and the probability of the group generating new entities is negatively correlated with the number of entities in the group. The logic of the function is like this:

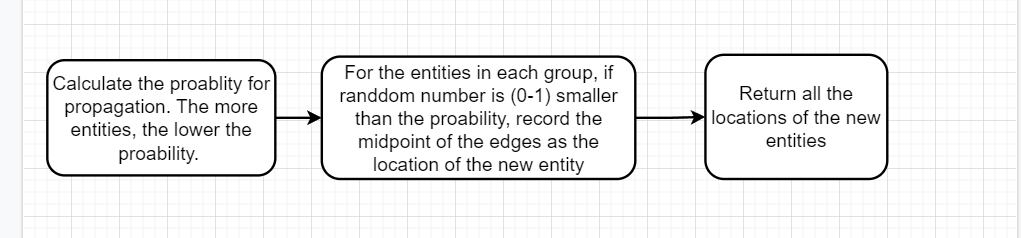


Figure 10 The logic flow of the GetNewPositions function

**4.3) PropagationFn:** The main purpose of this function is to generate the corresponding entities based on the location information returned by the function GetNewPosiotions. The logic of the function:

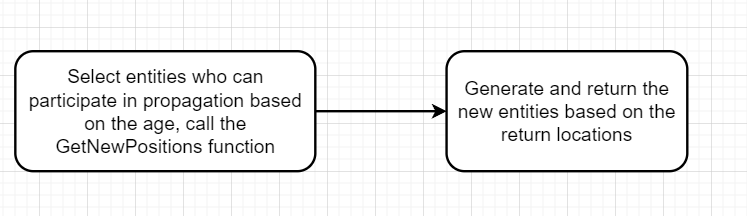


Figure 11 The logic flow of the PropagationFn function

**4.4) Propagate:** This function simply calls the function PropagationFn and displays the newly generated entities in the UI.

**Complexity Analysis:** Suppose the total number of entities is n. For the propagation mechanism, the latter three functions mainly iterate and process each entity in the group, so the time complexity of the three functions is about . For the first function Group, the time complexity is higher. First, we need to double traverse all the entities to find all the edges. The time complexity of this process is . Then we put these edges into the Min Heap for processing, and the time complexity of this process is in the worst case. Finally, we need to process the General Tree, which we need to facilitate each entity, and the time complexity is . In general, the time complexity of the implementation of propagation is in worst case.

1. **Escape:**

**The realization of the escape function for the entities in second trophic level:**

In this function we need to control the escape mechanism. In our idea, we mainly deal with all the entities who want to prey on it within the entity's field of vision, and determine the predator most likely to prey on the current entity based on the comprehensive consideration of the distance and the state of the predator. Then the entity will choose to move away from this predator. One problem we need to consider is that our maps have boundaries, and therefore need to keep entities from moving beyond those boundaries. In order to solve this problem, we don't let entity blindly moving away from predators, and need to determine whether the movement will lead to crossing the boundaries. If so, the entity should choose an angle that is close to the direction but does not cause it to cross the border. The logic of the function:

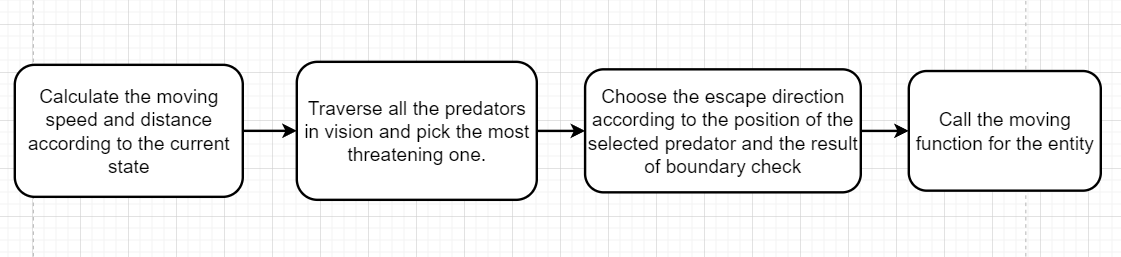


Figure 12 The logic flow of the escape mechanism

**Complexity Analysis:** The main part of this process is traversing the predators in the vision, and the time complexity of this process is . In addition, the time complexity of the other parts is . Therefore, the total time complexity is . Here m is the number of predators in the vision.

1. **Random movement:**

In this part, we define how the second trophic level animals and the third trophic level animals run aimlessly when they have no a deterministic action choice to hunt, rest or escape from hunter. The running velocity is a random value in a range from a predefined minimum to a predefined maximum, which is determined by the current state of the current animal. And the distance of movement is randomly selected from a range which is determined by the minimal velocity and the maximal velocity. Besides, in order to avoid a situation where the current animal runs back and forth without rationality, we add one constraint to the value of the current angle. Namely, we take the previous running angle of the current animal into consideration so as to make the movement of the animal more reasonable. After all the constraints above taken into account, we can obtain the ultimate movement which is based on the horizontal movement and vertical movement. In this way, we implement the function of random movement when the current animal has no a deterministic action choice.

**Complexity Analysis:** In this function, we mainly apply random mechanism and no loops are included. Then the time complexity is

**Class Introduction:**

Table 1 class **Environment**

|  |  |  |
| --- | --- | --- |
| **Member** | **Type** | **Introduction** |
| Weather | enumeration | To store all weather types (sunny, rainy, windy, snowy) |

Table 2 class **FirstNutritionalLevel**

|  |  |  |
| --- | --- | --- |
| **Member** | **Type / Return Type** | **Introduction** |
| Count | static int | The total number of plants in the map currently |
| MAX\_AGE | const int | The lifetime of the plants |
| THRES\_ILLUMINATION | const int | The least illumination storage for propagation |
| THRES\_MOISTURE | const int | The least moisture storage for propagation |
| ENERGY\_BRINGTO\_STL | const double | The energy provided to the herbivores when the plant is eaten |
| flag\_OKToBreed | bool | To indicate whether the plant has fertility ability |
| currentWeather\_Flag | int | The current weather |
| Age | int | The plant’s current age |
| Illumination | int | The plant’s current illumination storage |
| Moisture | int | The plant’s current moisture storage |
| StateUpdate() | void | To update the plant’s relevant attributes according to the weather |

