Task 12 - Spike Summary Report

Spike: Task_12
Title: Command Pattern
Author: Thomas Horsley, 103071494

Goals & Deliverables

Aim: Demonstrate a functional command pattern within Zorkish by using commands to manipulate game data during.

Deliverables:

- Functioning code
- Spike summary report
- Git commit history

Technology, Tools and Resources

Tech and Tools

Resources

- Commands - Refactoring Guru: https://refactoring.guru/design-patterns/command
- The Factory Design Pattern
  See: https://www.youtube.com/watch?v=usmdZniV_Yw&t=714s

💡 The project was scripted in C++ 17 using Visual Studio Community 2022.

UML’s and charts are made with *www.Lucidchart.com*

Source control is handled using Git.

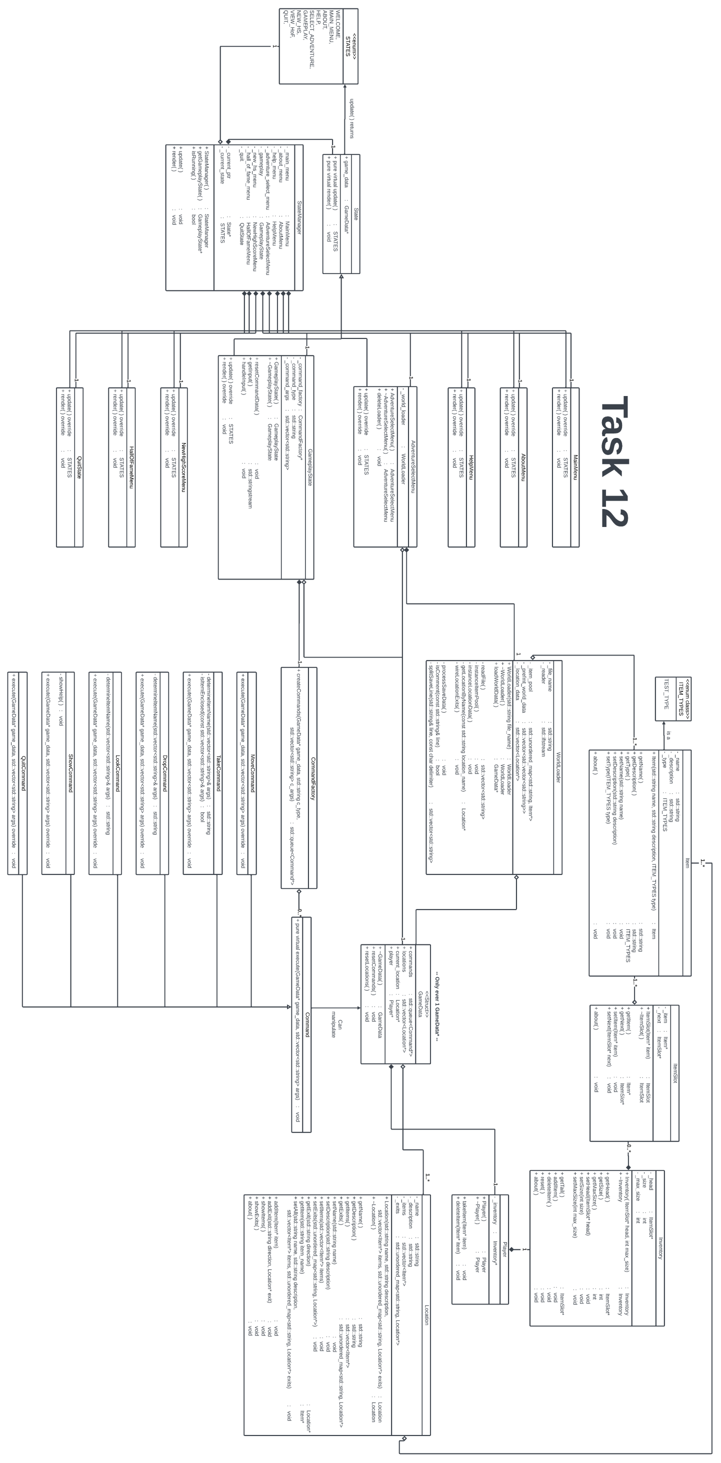
- The Command Pattern
  See: https://www.youtube.com/watch?v=yDkIK3JfHkw&t=1105s

