Proposal For the Final Project of CS 3110

Wentao Guo (wg247), Michael Zhou (mgz27), Ruiqi Song (rs927)

1. Regular Meeting Schedule:  
   We will primarily meet during weekends for 3 to 6 hours, favorably on Sunday afternoon or Saturday afternoon inside Carpenter Hall, and for each weekend meeting, we will inspect our recent progress via peer reviews and establish a work plan for the next week collaboratively. In addition, we will meet once or twice in both/either Wednesday afternoon and/or Friday afternoon inside Carpenter Hall. That meeting will be about 2 hours and more intended to discuss some technical issues that require collaborative inspection.

Besides these regular meetings, we will also hold informal one-to-one conversations every two days. The locations and schedule for such conversations depend on each one’s availability.

1. Key Features about Our Project

Our project is to build a [todo - narrative description, about one page long]

**Functionalities**

* Main Storyline while Tasks-Oriented
* Bosses in Different Locations
* Simple Fighting
* Armors / Weapons / First-aid Kit
* Health Points / Strength
* Special Skills

1. System Roadmap [might need to recategorize the scopes]

**MS1: The Basic Structure**

Goal: Design the game’s main functionality that runs via text I/O.

Satisfactory Scope:

* The OCaml data representation type for a map is parsed correctly from JSON.
* The OCaml data representation type for initial states is parsed correctly from JSON.

Good Scope:

* The state representation type updates correctly
* Start planning roadmap for bosses’ actions/features
* Define the score for each player
* Define and implement the commands
  + If a command is malformed, or if an action is illegal, the user is known of this and the system does not crash. The state should remain unchanged.
  + If an action is legal, the state should update correctly
  + Corner cases – what happens if you try to go left or right but it’s out of bounds – wrap arounds or not?
* Define Rules for scoring
  + How much gain if you attack an enemy?
  + How much lost if get attacked by enemy?
  + How much when pick up item – weapon, armor, first-aid kit item?
* Winning/losing condition

Excellent Scope:

* Implement the enemies’ actions as defined in MS1

**MS2: GUI Prototype**

Goal: Refine the basic version of the text prototype, adding additional features and refining some existing features. Implement the GUI component with basic functionality using the OCaml native Graphics library.

Satisfactory Scope:

* Initialization - GUI draws the map grid correctly without crashing, based on the defined adventure input file.
* The engine with GUI does not crash
* Debug/change some existing features/definitions from MS1.

Good Scope:

* Update – GUI updates the map state correctly based on the data representations of the game state.
* Implement enemies’ actions – similar to players, but could be defined differently in the JSON file
* Add more verbs to the text console and handling of these verb。

Excellent Scope:

* Additional functionalities for the GUI, such as hotkeys, mouse tooltips, etc.

Note: The GUI is completely decoupled from the text I/O version from MS1. We are implementing two separate versions of the game (text I/O and GUI) displaying basic functionality before extending each version with additional functionalities. We plan to start the GUI implementation early on just to check that the GUI works correctly with basic features.

**MS3: Additional Features/Further Polishing**

Goal: Polish the game functionality with

Satisfactory Scope:

* Can use up, down, left, right keys to move the player
* Game does not crash on any user action

Good Scope:

* Game does not lag – user does not wait infinitely for a GUI to update.
* User can click on a tile to add a new item, new enemy [Don’t know if we choose to do this though]

Excellent Scope:

* Control game running speed.
* Difficulties of game – easy, medium, hard
* Cool graphics – color changes to the player as its health changes: red – low health, yellow – good health level.
* An enemy moves faster as it moves closer to the player

Note: It is very difficult to decide which features would count as good vs excellent scope in this milestone at this stage. This is just a brainstorm of the features that could be implemented.

1. Design Sketch

Basic Structure:

* Adventure Map - the initial state of the map when loaded via JSON. Data-driven.
  + Dimensions of map
  + Locations of players/items
  + The initial state of each system—such as health, strength, level, and experience point
  + The information of story, such as conversation and plot
  + The skill system
  + And more system that could make the game more playable
* Game State -
  + Current Item list – armors, weapons, first-aid kit
  + Player(s) – main player, enemies
    - Health points
    - Strength, ability
    - Special skills
    - Level
    - Weapon and equipment
* Main Engine Interface (via I/O)
  + Input/output design
* Interactive shell interface (game without GUI)
  + Input/output design
  + Unique input design that makes terminal game compatible with GUI game
* GUI – via Graphics library (OCaml native)
  + Graphical design—image stored as color matrix

Data structure:

* The maps of the game will be stored in JSON file which contains the basic design of the map, such as the starting point of item, NPC, enemy
* Artistic material of the game will be store either in JSON file or ML file in the form of color matrix due to the limitation of the Graphic module of Ocaml.
* Similar to A2+A3, the current data will be pasted as “state” within the program and will not the stored in the file (However, a saving feature may be implemented based on user experience)
* If a saving feature is achieve, a JSON file that record the key data (such as level, inventory, progress in the main story) will be created

Third party library:

This game will use libraries that were pre-installed with Ocaml at the beginning of the semester, so no installation of third-party library will be required.

Test:

At the stage of MS1, the main tool of testing is OUnit test to check the reliability of the game. The tests will be checking the state of each system, such as player’s health, the inventory or the progress of the story.

After MS1, the main tool of testing will be human because of the nature of GUI. We will play through the game to make sure the graphical interface is in good shape.

Team members will randomly check the codes written by others to make sure that the person is writing the correct code. So, an open-source control system-github is essential to this process.