CS-224 Object Oriented Programming and Design Methodologies

Assignment 02

September 10, 2018

1 Guidelines

You need to submit this assignment on 24th of September at 630 pm as the next assignment will be given on the same day. Some important guidelines about the assignment are as following:

- You need to do the assignment alone
- You will submit your assignment to git-hub
- You need to follow the best programming practices like the previous assignment. Failure in doing so will have your marks deducted.
- Submit assignment on time; late submissions will not be accepted.
- It is strongly advised that you start working on the assignment the day you get it. Assignments WILL take time.
- Follow git guidelines as shared on workspace as given here.
- DO NOT send your assignment to your instructor, if you do you will be given ZERO for not following clear instructions.
- You can be called in for Viva for any assignment that you submit

2 Text Based Adventure

In this assignment you will be creating a text based dungeon crawler game. You have been given a source file `main.cpp` which contains the overall structure of the code that you need to write. There is also `Assignment2.exe` given to you so that you can play the game and judge how it works. Keep in mind that the dungeon needs to be hidden in your final submission. It is shown here for debugging purposes. The functions given here are all that you will need but you can make your own new functions. Details about the assignment are as following:

- The dungeon should not be smaller than 8x8 grid.
- P marks the position of player
- X marks the position of Exit
- E represents enemies on map
- F represents food on map
- T represents trap on map
- H represents health on map
- w represents wall on map

2.1 Dungeon Creation

You will create a dungeon of size **width** x **height**. What this means is that you will need to create a single dimensional dynamic char array of size **width** x **height** as given by the user. This should be done inside the **CreateDungeon** function. You may need to store the starting point and the exit point by using the **Point** structure. The **Traversal** function will then use char array to help with the traversing. Here are the guidelines as to how the dungeon should be created:

- You should place walls around the borders of the dungeon
- Traverse the empty spaces one by one and place objects with a 20% probability. If the probability is met then you should generate another random integer. If the value is between 0 and 15, place an enemy. 15 and 30, place health. 30 and 45 place trap. 45 and 60 place food and a wall if it is beyond 60. This will create a dungeon with a reasonable randomness. The numbers can vary if you want to change the difficulty of your dungeon.

- Now traverse the dungeon again and place the player's starting position at random right next to the left edge of the dungeon. In case you traverse and are unable to place the starting position, place it on the top-left corner by default.
- Place the exit the same way right next to the right edge of the dungeon. Again, if you are unable to place the exit at random, place it at the bottom right corner by default.

2.2 Dungeon Traversal

Rules for traversing the dungeon are as following:

- You will move left, right, up or down every time
- Each step will consume 1 food
- If you encounter a wall or an edge, you will not move and lose 1 food
- If you reach H, you will gain 1 health.
- If you reach T, you will lose 1 health. You will call the **TrapStatements** function at this point which will show one of the three random statements.
- If you reach F, you will gain food for 4 to 8 days. You will call the **FoodStatements** function at this point which will show one of the three random statements.
- If you reach E, you will fight 2 to 4 enemies. You will call the **Combat** function at this point which will play the combat automatically. In combat you will attack the enemy with a 30% probability to score a hit. If you hot the enemy, You will call the **HitStatements** function which will show one of the three random statements. The enemies will attack you as well with a 10% probability each for making a hit. If they hit you, you will call the **GetHitStatements** function which will show one of the three random statements. This will go on till either the enemies die or you die.
- If you reach X, you will win the game
- If you run out of food, you will die
- If you choose 'x' as input for your turn, you will die

3 Tips

Here are a few tips that can help you in doing this assignment:

- Do not do this assignment in one sitting. It will overwhelm you.
- You will need to do a bit of calculations to make the single dimensional array to behave like a multidimensional array. Keeping the array size small in the beginnnig will help.
- Do not forget to deallocate memory locations when they are no longer in use or if your program is ending.
- Showing the dungeon in every step will have you debug easily.
- Run the given executable many times to see how the program should work. Many of your questions would be answered that way. It is however missing a few checks that you will need to add yourself.

4 Further Enhancements (Not Required)

If you end up completing the assignment and had fun, you can make it even better by doing the following additions:

- You can have enemies with different health pools. Basically, they will be created as player objects.
- You can have more than 3 statements in every function. This will make your adventure story less redundant and more fun.
- You can have your enemies move every turn too (randomly or with your defined AI)
- You can have a series of rooms. As you reach an exit, you move to a new room
- You can add story elements as well for example, you find notes. As you find all the notes a story becomes clearer.
- You can have the game play itself, which would mean that you will define player AI.
- Let your creativity run wild and you can add much much more to this text based adventure.

–THE END–