CS 2340 — Milestone 4: Project Iteration 3

Time Progression, Watering, Robustness Diagram and Package Diagram

BACKGROUND: For the third iteration of the project, your team will implement additional farm interactions along with time progression. For the design component, your team will create a robustness diagram. Robustness diagrams are helpful for visualizing the connections between major components inside a system. These diagrams can help you ensure that your system is adhering to a software architecture that reasonably separates responsibilities and visibility. Package diagrams are useful for illustrating the logical architecture of a system. They provide a visualization of the structure of the system and the dependencies between its modules.

PURPOSE: Planning is key when designing large systems. This milestone expands upon the planning started by M2 and M3 through the creation of a robustness diagram and a package diagram.

TASK: This milestone has two team design deliverables, a testing deliverable, and a required feature set for implementation. Other than the requirements outlined below, the details of your implementation are up to you. Your app implementation/functionality will be graded during a demo, which will occur the week after milestones are due.

Design Deliverable 1: Robustness Diagram

1. For the character creation we required you to implement in M2, generate a robustness diagram based upon this system.
   - Specifically, ensure that the Welcome Screen, the Player Configuration Screen, and the Farm Screen are components of the diagram.
2. Your diagram must include any necessary controller classes needed to make the system operate.
3. Your diagram must include an entity class to store the Player data after creation and for retrieval.

Design Deliverable 2: Package Diagram

1. Create a diagram showing how the various elements of your system are grouped into packages.
2. Show any hierarchical organization of packages.
3. Show relationships between packages, namely dependencies.

Implementation Requirements

1. Time progression
   - There must be a way to **advance time** in the game.
     - For example, this can be implemented as a button on the farm screen which advances the game by one day.
2. Implement the ability to **plant seeds**.
   - Player should be able to move seeds from their inventory to a farm plot.
   - Planting a seed should update the farm UI. The farm plot should display its contents either graphically or textually.
   - You can choose to allow players to pick a plot for planting or simply use the first available plot.
3. Develop a **growth cycle**. Plants should advance through their growth cycle when the game is moved forward a day.
   - You may define these growth stages however you want, but there must be at least 4 of them, one of which should be a dead stage.
   - For example:
     - Day 0: Seed
     - Day 2: Immature Plant
     - Day 3: Mature Plant

     - Day 0: Seed
     - Day 2: Immature Plant
     - Day 3: *(with insufficient watering)* Dead Plant

4. Implement **watering** and develop and display **water levels** for each plot (either graphically or textually).
   - Watering a plot should increase the plot's current water level.
   - Plots should have a maximum and minimum water level.
   - Too much or too little water should cause the plot's plant to die.
5. Maintain the ability to harvest mature plants.

Testing Requirements

1. Write **unit tests** to verify the functionality the newly implemented features.
   - There is no code coverage requirement, but you should make sure that your unit tests cover meaningful functionality of the implementation requirements.
   - *You should have at least 5 unit tests.*
2. **Testing Deliverable**: Include with your submission a brief writeup describing your testing process for the milestone. Explain which components were chosen for testing and why. Additionally, explain how your tests verify that the code functions as expected.

Checkstyle

During demo your team will be required to run the checkstyle script (located under files>checkstyle>Java Guide.pdf). This script will give your project a score out of 10 and will account for 10 points of your M4 final grade. Be sure to run the checkstyle script prior to submission to avoid unforeseen deductions.

Milestone Tagging

Tags are a way of marking a specific commit and are typically used to mark new versions of software. To do this, use "git tag" to list tags and "git tag –a tag_name –m description". You are required to tag the latest commit before the deadline which is to be graded during demo. You will be required to pull this commit during demo.

Submission Requirements

In addition to your diagrams, ensure that you include a link to your GitHub repository in your submission. Also, ensure that you have added your grading TA(s) as collaborators so that they may view your private repository. **Repositories must be located on the Georgia Tech GitHub and must be set to private!** Points may be deducted if these guidelines are not followed!

CRITERIA: Groups are required to demo in order to receive credit for the features they have implemented.