Table 3 class **FNLHelper (Agent of FirstNutritionalLevel entity)**

|  |  |  |
| --- | --- | --- |
| **Member** | **Type / Return Type** | **Introduction** |
| entity | FirstNutritionalLevel | The entity of this agent |
| shape | Shape | The animation shape of the entity |
| location | Location | The current location of the entity |
| ShowInformation | void | The trace and visualization of the properties and states of the entity |

Table 4 class **SecondTrophicLevel**

|  |  |  |
| --- | --- | --- |
| **Member** | **Type / Return Type** | **Introduction** |
| Count | static int | The total number of herbivores in the map currently |
| MAX\_AGE | const int | The lifetime of the herbivores |
| MAX\_ENERGY | const double | The upper limit of herbivores’ energy |
| MAX\_TIREDNESS | const double | The upper limit of herbivores’ tiredness |
| THRES\_ENERGY | const double | The boundary energy of whether it is hungry or not |
| THRES\_TIREDNESS | const double | The boundary tiredness of whether it is tired or not |
| ENERGY\_BRINGTO\_TTL | const double | The energy provided to the carnivores when the herbivore is eaten |
| Age | int | The herbivore’s current age |
| Energy | double | The herbivore’s current energy |
| Tiredness | double | The herbivore’s current tiredness |
| State() | void | To calculate the herbivore’s current state based on its age, energy and tiredness |
| ObjectInSight | List | To store all the plants and carnivores in its sight |

Table 5 class **STLHelper (Agent of SecondTrophicLevel entity)**

|  |  |  |
| --- | --- | --- |
| **Member** | **Type / Return Type** | **Introduction** |
| entity | SecondTrophicLevel | The entity of this agent |
| current\_action | Action | The current action of the entity |
| shape | Shape | The animation shape of the entity |
| location | Location | The current location of the entity |
| ShowInformation | void | The trace and visualization of the properties and states of the entity |
| angle | int | The angle of movement of the entity |
| flag | bool | Recording whether the entity captures its food |
| offset | Location | The offset of the current location of the entity |
| originLocation | Location | The original location of the entity |
| storyboard | Storyboard | An auxiliary component of animation of the entity |
| doubleAnimationX | DoubleAnimation | The horizontal coordinate of animation of the entity |
| doubleAnimationY | DoubleAnimation | The vertical coordinate of animation of the entity |
| STLHelper() | NONE | The default constructor of this agent |

Table 6 class **ThirdTrophicLevel**

|  |  |  |
| --- | --- | --- |
| **Member** | **Type / Return Type** | **Introduction** |
| Count | static int | The total number of carnivores in the map currently |
| MAX\_AGE | const int | The lifetime of the carnivores |
| MAX\_ENERGY | const double | The upper limit of carnivores’ energy |
| MAX\_TIREDNESS | const double | The upper limit of carnivores’ tiredness |
| THRES\_ENERGY | const double | The boundary energy of whether it is hungry or not |
| THRES\_TIREDNESS | const double | The boundary tiredness of whether it is tired or not |
| Age | int | The carnivore’s current age |
| Energy | double | The carnivore’s current energy |
| Tiredness | double | The carnivore’s current tiredness |
| State() | void | To calculate the carnivore’s current state based on its age, energy and tiredness |
| ObjectInSight | List | To store all the herbivores in its sight |

Table 7 class **TTLHelper (Agent of ThirdTrophicLevel entity)**

|  |  |  |
| --- | --- | --- |
| **Member** | **Type / Return Type** | **Introduction** |
| entity | ThirdTrophicLevel | The entity of this agent |
| current\_action | Action | The current action of the entity |
| shape | Shape | The animation shape of the entity |
| location | Location | The current location of the entity |
| ShowInformation | void | The trace and visualization of the properties and states of the entity |
| angle | int | The angle of movement of the entity |
| flag | bool | Recording whether the entity captures its food |
| offset | Location | The offset of the current location of the entity |
| originLocation | Location | The original location of the entity |
| storyboard | Storyboard | An auxiliary component of animation of the entity |
| doubleAnimationX | DoubleAnimation | The horizontal coordinate of animation of the entity |
| doubleAnimationY | DoubleAnimation | The vertical coordinate of animation of the entity |
| TTLHelper() | NONE | The default constructor of this agent |

**Relation between Data Structure:**

Table 8 List data structure

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Scope** | **Type of Elements** | **Introduction** |
| AllAnimal | Global | <object> | To store all the livings’ entities linearly |
| FNLStatistics  STLStatistics  TTLStatistics | Global | <int> | To store the number of plants / herbivores / carnivores at each tick (required in the statistics report) |
| all\_Loc | Local | <LocationAndChoice> | To store all the livings’ initial locations when they are randomly generated |
| ObjectInSight | Local | <object> | To store all the relevant livings’ entities in the animal’s sight |
| locationsByType | Local | List<Location> | To store the locations of newly generated entities in different nutrient levels. |
| visited | Local | <bool> | To record if an entity has been visited to avoid repeated access when performing the propagation mechanism. |
| visitedIndexByOrder | Local | <int> | To store the order in which entities are visited. The entities at the front of the list are closer to each other and the probability of propagation is higher. |

Table 9 Dictionary data structure

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Scope** | **Type of Elements** | **Introduction** |
| result | Local | <int, List<Location>> | To store the locations of entities in different groups of the same species. |

Table 10 Priority queue data structure

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Scope** | **Type of Elements** | **Introduction** |
| priorityQueue | Local | <Edge> | The min heap used in our data structure is encapsulated. The Push function will put the new element into the heap, and use the Shift Up function to rebuild the min heap according to the sorting rules. The Pop function will get the first element in the heap and delete it. The purpose of this data structure is o store all the edges constructed by two entities with the distance between them smaller to the given radius. The edge with shorter distance will have higher priority. |