Tasks Undertaken

Planning

Diagrams and Charts

Task 12


STATES
State
StateManager
MainMenu
AboutMenu
HelpMenu
AdventureSelectMenu
GameplayState
NewHighScoreMenu
HallOfFameMenu
QuitState
WorldLoader
ITEM_TYPES
Item
ItemSlot
Inventory
Player
Location
GameData
CommandFactory
Command
MoveCommand
TakeCommand
DropCommand
LookCommand
ShowCommand
QuitCommand

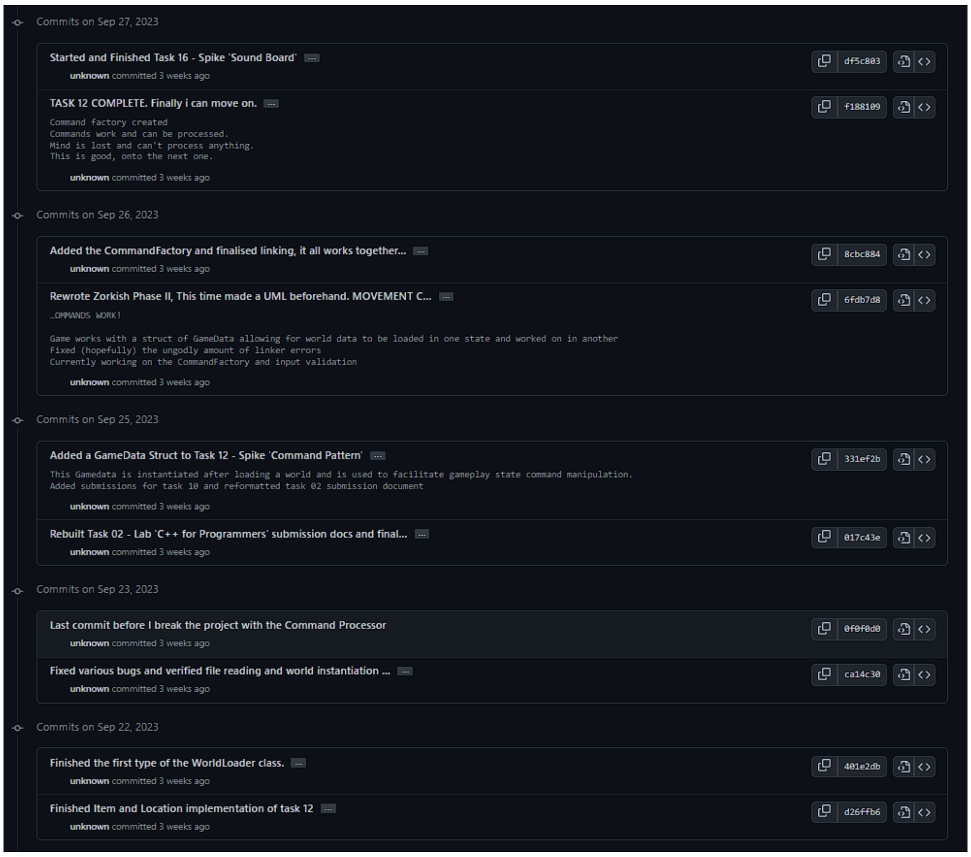
Class Descriptions and Notes

The Commands themselves are quite simple, each command is responsible for manipulating a specific element of the Game Data. The commands take a set of verified string arguments from the command factory (and in turn the WorldLoader object). The factories only responsible for returning a vector of commands (this will allow support for chain commands).

Our Game Data is struct contains a vector of location pointers, our current location pointer and a reference to the player. Additionally, the Game Data has the capacity to clean it’s memory once destructed. In Task 13, the Game Data is reformatted to a PoD struct who’s memory allocation and deallocation is handled through commands and it’s owning State child.

