CS 467 Capstone Course
Project Plan
Winter 2018

Adventure Team Project Plan

Adventure Team Members

- Michael Heinemann
- Stephen Liu
- Heidi Uphoff

Introduction

The Adventure Team will be completing a command line adventure game modeled from classic text adventures like Colossal Cave and Zork. This project will enable the team to polish coding and software engineering skills developed during their time as OSU students.

User's Perspective Description

This is an escape game. Users will be placed in a starting room and must navigate through the maze of rooms to the end room. Within each room users will be presented with descriptions, options, and an enemy or obstacle to defeat. Failure to defeat the enemy or overcome the obstacle will result in the game ending. Users interact with the room using natural language commands such as "look out window," "open up chest," or "go through north door."

Throughout the rooms there are eight objects that users can pick up and place in their inventory. They can dispose of objects in the inventory in any room. Objects must remain in rooms in which they are disposed. Some objects can be used to overcome specific room enemies/obstacles.

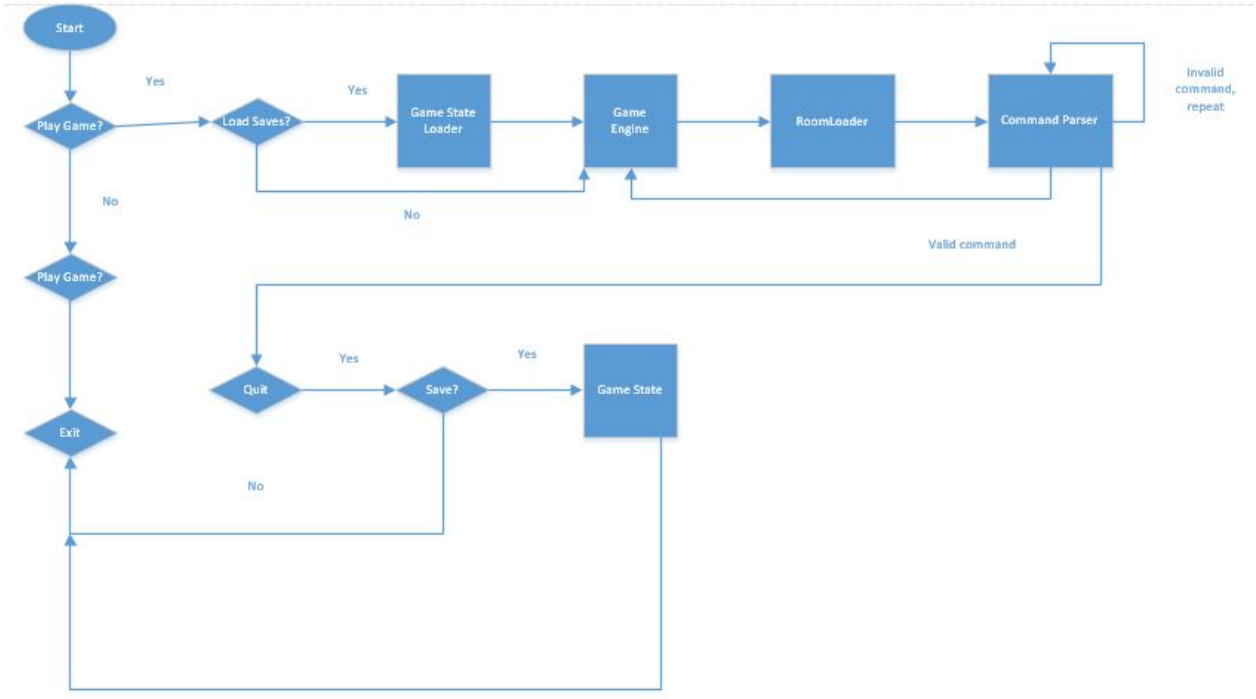
In order to defeat the room's enemy or overcome the room's obstacle, users are given three choices: try to flee, fight back, or use an object from their inventory. If they try to flee or fight back, the game rolls a dice to see if they succeed. If successful in fleeing, users are randomly placed in an adjacent room. If successful in fighting back, a user remains in the room and can decide what to do next. Items from the inventory may or may not work to defeat the enemy depending on the room.

