SC02 Group E Algorithm Maze Game Documentation

Members:
Tan Jasmine/1005312
Tay Jia Jiun Joshua/1005904
Nathan Aldrich Wiryawan/1006275
Ng Sue Chi/1006112
James Leo Wei Shaun/1005997

Description

Scenario. This game is aimed at school students, especially primary and secondary school students, who want to get a headstart in learning the basics of programming but are intimidated by a text-based interface, such as that of common programming languages on text editors or IDEs. As digitalization has gotten more ubiquitous in our lives, we realized the importance of embedding the ideas and logic of computational thinking from an early age.

Description of the game. In this game, the player has to solve a maze by moving a certain object (player object) to another end goal object (treasure object). However, instead of directly moving the player object, the player needs to queue the commands and press an execute button before the player object can move based on the commands. Therefore, it teaches the person playing the game to think systematically and algorithmically while planning to give instructions to the object. If the player object hits a wall object or doesn't reach the treasure object, a message will pop up saying that the player hit a wall or didn't reach the treasure, and the level repeats.

Documentation

Libraries required for this game are:

- turtle
- tkinter
- math
- time

Window Creation:

The window is created from tkinter's Tk() function, which creates a window. Within the window, we have a few essential components: the

main canvas where the game will run on, the buttons where the player may give commands to the turtle and the canvas where the commands will be displayed. When the RawTurtle objects (Pen, Player, Treasure) are created, they are instantiated inside the main canvas. When the CommandPen RawTurtle object is created, it is instantiated inside the commands canvas.

Game Objects:
There are four types of game objects – wall, player, treasure, and icons – which have different instantiation properties. Therefore, we use different classes for each of them. Since each of them requires the usage of methods of the **Turtle** class, thus we inherit **turtle.Rawturtle** in the creation of each class.

Pen - Wall Creation

- Instantiated with:
  - Square shape
  - Blue color
  - turtle penup method
  - 0 turtle speed
- It contains no methods as this object just needs to be stationary.

Player

- Instantiated with:
  - Original turtle shape
  - Red color
  - turtle penup method
  - 0 turtle speed
  - An empty list of commands - self.commands = []
- Methods:
  - forward(self) - append self.commands to move player object turtle forward
  - turn_left(self) - append self.commands to turn player object turtle 90 degrees to the left
  - turn_right(self) - append self.commands to turn player object 90 degrees to the right
  - start_loop(self) - append self.commands as indication of a command looping start

    - end_loop(self) - append self.commands as indication of a command ending start
    - is_collision(self, other) - a method to validate whether an the player object clashes with each other or not, returns boolean value.

Treasure

- Instantiated with:
    - Circle shape
    - gold color
    - turtle penup method
    - 0 turtle speed
    - Coordinates of (x,y)
- Methods:
    - destroy(self) - destroy the treasure object

CommandPen

- Instantiated with:
    - Square shape
    - yellow color
    - penup turtle method
    - 0 turtle speed
- Methods:
    - frontstamp(self) - creates a yellow triangular icon facing upwards
    - leftstamp(self) - creates a blue triangular icon facing the left
    - rightstamp(self) - creates a red triangular icon facing the right
    - startloopstamp(self) - creates a black arrow icon facing the left
    - endloopstamp(self) - creates a black arrow icon facing the right
    - newline(self) - moves the icon to a new line

Functions:

setup_maze(level)

- Each level has been previously stored in a list containing equal length strings which consists of “0”, “P”, “T”, “ ”. Each of those characters represent the location of the wall, player, treasure, and empty space respectively.

