Nick Crescentini

SNHU - CS499

Milestone Three – Enhancement 2 Narrative

1. **Briefly describe the artifact. What is it? When was it created?**

This artifact is the final project I submitted for IT-140 in August 2023. The program is a text-based adventure written in Python where the player explores a castle to collect items that will help you defeat the dragon hiding somewhere within. While this program is relatively simple, it has useful features such as navigation, an inventory system, win/lose conditions, and more.

1. **Justify the inclusion of the artifact in your ePortfolio. Why did you select this item? What specific components of the artifact showcase your skills and abilities in software development? How was the artifact improved?**

The main enhancement was the replacement of the existing item detection logic, which used a long chain of if/elif statements to determine if an item is in the current room. While this system is functional, it is inefficient, difficult to maintain, and difficult to scale. Adding new rooms would require multiple updates to the logic, and moving multiple items to different rooms would require a significant effort.

These limitations were addressed by refactoring the item detection logic to use a new algorithm that leverages the existing rooms dictionary. This logic can dynamically check for items in the room and against the player’s inventory. This system is more efficient, much simpler to maintain, and allows for easy scalability while reducing the entire logic to just a couple of lines. Adding 10 new rooms is now as simple as updating the dictionary.

This artifact demonstrates the improvements I’ve made in writing efficient and maintainable code since beginning the computer science program. It also highlights the progress I’ve made understanding algorithms and data structures. Finally, it showcases my ability to find weaknesses in existing code, design a clean solution, and implement the improvement successfully.

Additional minor improvements:

* Reformatted the rooms dictionary for enhanced readability and improved maintainability

Before:

A screen shot of a computer

AI-generated content may be incorrect.

After:

A screenshot of a computer program

AI-generated content may be incorrect.

* Added a new function to display a victorious knight using ASCII art if the player meets the win condition. This does not change functionality but adds to the user experience with a new end game reward that differs from the lose condition.

A screenshot of a computer screen

AI-generated content may be incorrect.

1. **Did you meet the course outcomes you planned to meet with this enhancement in Module One? Do you have any updates to your outcome-coverage plans?**

Outcomes 3, 4, and 5 were what I had originally planned, and I believe I met that goal. Outcome 3 was met by refactoring the inventory system with a dynamic algorithm solution to improve efficiency and address the issues of scalability and maintainability. Outcome 4 was met by demonstrating my skills in the implementation of this solution, delivering value by identifying an issue, designing a solution, and successfully refactoring. Outcome 5 was met by simplifying the item detection logic, thereby reducing the likelihood of logic errors during manual updates. This aligns with a security minded approach as errors in logic can potentially lead to unpredictable program behavior, exploits, or vulnerabilities.

1. **Reflect on the process of enhancing and modifying the artifact. What did you learn as you were creating it and improving it? What challenges did you face?**

As I worked through this enhancement, I was reminded of my very first SNHU class, and the hours spent trying to get this program to work. I’ll never forget lying in bed and suddenly having an “aha!” moment, when a solution for my inventory problem popped into my head. It was a moment in my learning journey that gave me such a sense of accomplishment and one that I will not soon forget. Now, in my final term, that same code block that I once thought was so clever immediately stood out as needing to be refactored.

This process also gave me greater appreciation for the different approaches needed for writing new code and those needed for refactoring existing code. With new code you can plan and design from the ground up. When refactoring, you need to design a solution that fits within the existing framework and any existing errors or limitations.

This leads me to another lesson learned, the importance of well-designed comments. It had been almost two years since working with this code, so having the comments to help refamiliarize myself with the codebase was invaluable and allowed me to dive right back in. For programs that are much larger, more complex, and have multiple developers working on them, the value of efficient commenting is even greater.