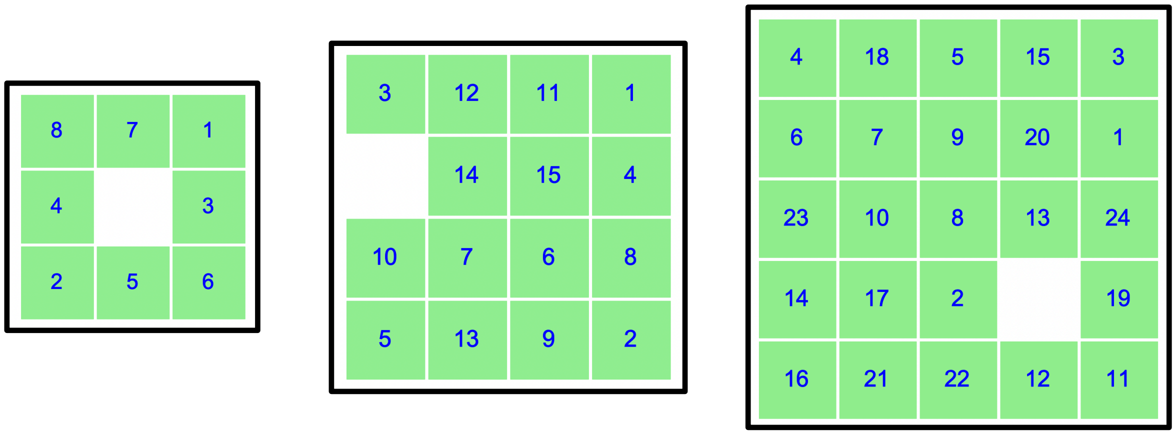
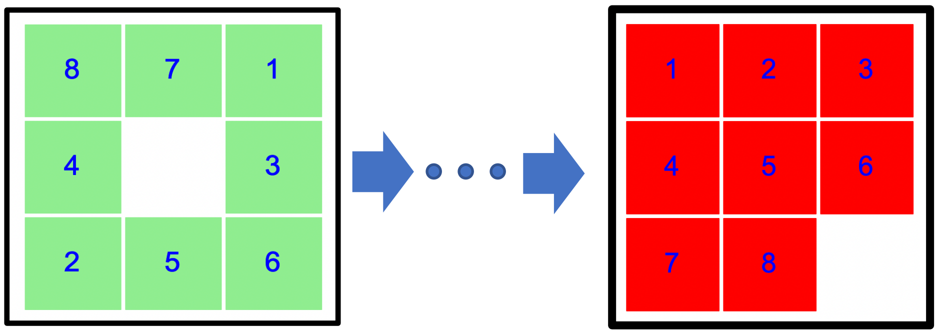
Sliding Puzzle GUI - 2024



OVERVIEW

In this assignment, you are going to design and develop a GUI based sliding puzzle game using Turtle Graphics for three different sizes: 3x3, 4x4 and 5x5 as depicted above. The objective of this game remains the same as the console-based version of Sliding Puzzle. However, the interaction between the game and the player is based on the Mouse Click events and the player is not prompted to enter the move. Instead, the player simply clicks one of the tiles adjacent to the empty white space to select the move.

The figure below illustrates an 8-number puzzle, with "INITIAL" as the starting point of the game. The player must slide one adjacent tile at a time into the unoccupied space (empty space) repeatedly until all numbers are in sequential order from left to right, top to bottom, as shown in “FINAL".



INITIAL **FINAL**

SCOPE

1. At the start of the game, prompt the user to enter the size of the game to play: 3, 4, or 5.
    a. use Screen's numinput() to prompt the user (see Sample Output)
2. After successfully completing the prompt, generate a randomized, SOLVABLE puzzle accordingly and display it on the screen using Turtle graphics. Display each tile as a square shape with a minimum width of 80 pixels.
3. Wait for the mouse click event from any of the tiles adjacent to the empty white space.
4. Animate the movement of the chosen tile towards the empty white space. The number on the tile will be displayed after the animation is completed. Essentially, while the tile is in motion, the number is not visible.
5. When the puzzle is solved (i.e. the numbered tiles are in sequential order, left to right, top to bottom), change the background color of all tiles to red as shown in the figure in the overview section and leave the program in running mode.
    a. You are not required to prompt the player to start another new game.
6. Choose one color for all the numbers displayed on the tiles, ensuring that the selected color is not the background color of the Turtle's Windows. A penalty of 10% to Functionality will be imposed if the Window's background color is used.
7. Use the standard turtle square shape to represent the tiles without incorporating custom, external image files.

