

Task 8 - Spike: Game State Management

CORE SPIKE

Context: Game state management is a common feature of games.

Knowledge/Skill Gap: The developer is not aware of implementation methods for flexible game state (“stages” of a game) management.

Goals/Deliverables:

[DESIGN DOC] + [CODE] + [SPIKE REPORT]

You need to create the “Zorkish Adventure” game (Phase 1), as described in the specification document available on the unit website.

You will need to deliver the following items:

1. A simple documented plan for your code design. **(REQUIRED!)**
2. Create a simple console program that implements the “Zorkish: Phase I” game using a flexible (extensible) game state management method of some kind. It should have no gameplay yet! The OO State Pattern is the strong suggestion. The implementation must demonstrate the following game stages (states):
 - a. “Main Menu”: Allows the user to select other stages.
 - b. “About”: Remember to include your own details here.
 - c. “Help”: List a summary of commands – simple hard-coded text is fine.
 - d. “Select Adventure”: Use a hard-coded list and the title of your test game.
 - e. “Gameplay”: Placeholder only. A test “stage” which only accepts “quit” and “hiscore” commands.
 - f. “New High Score”: Allows user to enter their name, but doesn’t work (save details) yet.
 - g. “View Hall Of Fame”: Shows a list of name/score. Simple hard-coded text is fine.

NOTE: You must have a documented plan. You can do this on paper and include a photo in your design document. A strong suggestion is a UML class diagram representing a state pattern you would need specific to the Zorkish game as you are doing it. If your final implementation different from the initial design that is okay. Make a note about it.

Recommendations:

- Read the complete Zorkish game specification document.
- If not familiar (or you need a reminder) research/read about the state pattern used to represent each “stage” of the game.
- Use an “agile” (test and commit often) approach as you go. Don't just commit once at the end when all work is finished. For example, implement a single state then test and commit. Next add another state and test changing between the two, and commit. Repeat until the minimum work to complete the spike it done.
- Do NOT overengineer this! (Leave fancy extensions and complete features for extension work.)
- If possible, leave complex issues or issues that might distract from the essential “core” until last.

NOTE: Stay focused on the main points of this spike – state management! Do **NOT** implement a complex gameplay, a command parser, the “Hall of Fame” file IO, scoring features etc. Compare Phase 1 and Phase 2 and read later spikes to see why! Focus on the minimum to get this spike done! If you are not sure, check with your tutor.