**Pet Rock: The Game (working name)**

**Requirements**

* The game shall be intuitive/accessible to players of all skill levels
* The game shall use webcam/gesture controls as the primary form of input
* The game shall be capable of running without any performance issues on any computer or laptop (provided that the machine has a webcam)

**Design**

* The main purpose of the game is to care for a pet rock. The type of rock the player receives on starting the game is selected randomly. Depending on the type of rock, the player will have to interact with it in specific ways to keep it happy, since each rock type has a specific group of likes and dislikes.
* The player will use the webcam to interact with the game environment and interface. Through these means, the player can:
  + Check their pet’s stats – Happiness, hunger, health. (More to come if time allows)
  + Feed the pet
  + Clean the pet
  + Nurse the pet if it is sick
  + Clean up pet droppings
  + More if time allows
* The pet will not die if the user does not take good care of it (after all, pet rocks can’t die). However, the lack of care will be reflected in the stats page, and the pet will not be as responsive and may even misbehave due to being neglected.
* Specific control information will be added to this document as development continues. Currently, the plan is to let the character select their form of interaction by pointing to the desired option, and then waving their hand over the pet in a specific way to perform the selected action.
* If time allows, functionality for pet customization and interactions that use facial emotion recognition may be included at a future date.

**Implementation**

* The game itself will be implemented using Java and LibGDX. A basic algorithm will be used to determine how the pet rock’s mood/health is set throughout gameplay. The various stats will be set as integers, and will be used to determine the rock’s behavior. Before motion controls are applied to the game, the game will be programmed within LibGDX to allow interaction via the mouse/keyboard. This control scheme will likely be used only for testing purposes.
* The motion controls will work through OpenCV. Based on my current research, it is possible to use motion tracking to emulate keystrokes and mouse input. The game will make use of this functionality, replacing the default mouse/keyboard controls with the camera based control scheme.
* As mentioned in the design section, if time allows, the Microsoft Emotion API will be used for facial emotion recognition. However, this feature is not fully planned yet, as the main focus of the project is currently integrating the game with motion controls.

**Evaluation**

The goal of the project will be considered “achieved” if the player, regardless of skill level and without extensive instruction, is able to:

* Navigate the UI in its entirety
* Perform all the listed actions in the design section
* Succeed in achieving the game goal (maintaining a happy pet)
* Initiate and terminate the game session

Other evaluations of each development milestone will be performed, and are listed with each of the corresponding milestones in the schedule below.

**Milestones**

* **Write up small test programs** that simulate various actions, events, and other individual components of the full game **using LibGDX** to ensure that each of these pieces is possible to include in the game itself.
  + Completion – the test programs are written and function as expected
* **Assembly of game components**:
  + **Write the game logic**. This consists of the functions used to alter the pet’s various stats upon interacting with the interface.
    - Completion – Some tests will be written in which action functions are called. If the pet stats change in the expected way for each test, this step is considered completed.
  + **Build the game interface.** This involves displaying a sensible UI layout on the screen that can be used to call each function and interact with the game. Until later in development, this interface will use simple shapes and text to represent the various buttons, as well as the pet rock itself.
    - Completion – If it is possible to achieve the same outcomes from the “game logic” tests via the UI, this step is considered completed.
* **Create test programs** that utilize **OpenCV** in which feedback is given to the user based on motions and gestures that will be used in the game.
  + Completion – If the user is able to get the ideal feedback from performing the test motions, then the gestures are ready to be tested with the game itself.
* **Apply motion controls to the game –** This involves taking the feedback from the OpenCV motion detection and using it to emulate mouse and keyboard input associated with the game. Each gesture will be added to the game gradually and tested to ensure there aren’t any issues or conflicts between gestures.
  + Completion – A similar test process to the UI test process will be used once again to ensure the same series of inputs lead to the same expected outcomes as the original logic tests. If these tests all pass, then this step is complete, and the game is considered playable with motion control.
* **Polishing game appearance –** Replacing the placeholder assets with cleaner graphics and adding sound effects to certain in-game actions. Some play-testing will be carried out to ensure these assets are functional during gameplay.
  + Completion – If the gameplay remains unaffected (aside from cosmetics) after the addition of the sounds/graphics, this step is considered successful.
* **Play Testing –** This will occur throughout the development process, and is considered more of a regular procedure than a milestone. Other players of varying skill levels will be recruited to test the game to ensure playability and provide feedback if there are any components of the game that need improving. Play testing will be considered “complete” if all of the play testers’ feedback is addressed and there is no further issue with the game from the player’s perspective.

**Updated schedule coming soon.**