****

Data Structure   
Project Report

**Topic：Tomb raider**

**Faculty School of Computer Science**

**and Engineering**

**Program Computer Science and Technology**

**Full English Creative Class**

**Name 周子昕 刘嘉麒 田嘉琪 吕曼**

**Student ID 201730601080 201730613496**

**201730600489 201730600410**

**Lecturer 陈百基**

**Course Code 045101681**

**Credit 1**

**Start/End Date 2019/02/25-2019/06/24**

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

**Data Structure Project: Tomb Raider**

# Background

The project we select is "Tomb raider". The background is set in Passionless Valley, a dangerous and mysterious place described in Jin Yong’s novel. The objective of our project is to make a stand-alone game that let the player play the role of Yang Guo, going through traps and beating enemies to find love flower and save Xiaolongnv.

# Design Principle

## Assumptions

### Map

The game has one 18\*17 map. The map consists of boxes, stones, trees, flowers and green grass, with the love flower located in the middle for the first level and a house at the corner for the second level.

The boxes can be destroyed while the stones, trees and flowers cannot. After the boxes are blown up, items may be produced.

### Characters

The game includes a player-controlled character and some enemies. Yang Guo is controlled by the player while the enemies are controlled by the computer. When Yang Guo meet the enemies, the hit point of him will decrease.

### Rule

In the first level, if the player gets the love flower in the center of the map, the player will enter the next level. In the second level, on arriving at the house at the lower right corner of the map, he wins.

If the player dies under attack, he loses the game.

## Manipulation

Players use the keyboard to manipulate the character to move, place bombs and pick up items.

The keys and corresponding actions are shown in table 1.

Table 1

|  |  |
| --- | --- |
| Key | Action |
| arrow key “↑” | move up |
| arrow key “↓” | move down |
| arrow key “←” | move left |
| arrow key “→” | move right |
| space key | put bomb |
| key “A” | pick up items |

## Items

Book: Increase the attack of the bomb and the scope of the explosion.

Shoes: Increase the speed of the character.

Bubble: Increase the number of bombs that can be placed at the same time.

Bomb: Initially, the player can only place one bomb at the same time. The scope of the bomb explosion includes the unit length in the four directions of up, down, left and right with the bomb as the center. The bomb will do damage to any box or character in the scope of the bomb explosion.

Food: Increase the hit point of the player.

Trap: Reduce the player’s speed.

## Initial values

Table 2

|  |  |  |
| --- | --- | --- |
| **Name** | **Property** | **Value** |
| Player | Hit point | 1000 |
| Speed | 10 pixels/move |
| Maximum number of bombs that can be put at the same time | 2 |
| Dog | Hit point | 200 |
| Speed | 3 pixels/60ms |
| Attack | 60 |
| Wolf | Hit point | 400 |
| Speed | 3 pixels/60ms |
| Attack | 80 |
| Spider | Hit point | 600 |
| Speed | 3 pixels/60ms |
| Attack | 100 |
| Box | Hit point | 600 |
| Others | Number of enemies | 3\*i for each type, where i represents the difficulty level chosen by the player. |

## Functions

### Random map

A certain part of the map is fixed, and the rest of it can be generated randomly. The original positions of the characters and the items are also distributed randomly.

### Picking up items and get the corresponding rewards

By picking up the items, the player’s properties may change, for example, his speed increases when he picks up a pair of shoes.

### Collision detection

If the characters meet bombs, stones, trees, flowers or boxes, their coordinates will be set to the coordinates before they intend to move. Also, the player can’t put a bomb if there is a bomb, a stone or a box.

### Simple AI

The enemies can decide to chase after, run away from the player or stay still. The decision is made taking multiple factors into consideration, including whether there is any bomb, the distance between the enemy and the player and the hit point of the enemy.

When chasing after the player, they are able to find the shortest path to the player, avoiding the obstacles.

### Multimedia

The player can play the game with beautiful music on, which adds up to the entertainment.

# Model Discussion

## Model

### The general model

The general model is shown in Figure 1. To be more straightforward, we simplify the model and get the result shown in Figure 2.