Table 11 General tree data structure

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Scope** | **Type of Elements** | **Introduction** |
| tree | Local |  | This data structure we have learned in our class. It is mainly used to generate a tree structure with more than 2 child nodes and encapsulated by array. Each position in the array stores the root node of the tree to which the element belongs to. The Find function returns the root of the tree to which the element belongs to. The Differ function determines whether two elements belong to the same tree. The Union funtion adds two elements to the same tree. The purpose of the data structure is to justify whether two entities are in the same group (If so, these two entities will be attached to the same tree. ) |

Table 12 Struct data structure

|  |  |  |
| --- | --- | --- |
| **Name** | **Scope** | **Members** |
| Edge | Global | (int) from  (int) to  (double) distance |
| LocationAndChoice | Global | (Location) Location  (int) choice |
| Location | Global | (double) Left  (double) Top |
| Motion | Global | (double) X  (double) Y  (double) Velocity |

1. **Demonstration**
2. **Install dependency library**

In our project folder, we provide the dependency library file that should be installed before run the main program. This program is called windowsdesktop-runtime-6.0.2-win-x86.exe, which will provide runtime of C# and windows desktop.

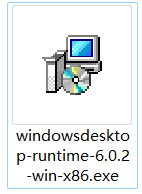


Figure 13 The exe file

1. **Running the program**

After install the dependency, we can double click the main program called Ecosystem.exe. This is the first screen of this program:

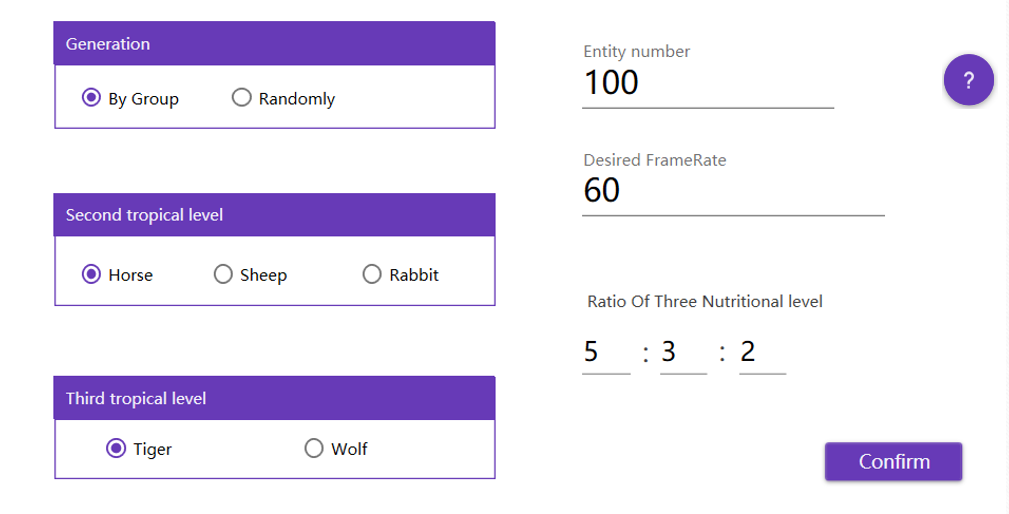


Figure 14 The first screen of this program

And we can observe that there is a question mark icon in the above figure. When the user clicks on this icon, a user help page will pop up. This page will prompt users how to set various parameters of the ecosystem and the impact of these parameters on the ecosystem to help users to customize the initial state of the ecosystem.

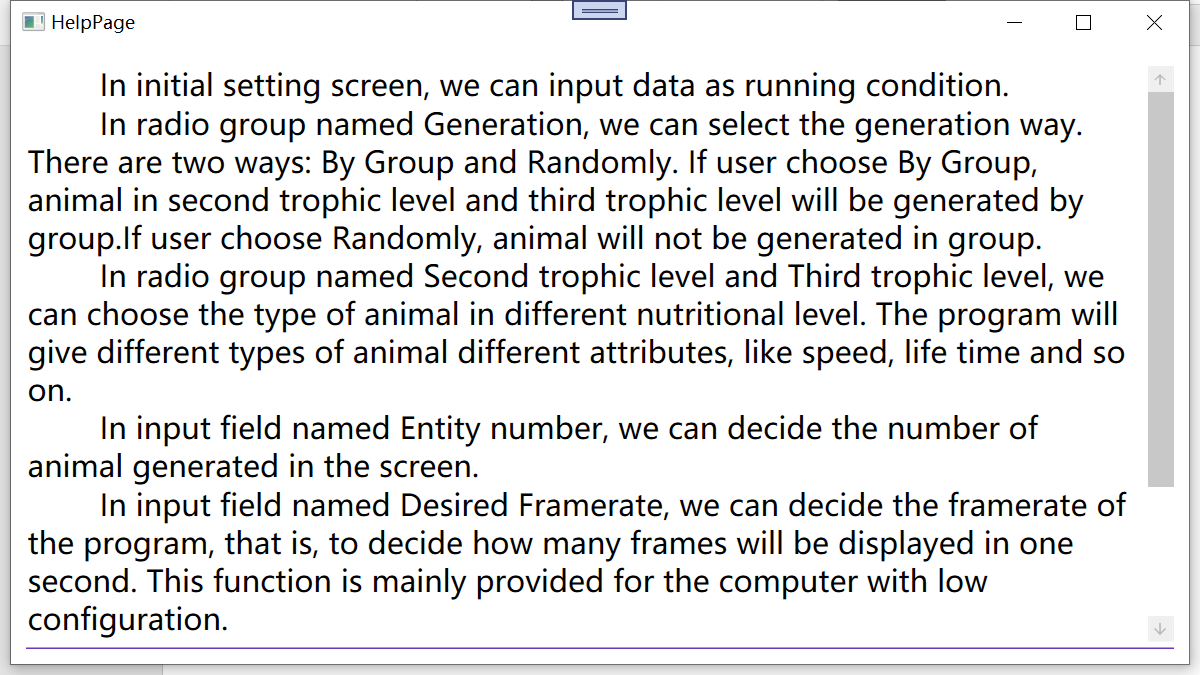


Figure 15 The help page

In the first screen, we can input the data as running condition:

In radio group named Generation, we can select the generation way. There are two ways: By Group and Randomly. If user choose By Group, animal in second trophic level and third trophic level will be generated by group. In addition, in order to show the weather in the current ecosystem and help users better understand changes in the environment, we add a dynamic weather icon to update the weather in the ecosystem in real time so as enrich the UI, just as the following graph shows:

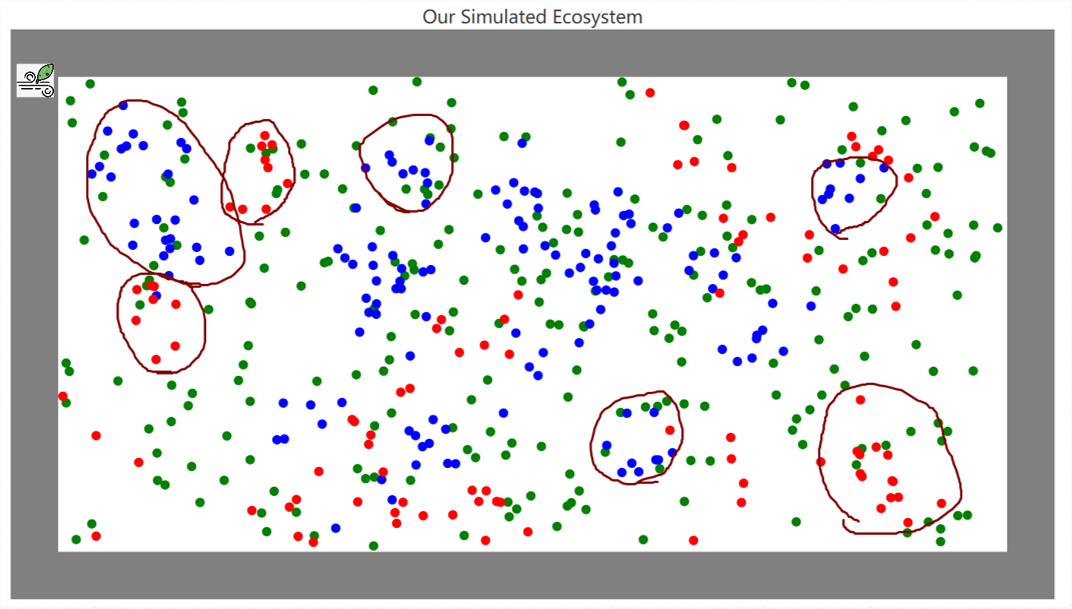


Figure 16 The screen of the initial interface by group

From this graph we see that there are some evident groups in the ecosystem. Considering that animal in first nutritional level is always in uniform distribution, we don’t make them in groups.

If user choose Randomly, animal will not be generated in group, as the following graph shows:

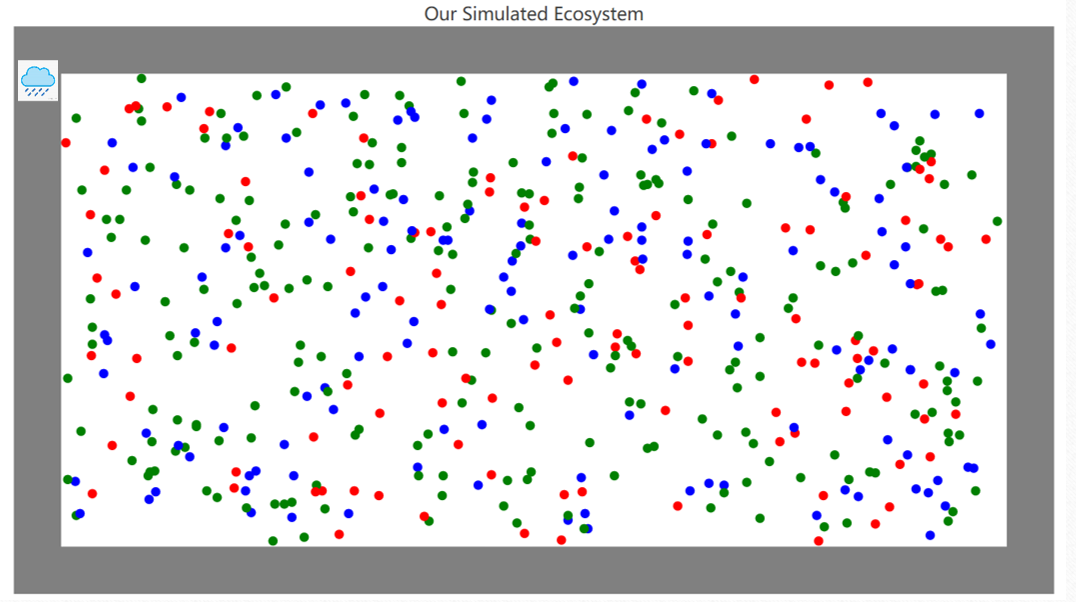


Figure 17 The screen of the initial interface by random

In radio group named Second trophic level and Third trophic level, we can choose the type of animal in different nutritional level. The program will give different types of animal different attributes, like speed, life time and so on.

In input field named Entity number, we can decide the number of animals generated in the screen.

In input field named Desired Framerate, we can decide the framerate of the program, that is, to decide how many frames will be displayed in one second. This function is mainly provided for the computer with low configuration.

In input field named Ratio of Three Nutritional Level, we can decide the ratio of three nutritional level in the screen. For example, if the number of animals is 100, the ratio is 5:3:2, then numbers of three nutritional level are 50, 30 and 20.

