



Task 08 - Spike Summary Report



Spike: Task_08

Title: Game State Management

Author: Thomas Horsley, 103071494

Goals & Deliverables

Aim: To implement Zorkish Phase I, as described in the specification document.

Deliverables:

- Design Documentation
- Functional Zorkish Phase I Solution
- Spike Summary Report
- Git Commit History

Technology, Tools and Resources

Tech and Tools



The project was scripted in C++ 20 using Visual Studio Community 2022.

UML's and charts are made with
www.Lucidchart.com

Source control is handled using Git.

Resources

- Lecture “*Game States and Stages*”
- Lecture “*Design Patterns Introduction*”

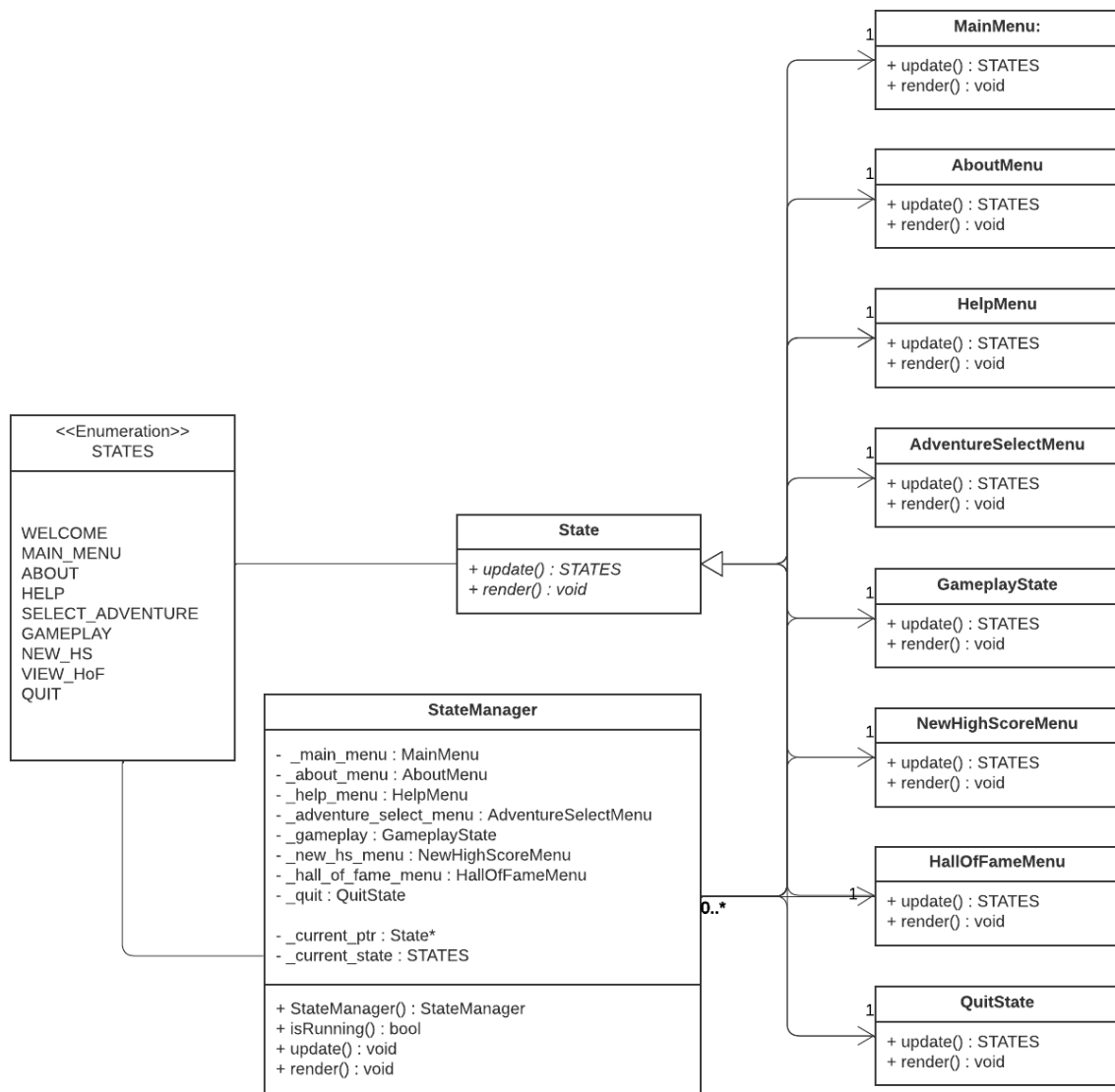


Task 08 - Spike Summary Report

Tasks Undertaken

Planning

Diagrams and Charts





Task 08 - Spike Summary Report

Class Descriptions and Notes

The Zorkish StateManager class was implemented using the state management pattern. An enum of states is contained within the StateManager header, each of which correlates with a State Object child class containing the relevant state functionality.

The StateManager will construct a single instance of each State child and (depending on the value of `_current_state`) will wrap the appropriate state functionality within its own `update()` and `render()` methods before presenting this to the `main()` method.

Implementation

Git Commit History

Commits on Aug 21, 2023		
Refactored the working inventory code	unknown committed yesterday	80df400
Added InventorySlot (Node) class.class	unknown committed 2 days ago	4c471dd
Commits on Aug 20, 2023		
Started transition from Vectors to Linked Lists.	unknown committed 2 days ago	dfc4074
Commits on Aug 19, 2023		
Started Task 09 - Spike 'Game Data Management'.	unknown committed 4 days ago	c9204ed
Started Task 09 - Spike 'Game Data Management'.	unknown committed 4 days ago	d274568
Commits on Aug 16, 2023		
Finished Task 08 - Spike 'Game State Management'	unknown committed last week	eb85a6f
Nearly finished, bug with menu's double rendering.	unknown committed last week	53dd9f7
Completed structuring of the OO Gamestate Handler	unknown committed last week	59d91e2