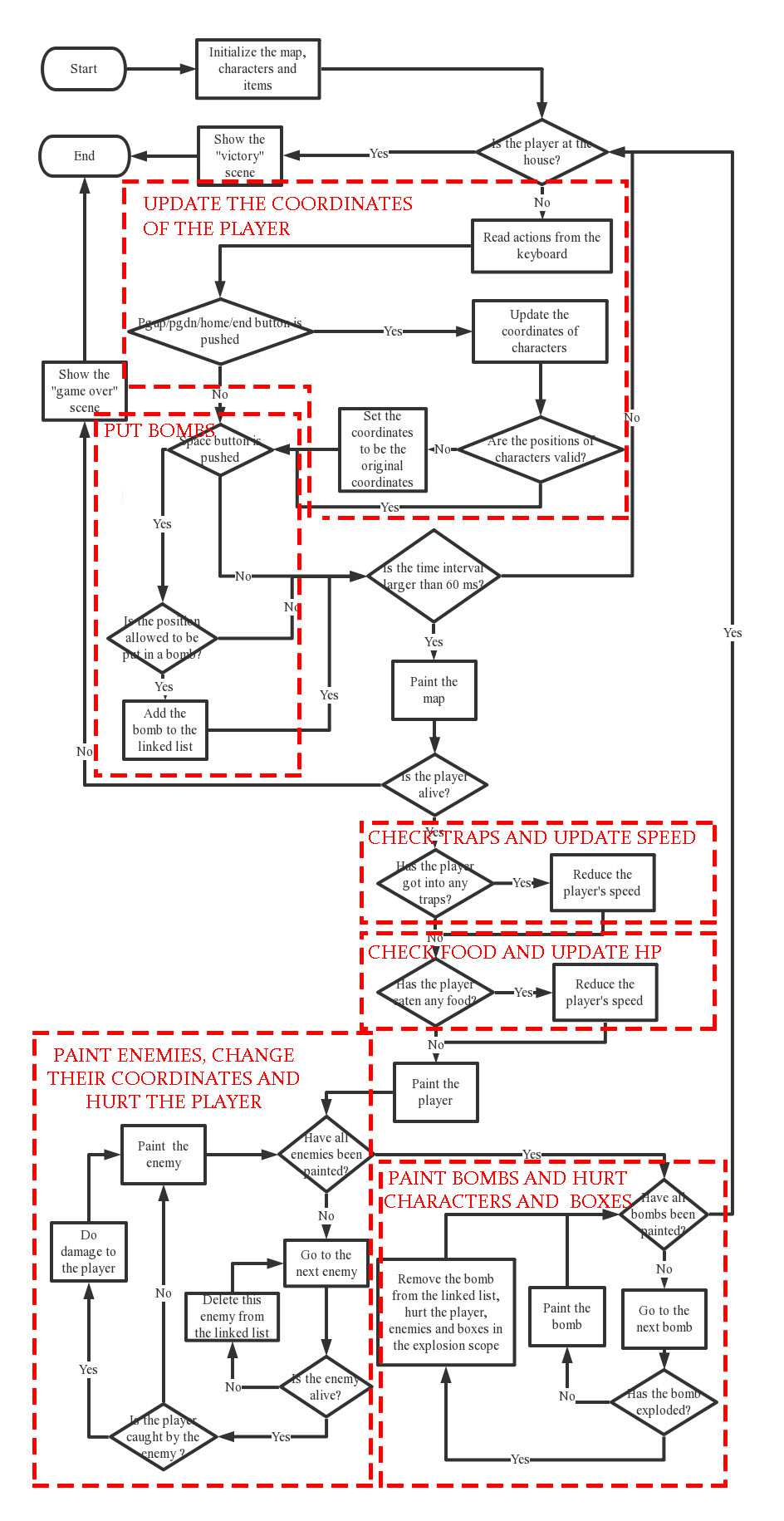


Figure 1

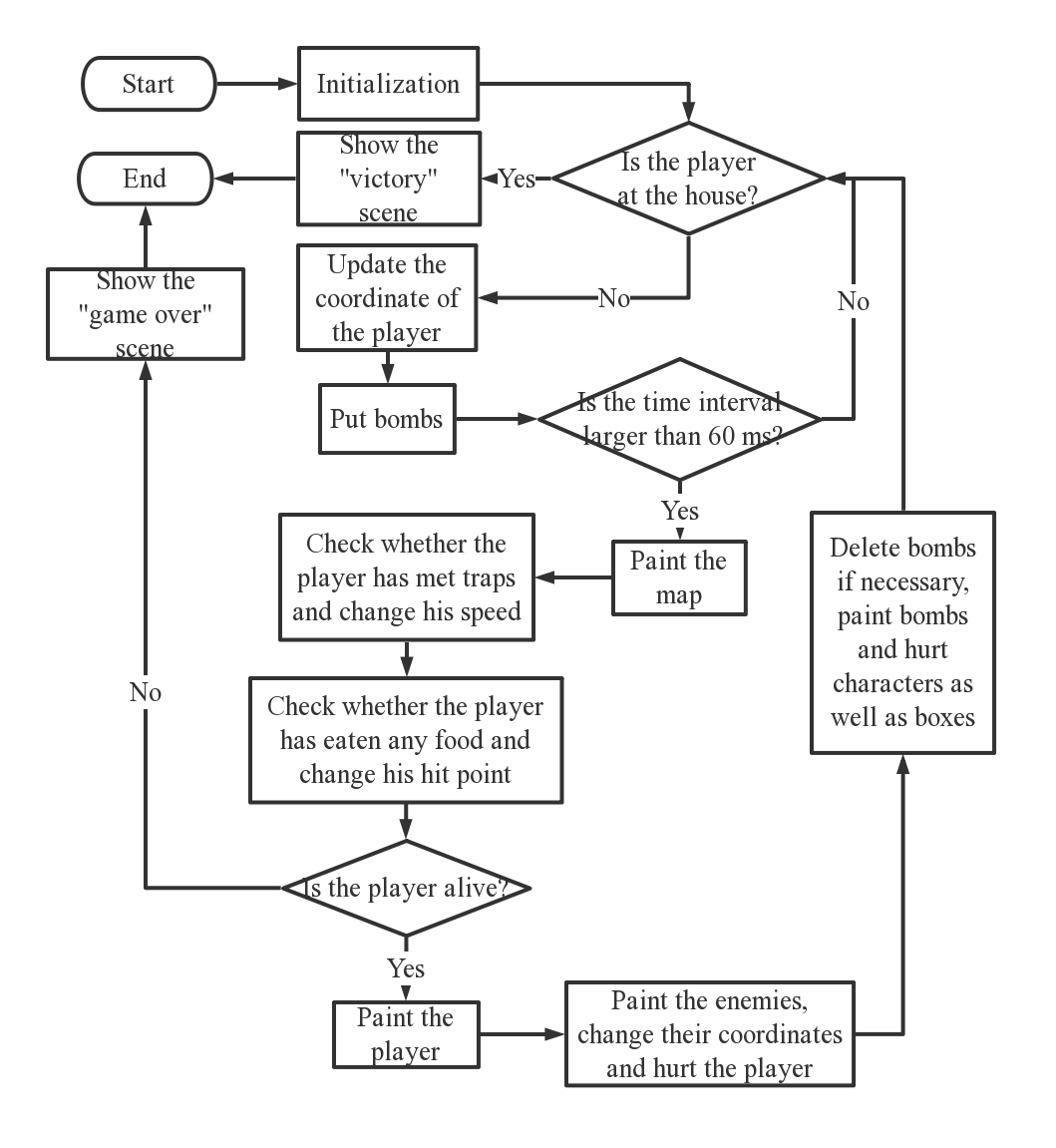


Figure 2

To start the game, we first initialize the map, characters and items. Then we check whether the player is at the house. If he is, he wins. If not, the process continues.

After this, we update the coordinate of the player and put bombs if the system has received the corresponding action.

The next step will be painting all items. Firstly, we paint the map. Secondly, we check whether the player has met any traps and reduce his speed if he has. This will be followed by checking whether the player has eaten any food and increase his hit point if he has.

Then we check whether the player is alive. If not, he loses the game. If he is still alive, we paint him.