- As the entries in the list is a multi-length string, we can also see it as a two dimensional list if we want to call each of the characters respectively. Hence, we use a nested-loop to see each entry.
- If the entry is “0”, then the instance of Pen() will stamp a wall at that location. If the entry is “P”, then the instance of Player() will go to that location. If the entry is “T”, then we will instantiate Treasure(x,y) in its given coordinates.

show_commands()

- Player class was given self.commands as a list containing the string of commands we want and a few other methods to append the commands into this list.
- This function serves to display the icons used as the commands.

execute_commands()

- Function reads the entries of self.commands in Player class. If it is ‘f’, it moves forward, relative to the direction it is looking at. If it is ‘tr’, it will turn right 90 degrees. If it is ‘tl’, it will turn left 90 degrees. If it is ‘sl’, its index and the other index with ‘el’ will be searched and it will run a function loop_func(sl_idx, el_idx).
- This function also contains the Popup messages if the player object doesn’t collide with the treasure object or it collides with the wall object. It will give a message showing that the player made a mistake and repeat the level using repeat_maze() function.

loop_func(sl_idx, el_idx)

- It stores the entries between the index and runs it once with the same conditions as execute_commands() but will only read ‘f’, ‘tl’, ‘tr’. Therefore, it won't be an infinite loop.

repeat_maze()

- As the name states, it regenerates the current level.

clear_commands()

- Resets the self.commands list in the player object to be empty again.
- It also clears the pen stamps in the command screen window.

next_level()

- Goes to the next level if the treasure is already collected.

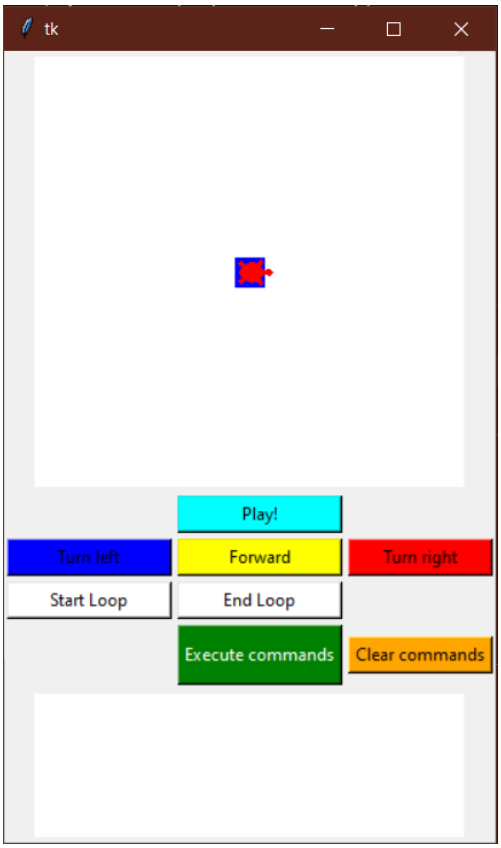
Other Components

Main game loop:

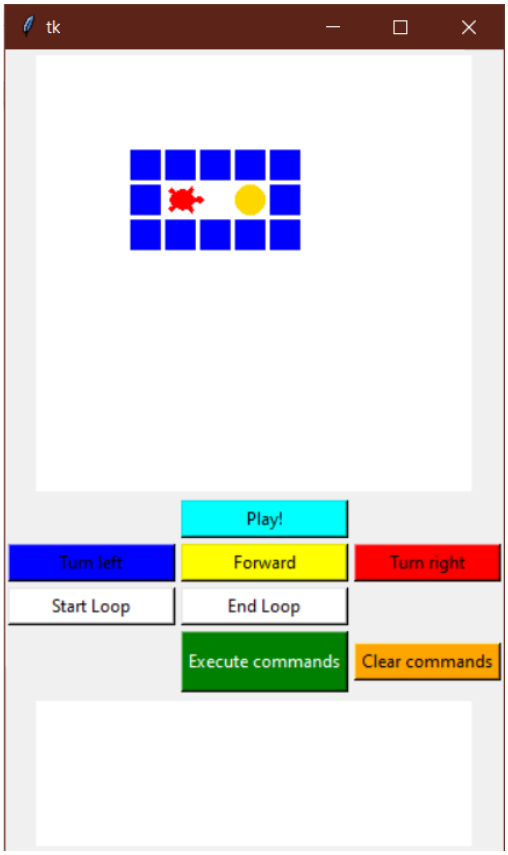
1. Player renders the level by pressing the play button
2. Player issues commands for the turtle to follow by pressing the buttons (Forward, Turn left, Turn right, Start loop, End loop)
    a. For each command pressed, the respective command will be added to the commandslist list.
3. If the player makes a mistake, he may press “clear commands” to restart the commands.
4. Player presses “Execute commands”
5. For each command in the list of commands given by the user:
    a. If the command is to move forward, the turtle will move forward unless there is a wall in front of it
    b. If the command is to turn, the turtle will rotate
    c. If the command is to start a loop, the turtle will execute the loop_func()
    d. If the turtle attempts to move forward into a wall, the level will be restarted.
6. If the turtle hits the treasure, the level is completed and the player can advance to the next level.
7. Otherwise, the level will be restarted.

