

Project Doc

C++ Dungeon-Crawler-4

10.12.2022

Henry Hsu 1011849

Enchao Jiang 1010222

Hedi Jaafar 100694170

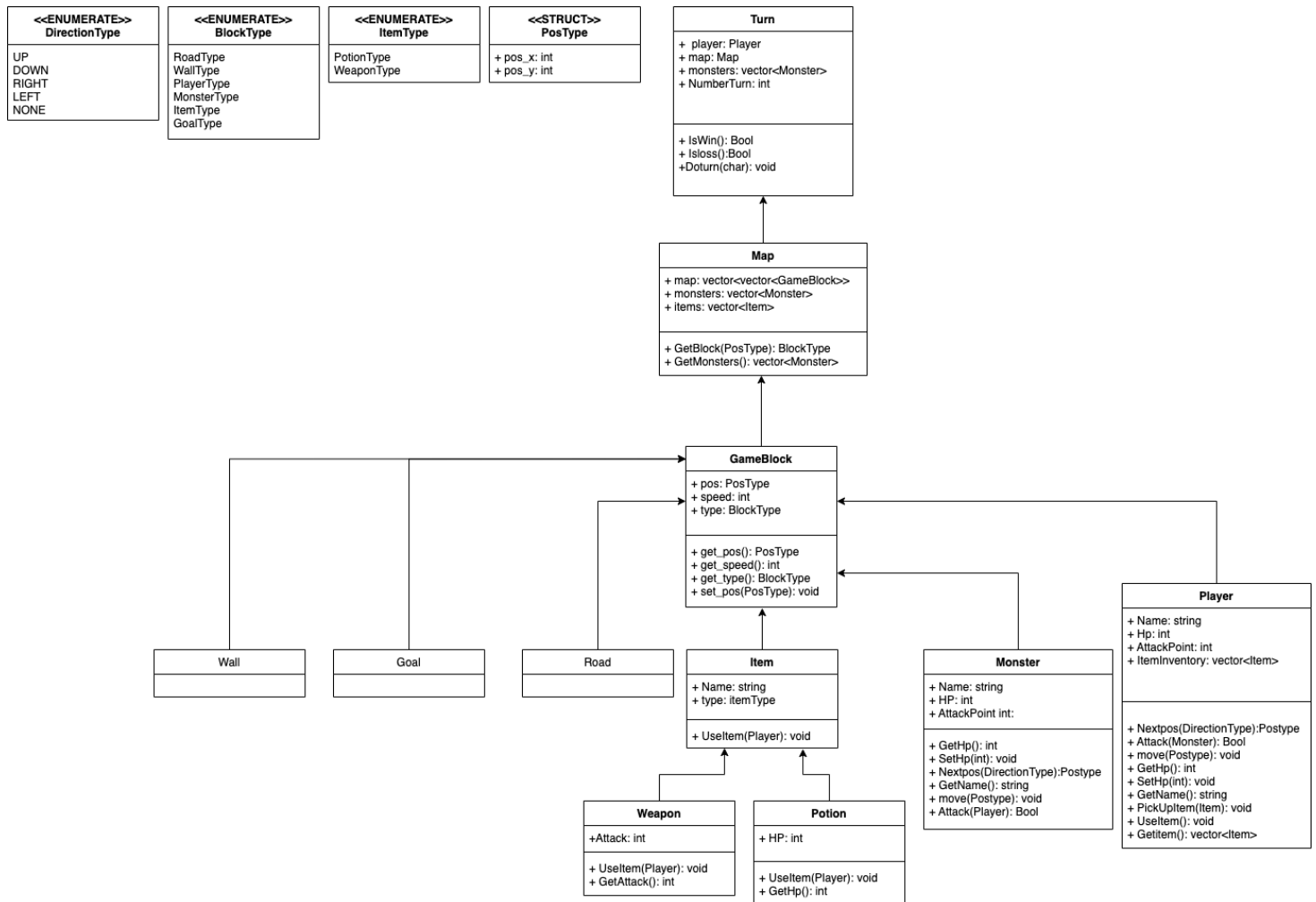
Vladimir Kurazhev 587345

Overview:

The game is a 2D game that the player is in control of the character from the overlook view of the map, which has enemies, items, the goal, and the starting point generated on the map. The monster will move randomly on the map while items will not. The character is controlled with keyboards(e.g. WASD), moving up, down, right, and left. The game process is turned-based. Nothing happens when there are no inputs, after each input the map refreshed and showed the result after the input. When the character moved into a monster the monster is been attacked and losses its HP with the amount of attack point the player has, in the same way, when a monster moves into a player's position the player's HP is reduced by the amount of attack points that the monster has. For now, there are going to be two types of items that will be generated on the map, weapons, and positions. When the player moved into the item is been picked up and stored in the player's inventory, the player could only use the first item in the inventory, however, there are commands that allow the player to circle through the items in the inventory and make the item they want to use the first and then use it. After using the position the player will regain health, after using the weapon the player's attack point then be the attack point of the weapon. If the player moved on to the goal the player wins, but if the player's HP becomes 0, the player losses.