After that, we deal with the enemies. If this enemy is alive, we check if the player is caught by it and hurt the player if so, and then decide how it should behave. If this enemy is dead, we delete it from the linked list and then go to the next enemy until all enemies have been painted.

Lastly, we delete the bombs if it has existed for enough time, then paint the bombs and hurt the characters and boxes in their scope of explosion.

After all these steps, we go back to check whether the player wins. If not, we continue the above process. If the player wins or loses, the game ends.

### Initialization

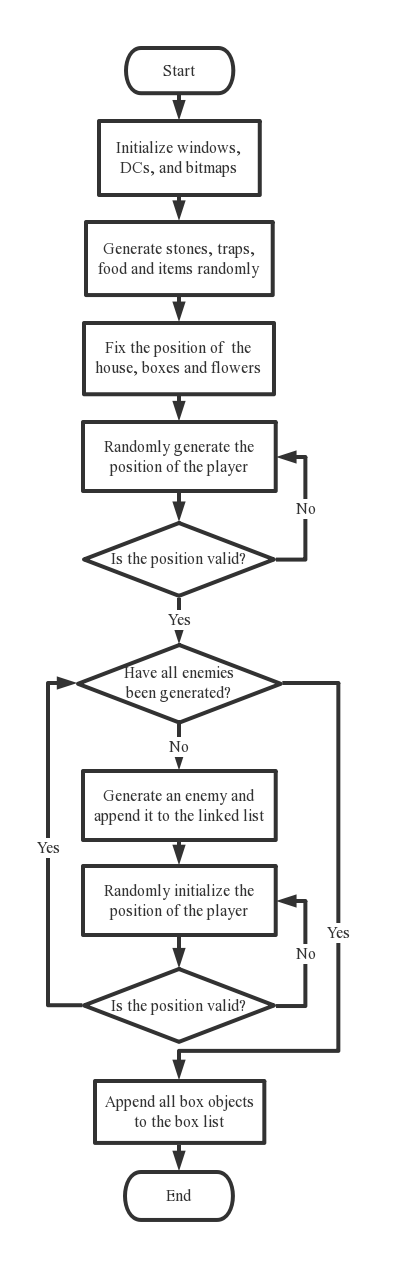
****

Figure 3

To set up the game, we first initialize the windows DCs and bitmaps. Then, we generate stones, boxes and items randomly. After this, we will generate objects of enemies, assign positions to the characters and make sure they are valid. Then initialization ends.