How to Play

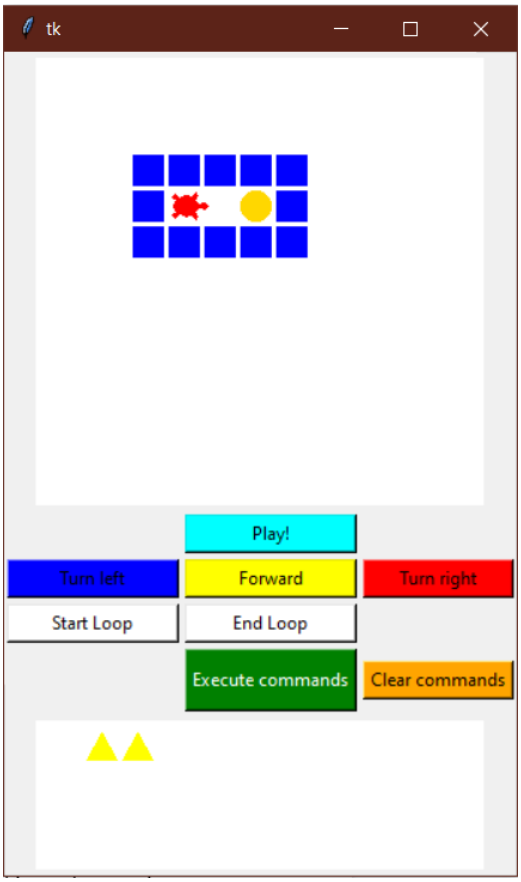
Right as we start the code this will appear. There are no mazes, so we have to click “Play!” to start the maze.



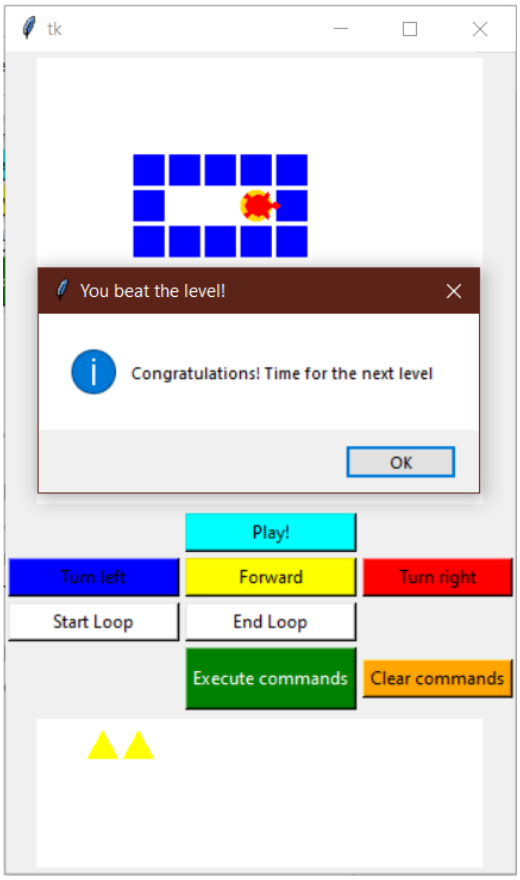
As soon as the “Play!” button is pressed it will generate a maze starting from the first level.