Fig. 1 Gameplay Mockup:

This gameplay mockup is one example of how the game might be played. Format of the gameplay is likely to be changed/updated as the projects proceeds.

```
You have entered a basement.
The air smells damp and moldy. There are three ways out, back from where you came,
a door on the opposite end, and a window. Not much in this room besides a desk and chair.
Better look around a bit and figure out what to do next...

Room: Basement
Exits: 2 doors, 1 window
Objects: Desk, Chair

Action: look at desk

The desktop is empty, but there is a drawer. I should see what's inside.

Action: open drawer

There is a key inside! It might be useful along the way.

Action: take key

Key was put into your inventory.

Action: look back at room

Room: Basement
Exits: 2 doors, 1 window
Objects: Desk, Chair

Action:
```

Fig. 2 Game Architecture:



CS 467 Capstone Course
Project Plan
Winter 2018

User Requirements:

- Rooms
    - There are 15
    - Each contains two features that can be examined. One of these features turns into an enemy or an obstacle that the user must defeat. The feature may also be something that the user can interact with that advances or does not advance the game.
    - Each has a long description for the first time a user enters the room
    - Each has a short description for subsequent visits to the room
    - Exits from a room are described in the long and short description and have a cardinal direction (north, south, east, or west)
- Objects
    - There are 8
    - They can be taken by a player and placed in their inventory
    - They may be needed to defeat certain enemies
    - They may be necessary to enter other rooms
    - They can be dropped in other rooms
- Verbs
    - There are 10 primary action verbs that players can use to interact with room features
        - look - This repeats the long form explanation of the room
        - look at - This gives a detailed explanation of one of the room's features
        - go - This sends a user through an exit. User must specify the exit either by cardinal direction (north, south, east, west) or by the description given
        - take - User acquires and object and places it in their inventory
        - drop - User drops an object in their inventory
        - help - list set of verbs for user
        - inventory - displays the contents of the user's inventory
        - hit - user hit an object/enemy
        - open - user tries to open item in room
        - move - user tries to move item in room
- Natural Language Parser
    - Will allow users to type using full sentences, prepositions, and synonyms
    - Must recognize common synonyms for the verbs
    - Can handle prepositions such as about and on
    - Identifies the primary and secondary nouns in the user's command
- Rooms are loaded into the game from text files
- The game will be saved when a user types savegame
- A saved game will be loaded when a user types loadgame

CS 467 Capstone Course
Project Plan
Winter 2018

Listing of Software Libraries, Languages, APIs, Development Tools, Servers
The game will be built using the C programming language. The team will collaborate and version control using git and github. All team members will use OSU's flip servers for development.

Description of Software Structure
Rooms and saved game states will be stored in text files in sub-folders in the same folder as the executable program code. When a new game is started, this data will be loaded into structs.

./
  adventure-game *(executable C file)*
  /rooms *(sub-folder to store room files)*
    room-1
    room-2
    room-3
    room-4
    room-5
    room-6
    room-7
    room-8
    room-9
    room-10
    room-11
    room-12
    room-13
    room-14
    room-15
  /gamestates *(sub-folder to store saved game state information)*
    gamestate-1
    gamestate-2
     ….

The executable code will come from the following 4 source files (with header files). These 5 files may be expanded as code is refactored. There will also be a make file for easy compiling.

- RoomLoader.c *(Code to load room information stored in text files)*
- RoomLoader.h
- GameStateLoader.c *(Code to load saved games in text files)*
- GameStateLoader.h
- GameEngine.c *(Code the runs game)*
- GameEngine.h

- GameEngine_helpers.h *(Header file that keeps global variables and structs for rooms and gamestate)*
- CommandParser.c *(Code that parses command line input using natural language parsing)*
- CommandParser.h
- makefile

The room struct will contain the following data elements to start:

- Room name
- 2 features that can be examined (one will turn into an enemy or obstacle)
    - These can be examined using the ten verbs, need to have a response written (string to show user) or an action for both features for each of the ten verbs
- The object that can defeat the enemy (if any)
- A description for when the player enters the room the first time
- A description for when the player enters a room the second time (or more)
- Exits – exits need a short name, and a cardinal direction because users can use either to provide instructions

