Task 19 - Spike Summary Report

Spike: Task_19
Title: Messaging and Blackboards
Author: Thomas Horsley, 103071494

Goals & Deliverables

Aim: Develop a messaging system with either a dispatch or blackboard architecture. This system must be extensible enough to be implemented into Zorkish.

Deliverables:

- Functioning message dispatch system
- Spike summary report
- Git commit history

Technology, Tools and Resources

Tech and Tools

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.

Resources

- Echo360 Lecture "*Topic 7.1 - Messaging Systems and Architectures*"

Task 19 - Spike Summary Report

Class Descriptions and Notes

> 💡 For the “post-office” structure, I've chosen to implement a dispatcher for the messages rather than a blackboard as I believe this will suit my pre-existing code better.

Three notable extensions to Zorkish are the InputHandler, InputData and EventDispatcher classes. These objects provide us the tools to take input, format the input into a message object capable of being used as a context for the commands, and dispatch events when necessary.

Implementation

Git Commit History

Commits

History for COS30031-2023-103071494 / 19 - Spike - Messaging_ Annoucements and Blackboards

Commits on Oct 30, 2023

Completed task 19 - Spike 'Messaging Announcements and Bloackboards' ...

2046570

```
Created UML and planning documentation
Modified zorkish to contain EventHandler, InputHandler and InputData classes which act as the packaging and post office system respectively

Began creating the learning summary report for portfolio completion
```

unknown committed 1 minute ago

Commits on Oct 19, 2023

Best time for a commit is when you've just finished breaking everything ...

7b4609f

```
Pulled commands into their own subfolder (these are our events)
Created an eventHandler class which takes an InputData object and pushes the relevant commands to an events queue.
The eventHandler adds and removes subscribing components based on what is the immediate area. This data is cached for fast gamedata manipulation via
commands
```

unknown committed 2 weeks ago

Commits on Oct 18, 2023

Started Task 19 - Spike 'Messaging Announcements and Blackboards ...

8e79e4e

```
Pulled over all the files from task 13
Planned the inputHandler and EventDispatcher classes

Began creating the inputHandler class.
```

unknown committed 2 weeks ago

Commits on Jul 31, 2023

Initial Commit ...

Verified

0d3f501

KingSchlock committed on Jul 31

End of commit history for this file

```cpp
class EventDispatcher {
private:
    GameData* _game_data = nullptr;

    MoveCommand* _move_command = nullptr;
    TakeCommand* _take_command = nullptr;
    LookCommand* _look_command = nullptr;
    ShowCommand* _show_command = nullptr;
    QuitCommand* _quit_command = nullptr;

public:
    // Create all the commands within the _commands vector
    EventDispatcher();
    ~EventDispatcher(); // Delete commands

    void setGameData(GameData* game_data);
    void filterLocalComponents();
    void resetComponents();

    // Call the onEvent() method for the relevant commands parsing the relevant args
    std::queue<Command*> processEvents(InputData* input_data);

private:
    void getEntityComponents(const char UEID);
    C_Render* getRenderer(const char UEID);
    C_Inventory* getInventory(const char UEID);
    std::vector<C_Portal*> getPortals(const char UEID);

    std::string getExitUEIDFromDir(std::string direction);
};
```

EventDispatcher acts as the post office for our messaging system.

```cpp
#include "../../commands/hdr/Command.h"

#pragma once
struct InputData {
    CommandType c_type = CommandType::INVALID;
    std::vector<std::string> args = {};

    C_Inventory* src = nullptr;
    C_Inventory* dest = nullptr;
    Entity* item = nullptr;
};
```

The message context is pre-formatted and contained within the args. With other data being stored independently.

Additionally, each command has been modified to take an InputData message to provide context for it's triggerEvent( ) method.

The InputHandler object takes the user input and is responsible for formatting the input appropriately.

Once formatted, the InputHandler will provide the EventDispatcher with the InputData.

```cpp
#pragma once
class InputHandler {
private:
    InputData* _input_data = nullptr;

public:
    InputHandler(InputData* input_data = nullptr);
    ~InputHandler();

    InputData* handleInput(std::stringstream& raw_input, GameData* game_data);

private:
    void resetInputData();
    CommandType validateCommandType(std::string raw_c_type);

    std::vector<std::string> formatArgsForType(std::string raw_c_type,
        std::vector<std::string> args, GameData* game_data);

    std::string getItemName(std::vector<std::string> args, std::string safety_word = "");
    bool takeOrDrop(std::string take_modifier);
    std::string formatTakeOrDrop(std::string raw_c_arg);
    std::string extractUEID(std::string ucid);
};
```

```cpp
    getline(std::cin, input);
    std::stringstream input_stream(input);

    // The post office
    std::queue<Command*> events = _event_dispatcher->processEvents
        (_input_handler->handleInput(input_stream, _game_data));

    // Consider this the mailing system
    while (!events.empty()) {
        events.front()->triggerEvent();
        events.pop(); }

    if (frame_start_location.front() != _game_data->current_location.front()) {
        _event_dispatcher->filterLocalComponents(); }

    return STATES::S_GAMEPLAY;
}
```

The GameplayState update( ) method utilizing the messaging system.

Task 19 - Spike Summary Report

What was Learned?

> 💡 This Spike involved understanding the “Post Office” pattern and implementing a messaging system robust enough to support the basic ECS and command pattern structure within my Zorkish implementation.

Throughout the duration of the project, I found that similar messaging systems are extensively used in both games and game engines for allowing communication between largely decoupled components and systems. I plan to use a more robust equivalent system within my own custom project.