### Manipulations of items stored in linked list (bombs as an example)

#### Adding a bomb

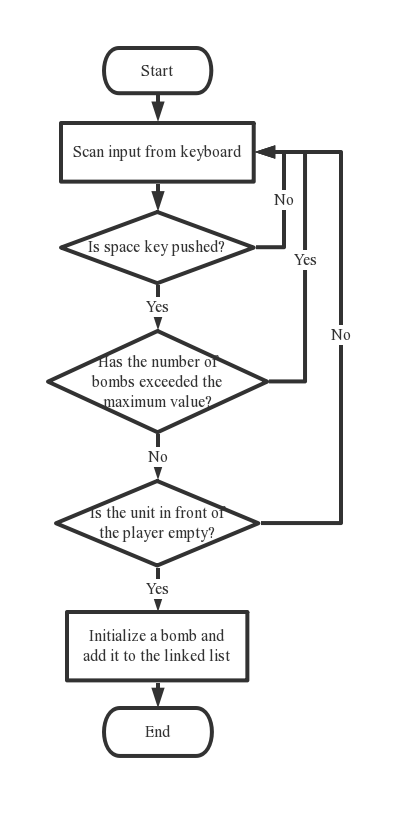
****

Figure 4

We scan input from the keyboard, if the space button is pushed, we’ll check whether the number of bombs exceeds the maximum value. If not, we’ll check whether the present position is valid for a bomb. If it is valid, we generate a bomb object, initialize it and add it to the linked list.

#### Tracking the state of bombs

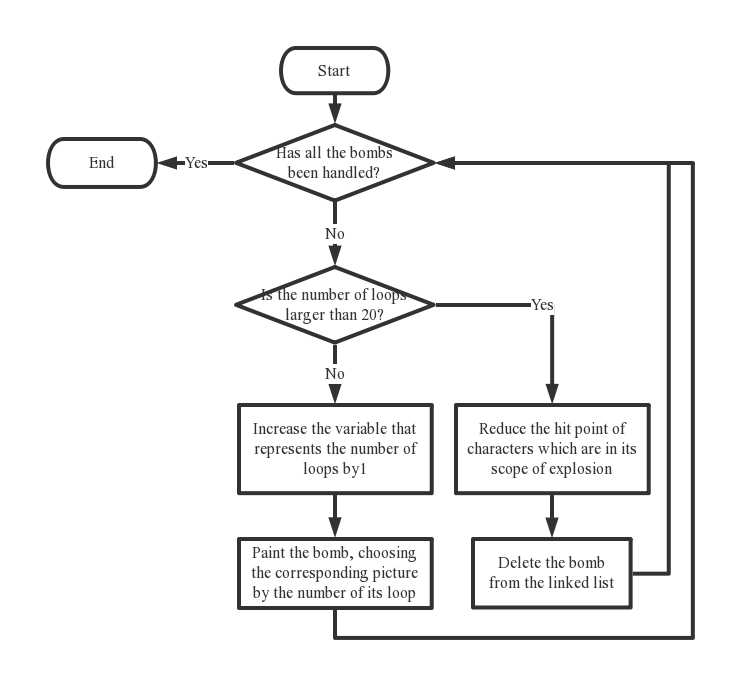
****

Figure 5

We handle the bombs one by one, accessing them by using fence defined in the linked list class. We use a variable to record the number of loops the bombs has gone through.

If the number of loops is not larger than 20, we increase it by 1 and then paint the bomb according to the number of loops. If not, we check if there is any character or box in the bomb’s scope of explosion. If any, we reduce their hit point. Then we delete this bomb from the linked list and go back to see if we have traversed all the bombs.

### Collision checking

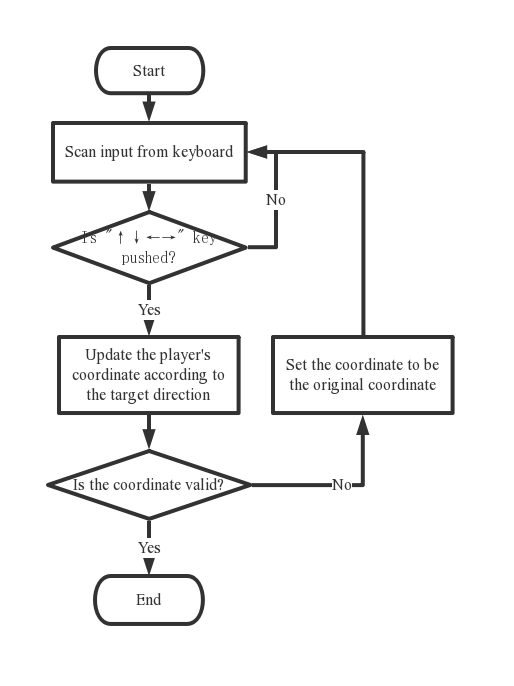
****

Figure 6

We scan input from keyboard, if the player is pushing keys to make the character move, we update the player’s coordinate according to the target direction, and then see if the coordinate is valid. If not, we will set the coordinate to be the original one.