The gamestate struct will contain the following data elements:

- Room player is in
- Rooms player has visited
- Inventory
- Objects – and their present location
- Information if a user as defeated the enemy in each room in case they go back

Team Member Task Timeline

Major Team Milestones:

- **Week 3 -** *January 21 - 27*
    - Project Plan Due - 22
    - World building - write what happens for each room
- **Week 4 -** *January 28 - February 3*
    - Create and link rooms
    - Load rooms into game
- **Week 5 -** *February 4 - 10*
    - Working natural language parser
    - User can start game, see options, move between rooms
- **Week 6 -** *February 11 - 17*
    - Natural language parser is polished
- **Week 7 -** *February 18 - 25*
    - Mid-Point Report Due - 20
    - Mid-Point Code Due - 20
- **Week 8 -** *February 26 - March 3*

    - User can interact with features (defeat enemies)
- **Week 9 -** *March 4 - March 10*
    - User can save and load a game
- **Week 10 -** *March 11 - March 17*
    - Final Report Due - 17
    - Final Code Due - 17

Michael Heinemann's Workload (Focuses on the Game Engine):

- **Week 3 -** *January 21 - 27*
    - Create stories for assigned rooms - 3 hours
    - Create basic data structures for the rooms, and objects - 4 hours
    - Start work on game engine - 5 hours
    - **Hours:** 12
- **Week 4 -** *January 28 - February 3*
    - Input room stories into appropriate format - 2 hours
    - Start code for room creation - 5 hours
    - Start code for adding room connections - 5 hours
    - Testing/debugging - 2 hours
    - **Hours:** 14
- **Week 5 -** *February 4 - 10*
    - Work on player movement through rooms - 6 hours
    - Work on user interface - 5 hours
    - Testing/debugging - 2 hours
    - **Hours:** 13
- **Week 6 -** *February 11 - 17*
    - Start work/research on including objects/features within the rooms - 6 hours
    - Start mid-point report - 4 hours
    - Testing/debugging - 2 hours
    - **Hours:** 12
- **Week 7 -** *February 18 - 25*
    - Help finish up mid-point report - 2 hours
    - Clean up/prepare code for mid-point submission - 3 hours
    - Work on object/feature interaction within game - 3 hours
    - Work on in-game obstacles - 3 hours
    - Testing/debugging - 2 hours
    - **Hours:** 13
- **Week 8 -** *February 26 - March 3*
    - Help finish up code for object interaction - 2 hours
    - Help teammates with any needs - 2 hours
    - Test/debug/polish code - 4 hours
    - Start working on code to save game state - 4 hours
    - **Hours:** 12
- **Week 9 -** *March 4 - March 10*
    - Continue work on game state code - 4 hours
    - Testing of gamestate code for multiple scenarios - 4 hours
    - Polish user interface for final submission - 3 hours

    - Testing/debugging - 1 hour
    - **Hours:** 12
- **Week 10 -** *March 11 - March 17*
    - Final Project report - 4 hours
    - Polish code - 4 hours
    - Testing - 4 hours
    - **Hours:** 12
- **Total Hours:** 100

Stephen Liu's Workload (Focuses on the text files and data loading):

- **Week 3 -** *January 21 - 27*
    - Create stories for my assigned 5 rooms - 3 hours
    - Start code to create a series of files that hold descriptions of the in-game rooms and how the rooms are connected - 5 hours
    - Help Heidi with the natural language parser - 2 hours
    - Help Michael create format for storing data in game engine - 2 hours
    - **Hours:** 12
- **Week 4 -** *January 28 - February 3*
    - Input my room stories into appropriate room format - 2 hours
    - Start code for the text loading software (load rooms from text files into the game) - 5 hours
    - Start code for the text parsing software - 4 hours
    - Testing/debugging - 2 hours
    - **Hours:** 13
- **Week 5 -** *February 4 - 10*
    - Complete a working version of writing into data file results of the command line parser (which room the user entered into and the path the user traveled) - 6 hours
    - Complete a working version of loading from data file (load which room the user entered into and the path the user traveled). - 5 hours
    - Test and review code for the text loaded and parsed in from data file - 1 hours
    - **Hours:** 13
- **Week 6 -** *February 11 - 17*
    - Start on mid-point report - 4 hours
    - Start work/research on incorporating read/write of room features, objects, and user interactions in the room - 6 hours
    - Test and review code for game engine - 4 hours
    - **Hours:** 14
- **Week 7 -** *February 18 - 25*
    - Polish mid-point report - 4 hours
    - Review code for mid-point submission / make corrections as necessary for instructor - 3 hours
    - Start working on incorporating read/write of room features, objects,