Implementation

Git Commit History



Commits on Sep 27, 2023

Started and Finished Task 16 - Spike 'Sound Board' ... df5c803
unknown committed 3 weeks ago

TASK 12 COMPLETE. Finally i can move on. ... f188109

```
Command factory created
Commands work and can be processed.
Mind is lost and can't process anything.
This is good, onto the next one.
```

unknown committed 3 weeks ago

Commits on Sep 26, 2023

Added the CommandFactory and finalised linking, it all works together... ... 8cbc884
unknown committed 3 weeks ago

Rewrote Zorkish Phase II, This time made a UML beforehand. MOVEMENT C... ... 6fdb7d8

```
…OMMANDS WORK!

Game works with a struct of GameData allowing for world data to be loaded in one state and worked on in another
Fixed (hopefully) the ungodly amount of linker errors
Currently working on the CommandFactory and input validation
```

unknown committed 3 weeks ago

Commits on Sep 25, 2023

Added a GameData Struct to Task 12 - Spike 'Command Pattern' ... 331ef2b

```
This Gamedata is instantiated after loading a world and is used to facilitate gameplay state command manipulation.
Added submissions for task 10 and reformatted task 02 submission document
```

unknown committed 3 weeks ago

Rebuilt Task 02 - Lab 'C++ for Programmers' submission docs and final... ... 017c43e
unknown committed 3 weeks ago

Commits on Sep 23, 2023

Last commit before I break the project with the Command Processor 0f0f0d0
unknown committed 3 weeks ago

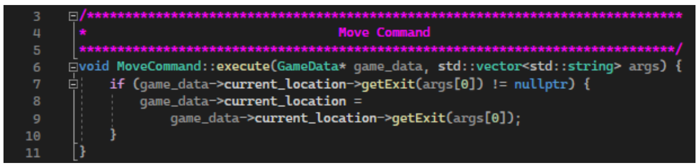
Fixed various bugs and verified file reading and world instantiation ca14c30
unknown committed 3 weeks ago

Commits on Sep 22, 2023

Finished the first type of the WorldLoader class. ... 401e2db
unknown committed 3 weeks ago

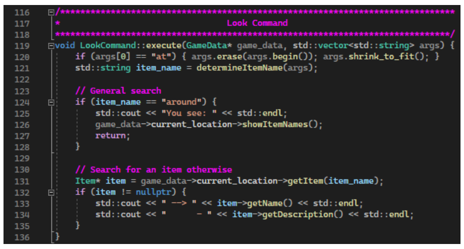
Finished Item and Location implementation of task 12 ... d26ffb6
unknown committed 3 weeks ago

Code



```cpp
/*****************************************************************************
*                              Move Command
*****************************************************************************/
void MoveCommand::execute(GameData* game_data, std::vector<std::string> args) {
    if (game_data->current_location->getExit(args[0]) != nullptr) {
        game_data->current_location =
            game_data->current_location->getExit(args[0]);
    }
}
```

Simple Move Command which sets the current location to a direction specified

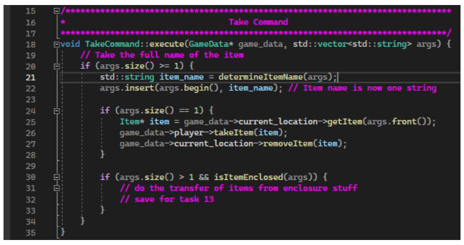


```cpp
/*****************************************************************************
*                                Look Command
*****************************************************************************/
void LookCommand::execute(GameData* game_data, std::vector<std::string> args) {
    if (args[0] == "at") { args.erase(args.begin()); args.shrink_to_fit(); }
    std::string item_name = determineItemName(args);

    // General search
    if (item_name == "around") {
        std::cout << "You see: " << std::endl;
        game_data->current_location->showItemNames();
        return;
    }

    // Search for an item otherwise
    Item* item = game_data->current_location->getItem(item_name);
    if (item != nullptr) {
        std::cout << " --> " << item->getName() << std::endl;
        std::cout << "      - " << item->getDescription() << std::endl;
    }
}
```