### Decision making and path-finding

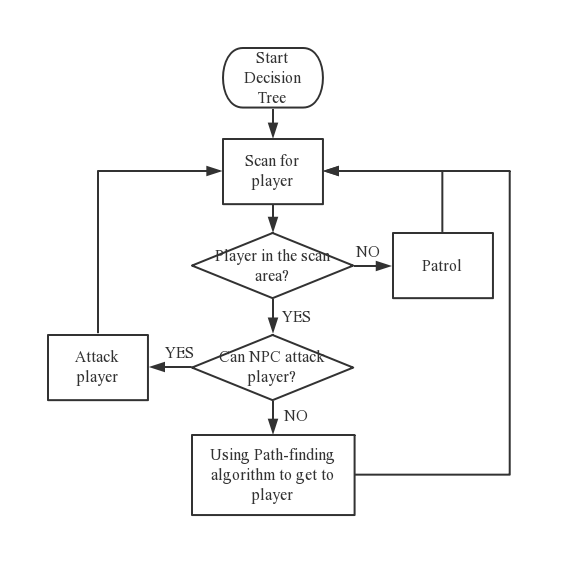


Figure 7

If the player is not around the enemy, the enemy will keep patrolling. If is, we will see if the enemy can attack the player. If it can, we will do so. If it can’t, we use A\* path-finding algorithm to let the enemy approach the player.

### Changing characters’ properties

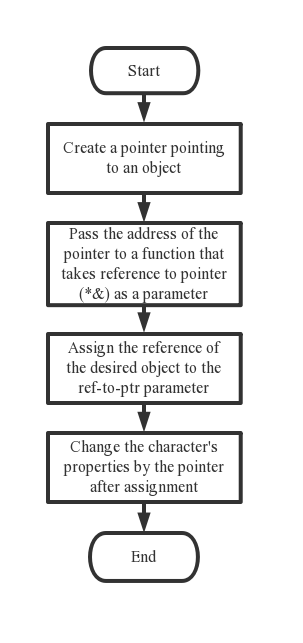
****

Figure 8

Changing the characters’ properties is done by the following steps. Firstly, a pointer pointing to that object is created. Secondly, we pass the address of the pointer to a function that takes a reference to pointer (\*&) as parameter. Then, we call that function, assigning the reference of the desired object to the ref-to ptr parameter.

### Painting the characters

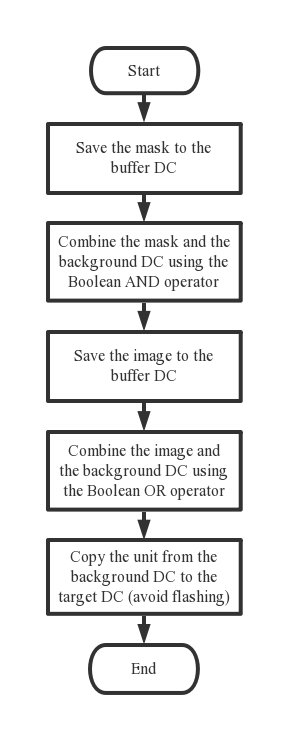


Figure 9

Table 3

|  |  |
| --- | --- |
| Mask |  |
| Image |  |
| Result |  |

We use double-buffer method to avoid flashing. Firstly, we save the mask to the buffer DC and combine the mask and the background DC using the Boolean AND operator. Secondly, we save the image to the buffer DC, combing the image and the background DC using the Boolean OR operator. Finally, we copy the unit from the background DC to the target DC.

## Complexity Analysis

### Initialization

Initialization doesn’t depend on any input from the user, the operations are fixed. Time complexity is O(1).

### Linked list

In our project, linked list is used to store objects that need to be accessed, inserted and deleted frequently, for example, bombs and ghosts.

Therefore, we analyze the complexity of functions that implements the operations on linked lists, including accessing elements, insertion, deletion and searching by key. The result is shown in table 2.

In order to find the element with a specific feature, we search by key. For example, we need to find characters in the scope of a bomb when it explodes.

We apply sequential search, because the number of items to be searched is relatively small (less than 9).

Table 4

|  |  |  |  |
| --- | --- | --- | --- |
|  | Best case | Worst case | Average case |
| Accessing elements | O(1) | O(n) | O(n) |
| Insertion | O(1) | O(1) | O(1) |
| Deletion | O(1) | O(1) | O(1) |
| Searching  by key | O(1) | O(n) | O(n) |

### Path-finding algorithm

**Space:**

We use a struct to store maps, one struct node takes 4\*5+1\*2=22 bytes

pmap is a m\*m array: m\*m\*22 bytes. When m = 18, the space required is 18\*18\*22 = 7128 bytes = 7 KB

**Time:**

For function add, calculate and checkGCollision, time complexity is O(1).

