F09 Group 8 – MOns-T.E.H. Run

Members: Michael Chun Kai Peng, Stephanie Pang, Novian Tan

Description

Mons-T.E.H Run was inspired by a minigame in “Fall Guys” called “Tip Toe”. Using the same concept of a mix of real and pseudo tiles to challenge the player trying to cross the bridge, we added our own story line to make it interesting. The story is as follows:

You awaken by a series of rumblings, confused and lost. An NPC, ‘PW’, orientates you and explains that a evil character by the name of ‘MOns-T.E.H’ is chasing you, and the only way to escape is through a series of bridges. However, these bridges were designed to trick the users, as there are tiles which are fake. While falling in once or twice will respawn you at the start of that bridge, falling in too many times will allow the MOns-T.E.H to catch up with you, and you wouldn’t want to know what happens next.

A grid is shown to provide you visual aid. The green tile indicates where you must reach to escape, while the red tile indicates your current position. To track your progress, the orange tiles indicate the tiles you’ve been on. To navigate through the bridge, key in “Up”, “Down”, “Left” or “Right” (not case-sensitive) to move accordingly. After each move to a real tile, the grid update. However, if you step on a fake tile and fall, the grid resets, so its important to remember the ‘real tiles’. Finish all 3 stages to escape the MOns-T.E.H.

If you wish to just play a single stage, ‘Arcade’ mode is available. You can choose the difficulty level and play that stage alone.

Documentation

The game requires libraries: **Time**, **Matplotlib** and **Random**.

introduction(). This function returns Boolean determining game mode.
It first prompts the user to select a game mode. If story mode is chosen, it will print the introduction messages and game instructions. If arcade mode is chosen, the cutscene will be skipped.
An invalid input will send an error message to user to input a new instruction.

difficulty_level_selector(). This function returns integer which indicates board size.
This function only runs on arcade mode.
It prompts the user to input a difficulty level, Easy, Normal, Hard or Extreme, and assigned the **board_size** to a respective integer. The harder the difficulty level, the larger the board.
An invalid input will send an error message to user to input a new instruction.

generate_board(board_size). This function returns a nested list that represent the playing board.
It takes an integer value board_size, which will determine the number of elements in the lists which represent the rows and columns of the board. The contents of **board** are

randomly chosen with 33.3% change of "**T**" and 66.6% change of "**F**". Start and end points are also specified at the corner locations.

is_there_path(board,board_size). This function returns a Boolean that determines if the board has a path from start to end.
Pathfinding code that was developed with the assistance of Prof Matthiew.
It takes in the nested list, **board**, and integer, **board_size**, and checks at each instance of the path if there is a neighbouring "**T**" that thus connects start to end.

create_display_board(board_size). This function returns a nested list with integers.
This function takes in board_size to create a nested list. It represents the empty board of the player when the game starts.

display_board_progress(coord_next,display_board,board_size). This function takes in display_board and appends it according to coord_next. This updates display_board after the player makes a move to a new coord_next.

gameplay(board,board_size,display_board). This function is the main logic of the game.
It takes in **board**, **board_size** and **display_board**. It prompts the user to key in an action move, which will be translated into a coordinate of the board. The coordinate is then used to check if it is a valid move and if it is a real tile or a fake tile. If it is a real tile, the player can continue the game, otherwise the player must restart the round. After each move to a real tile, display_board is updated and shown to the player. It works by the following logic:

```
1. While game not won:
    a. User input action
    b. If player moved to end point,
        i. Game won
    c. If player moved to real tile,
        i. Update display board
        ii. Set current coordinate as new coordinate
    d. If player moved to fake tile,
        i. Reset board
```

main(). This function orders the subfunctions of the game and runs the game.
It contains a valid board generation loop:

```
1. While there is no valid board,
    a. Generate board
    b. If board has valid path, there is valid board
```

It orders the functions by:

```
1. Choosing game mode. If arcade:
    a. Ask player for difficulty level
    b. Valid board generation loop
    c. Gameplay logic
2. If story mode:
    a. Set easy board
```

b. Valid board generation loop
c. Gameplay logic
d. Set normal board
e. Valid board generation loop
f. Gameplay logic
g. Set hard board
h. Valid board generation loop
i. Gameplay logic