Task 3 - Spike: GridWorld

CORE SPIKE

Context: Games are commonly driven by some form of game loop.
Knowledge/Skill Gap: The developer is not familiar with basic game loops and using a code repository.

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

You need to create the "GridWorld" game, as described in a simple specification document available on the unit website. The game will demonstrate the use of a simple game loop, separation of update/render code, and use of game data (both for the world map and players current location).

You will need to deliver the following items:

1. A simple plan (sketch or structure design) for your code design. (Yes, a simple functional design is fine – just as long as you can demonstrate that you did actually plan first before coding!)
2. Create a simple console program, using C++, that implements the "GridWorld" game using a simple game loop. The game must demonstrate the separation of:
    a. processing of input (text commands from the player),
    b. updating of a game model (where the player is and their options),
    c. display (output current location and options) of the game to the user.
3. Evidence in your repository of incremental code commits as you progress with this task. Do this by including a screen capture in your spike report of your commit log summary for this task.
4. Spike Outcome Report. See the spike report template for details.

Notes: Point 1 will not be included from now on it is always required. Point 5. "Spike Outcome Report" is always a required as a "deliverable" for a spike task. It will not be repeated in future spike tasks as it is assumed.

Recommendations:

- Read the "GridWorld" game details and do a quick sketch/design of how you will organise your code. (No formal design standard specified… just use something that would work to help you explain your design to another programmer.)
- Look at the Spike Outcome Report template – note what you need to record for later.
- Use an IDE for C/C++ development that you already know. (We'll look at Visual Studio later so don't get distracted from the main point of this spike!)
- Begin small, test often. Use simple print-outs to check values (or the debugger if you already know that and are comfortable). Don't try fancy debugging yet if you don't know it – that's a later spike if it's something you need to work out. Stay on target!

Tips:

- Consider a DEBUG macro condition if extra code is needed for testing purposes.
- You do not need to show the "map" or the current player location – but it is useful.
- If you find some useful resources that helped you to get your code working, then remember to note them down and include them in your spike outcome report. Google, books, blogs, classmates, etc. all go in the spike report.
- Have a plan for when you present your outcomes to the tutor. (They will probably say something like "Right – show me"… and it's up to you to show-off what you've done.)
- Many of these tips also apply to spikes in general and are not exclusive to this spike.

DO NOT

- Make things any more complicated than you need – just get it working.
- Create complex data-structures. A simple 2D array is fine (and expected).
- Add all the fancy features you can think of! Save ideas for later and perhaps develop them as a portfolio item. (You should still document/note them down though.)
- Load the map data from a file. Simply hard code the map data.