The World of Supercalifragilisticexpialidocious

by Cohort Class 5, Group 6: Eunice Kwok(1005469), Guo Ziniu(1004890), Sarah Chua(1005522), Xavier Leung(1005808)

The game our group programmed is inspired by early 2000s web browser flash games, where simple commands of directional inputs and possibly 1-2 additional buttons are used to play the game. In our game, we chose to have a dungeon crawler game, similar in concept to Mystery Dungeon (DeVries, 2008).

As quirky as the name suggests, the player will be on a quest to explore the map alone. Any step he takes may encounter a surprise event. The player starts out in the middle of the map, where he needs to navigate the map to trigger event spaces. The path of the player will then be lighted up. In these event spaces, the probability is evenly distributed to trigger a combat event, a quiz event, or a non-combat event. The game isn't designed to be consistent that the player should clear the game every try playing optimally, rather each play is meant to be different from the last, which is a game concept from rouge-like games. This uniqueness is done by randomization.

The objective of this game is to reach a certain points before the player loses all its HP or had explored the entire map. Players will gain and lose HP points through the series of events they encounter.

A combat event is where the player enters a fight with a monster. A race against time- While the monster automatically deducts HP points from the player, the player will need to continually press the “G” key as fast as they could to fight against the monster. The first to reach zero HP will lose the fight. The player is immediately healed after every combat and will receive rewards and penalty accordingly.

A quiz event is an event where the player attempts to solve either a generated math question or a predetermined word puzzle. Should they be able to solve it in one try, they would receive a boost to their attack, otherwise, nothing happens. However, in the event they choose to give up, a penalty will be given as our game does not encourage people to give up.

Finally, a non-combat event is an arbitrary event where the player is either rewarded or penalized in their stats.

Our code uses turtle to visualize and give real time feed back to the player as to where they are in the room. We adopted the use of pop-up messages rather than event prompts to enhance the gaming experience. The functions are kept to a minimum so that the code would hopefully be easier to read.

Documentation:

In this game, we will need to import Turtle, random, tkinder, time, threading, os.

non_combat() ….. This function plays the role of a treasure chest where upon discovery, either rewards, penalties or nothing will be given to the player on random. A message reflecting the result will pop up (through the use of the tk.messagebox.showinfo(title,message) function), followed by the updated information of the player's current attributes.

math() ….. This function generates 2 random integers, between 1 and 100, and 1 math operator to form a basic arithmetic math question for the player to solve. The operator is chosen by randomly choosing an integer between 1 and 4 where ' + ', ' – ', ' * ', ' // ' are represented by integer 1,2,3, and 4 respectively using the randint(start,end) function.

gibberish() ….. This function generates 1 random word puzzle from a dictionary of questions. A new duplicated list named "unanswered question" is created so that answered questions will be removed from this dictionary, preventing the same questions from appearing again. As while loop is used, "break" is used to stop the loop after an answer is given.

The code used for gibberish() function is made with reference from https://stackoverflow.com/questions/43206277/how-can-i-make-my-quiz-randomly-select-unseen-qs-in-python

For both math() and gibberish(), textinput method is used for users to enter their answers. The code is written such that rewards will be added accordingly if correct answers are given and the player's information will be updated immediately.

fight() ….. This function is triggered when player triggers a combat event, prompting the player to either fight or run from the combat space. This function pop-up a message providing the instructions for the game. The function for the mechanism of the fight [showfighting()] is used inside this function. Also, it resolves the result of either choice of the player.

showfighting() ….. This function activate the "g" key for the player to attack the monster. The amount of destroy each attack carries depends on the difference between the number of strength and defense points the opponent has. This simulates the combat between the player and the monster during a combat event.

showplayer() ….. This function shows the location of the player and the current player state using turtle commands. It also draws all the event grids and update the grid with names of the events using turtle commands. n_position refers to the grids that has treasure chest as the event, f_position, m_position and g_position for combat, math and gibberish event respectively.

Job(threading.Thread) ….. This class creates another thread, so that the monster can attack the player automatically at the same time through the use of time.sleep() function. As parameter given was 2, the monster will attack the player every 2 seconds with a power of k, where k is the difference between the monster's strength points and the player's defense points.

The code for this class that control the other thread was taken from https://www.cnblogs.com/scolia/p/6132950.html

eventspace() ….. Assigns 20 XY coordinates randomly within the game space where an event will spawn.

check() ….. This function checks for the position of the player constantly to see if the player lands on an event space. It also determines with equal probability which of the 3 main events will spawn.

init() ….. This function sets up the initial setting at the start of the game. We set the map size to be of 800 x 800 pixels. It also triggers the initial message, explaining the basic concepts and controls of the game.

gameloop() ….. This function updates the player position and vision when the player moves in the game space. It draws all the explored grids using the square() function where it changes the box color to white. It also concludes and ends the game when the player finish exploring the entire game space, achieve the winning condition, or when player health (HP) is reduced to 0.

explore_w() ….. This function moves the player up one space when the "w" key is pressed. Along with similar functions for the other keys, he can then move around in the map by boxes of size 40x40 pixels. Similarly, the other explore_* functions move the player accordingly.

square(x ,y , size, color_name) ….. This function draws the squares on the map.

tk.messagebox.showinfo(title,message) ….. This function will generate a pop up message with the title and message given as parameters.

Citations

DeVries, J. (2008). Mystery Dungeon: Shiren the Wanderer Review. Retrieved December 08, 2020, from https://www.ign.com/articles/2008/03/04/mystery-dungeon-shiren-the-wanderer-review