After we complete the settings, we can click the confirm button, and then we can see two windows of this program:

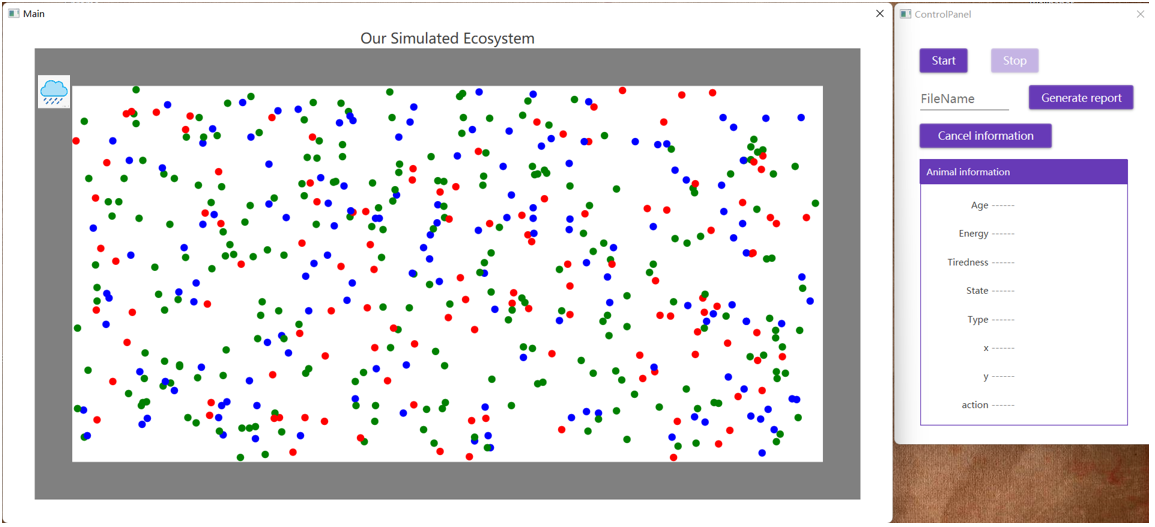


Figure 18 The main program and control panel

The left one is the main program, which is used to display how animals in the ecosystem evolve. The right one is the control panel, in which we can control the process of the program.

In the control panel, we can start and stop the process by click the corresponding button. After some time, we can generate the statistic report for the whole process. Meanwhile, we can see the information of any entity in the main screen by click any circle in the main program.

When we click start button, the process starts. When we click one entity in the main screen, we can see that this entity is wrap by a circle, and its information is display in the control panel:

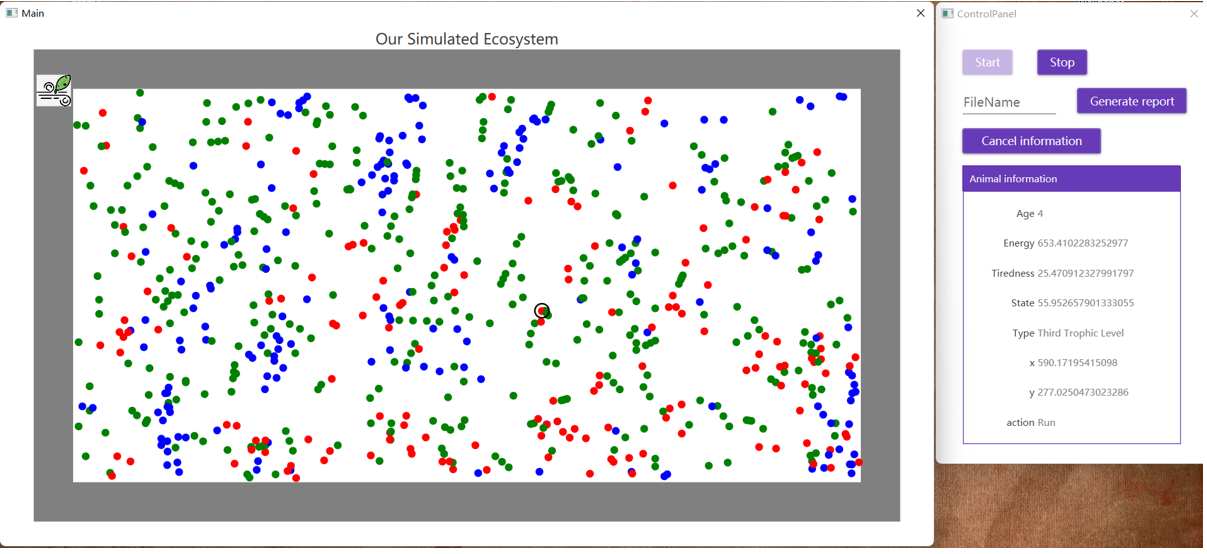


Figure 19 The main program and control panel

Age is the age of this entity, energy represented the remained energy of this entity, and energy can only be obtained by eating prey. Tiredness represents the tiredness of this entity. If this animal run too fast, it will become tired. Type represents its type among first nutritional level, second trophic level and third trophic level. x and y represent the x and y position of this entity. Action represents the action this entity does now.

After running for some time, we show the main screen in the following graph:

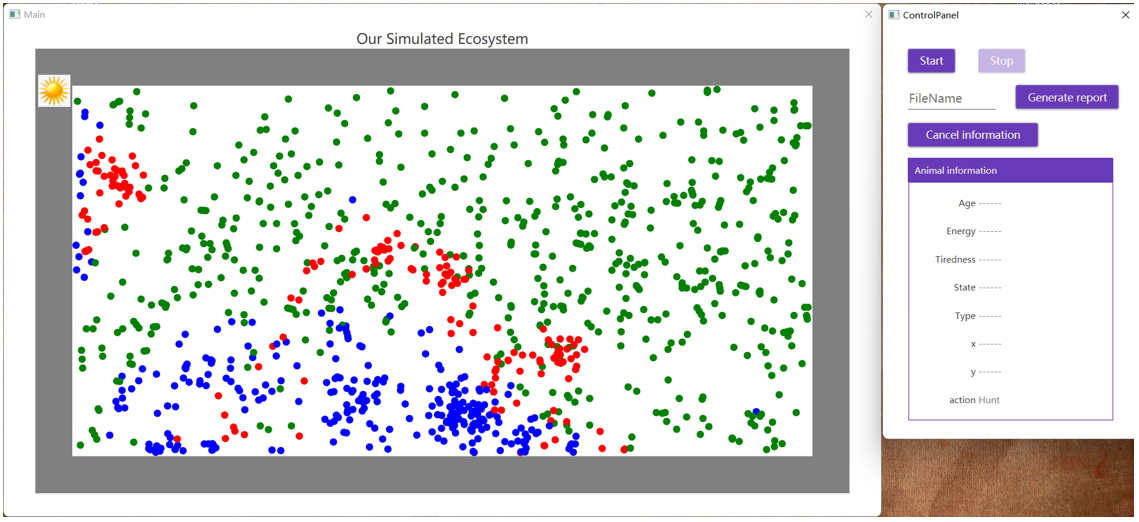


Figure 20 The main program and control panel

in this graph we can see many populations.

