Sketch of the game

Liang Chen 梁晨 3180102160

I.What's This for?

In this report I briefly summarize the basic elements of the game, the story line of the game, the relations between classes of the source code and the challenges I faced when I implemented specific functions of the game.Hopefully, this sketch can be a helpful complement to my source code with comments.Moreover, the further possible research we can do about the game is discussed in the last part of this passage.

II.Basic Elements and Story Line

In the game, a castle of n floors with n rooms in each floor except for the lobby floor is built in the first place.Note that the lobby floor has only one room.For the number n is user-defined, **n*n-n+1** rooms and the castle are **dynamically allocated using new operator.**Originally the warrior is set in the lobby floor and he needs to search around the castle to find the trapped princess and take her back to the lobby floor to successfully complete the rescue.The warrior may encounter monsters in the castle and die immediately.Note that the princess and monsters **are randomly set** in the castle.

To increase the playability of the game, I inject the original game with some additional elements.Firstly, the player can find other objects besides the princess and monsters in the castle.**Medicine and special props** are available in the castle.One potion of medicine can get you through the attack of one monster and with the help of special props, you can win the love of your princess after successfully rescuing her.Note that the medicine and special props are also randomly set in the scene.Moreover, **5 different princesses** are available in the game among which the probability of encountering one of the them is really small while the probability of meeting the other is equal and relatively large.**Once you win the love of all 5 princesses, all the missions are accomplished in the game.**In addition, the warrior can

go to another floor only with an elevator connecting two different floors. (Honestly, it's more suitable to call the connection escalator, but for I define it as elevator in the source code, I prefer to refer to it with the name elevator.) And the elevators **are randomly set in each floor.**

The amount of each type of objects(including elevators)is showed below:

·Princess *1
·Special Props *1
·Monsters *n
·Medicine *n/2
·Elevator *n/3 each floor

The player can control the warrior with A,S and D following the instructions.A complete process of a round of the game is attached below:

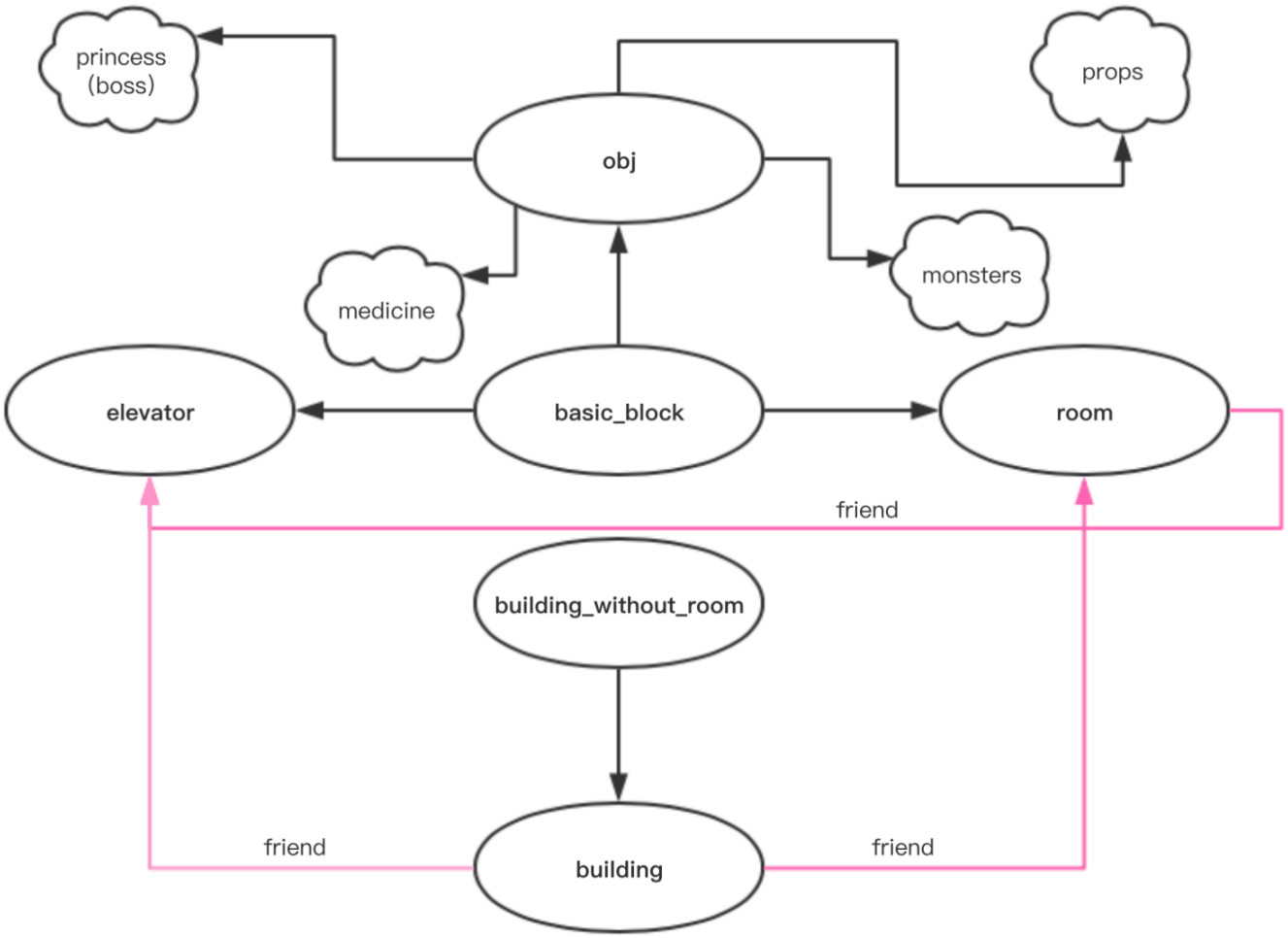
```
Welcome to the Programmer Towers!
You as a programmer are destinied to find your princess trapped in these towers.
And you need to take her back to the lobby.
Press A or D to move left or right.
Press W to enter the elevator to go up and down.
Monsters may get in your way!

Now choose the index of the tower from 3 to 9.
If the index of the tower is n, then the tower has n floors with n rooms in each floor except for the lobby.
However, a tower with higher index isn't necessary to be a tower harder to get out.
Now choose!

6
Press S to start a new game.

s
Welcome to room 0 on the lobby floor.
This floor only have one big room.
The billboard in the hall says:
Now you have won the love of 0 princess(es)

You can take the elevator.
The elevator would take you to the 2nd floor.
w

A few seconds later...
Welcome to the 2nd floor.
Press W to get out of the elevator.

w
Welcome to room 6 on the 2nd floor.
You can go left.
And you can take the elevator.
The elevator would take you to the lobby floor.
```

```
You've found the magical medicine.
It can help you recover from a fatal disease.


d
a
Welcome to room 5 on the 2nd floor.
You can go either left or right.

And you can take the elevator.
The elevator would take you to the 3rd floor.
w

A few seconds later...
Welcome to the 3rd floor.
Press W to get out of the elevator.

w
Welcome to room 5 on the 3rd floor.
You can go either left or right.

And you can take the elevator.
The elevator would take you to the 2nd floor.
a
Welcome to room 4 on the 3rd floor.
You can go either left or right.

And you can take the elevator.
The elevator would take you to the 4th floor.

The gorgeous princess Pony Ma is standing in the centre of the room!
It seems like the princess has something to talk to you.

Pony Ma:How can you get stronger with out topping up?

Okay...Anyway, you're half way to success!
Take Pony Ma to lobby to complete the rescue Pony Ma.


d
Welcome to room 5 on the 3rd floor.
You can go either left or right.

And you can take the elevator.
The elevator would take you to the 2nd floor.
w

A few seconds later...
Welcome to the 2nd floor.
Press W to get out of the elevator.

w
Welcome to room 5 on the 2nd floor.
You can go either left or right.

And you can take the elevator.
The elevator would take you to the 3rd floor.
d
Welcome to room 6 on the 2nd floor.
You can go left.
And you can take the elevator.
The elevator would take you to the lobby floor.
W


A few seconds later...
Welcome to the lobby floor.
Press W to get out of the elevator.

W
You succesfully rescue the princess!
But you don't win the love of the princess.
She leaves you without a word......
```