NOTE:

- Keep your entire source code in ONE SINGLE file.
- Use only python modules as specified in "Permitted Modules" session.
- In your design stick ONLY to functions, in other words, no class objects of your own.
    - Furthermore, the lines of code containing the sub-function(s) defined within another function will be counted as part of the parent function.

STARTUP OPTIONS

Not applicable

SKILLS

In this assignment, you will be trained on the use of the followings:

- Refactoring - logic reuse based on the previous assignment.
- Problem Decomposition, Clean Code, Top-Down Design
- Functions (with parameters and return) for program structure and logic decomposition
- Standard objects (strings, numbers & lists)
- Variable Scope
- GUI-based programming: mouse click event, callback, asynchronous-style programming
- Turtle Graphics - a basic graphics drawing tool utilized as the building block for creating the visual representation of the puzzle game.

PERMITTED MODULES

Only the following python modules are allowed to be used:

- random, turtle, math, numpy, functools.

DELIVERABLES

1. Program source code (A2_School_StudentID.py)

where School is SSE, SDS, SME, HSS, FE, LHS, MED and StudentID is your 9-digit student ID.

Submit the plain (zip not required) python file by due date to the corresponding assignment folder under "Assignment (submission).

For instances, a SME student with student ID "119010001" will name the python file as follows:

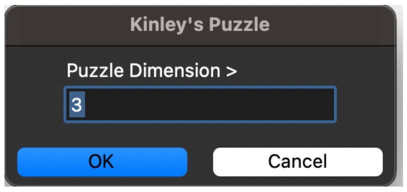
- A2_SME_119010001.py:

5% will be deducted if file is incorrectly named!!!

TIPS & HINTS

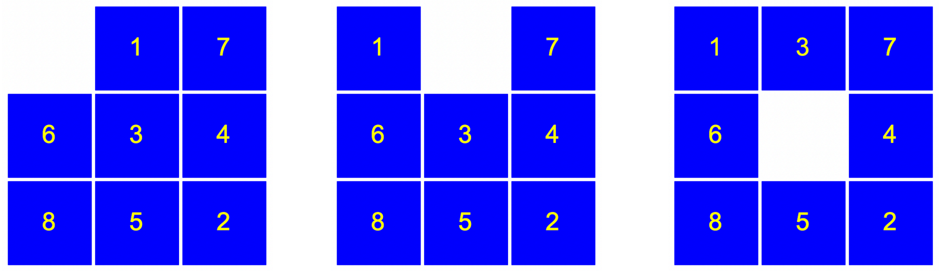
- Beware of variable scope as you might keep a few variables as global such as puzzle
- Refer to python website for program styles and naming convention (PEP 8)
- Use turtle objects to layout the tiles
- Use onclick() to capture the mouse click event for each tile
- Pair each tile with separate turtle object to write the number on each tile
- Review the lecture slides on “Auto vs Manual” screen refresh for both turtles and text display
- Use goto() to animate the movement of the turtle instead of forward() or backward() to avoid changing of turtle’s heading.

SAMPLE OUTPUT


Kinley's Puzzle

Puzzle Dimension >

3



MARKING CRITERIA

- Coding Styles – overall program structure including layout, comments, white spaces, naming convention, variables, indentation, functions with appropriate parameters and return.
- Program Correctness – whether or the program works 100% as per Scope.
- User Interaction – how informative and accurate information is exchanged between your program and the player.
- Readability counts – programs that are well structured and easy-to-follow using functions to breakdown complex problems into smaller cleaner generalized functions are preferred over a function embracing a complex logic with many nested conditions and branches! In other words, a design with clean architecture with high readability is the predilection for the course objectives over efficiency. The logic in each function should be kept simple and short, and it should be designed to perform a single task and be generalized with parameters as needed.
- KISS approach – Keep It Simple and Straightforward.
- Balance approach – you are not required to come up a very optimized solution. However, take a balance between readability and efficiency with good use of program constructs.

ITEMS	PERCENTAGE	REMARKS
CODING STYLES	20%	0% IF PROGRAM DOESN'T RUN
USER INTERFACE	20%	0% IF PROGRAM DOESN'T RUN
FUNCTIONALITY	60%	REFER TO SCOPE

DUE DATE

April 2nd, 2024, 11:59:59PM