Now we generate the statistic report, and user’s default browser will be opened automatically.

Report is in the root directory:



Figure 21 The html file

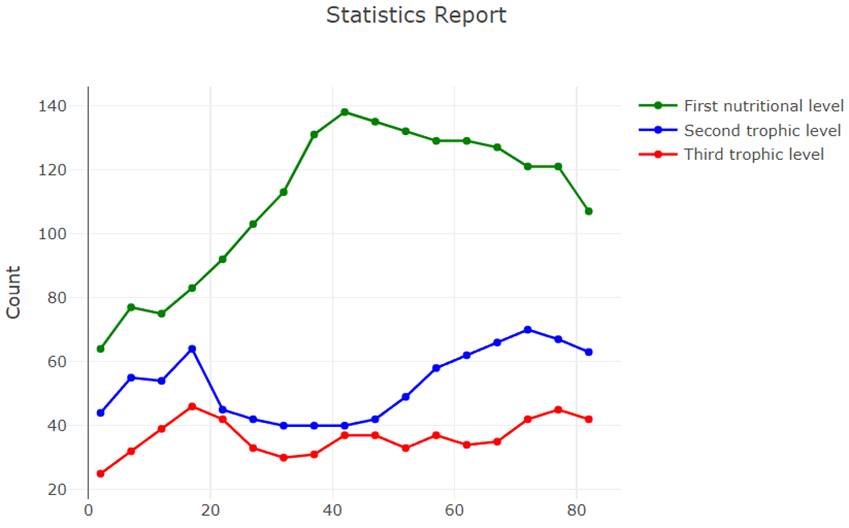


Figure 22 The graph of the curves

In the statistic report, x-axis is ticks passed by, y-axis is the numbers of animals in three nutritional level.

1. **Conclusion**

**1. Function:**

Our project has accomplished all the functions required by the course. Including the establishment of the model of the three trophic levels, the realization of predation mechanism and so on. We also complete the GUI rendering of the ecosystem. In addition, there are many other functions that we supplement to the project to improve the ecosystem that we build. For example, in addition to the predation mechanism, we also realize the propagation and aggregation mechanism, and the influence of weather on the ecosystem is considered, which makes our system more consistent to the actual ecosystem. We also improved the GUI rendering and added animations to reflect the behavior of the entities, which will help users better understand the ecosystem.

**2. Program performance:**

In the process of designing the program, we use some algorithms to reduce the complexity of the program as much as possible, which makes our program have great performance. When the number of entities in the ecosystem is below 1500, the program still runs smoothly. There is some lag when the number of entities exceeds 3000, but the animation is still fairly smooth. The maximum number of the enitities is around 10,000, at which point the program becomes significantly stuttered.

**3. Rationality:**

The ecosystem model established by us is relatively stable, and generally there is no phenomenon of ecosystem collapse caused by too few or too many entities at a certain trophic level. But the ecosystem will collapse when users enter improper initial values. For example, if the number of tigers entered by user is too small, it will lead to the extinction of tigers, which is also consistent with the actual situation Therefore, the ecosystem model established by our project is reasonable.

**4.Features:**

We add some of our special features to the project. For example, we supplement the mechanism of propagation as mentioned above. In addition, we design some features in the UI. For example, we add a help page to make it more clear to users to know how to operate our application. There is also a real-time display of the state of entities and a real-time curve of the number of entities to help users better understand the development of the ecosystem. When the program is running slowly, users can also adjust the frame rate to make the program smoother.

**5. Acquisition:**

Through this project, we have a better understanding of what we have learned in the data structure course. We know how to use data structures to help us efficiently design programs and algorithms to solve problems. At the same time, we also learned how to work better as a team, and how to design a project and make arrangements for its implementation properly.

1. **Appendix——Individual Report**

**1. 杨浩斌**

**Individual Report**

杨浩斌 202030430240

**My responsible part**

1. Construct the general idea
2. Create motion for each animal
3. Design user interface.
4. Provide the function of data analysis

**Difficulties**

I have encountered many difficulties. The first one is that how I can describe the motion of animals. The second one is how to make the program fast. The third one is how to control the process.

1. How to describe the motion of animals

In reality, the motion of animal is complex and is hard to describe clearly. For simplification, we just divide the continuous motion into concrete parts. In our design, each part has a time interval called tick. In each tick, the motion just contains two statuses: the initial position and the destination, and the motion is that the entity will run from the initial position to the destination straightly. After on tick, the entity will consider next destination and go to it.

1. How to make the program fast

The first thing that slow program down is the UI operation.

In our program, each tick the animal will do these two things: detection and action. In detection, each entity will detect the animal within its vision range, so it is inevitable to compute the distance from one entity to another. In the beginning, we call the function of each shape to compute the distance. Using this method, we found that it will cost lots of time when the number of animals is very large because it is a UI operation. Afterward, we use a variable to save the original position of each entity and a variable to record the offset of each entity relative to the original position. When computing the distance of two entity, we just need to use these variables to compute. This method actually can’t accurately compute the real distance of two entity, but we don’t care because the accuracy of whether one entity is in another is not so important.

The second thing that slow program down is the computation time of each function.

In reality, each animal thinks and acts independently. But in the program, if it run in single thread, each entity will act in sequence. When the number of animals is large, this side effect will be more evidently. To avoid it, we provide each entity with a independent running time so that they can run in parallel.

The other things include some time-consuming operations that may block the main thread and UI thread. We need to arrange them to run asynchronously so that they can’t affect the running of each existing thread.

