**RPG Project**

The goal of this project is demonstrate personal competency in several areas of programming, system design and project management. The result of this project will be a playable turn-based roleplaying game made with Game Maker Studio 2 that can serve as the basis for more specialized versions of a turn-based combat system.

Particularly, code readability, maintainability and modularity are the focus of the development process.

A version of Test Driven Development (TDD) will be used, in a more literal sense however. Tests will be designed to ensure code doesn’t regress and fail to deliver on requirements.

**Systems Design**

A turn-based RPG can be described as a collection of systems working in harmony to deliver an interactive, engaging experience. There are several systems that can be designed and coded in isolation, but will interact with each other. Videogames follow the MVC design pattern very closely. The underlying model consists of the game’s logic. The view is what the player sees on the display. The control is the way which the player interacts with the model. By keeping this in mind and separating model and view logic, creating healthy, clean code should come naturally.

**The PWI Paradigm**

In an RPG, there is an important pause-wait-interact paradigm. There will be times when control is taken from the player and input is not accepted. This is for events such as cutscenes, timed dialogue boxes and battle animations.

Other times, the game logic is listening for control input. This is “waiting.” Waiting continues under 2 conditions. Either the player must press a specified button or issue a command to the game logic, or a period of time passes such that a previously set timer expires and the game logic continues.

Interact, where the game spends most of it’s time, is when control of the game is in the hands of the player. This is when the player is navigating the play field, menus or is selecting commands during a combat sequence.

The implement a PWI paradigm, the logic will track what state the game is in currently. A policy decision point will be placed where necessary to determine what things are allowed to happen and when they are suspended. All systems are dependent on what the PWI paradigm allows to happen.

**Display Paradigm**

There will be various layers of graphics that are rendered to the display. The three most important layer types are:

* The interface layer: The layer all GUI elements are drawn. This is the top most layer.
* The foreground layer: Where environment objects are drawn that are visible above the characters sprites.
* The sprite layer: The layer where all character and enemy sprites will be drawn on.
* The background layer: Layers of the display that background tiles and images are drawn to. These are beneath the sprite layer.

**Interface**

The interface is what the player uses to access information about the game logic. This includes text boxes, menu windows, damage numbers. We will design the interface to utilize basic “building blocks.” This is done by breaking the elements of the interface into discreet components.

Text box: Contains any text that is to be read by the player and closed when there is no more text to be displayed.

Menu box: Contains text that is to be persistent on the screen. It’s visibility is determined by the game logic (view module).

Decision box: A text box that allows the player to make a decision. This represents branching paths within the game logic’s narrative flow. This box makes the game wait.

Options box: A box used to navigate to menus or issue battle commands.