Assume that p is the final number of closed lists. Then time complexity is

T(n) = c + m\*c + p\*(c + m\*m + p\*c) ≈ p\*m\*m

p can be estimated as：

Best case: p =

Worse case: p = m \* m

Average case: Since NPCs only pursuit when player is in their alert-area, we can exclude the worse case and assume average case is similar to best case.

In conclusion, time complexity is O(m2)

## Relation between Data Structure

### Linked list

In our project, there are a few types of elements that need to be inserted and removed frequently. In order to meet this demand, we apply linked list to organize these elements. For example, a bomb is inserted to the linked list when the space bar is pressed. After 1.2 second, it is removed. Also, we can use the find function to access the element with the required key. It’s the same with the enemies and boxes.

### Tree

The enemies are designed to be smart in this project. They can make an attack on the player, hide from the player or stay still. The decisions are evaluated by a decision tree, taking multiple factors into consideration.

### Graph

When the enemies decide to chase after or run away from the player, they need to decide which road to take. In this situation, we need to find out the shortest path for the enemy to approach the player. This situation can be modeled as finding the shortest path from one point to another in an n by n matrix, with weighted path. We treat this problem as a graph problem, which can be solved by applying A\* path-finding algorithm.

# Demonstration

## Starting the game



Button 1 (Group)

Button 2

Button 3

Rules

Figure 10

This is the starting interface of our game. Choose one level and click the corresponding button of button group 1 to start the game. The difficulty varies from level to level according to the number of enemies. If button 2 is clicked, the starting interface will disappear. If the user doesn’t want to play the game with background music on, he can click button 3. When this button is clicked for another time, the music will be on again. The rules of the game are shown in the lower left corner.

## Manipulating the character

The user can use “↑↓←→” keys to manipulate the character’ s movement.

Bombs can be put by pressing the space key.



Figure 11

When an item appears, the user can pick it up by facing it and press the “A” key.

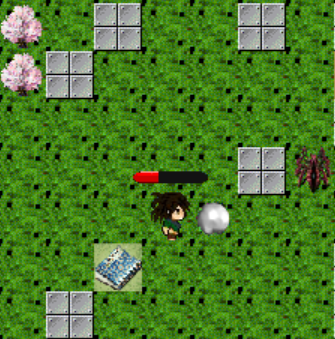


Figure 12

## Items

All the items and their functions are shown in the table4.

Table 5

|  |  |  |
| --- | --- | --- |
| **Item** | **Picture** | **Function** |
| **Bomb** |  | Hurtboxes, enemies and the player in its scope of explosion. |
| **Shoes** |  | Accelerate the player by 2 pixels per movement. |
| **Pearl** |  | Increase the maximum number of bombs can be put at the same time by 1. |
| **Book** |  | Increase the attack of the bomb by 20. |
| **Love Flower** |  | Enable the player to enter the next level. |
| **Food** |  | Increase the hit point of the player by 200. |
| **Trap** |  | Reduce the player’s speed by 6 for 7.2 second. |

## Enemies

When the player is caught by the enemy, his hit point decreases. There is certain chance that the user may escape from the enemy.

Table 6

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Picture** | **Property** | |
| **Dog** |  | Hit point | 200 |
| Speed | 3 pixels/60ms |
| Attack | 60 |
| **Wolf** |  | Hit point | 400 |
| Speed | 3 pixels/60ms |
| Attack | 80 |
| **Spider** |  | Hit point | 600 |
| Speed | 3 pixels/60ms |
| Attack | 100 |

## Aim

For the first level, the player should get the love flower to enter the next level.



Figure 13

For the second level, the player needs to find Xiaolongnv, who lives in the house in the lower right corner of the map. When the player reaches the house, he wins the game.



Figure 14

## Ending the game

### Victory

When the player gets the love flower, this scene will be presented.



Figure 15

Then the player enters the second level.

