Task 9 - Spike: Game Data Structures

CORE SPIKE

Context: Game developers will often encounter a variety of different types of data and access/usage scenarios, even in a single project. It is essential, therefore, that developers be aware of the different data types available to them. In particular they should know what collection types are available, the advantages and disadvantages, and their suitability for different jobs in a game project.

Knowledge/Skill Gap: The developer is not familiar with common data types and collection types, their various strengths and weaknesses, and their applicability in common game scenarios.

Goals/Deliverables:
[CODE] + [SPIKE REPORT] + [SHORT REPORT]

1. Research and evaluate four different data structures that could be used to create the player inventory for the Zorkish game. At a minimum, you must show your awareness of advantages and disadvantages for this application. Document your evaluation criteria and results in a **short report**.

2. Using your decision (as documented in your short report), create a working inventory system demonstration program. Your work must demonstrate (bug free) inventory **access** (view), **addition** and **removal**.

Note: The short report is separate from the spike report. Ask your tutor about this if you aren't sure what to put in each. Your work won't be accepted unless you have all deliverables ready.

Note: All work should be in your repo! Upload the spike report (pdf) and the short report (pdf) to Doubtfire.

Recommendations:

- A nice overview of different data structures is available in the C++ STL is here: http://en.cppreference.com/w/cpp/container. You may use other libraries, but the STL is recommended.
- Keep the short report SHORT - very focused and concise. No padding or fluff!
- If you do not yet have a specific Player Character class/object for Zorkish, now might be a great time to create one. (The inventory is typically a part of the player.)
- Do NOT over-engineer this! You do NOT need a command processor for this.
- Your program can be non-interactive, and simply execute the operations required in the program main() and show simple text output as a result.
- Separate demo or Zorkish add-on? Start with separate. (Only add it in to Zorkish when it's needed.)

Tip: Your code does NOT need to be part of the other Zorkish code that you've created so far (previous spike). It might actually make your spike code for this task less concise and less clear. So, if you do include it to your existing code, refrain from adding any distractions – keep the "spike" work pointy and focused on the "gap"!