The Look Command has the ability to determine if the player wants to look in their general area or at something specific

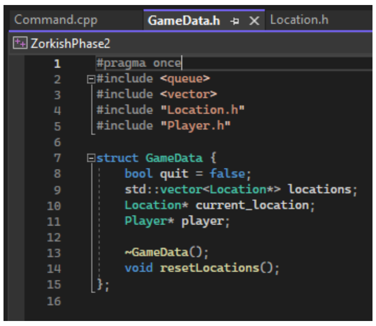


```cpp
/*****************************************************************************
*                                Take Command
*****************************************************************************/
void TakeCommand::execute(GameData* game_data, std::vector<std::string> args) {
    // Take the full name of the item
    if (args.size() >= 1) {
        std::string item_name = determineItemName(args);
        args.insert(args.begin(), item_name); // Item name is now one string

        if (args.size() == 1) {
            Item* item = game_data->current_location->getItem(args.front());
            game_data->player->takeItem(item);
            game_data->current_location->removeItem(item);
        }

        if (args.size() > 1 && isItemEnclosed(args)) {
            // do the transfer of items from enclosure stuff
            // save for task 13
        }
    }
}
```

Take command transfers objects from one inventory to another (drop command works very similarly).



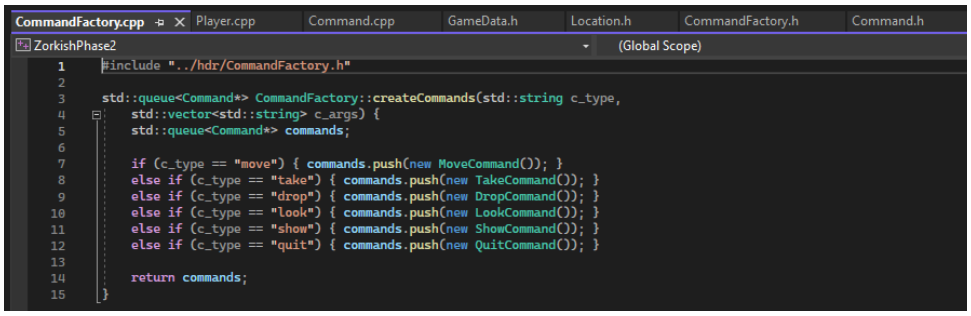
```cpp
#pragma once
#include <queue>
#include <vector>
#include "Location.h"
#include "Player.h"

struct GameData {
    bool quit = false;
    std::vector<Location*> locations;
    Location* current_location;
    Player* player;

    ~GameData();
    void resetLocations();
};

```

The Game Data is the struct whose components are manipulated through the commands. This data is instantiated by the AdventureSelectMenu and manipulated via the GameplayState.



```cpp
#include "../hdr/CommandFactory.h"

std::queue<Command*> CommandFactory::createCommands(std::string c_type,
    std::vector<std::string> c_args) {
    std::queue<Command*> commands;

    if (c_type == "move") { commands.push(new MoveCommand()); }
    else if (c_type == "take") { commands.push(new TakeCommand()); }
    else if (c_type == "drop") { commands.push(new DropCommand()); }
    else if (c_type == "look") { commands.push(new LookCommand()); }
    else if (c_type == "show") { commands.push(new ShowCommand()); }
    else if (c_type == "quit") { commands.push(new QuitCommand()); }

    return commands;
}
```

Rudimentary CommandFactory

What was Learned?

💡 Though this spikes outcome speaks only to the command pattern, I found the most challenging aspect of this spike to be all the small issues which arise when building less trivial code solutions. Issues such as linker errors, transferring data between objects, speed of operations and memory validity all reared their ugly heads.

Additionally, I spent a fair amount of the researching phase for this Spike understanding the software patterns required to instantiate commands at during runtime and have them manipulate data. Whilst the command pattern was prevalent (obviously), so too was the factory pattern. Once I understood how the factory idea worked, I decided it to be too overengineered for this Spike.
