Victor Zuniga Salgado

CS499

Professor Bryant

11/22/24

Enhancement Two: Algorithms and Data Structure

For Enhancement Two: Algorithms and Data Structure I chose to use a text-based game titled “Battle at Mystery Manor Text Adventure Game” that I created for my final project in my IT-140 Intro to Scripting class. IT-140 was the first class that I took after transferring to Southern New Hampshire University. Prior to this, I had not been in college for two years. There was a lot of information to take in, but I enjoyed what I learned as time went on. Python was the first programming language that I got to use, and it’s remained my favorite programming language this whole time. The text-based game is an adventure-based game that contains different rooms that each contain an item. Players will need to collect every single item prior to reaching the final room to defeat the boss and win the game.

The ”Battle at Mystery Manor Text Adventure Game” is an excellent artifact to include in my ePortfolio because it highlights several key aspects of software development that showcases my creativity, problem-solving abilities, and a great understanding of programming concepts. The artifact demonstrates proficiency in Python such as the use of functions, dictionaries, loops, and conditionals. Demonstrates the ability to build a user-focused application with an emphasis in engagement and usability. It also demonstrates my ability to think creatively and solve problems effectively by designing a room-based game world that has specific rules and objectives.

Enhancements were made to the original artifact to demonstrate a well-rounded application of algorithms, data structures, and software design principles. Data structures are used effectively through the use of dictionaries such as “RoomGraph” which stores rooms and their connections as well as “Room” objects. Queues such as “deque” implements the shortest path search using the breadth-first search (BFS). The shortest path function (shortest\_path) uses BFS to navigate the graph representation of the game map, showcasing knowledge of graph traversal algorithms.

Implementation of BFS also demonstrates algorithmic thinking. The priority queue for handling game events adds complexity and demonstrates the ability to manage time-sensitive or priority-based events in real-time systems. The (deque) data structure provides quick append and traversal functionality, simulating a linked list for item history management. Modular code with classes (Room, RoomGraph) encapsulates functionality and promotes reuse and scalability. The use of methods like (add\_neighbor) and (show\_item\_history) showcases a thoughtful, reusable API for interacting with objects. Other enhancements include added commands for checking inventory, room item history, and the shortest path to specific goals (ex., "Library"). Optimal data flow through: Ensured that room connections are bidirectional for a realistic map experience.

Reduced unnecessary complexity by leveraging built-in data structures like (deque) and (heapq). The inclusion of meaningful algorithms and data structures underscores proficiency in foundational skills critical to advanced software development.

The course outcomes that enhancement aligned with was: **Demonstrate an ability to use well-founded and innovative techniques, skills, and tools in computing practices for the purpose of implementing computer solutions that deliver value and accomplish industry-specific goals. Design and evaluate computing solutions that solve a given problem using algorithmic principles and computer science practices and standards appropriate to its solution while managing the trade-offs involved in design choices.**After reviewing the enhancements made I’m certain that the enhancements made meet the course outcomes that were planned.

Enhancing this artifact reinforced my understanding of data structures and algorithms. Creating an intuitive command-based interface taught me the importance of considering user interaction and experience. I realized the importance of clean, modular code. Encapsulating functionality within classes and methods made it easier to add features like item history or new room connections without introducing errors or redundancy. There were some challenges that I encountered when make the enhancements. First, implementing BFS functionality into the game broke the game at first, so I had to break the algorithm down into smaller tasks while testing each step to ensure correctness. It’s interesting to see how differently the game could be built or expanded which I didn’t think was possible when I initially built it.