and user interactions in the room - 5 hours
    - **Hours:** 12
- **Week 8 -** *February 26 - March 3*
    - Finish up code on incorporating read/write of room features, objects, and user interactions in the room - 4 hours
    - Start working on code to write and load saved game text files - 5 hours
    - Fix bugs in game engine and fix bugs reported by team members in the text writing and loading software- 3 hours
    - **Hours:** 12
- **Week 9 -** *March 4 - March 10*
    - Start on final project report - 3 hours
    - Start on walkthrough document for instructor to use when grading - 3 hours
    - Test code against walkthrough document - 2 hours
    - Fix bugs identified in testing - 4 hours
    - **Hours:** 12
- **Week 10 -** *March 11 - March 17*
    - Test code that allows users to save and load game - 2 hours
    - Finish final project report - 4 hours
    - Final testing and code review - 6 hours
    - **Hours:** 12
- **Total Hours:** 100

<u>Heidi Uphoff's Workload (Focuses on the natural language parser):</u>

- **Week 3 -** *January 21 - 27*
    - Project Plan - 2 hours
    - Set up github project complete with file names and stub code - 1 hour
    - Create stories for my assigned 5 rooms - 3 hours
    - Research natural language parsing - 2
    - Help Stephen create format for storing data in text files - 2 hours
    - Help Michael create format for storing data in game engine - 2 hours
    - **Hours:** 12
- **Week 4 -** *January 28 - February 3*
    - Input my room stories into appropriate room format - 2 hours
    - Research natural language parsing - 2 hours
    - Start on the command line input parser - 6 hours
    - Test and review code for room data loading, report bugs to team - 2 hours
    - **Hours:** 12
- **Week 5 -** *February 4 - 10*
    - Complete a working version of command line parser - 6 hours
    - Test and review code for command line parser - 3 hours
    - Research new features to polish natural language parser next week - 4 hours
    - **Hours:** 13
- **Week 6 -** *February 11 - 17*
    - Start on mid-point report - 4 hours
    - Polish natural language parser - 6 hours

    - Test and review code for game engine - 4 hours
    - **Hours:** 14
- **Week 7 -** *February 18 - 25*
    - Polish mid-point report - 4 hours
    - Review code for mid-point submission / make corrections as necessary for instructor - 3 hours
    - Work on code that allows users to interact with features and defeat enemies - 6 hours
    - **Hours:** 13
- **Week 8 -** *February 26 - March 3*
    - Test and review code that allows users to interact with features and defeat enemies - 4 hours
    - Fix bugs in game engine - 4 hours
    - Fix bugs reported by team members in the natural language parser - 4 hours
    - **Hours:** 12
- **Week 9 -** *March 4 - March 10*
    - Start on final project report - 3 hours
    - Start on walkthrough document for instructor to use when grading - 3 hours
    - Test code against walkthrough document - 2 hours
    - Fix bugs identified in testing - 4 hours
    - **Hours:** 12
- **Week 10 -** *March 11 - March 17*
    - Test code that allows users to save and load game - 2 hours
    - Finish final project report - 4 hours
    - Final testing and code review - 6 hours
    - **Hours:** 12
- **Total Hours:** 100

Conclusion

The Adventure Team will complete their text command line adventure game within seven weeks. Each team member will spend at least 100 hours developing the software for a total of 300 development hours. Deliverables will include a mid-term report, a final report, a walkthrough/cheat sheet for the instructor to use for grading, and source code.

References

Client requirements -
https://oregonstate.instructure.com/courses/1662181/pages/cmd1-adventure
Natural language parser example - https://www.youtube.com/watch?v=II3O1CJA-x8