Commits on Aug 13, 2023		
started Task 08 - Spike 'GameState Management'	unknown committed last week	58f2676
Commits on Jul 30, 2023		
Initial Commit	KingSchlock committed 3 weeks ago	0d3f501



Task 08 - Spike Summary Report

Code Snippets

```
1 // Task 08 - Game State Management
2 // Author - Thomas Horsley (103071494)
3
4 // The state manager was built implementing the OO State pattern.
5 // See spike report! */
6
7 #include <iostream>
8 #include <string>
9 #include "StateManager.h"
10
11
12 int main() {
13     StateManager _game_manager;
14
15     while (_game_manager.isRunning()) {
16         _game_manager.update();
17         _game_manager.render();
18     }
19
20     return 0;
21 }
```

main() with an instantiated StateManager object

```
70 class StateManager
71 {
72 private:
73     MainMenu _main_menu;
74     AboutMenu _about_menu;
75     HelpMenu _help_menu;
76     AdventureSelectMenu _adventure_select_menu;
77     GameState _gameplay;
78     NewHighScoreMenu _new_hs_menu;
79     HallOfFameMenu _hall_of_fame_menu;
80     QuitState _quit;
81
82     State* _current_ptr = &_main_menu;
83     STATES _current_state = STATES::MAIN_MENU;
84
85 public:
86     StateManager();
87     bool isRunning();
88     void update();
89     void render();
90 };
91
```

StateManager definitions

```
1 #pragma once
2
3 enum STATES {
4     WELCOME,
5     MAIN_MENU,
6     ABOUT,
7     HELP,
8     SELECT_ADVENTURE,
9     GAMEPLAY,
10     NEW_HS, //New Highscore
11     VIEW_HoF, //View Hall of Fame
12     QUIT,
13 };
14
15 class State {
16 public:
17     virtual STATES update() = 0;
18     virtual void render() = 0;
19 };

```

STATES enum and State class definitions



Task 08 - Spike Summary Report

What was Learned?



This task involved creating a scalable state management system for the Zorkish Phase I prototype. The Spike involved manipulating custom classes and return types to effectively manage a state driven game loop. Additionally, a state driven design allows the developer to encapsulate specific state functionality within a manager object, further abstracting the management system from the user.