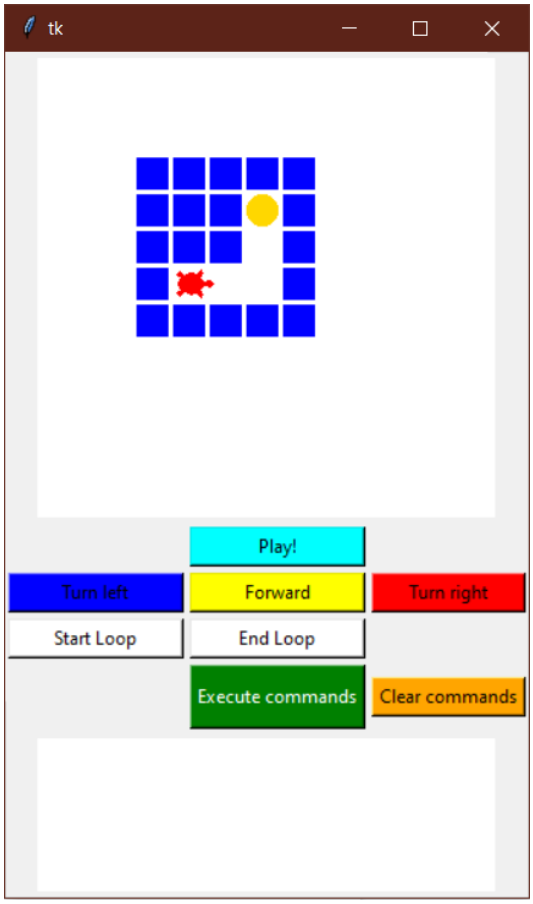
The player should then key in the commands for the player object (red arrow) to reach the treasure object (gold/yellow circle). In this case, it is the “Forward” button twice.



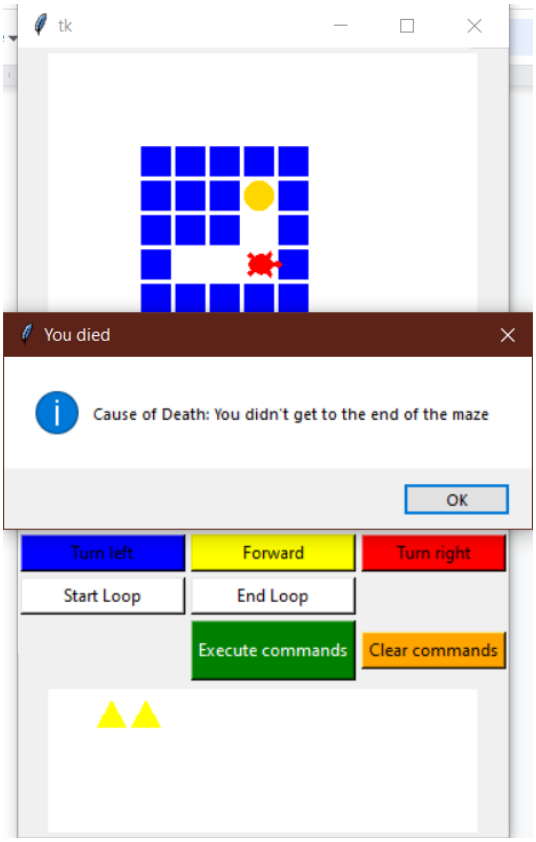
Then, the button “Execute commands” should be pressed. When it is pressed, the player object should move according to the commands the player has keyed in.



