Ashton McBride

Professor Sipes

Game Scripting

2 May 2025

Python Post Mortem

While creating my final project in Python, there were a few things that went right and wrong. I found coding in a language that I am not very familiar with can be a very effective and worthwhile learning experience. One thing that went really well was coming up with the idea for the project. After spending a lot of time solving slider puzzles lately, I knew immediately that I wanted to try programming one. I ended up utilizing a tutorial from a book titled *Making Games with Python & Pygame* by Al Sweigart. There are a lot of in-depth tutorials in this book that cover the pygame library and include source code for many different types of games, making it easy to follow and understand how a programmer would create them.

One obstacle I faced during the project was deciding how to apply pieces of code that I learned in class with code I was referencing from the tutorial. I was used to having a main while-loop in previous assignments, while this tutorial treated main as its own function. I wanted to add a title screen and a win screen, but I decided against it because there was no clean way to do so without messing up the structure of the program. I also felt that having a win screen in this game did not feel entirely necessary. I wanted the player to move to the next puzzle immediately after solving one in order to keep them engaged and motivate them to keep going. If I were to improve upon this game in the future, a main menu screen would definitely be something I would add.

An important lesson I learned from this project was understanding how even the simplest ideas like a slider puzzle can have differing levels of complexity and often require additional parts that can easily be overlooked. For example, the board has to start in the solved state every time it is generated in order for the game to know what combination of tiles would result in a win. It creates two board data structures, one representing the current game state and another representing the tiles in the “solved” state. The slider puzzle program also has to keep track of the direction the player wants a tile to move in and validate every move they make. Another lesson I learned was to put myself in the player’s shoes. The player should be able to close the game at any point they want. The player should also only be able to move tiles that are next to the empty space. Small details like these can add up into a larger, more complex system.

In conclusion, I enjoyed programming in Python. Comparing this language to ones like Lua was exciting to me. I liked being able to expand more on the slider puzzle tutorial and add my own spin on it in order to demonstrate my understanding of how the program worked. A few things I added myself were different board sizes that change after solving each puzzle and a counter that increases every time the player moves a tile. It is interesting seeing how increasing or decreasing certain values in a program affects the user experience. For example, I decreased the amount of time it takes to move a tile and the amount of time it takes to generate a puzzle, which made the puzzles much more pleasant to solve. I enjoy having the opportunity to mess around with all kinds of values until I get a program to perform exactly how I want.

Works Cited

Sweigart, Al. *Making Games with Python & Pygame.* Al Sweigart, 2012. https://inventwithpython.com/pygame/chapter4.html