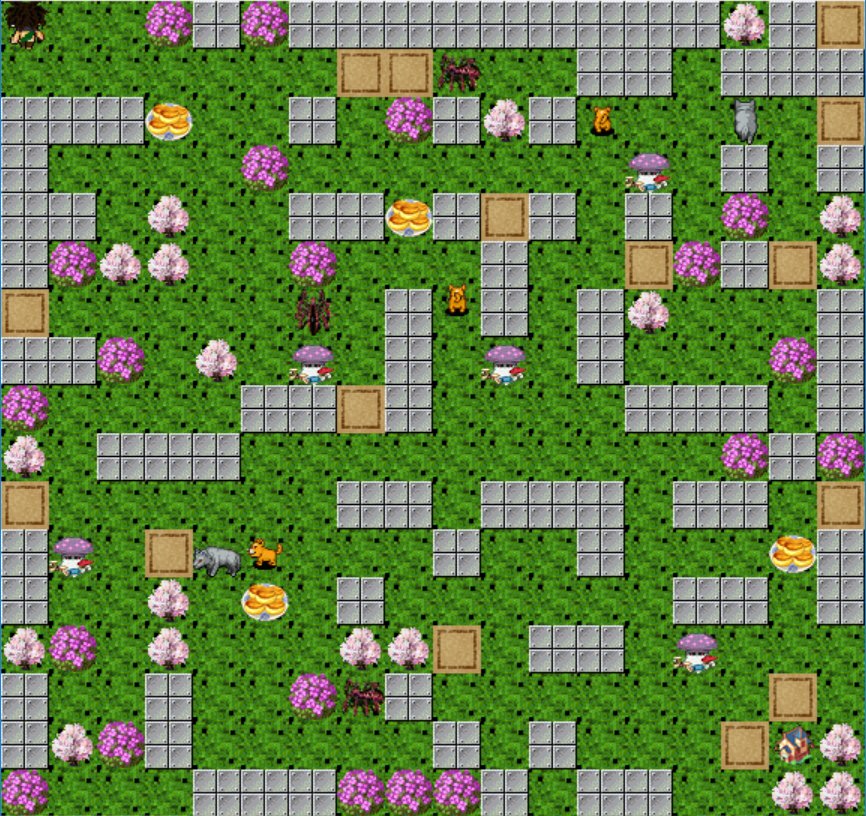


Figure 16

When the player reaches the house, the picture that signifies victory will be shown.



Figure 17

### Defeat

If the player’s hit point becomes less than 0 under attack, he loses the game.

The picture that signifies defeat will be shown.



Figure 18

## Results

This game can run smoothly, without any error. However, we find that sometimes the enemies don’t move as we design. This problem may be caused by the deficiency of the collision checking function.

# Conclusion

In this project, we made a stand-alone game that allows the user to fight against the enemies to reach the destination. It’s entertaining and exciting, with romantic settings, attractive GUI and easy manipulation.

In the implementation, we designed concise structure, put the knowledges of data structure in to practice and use smart algorithms.

# Appendix

* 1. **Individual Report**

1. 周子昕 201730601080

- Which part you are responsible

I am responsible for designing the program and implementing most of the functions. It takes me 20 days to accomplish the work.

- What are the difficulties

The working process is quite difficult. Firstly, the workload is way too large. To improve our programming skills, we decided to make this game a windows desktop application and implement the functions by using win32 api, without using any game engine. Therefore, we need to focus much energy on programming. For example, we need to implement collision-checking and items-picking-up by ourselves, which have been implemented in a game engine. Also, the large system requires immediate update when a new function is added. This can be challenging.

Secondly, it’s difficult to combine knowledges of data structure with a specific object. In labs we used to do, we just need to implement linked list and use simple application to show the correctness of the implementation. However, in this project, more complex application is required. For instance, I need to program to manipulate the objects stored in linked list via fence. At first, I didn’t write the corresponding constructors for the object classes, which led to an error. Then I forget to set the fence to the head of the linked list after a whole loop, causing a read/writing error. After solving many problems like these, I finally made it.

Lastly, debugging makes me upset at times, and I have to cheer up and try again. Since my programming skills are not good enough to make the program run correctly at my first attempt, I need to debug for many times. It requires patience, practice and persistence.

- Any special challenges

The most special problem is very tricky, on which I almost gave up. I need to change one data member of the element pointed by fence. At my first attempt, I passed a pointer as the argument. The value of the data member won’t change in this situation. I didn’t know where the problem lied on, so it took me two days to find out it. After locating the problem, I looked for solution on the Internet and eventually solved this problem by using pointer of reference, which changes the reference.

- What you have learnt

I have learnt a lot in this project. For one thing, my programming skills is improved by programming about 2000 lines. For another, I learn the design process of a large program. Moreover, I get a better understanding of team spirit.

1. 刘嘉麒 201730613496

Which part you are responsible:

I am responsible for implementing A\* algorithm (a path-finding algorithm), designing several mazes and testing the game to find whether there exist some bugs.

What are the difficulties:

The most difficult part when I am implementing A\* algorithm is to find adequate data structure: first, it should be able to record the three key elements in A\* algorithm: G (The distance that come so far), H (The estimation of the remaining distance) and F = H + G. In addition, it should be able to record the previous location so that we have the tool to record the path. What’s more, it should be able to be inserted into the Close list and the Open list (both are the concepts from A\* algorithm). Based on the above idea, I create a struct containing the elements mentioned above.

Another very difficult point is that how to implement A\* algorithm, how to expand the nodes. First I used some priority queue to maintain the Close list and the Open list. But it is simple too much code to write and difficult to debug. As a result, it did not work well and it is difficult for me to modify and optimize it. And then I used Heap to maintain this two list. It turns out the same problem as the previous one. In the end, I chose to use a two-dimension array because it is easy to maintain and good enough for our project size. Still, if we apply it to a very large map, it may not work so well. But in our case, it works very well.

