CS 467 Capstone Course
Project Plan
Winter 2018

Adventure Team Project Plan

Adventure Team Members

- Michael Heinemann
- Stephen Liu
- Heidi Uphoff

Submission Contents

- Adventure_Team_MidPoint_Check.pdf - Instructions for compiling and testing working code
- Adventure-game-master (directory) - Code to compile for running game
- Videos
    - Game demo
        - https://media.oregonstate.edu/media/t/1_mmr5pbq1
    - Natural Language Parser demo
        - https://media.oregonstate.edu/media/t/0_70vd59ij

Project Status

Game Engine - *Lead Developer Michael Heinemann*

The game engine is used to continually run the game from start to finish. The engine utilizes data loading files to read in the information pertinent to gameplay, as well as a natural language parser file, to process any incoming user command line input. In its current state, the game engine utilizes four functions. First is a function to display the game description as well as instructions to introduce the user to the game. Second, a function to catch user input from the command line. Third, a function that starts actual gameplay, which continually runs the game until completion. Lastly, a function used to examine current room contents, which is limited in its functionality for now, but will be expanded upon in the coming weeks. Currently, the game engine allows the user to navigate throughout different rooms, displaying room descriptions, names, and features. Room descriptions are presented in long form when first entering a room and short form when entering a room already visited once. Certain in room features can be "looked at" displaying their descriptions.

The game engine will be expanded on from here, to include objects and features that can be picked up or interacted with, and to include in room objectives to complete in order to move on. A player inventory will be created to hold items picked up throughout the game that can be used in different situations. We will also be adding abilities to save game states and load in games already in progress.

Data Files and Data Loading - *Lead Developer Stephen Liu*

The text loading and parsing software reads the data files for the game engine to run. For example, the game engine reads the room data files to retrieve information such as if a room has or has not been visited, the room name, long description, short descriptions, features,

connection rooms, etc and parses these information and loads it in the appropriate data structure that can be utilized by the game engine. Additional data files that are loaded to be used by the game engine includes which objects were placed in the player's inventory and the save files to retrieve a gamestate. These additional data files text loading and parsing software are currently in development.

Natural Language Parser - *Lead Developer Heidi Uphoff*
The natural language parser, used to interpret command line user instructions, has been implemented by the Game Engine. The parser takes input and processes it so that is more likely to catch the verbs and nouns the user intended. Input is lowercased, stripped of punctuation, and stripped of stop words such as prepositions and articles. First, the parser searches for verbs. There are ten verbs utilized by the game. The parser looks for these as well as catches some synonyms. Next, the parser looks for exits, both for cardinal directions and for exit names. After that, the parser seeks out any features and objects used in the game. Finally, the parser catches any other nouns. It returns to the Game Engine a verb and two nouns if they exist in the input. More testing and calibration will be required to ensure the parser matches what is needed for the Game Engine as its development continues.

User Instructions
Compiling and Running Game

```
make gameEngine
gameEngine
```

Starting The Game
At the start of the game the user will be given a brief description of the game along with some instructions. The user needs to type start game into the command line in order to begin.

This will be expanded upon to provide a more thorough description, with better instructions.

Moving Through Rooms
Movement through the rooms is handled from user input through the command line. To navigate through the game the user can enter commands such as "go north", "depart south", "exit east door". Directions to each door is provided in the room descriptions. Stop words can be handled, so including words like "through" and "to" should not inhibit movement through a door. The important thing is to remember to use an appropriate verb along with an exit noun. If any input is not recognized and appropriate error message should be displayed to the user.

In the coming weeks we will be adding more in game interaction with features and objects. Some objects may be required to navigate through certain rooms. There will also be certain tasks to complete and enemies to defeat in different rooms to provide more depth to the game.

Examining Features

Throughout the game, in each room the user will have available certain features that can be examined. As it stand if the user types in “look at” plus the feature name, a short description of that feature should be output to the screen. This function is in the early stages and might be a little buggy at the moment.

We will be adding more interaction with the features in the future. Certain features could be the key to defeating certain enemies, or might hold items to help along the way.

Other Verbs/Actions

Taking objects from a room is currently in development. We created an object.txt file that contains information about the eight objects of the game (the object’s name, room location, and description). Currently the program is able to read the object information from the object.txt file (this portion of the code is not included in this midpoint check assignment submission because of it being in development). The player can take an object in the room. For example, if the player is in the basement room, they can take the key by entering the command “take key”.