1. How to control the process

We need to control the start, the stop of the process. Besides, we need to observe the running of the process so that we can know the status of it. To implement them, we design a control panel to control and monitor the process.

**What I have learnt**

Through this project, I have learnt how to run the program in parallel and how to design a good user interface.

The programming language we use is C#, which can do the asynchronous task very easily and elegantly by using keyword such as async, await and so on. Besides, we use an interface design language called material design to design our user interface.

Furthermore, as the leader, I learn how to arrange task for every member in our group to complete the whole task.

**Some special challenges**

I have encountered many special challenges.

The first one is that when the number of animals is very large, the program will run slowly and even crash. From internet we know that the framework we used called WPF requires hardware with high configuration.

The second one is that it is hard to make sure that the animal can survive for a long time so that the ecosystem can keep stable.

The third one is that some apis are not act as expected. The running time we use is dotnet6, which is the latest version. The documentation of it is not complete so if some bugs exist, we may not easily solve them by surfing the internet.

**2. 赵旻昆**

**Individual Report**

(赵旻昆 202030430394)

1. **The parts that I am responsible for**
2. Initialization of all livings' generation by groups

(I realized the function that returns all animals’ and plants’ locations when the user chooses “by group” generation mode.)

1. The definition of “FirstNutritionalLevel” class which is so-called plants class and the changing mechanism of plants’ attributes.
2. The decision functions of the 2nd nutrition level (herbivores) and the 3rd nutrition level (carnivores).

(I realized the decision-making functions that call a certain proper action function based the situation of on other livings in its sight and its own status)

1. The gathering action functions of the 2nd nutrition level (herbivores) and the 3rd nutrition level (carnivores).

(I realized the function that control the animals form several groups by getting together with other near companions)

1. Part of the mechanisms of status’ update.

(The codes in the “Updater” file and the function named “UpdateState” are written by me. While the other mechanisms of status’ update in those action functions are responsible by my teammates)

1. The function named “EnvironmentChange” which controls the weather in each tick.
2. Part of the hunting mechanism.

(I realized the hunting functions which select a certain proper prey as its hunting target, let the predator move towards its hunting target roughly and at last call the function named “chase21” or “chase32” which do the hunting action precisely. The “chase21” and “chase32” functions are written by my teammates.)

1. Adjusting quantities of parameters to make our ecosystem become relatively stable with my teammates.
2. **Difficulties and challenges that I encountered**
3. When I realized the function that initializes of all livings' generation by group, I found there were much more aspects that requires considering than what I thought before. For instance, the species not only ought to be looked like groups but also should not squeeze together. What’s more, the degree of randomness has to be very appropriate. If randomness degree is too high, the herbivores’ groups may be too close to the carnivores’, which is very detrimental to the survival of the herbivores. And if randomness degree is too low, the result if out of reality. So, I adjusted the various related parameters again and again.
4. When I wrote the decision-making functions of herbivores and carnivores, I found it is controversial to decide which action the animal is supposed to do when it is in “comfortable” state. That is to say, when the animal is neither hungry nor tired and there is no natural enemy in its sight, what action it does becomes vague. If it stays to rest in this case, there will be too many animals stopping in the map when the number of animals is not large enough which is out of reality. If it runs at a random direction in this case, we can find that most of its energy is wasted during the walking aimlessly which is out of reality as well. And if it gathers around in this case, the logic is farfetched and the effect of gathering cannot be exerted. As a consequence, I introduced the probability of choosing these actions in this case.
5. When I tried to balance the ecosystem, I found it is hard to let it be relative stable since there are large number of parameters requires adjusting. For example, the ages of each species, the breeding probability, the breeding frequency, the energy decline speed and the tiredness increasing speed, the assimilated energy provided by the preys, the relationship between the animal’s speed and the its status……
6. **What I have learnt**
7. The normal distribution function performs well in the initialization of all livings' generation by groups.
8. The heap is the best data structure when choosing the prey with the worst state as the predator’s hunting target.
9. The experience that one mechanism needs taking various aspects into account is accumulated.
10. When encountering thorny problems, we had better learn to be flexible. Sometimes, bypassing the problem is better than struggling to solve it.
11. Adequate discussion among team members is very significant in such group cooperation project.

**3. 张恭利**

**Individual Report**

(张恭利 202030430271)

**Which parts I am responsible for:**

1.The function named “STLChooser” which defines the action choice of the second trophic level.

2.The function named “STLSeek” which defines how the second trophic level animals seek their food when they are hungry and cannot find food in the field of vision.

3. The function named “TTLSeek” which defines how the third trophic level animals seek their food when they are hungry and cannot find food in the field of vision.

4.The function named “STLRun” which defines how the second trophic level animals run in different ways under different circumstances and constraints.

5.The function named “TTLRun” which defines how the third trophic level animals run in different ways under different circumstances and constraints.

6.Part of the functions which involves movement, from calculating the velocity to moving a certain distance in a certain direction.

**What are the difficulties:**

1. How to implement the function named “STLChooser” which defines the action choice of the second trophic level animals reasonably?

How to define such a function reasonably is indeed a difficulty for me for the reason that I need to consider all the relevant circumstances and limitations comprehensively just like a real ecosystem. For the second trophic level animals, the first factor affecting their action choice must be predators, namely the third trophic level animals. When there exist any predators in their horizon, they must escape. When they are too tired, they must have a rest. When they have no enough energy, they must look for their food. Otherwise, they are likely to get together with their companions. Only when all things considered can I ensure that my implementation of the action choice for the second trophic level animals is reasonable. Fortunately, I have implemented the function and it seems to be reasonable enough eventually.

2. How to implement the “STLSeek” and “TTLSeek” functions?

The “STLSeek” function and “TTLSeek” function define how the second and third trophic level animals seek their food when they are hungry and cannot find food in the field of vision. At this time, they need to expand their vision to the global and seek their food.