The last very annoying problem concerns parameters: How fast the NPC and the character go? How accurate should the result be? What should happen if the two character is close together? Which point in the character picture should be the point we use in A\* algorithm? These kind of questions are annoying because they are highly sensitive and without a good principle to decide how to deal with them.

What you have learnt：

I have learnt that the choice of data structure is extremely important in programming, not only should it have the ability to satisfy the needs of an algorithm but also it can have a huge impact on how you implement this algorithm. If you choose a data structure wisely, you can save a lot of strength.

1. 田嘉琪 201730600489

I am responsible for the design principle, including requirements analysis, the design of the overall functionality of the game and the details.

The difficulty of this part is not very high, because there are a lot of games on the market are similar to the topic. And we have played some similar games, so the general process of our game has not changed much.

Its type is regular, that is, the player controls the characters moving and destroying the mobs, wining, or being killed by the mobs, and after regardless of failure or victory, you can reset or quit.

During the design process, what needs to be considered is the type and function of the props, the player controlling role, the influence of the mobs on the character (the damage caused by the mobs to the character's life value), the conditions for the player to win and the failure.

In addition, consider the values of the design player and the enemy, such as initial health, speed and damage, and so on. Also, I need to design the map, the characters, the rules,etc.

From the process of designing the game, I can learn to think about the details and grasp the whole game.

In fact, although I was not responsible for the code writing part, before the task starting, I read the Visual C++ Game Programming Basics and the game code written by other programmers.

In the process, I learned more, for example, a part of the function of drawing the game screen, importing the resource picture to form a map, inputting message processing, that is, implementing basic functions of keyboard and mouse controlling through code.

In the process of writing the game, we used the page jump, jumped from the start page to the first level, and jumped to the second level after the player completed the first level. We placed buttons on the start interface and set the background music, which are relatively simple parts.

In our design, the player can get the props by blasting the box. The props can change a certain character of the character, so we randomly generate a partial map, including randomly generating the box in the map.

In our program, we use recursive, pointer and other knowledge points related to the data structure. By using it flexibly in the code, I learned to better understand and use the relevant knowledge more skillfully.

This provides a new idea and a practical approach for my future programming.

The biggest role of learning these is to let me know that the games can be written in Visual C++, including user interaction interfaces, without having to use Qt or MFC.

1. 吕曼 201730600410

Our project started in the winter vacation, so I first discuss with my partners about what the theme, background, structure and way to play are in the first few days in our winter(about 20th January). Then we got the game theme “Tomb Rider” which is from the story of Yang Guo and Xiao Longnv. And the purpose of the game is to rescue Xiao Longnv which is done by Yang Guo, and the way to do this is to arrive at the house located in the center of the map, and in the way there will be many monsters attack Yang Guo, the play should manipulate Yang Guo to eliminate them, if YangGuo dies under attack, then the game lose.

In the second period, just after we confirm the theme, background, structure and way to play, our group decide to realize our game in this following parts: Map, Character, Rules and Manipulation. And I was responsible for collecting materials and sent them to Zhou Zixin. And she completed the programming under the reference of the materials.

In the next period, just after Liu Jiaqi design the path-finding algorithm, and all the classes such as player, enemy, bomb, etc. Then I do the implementation of the simple AI (Decision tree used to decide the enemy’s action: The enemies can decide to chase after, run away from the player or stay at the place to gain some hit point.

They are able to find the shortest path to the player, too.).

In the last part of mine which is after we had a outlook of our project and the main programming and algorithm is done, I write part of the mid-term report (Introduction and objectives).Then our project was going to the last part detailing and connect each part together, debugging,etc.

During all the things i do, i also faced some difficulty, most of them are solved under the joint efforts of our group members, still some is unsolved but we replace the methods to get our purpose. For example, the map we write always has memory leak problem, so we decide to encapsulate to the operation of the linked list further and make up one more time on the basis of the original, finally it was solved.

For another problem, the map output result is alternating output of a line 1 and a line 0 (1 for path, and 0 for wall), the results clearly lost the randomness. Then we debugged each sentence, observe the execution result of each sentence, later we found that the sentence “while (range = counter > 0) “ is without parentheses, leading to no copy operation, the correct statement should be the while ((range = counter) > 0), at this point, random map part code completed. The period of debugging is painful, but the great happiness the successful debugging brought was absolutely more precious.

On the whole process of our project, from its theme first being confirmed to the last successfully running in the computer, I had a further understanding of the framework for the game and the encapsulation of all classes, a profound experience to the class of inheritance and polymorphism , also learned to the construction of a simple game. To sum up, I learned a lot.

* 1. **Reference**

1. 荣钦科技. Visual C++游戏编程基础［M］．北京：电子工业出版社，2005.32-96.