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 single user’s adventure game in a story setting.

*Adventurer* *X once lived in Forest Home. One day, his village was attacked by a group of monsters and he decided to defend his village and he began his adventure. There are a lot of [elves, goblins, and some bosses] on his road, and after defeating these monsters, he became [stronger]. Sometimes he would find [arrows, and metal armors] well-equip him when fighting with these monsters. He might also explore some [unknown areas] which might be dangerous but rewarding. He gradually became stronger but none knew what is awaiting him…*

**Functionalities**

* Main storyline while tasks-oriented
* Bosses in different locations and bonuses for defeating bosses
* Simple fighting controlled by users input
* Items like armors /weapons /first-aid kits
* Adventurer’s growth in health points and strength by finishing tasks
* Special skills rewarded after exploring unknown areas

**System**

We will build a RPG game that is based on GUI. The system is consisted of underlying data system and graphical interface at the front. Player can move using their arrow key or WASD keys on the screen. Player could also pick up items to increase his strength, to move on in the main story line or to recover his health. A degenerated command shell version of this game is also available with limited features.

1. System Roadmap

**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, boss list, items, and storyline are parsed correctly from JSON.

Good Scope:

* State representation type updates correctly
* Define the score for each player
* Define and implement the commands
  + If a user action (either move, fight, or drop item) is invalid, the user should notice this error and the system does not crash.
  + If an action is legal, the state should update correctly
* 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 some simple enemies’ actions as defined in MS1

**MS2: GUI Prototype**

Goal: Refine the basic version of the text prototype, add additional features and refine 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.

Good Scope:

* Update GUI with the new game state when the system receives a valid user command.
* 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 as in 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 to check the GUI’s correctness with different game states.

**MS3: Additional Features/Further Polishing**

Goal: add the game functionality with a user’s growth record (such as skills gain and more strength) and polish the original game

Satisfactory Scope:

* The game can proceed to different states based on user’s current growth.

Good Scope:

* Game should move smoothly. Lowest fps should be still above 20 during the game.
* User can choose the difficulty level of the game (the default difficulty should be easy).

Excellent Scope:

* Control game running speed.
* 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

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. We will 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 manual playing with the game by GUI.

Team members will randomly check the codes written by others to make sure that the person is writing the correct code. Therefore, GitHub is essential to this process.