Programming III

Milestone 02

Deadline Monday 26/12/2022 11:59 PM

Dear C++ Master, congratulations on passing level1 of the game! and collecting all of the C++ gems; skills, grades, while overcoming all of the obstacles that faced you throughout the semester. In the second and final level, you are required to add few extra features, for ensuring your full understanding of the C++ OOP concepts.

Description

You are required to add few enhancements to your game. After starting the game, the user should be welcomed and sent directly to level 2, in which you are required to implement the following:

In level two, there should be two champions, in which the user can choose to play with one of them, they are Mario, Luigi. Each of them has a super ability. Mario's ability allows him to move 2 steps in the chosen direction, so that he may jump over an obstacle without having any damage. On the other hand, Luigi's ability is that he can remove all obstacles in front of him in the selected direction (without moving).

Class **Champion**

- It should have two subclasses Class **"Mario", Class" Luigi"**
- The champion class have the method **useAbility()**, this method does nothing, it just print "ability x is called", that is overloaded in both it's subclasses and **can only be used twice in the game**.
- In Class Mario the **useAbility()** method allows you to move 2 steps at a time allowing you to jump over an obstacle without having any damage. It must print **"Mario Ability is called"**
- In class Luigi the **useAbility()** method allows you to remove all obstacles in front of you vertically or horizontally. It must print **"Luigi Ability is called"**
- **There should be a RemaningAbilityMoves counter printed along with the champion info, at each step.** The ability should be activated upon pressing **key x** prior the move direction key. Therefore, an informatory message should be printed to the user at each turn, saying so.

Class **Map**

- Has 3 basic attributes rows and columns representing the size of the map in addition to a dynamically created 2D array of **Cell** objects **pointers** called **board**
- The **board** itself should be created using the same conditions of milestone 01
- With the champion starting in position 0,0.
- The **board** shall have 20 obstacles divided into 10 bombs and 10 thieves randomly distributed in the map.
- The **board** shall have 40 gems randomly distributed in the map.
- The rest of the cells in the map shall be empty(basic) cells.

Class **Cell**

- It should have 3 main attributes char type, representing the type of cell, int x and int y representing the location of such a cell.

- The “Cell” class should have a constructor that initializes all the attributes as well as a default constructor that initializes x, y = 0, and type to be ‘**.**’
- **The classes champion, obstacle, gem should inherit class cell**

Class **Obstacle**

- It should be having 1 main attribute representing the amount of dmg the champion takes.
- The “**Obstacle**” class should have a constructor that initializes the attribute dmg_amount with a random number from 1 to 5.
- The **Obstacle class** shall have 2 sub classes **Class “Bomb”** and **Class “Thief”**
- The obstacle class has the method **execute(champion c)** that is overloaded in both subclasses
- In **class Bomb** the **execute** method will reduce the HP of champion c with the dmg_amount of the obstacle without going below 0, the method should also print **“ bomb excuted with dmg = dmg_amount(replace by dmg_amount number)** “
- In **class Thief** the **execute** method will reduce the gemCount of champion c with the dmg_amount of the obstacle without going below 0, it should print “**thief executed with dmg = dmg_amount(replace by dmg_amount number)** “

Class **Gem**

- It should be having 1 main attribute representing the amount of points the champion adds to his score or health.
- The “**Gem**” class should have a constructor that initializes the attribute **points** with a random number from 5 to 10.
- The **Gem class** shall have 2 subclasses **Class “Potion”** and **Class “Coin”**
- The **Gem class** has the method **execute(champion c)** that is overloaded in both subclasses
- In **class Potion** the **execute** method will increase the HP of champion c with the points of the gem without going above 100, the method should also print “**Potion executed with points = *points* (replace by the points number)** “
- In **class Coin** the **execute** method will increase the gemCount of champion c with the points of the Gem, it should print “**Coin executed with points = *points*(replace by points number)** “

Important notes

- The 2d array should include pointers of type cell, that should always point to one of **the dynamically created cell class children(**champion, obstacle, gem)
- Class champion should have a type of character “c”
- Class Obstacle should have a type of character “o”
- Class gem should have a type of character “g”

Game play

- Once you run your main method, you will have the choice to create either a Mario champion or a Luigi champion by pressing ‘M’ or ‘L’
- Once you are done with selecting the champion you press ‘1’ to start the game, ‘2’ re randomize map.
- A map should be randomly generated according to the description above, and you keep on moving around the map (similar to milestone 1) until you either die from hitting many obstacles or win by collecting all gems.
- The map should be updated and displayed with every action you take (similar to milestone 1)

- The Champions info shall be displayed showing his health, location and gems collected as well as his type (Mario or Luigi), ability moves remaining.
- All of the milestone 1 requisites must apply, except for the mentioned changes.

Important notes:

- Since we are using dynamic arrays and objects reaction you need to make sure you overload the proper destructors to prevent any memory leaks
- Use the virtual keyword to allow dynamic polymorphism
- Each cell has the “char type” attribute to help you check, typecast, print the appropriate character in the map
- **Bonus: Implement GUI for your game**
- **Submission information will be announced later.**