III.Relations of classes

Primitively, I defined **two abstract classes** basic_block and building_without_room.All other classes are derived from these two abstract classes. I draw a diagram to illustrate the relations.Note that princess, props, medicine and monsters are all objects of the class obj but not new classes, and I draw them in the diagram simply because I originally made them inherited classes of obj but soon realized it unnecessary.Furthermore, the “bigger” class should have the privilege to access the private variables of the “small” class, therefore I make building and room the friend class of elevator as well as make building the friend class of room.

In addition, it’s natural to think of using a **vector of pointers** to store the addresses of rooms in the castle.Besides, **polymorphism** is also applied as abstract classes and virtual functions are used.



IV.Challenges

As the rooms, objects(which means princess, monsters etc) and elevators are all dynamically allocated in the heap, the biggest challenge I meet is how to properly delete these dynamic objects in the destructor.Note that things get especially confusing when I try to delete elevators in the rooms because one elevator is shared between two rooms.It's very easy to **double delete** the elevators unless you are careful scrupulous enough.

Besides the destructor, there's another subtle point that can render your program a total mess if not treated properly-**the state of the other elevators in the same floor.**Once you get to another floor taking one elevator in a specific room, states of all the elevators connecting the previous floor and the current floor in other rooms should be flipped.You can realize the function with a simple for loop, but the challenge is that it's really easy to ignore this problem.

```
building::~building(){
    for(int i=size*size-size;i>=0;i--){
        room* tmp=mapp[i];
        mapp.pop_back();
        delete tmp;
    }
}
// delete the room from the last to the first(because you cannot pop_front)


room::~room(){
    for(int i=0;i<now_obj_amt;i++) {
        delete  obj[i];
        obj[i]=NULL;
    }
    if(e){
        b->assist_des((property_2[0]-'0'-1)*size-size+1+property_1-1);
        delete e;
        e=NULL;
    }//pay special attention! two rooms share one elevator!Calling assist_des is necessary
}



void building::assist_des(int t)
{
    mapp[t-size]->e=NULL;
}//pay special attention!set the elevator of the room beneath(because the rooms are destructed from the
    last to the first)to zero(because the elev has been released in the current room's destructor)
```

V.Further Research

Shortly after I completed my program, I realized a substantial amount of further research could be done about this game.In the first place, the essence of the game is **searching a undirected graph,** therefore **algorithms concerning graph including different searching algorithms and shortest path algorithms** can be applied to give the player a scheme of rescuing the princess and winning her love.Moreover, as the objects are randomly set in the building, **some questions of probability emerge.**You can calculate the probability of successfully rescuing the princess, the probability of having no way succeeding etc.

***Special Attention Please: I developed my program with Xcode on MacOs, therefore the exe file I attached can only be successfully run on MacOs with the help of terminator.**