Task 14 - Spike: Command Pattern

CORE SPIKE

Context: A text-based adventure game can create an immersive experience where a player feels like the game "understands" them. One part of this is a robust way to process user input (text commands) and turn them into game world actions. The skills used to create good parsers and management of commands are also very useful in many other games and software projects.

Knowledge/Skill Gap: The developer needs to know how to create, for a text-based adventure game, a text command parser that is robust and accepts typical human variations. It should include an effective way to design, manage and extend commands.

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

Building on the work of earlier Zorkish tasks, create a modular and robust text command parser and command manager for the game suitable for the Zorkish: Phase 2 game. (Refer to the game specification document on the subject website for details).

The commands are limited (do not change the game world at all), but are an essential building block for the Zorksih game engine. Your spike outcomes must be able to:

1. The loading of adventure files (text format?) that includes (partial support) of the game locations and some game entities. (Commands will look at but will not move or change entities).
2. A robust command processor support a minimum of the following commands
    a. HELP (lists known commands and their syntax details),
    b. INVENTORY (what the play has),
    c. LOOK, LOOK AT (but not LOOK IN yet)
    d. ALIAS (to remap commands)
    e. DEBUG TREE (of the game graph world).
3. A UML diagram that matches your final command pattern-related classes. Include ths in your spike report.

Recommendations:

- Use an OO command pattern.
- Read the game specifications again. Seriously. ☺
- Research the "command pattern". (You'll probably want Command objects and a CommandManager that can "run()" each command object when needed.)
- Research "maps" or "dictionaries" – collections that can access contents using string keys. (STL)
- Put designs and plans on paper. Think as much as possible before you code! (If you do this, be sure to include your paper design with your outcome report.)
- Create a new adventure file that contains a minimal game world description and some game entities.
- Update the adventure loading code (which you should already have) so that your game world (a graph) supports entities (like rocks, etc).
- Implement a simple command (but using the command pattern/manager) and test.
- Implement other commands – and test… extend… until done.