MacGyver Labyrinth

Link : https://github.com/hjung06/MacGyver-Labyrinth

The objective of the game is to escape from the labyrinth by collecting 3 items while avoiding a contact with the guard. The game ends if MacGyver collects all 3 items and escapes from the labyrinth or either MacGyver and the guard makes a contact.

Pygame module was used making this game.

Text file 'n1'

A maze was created with a text file containing 15 lines of 15 spaces which represents the total number of rows and the columns for the maze. A set of characters (x = wall, 0 = corridor, d = departing position, e = finishing position) were used to represent different sprites which are going to be used for the maze.

Labyrinth.py

Sets conditions for the game. Determines if player have won or lost.

Constants.py

Contains constant variables.

Classes.py

Contains class objects for labyrinth.py

Class Maze

Generates and calculates position of sprites to be displayed on screen.

Method 'generate()'

This function opens the text file **'n1'** and iterates each characer in lines and turn them into a list type variable called **'structure'**.

Method 'wall()'

This method takes the list **'structure'** and designates each character its position in **[row, column]** format, then calculates its position in pixels by multiplying 32(size of the sprite) to its coordinate value. It stores the positions of character 'e' and 's' to the **'exit_position' 'starting_postion'**

It also creates a list called **'corridor'** and stores all pixel values of character '0' from the text file.

Class Guard

Finds position of guard and display the guard on the screen

Method 'guardPosition()'

This method finds the position of the guard. From '**exit_position',** it stores the coordinates of blocks around the exit if it exists and stores them in to a dictionary in { 'direction' : coordinate} format. Than it excludes the blocks which has a value of 'x'(a wall) and creates a

list containing only key value [‘up’, ‘down’, ‘left’, or ‘right’]. It randomly chooses a value and calculates the position of guard and removes the postion from the list **‘corridor’**.

Class Item

Display items non-obtained. Counts Items obtained by MacGyver and display them on the score board.

Method ‘itemPosition()’

This method generates random positions for items by selecting three different positions from the list **‘corridor’** and appends them to **‘item_position’**.

Method ‘displayItem()’

This method removes items from the screen if MacGyver obtains items.

Argorithm used:

1. display items using the positions from **‘item_postion’**; 2. draw the times on window; 3. If position of MacGyver == position of item; 4. Add 1 to the score; 5. Remove the item from the list; 6. Repeat until all items are removed.

Class ‘Player’

Calculates position of player.

Method ‘move()’

This method maneuvers MacGyver if user selects a direction from keyboard. If a key is pressed. It takes a value from each key pressed and adds or subtracts 32 pixels from either X axis or Y axis from the current position of MacGyver and displays new position on the screen. If the postion exceedes boarder of screen and equals to the position of wall, nothing happens.

Conclusion

The hardest part of this project was creating the algorithm for the Guard. I would get an IndexError, each time when I would try to get the value of blocks next to the exit block. This problem was due to the fact that the blocks does not exist. I used **Try, Except** methods to bypass the error and only get the existing values and copy them to a dictionary.