3. How to implement the “STLRun” and “TTLRun” functions?

The function named “STLRun” and “TTLRun” define how the second trophic level animals run in different ways under different circumstances and constraints. For instance, what should the maximal velocity and average velocity of their running be? And what should the angle be? Besides, how should their physiological state change when they are running? To make our ecosystem more alike to a real one, these functions should be considered carefully. And after I refer to some materials on the internet, I solve this difficulty smoothly.

**What I have learnt:**

1.From this project, the most important thing I have learnt is teamwork. Our team has one team leader and three members. Thus we need to have a clear work assignment. Actually, in the winter vacation, we made several online meetings on the WeChat. We mainly talked about how to begin our project and how to assign our work. Teamwork is particularly significant as not only do we need to have a clear division of labor but we also need to integrate our work after we finish our work respectively. At the beginning, I am all a little confused, not knowing what to do exactly. After some time, my work becomes clear and I cooperate with my teammates smoothly.

2.From this project, I have also learnt C# language and how to use it to create a project like an ecosystem. And my programming ability has also improved a lot. What’s learnt from books is superficial after all. Therefore, we need to put our knowledge into practice. Only then can we know what our shortcomings are. One way to examine our knowledge is by doing project just like our ecosystem.

3.From this project, I have also enhanced my logical thinking. Just as I mentioned above, I have implemented the functions which require considering many circumstances and constraints. If some parts are ignored, it will lead to irrationality of the result. In this case, I searched the internet many times to learn how the animals act and what their habits are under different seasons and weather. And then, I would try to implement them in my code. Through this experience, I have promoted my logical thinking.

**Special challenges:**

1.I have not learnt C# such a programming language before so I need to study it in the winter vacation for several days. And when I put my knowledge into practice, I find it is still difficult for me to implement the functions I want using C#. Thus, I have to do the project while frequently searching the internet.

2.Just as teacher Chen has mentioned, we need to improve our algorithm used in our code to reduce the time complexity. But when I first implemented my functions above, I found that they were too slow that we could only regulate the number of animals to be less than several thousand. After trying and updating many times, eventually we successfully increase the capacity of our ecosystem by a lot.

**4. 陈冠宇**

**Individual Report**

(陈冠宇 202030430042)

**Which part I am responsible:**

1. The realization of the propagation mechanism.
2. The realization of the escape mechanism of the preys.
3. Adjust the parameters of the system to make the ecosystem as stable as possible (to prevent rapid death or mass reproduction of species at a certain trophic level).

**What are the difficulties:**

1. How to realize the propagation of species:

The first difficult I faced was how to realize the propagation of animals. If we deal with this problem from the individual aspect and divide the animals into males and females having to be together to mate, it will be very difficult for us to solve this problem and lead to high time complexity. Considering this problem, I use the idea similar to the minimum spanning tree we have learned in data structure and deal with the problem from a holistic perspective. I regard the location each entity of the same species as vertex. And I use a min heap to store all the edges between each two vertices. And then I use the general tree data structure to construct different trees of vertices. If the distance between two vertices is smaller than a given range, they will be bound to the same tree which means they belong to the same group. Then we can get a lot of entity groups. For each group, the number of entities will increase when propagating. The growth rate (propagation probability) is related to the number of entities in this group. The more entities there are, the higher the probability will be. Therefore, new entities are generated and the propagation mechanism is realized.

1. How to prevent the prey from escaping out of bounds:

In my part, I also need to implement the escape mechanism. In my design, when the escape mechanism is triggered, I first calculate the escape speed of the prey according to the state of the prey combined with the formula mentioned above. The prey then chooses to move away from the predator most likely to capture it (determined by the state and the distance of the predator). However, there is a problem. Our maps have boundaries, and if I don't include boundary checks, the prey will probably go off the map. In order to deal with this difficulty, after each time the prey decides which direction to escape, the prey checks if the target location is out of bounds. If so, it will change the escape direction. The modification method is to start from the original escape direction, and the new escape direction extends from the original direction to both sides of the search until the new direction will not lead to the crossing. The reason for this new orientation is that it can prevent crossing the boundary and keep the prey as far away from the predator as possible.

1. How to deal with the propagation of the first trophic level:

As I mention above, I use the idea which is similar to the minimum spanning tree to generate groups and realize the propagation of species based on this. However, there is a problem that for the entities in the first trophic level. As they can't move, if they only propagate in groups, then in some relatively empty areas, it would be very difficult to generate entities in the first trophic level, so that there may be always not entities in some empty areas. This is in conflict with our real life. So, in order to make the model more reasonable, I made a special treatment for the propagation of the first trophic level. I make it possible for the first trophic level to be randomly generated on the map, thus allowing plants to propagate as widely as possible in our model.

1. How to keep the ecosystem more stable:

After we initially finished our project, we found an important problem that our simulated ecosystem was very unstable. In order to solve this problem, we have to spend a lot of time adjusting some parameters, such as the propagation cycle and rate, the moving speed and life span, to make the ecosystem as balanced as possible, that is, trying to prevent the population of one species from getting too high or too low.

**What I have learnt:**

1. From this project, I understand that a complete plan must be made before a project, so as to achieve higher work efficiency. Because our group had planned many details of the program well at the beginning, our group was very efficient when actually writing the code.
2. Considering all aspects of the situation before writing programs to solve problems is very significant, otherwise there are likely to be many bugs in the later period and therefore need to repeatedly modify the program.
3. Having a deeper understanding of the data structures of Min Heap and General Tree, and learn how to skillfully use data structures to improve the efficiency of solving problem.
4. Learned how to work better with others to develop programs. For example, I need to use some meaningful identifiers and function names to help my teammates understand my program better.
5. Using algorithm to solve problems can not only be attached to one method. When using a certain algorithm can’t efficiently solve the problem, we need to learn to

think of a new way to solve the problem.

**Special challenges:**

1. In my section, I need to use two data structures, min heap and general tree, which are not predefined in the C# library. So, I have to define these two data structures by myself, and this challenge helped me deepen my understanding of these two data structures.
2. Because our program may encounter the situation of a large number of entities. Therefore, when writing the code, I need to reduce the time complexity and space complexity as much as possible, to prevent the phenomenon of program stalling.