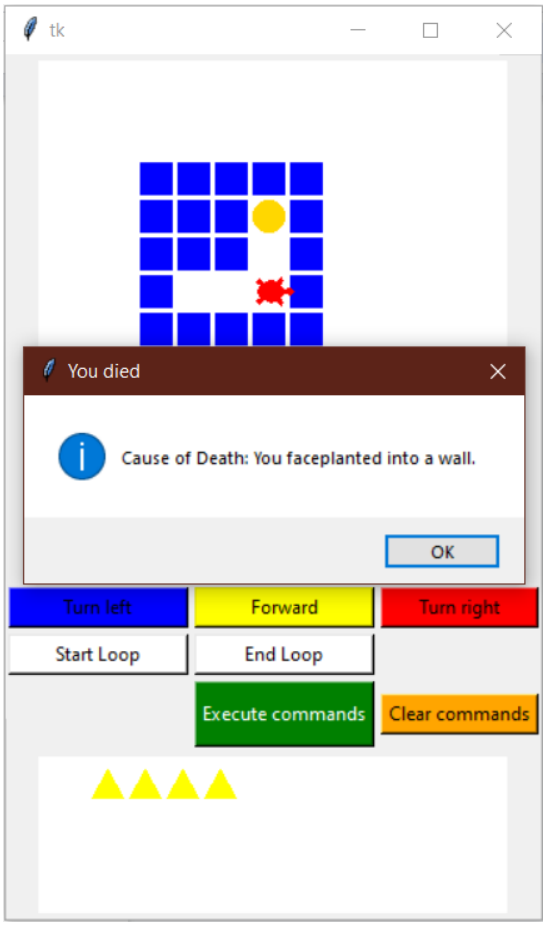
When the player object reaches the treasure object, a congratulation window will pop up, meaning that the player has succeeded in this level. Then, after pressing “OK” the game will proceed to generate the next level.



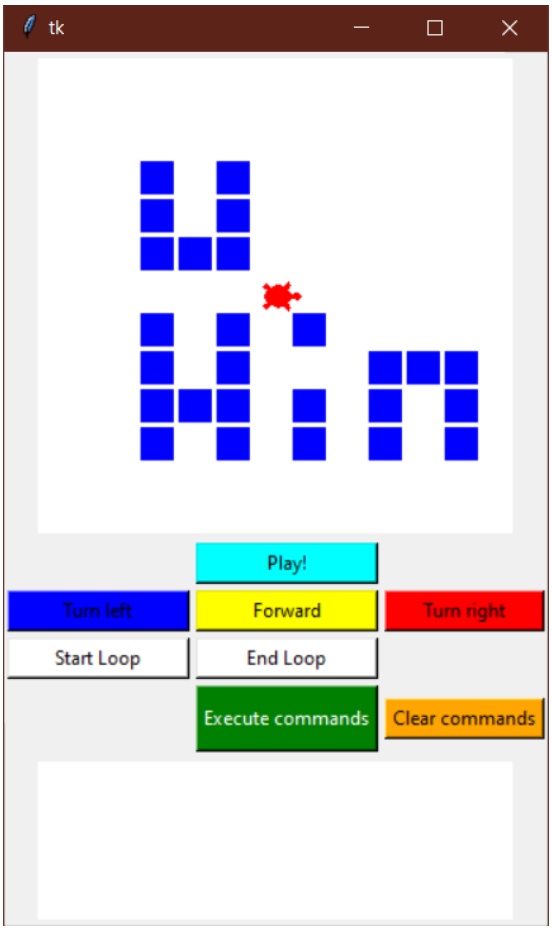
Every level has the same goal, which is to get the treasure, but is designed differently so that the player can experience different difficulties as the game goes on.



If the player keyed in the wrong commands and doesn’t reach the treasure, a window saying that the player doesn’t hit the treasure will appear and the maze would reset by regenerating the whole level once again.



If the player keyed in the wrong commands and hit the wall, a window saying that the player hits a wall will appear and the maze would reset by doing the same mechanism as the previous one.



Keep playing the levels until you reach this level which states that you have won the game (until we give more updates to the levels)