Software structure:

You can look at the refman.pdf file, or generate the HTML documentation for more precision.



GameBlock class:

Base class for all objects in the game which has a struct PosType which indicates its X, and Y coordinates on the map, its moving items, and speed, for walls, goal, and roads the moving speed should be 0, and each type of monster will have different moving speeds, for now, the player's moving speed will be constant. The BlockType indicates what type the block is, either wall, road, goal, monster, item, or player, allowing us to detect what the player is walking into and what should happen.

Map class:

A 2 dimension vector with the high and the width of the entire map, each element in the vector is a GameBlock item allows us to print out the map

Turn class:

The turn class acts like the control of the game, every time an input comes in the input will be checked and the DoTurn function is called with the input as the variable and takes the input into other objects and responds to the input.

The win condition is to touch the goal.

The loose condition is to have health = 0.

Turn :

- 1) Check Player Health (If zero -> Game End)
- 2) Player will choose a key
- 3) if key is a movement Check what is on next position
 - If wall, Player nothing
 - If road, move on it
 - If Goal, Win game end
 - If Item, pick up item
 - If monster, AttackIf key is UseItem() call the function UseItem of Player
- 4) Check all monster health
 - If zero -> GetPos of Monster, Monster is deleted from vector of map, and New Road object on Pos of dead monster
- 5) Monster choose direction randomly
- 6) Check what's on next position
 - If wall, Goal, Item, Monster nothing
 - If road, move on it
 - If Player, Attack
- 7) UPDATE MAP

Item class:

Base abstract class for weapons and potions

Weapon&potion:

Weapon will have attack points that will be the player's attack point when it is been used and Potion will have HP for the amount of health the player will recover when it's used.

Function UseItem(Player) will modify the stat of the Player (Add Health or Attack)

Player:

Will have attributes as the diagram, the NextPos function returns the next X and Y coordinates the player is heading according to the input, letting us to detect what type of GameBlock the player is walking into and make the the correct response, if the type is road, the move function is called and the player's coordinate is moved to the coordinate returned by NextPos. If the type is wall nothing happens, if the type is monster the Attack function is called and attacks the monster, if the type is item the item is been picked up and pushed into the inventory vector, if the type is goal the game ends.

Function NextPos(DirectionType) will return the next position following the direction. The directionType should be chosen with the keyboard.

Function Move(PosType) will update the position.

Function PickUpItem(Item) will add an item to the vector ItemInventory.

Function UseItem() will use the first item of the vector ItemInventory.

Monster:

Basically have the same feature as the player but without picking up items

and its NextPos will take a random input as a parameter letting it to move randomly.

Function NextPos(DirectionType) will return the next position following the direction. The directionType should be chosen randomly.

Function Move(PosType) will update the position.

Wall/Road/Goal:

Empty classes.

How to install and compile the program :

1. There are mandatory lib to install :

```
sudo apt-get install libsfml-dev
```

2. Optional lib for the documentation :

```
sudo apt install graphviz
```

```
sudo apt-get install doxygen
```

3. Clone the project :

```
git clone https://version.aalto.fi/gitlab/jiange1/dungeon-crawler-4.git
```

4. Build the programme :

```
cd dungeon-crawler-4
```

```
make game
```

5. Optional Build the doc :

```
doxygen doc/doxydoc
```

6. Lunch the game :

```
./game
```

How to use the program :

There are 5 controls.

WASD to move UP, LEFT,DOWN and RIGHT

U to use Item

Testing :

First part of the testing is of course with the debugger, and trying the program ourself.

Second part, we used the librarie GoogleTest to do the Unit Test of the program.

We tested each function with key value.

We end up with 100% line coverage, and all the test passing, so if someone try to change the base function we should be safe as long as we pass the test.

Work division :

Tasks that everyone participates in:

- Help with the debug at key point
- Attend meetings and give thoughts and ideas about the project
- General documentation

Specific tasks, and responsibilities of each member:

Henry Hsu:

Hosting, pushing the progress of the meeting, and writing meeting notes, Turn.cpp and help with SFML library

Enchao Jiang:

Everything around map.cpp, helping with all the base class

Hedi Jaafar:

UNIT Testing, base class under GameBlock and Documentation

Doxygen

Vladimir Kurazhev:

Response for familiarizing the SFML library, and graphic handle.

Work Log :

9/11 (Week 45)

First meeting, familiarize with the project.

10/11 (Week 45)

Going through some example code.

Setting the base of the project with UML diagram, and how things will work.

16/11 (Week 46):

Plan finished, and starting with the SFML library.

Base class under Gameblock mostly done.

23/11 (Week 47):

Starting with GoogleTest and Doxygen.

Adding function to the base class.

Turn class almost done, and start of Map class.

First example with SFML.

2/12 (Week 48):

Finishing the Turn class to suit exactly our utilization.

Finishing the map constructor, and testing SFML with our setup.

7/12 (Week 49):

Testing and debugging the full project.

Adding a Makefile so that it can compile for Linux machine, and adding the tutorial to make it work.