Phase 3 Report

Unit and Integration Tests

Features that need to be unit tested, and explanations:

1. We identified features of all different levels starting from the most basic features to the most complicated ones.
2. We tested the creation of arrays at different places in our program.
3. Testing if our program reads content from different files and resources.
4. Changes in the score due to different events by our main character.
5. Some of the tests are mentioned below:

InitRendering:

- After creating a window in the app, tests whether we can change the component given focus from the default FocusTraversalPolicy to JPanel component, which we use in the game.

initTimer:

1. initTimer tests whether our Timer instance is running. If the Timer instance is running, the application is able to execute a background thread for key inputs
2. https://docs.oracle.com/javase/7/docs/api/java/util/Timer.html

updateTestLogic:

- Tests all game ending conditions
    1. Game is cleared by collecting all required rewards and reaching the goal location.
    2. Game is over through reaching a negative overall score.
    3. Game is over when the enemy encounters the player at the same location.

updateInteraction:

1. Testing if the bonus rewards work in a way where their life expires after a certain duration.
2. The bonus rewards populate at a random position on the map.
3. If a bonus reward goes away, then another one appears right after in a different location.
4. We are checking to see if the results of collecting the rewards, bonus rewards, and penalties are good or not.
5. We check to see that there are changes happening to the player's score.

debugBoardOutput:

- Tested the board to see if specifications are up to requirements

updateMovables:

1. Test if all movables have updated positions
2. Test if all the movables can go over the different paths

initControls:

- We test to see with we have successfully created key input receiver

initDebugEntities:

- We test to see if the placements of objects within the cells are created and at the correct positions

Interactions tested:

We checked for all the essential interactions.

- One of them was the Player vs the Enemy
- Another one was the Player vs the Rewards and the Bonus Reward
- And similarly the Player with the Penalties

How we ensured the test quality

We made sure that we covered all the essential parts of our program that needed to be tested.

Line Coverage:

Total Lines = 2000
Lines covered = 950
Line Coverage = 950/1700 X 100% = 55%

Branch Coverage:

initTimer:
Total branches = 1
Branches covered = 1
Branch Coverage = 1/1 x 100% = 100%

updateTestLogic:
Total branches = 15
Branches covered = 11
Branch Coverage = (11/15) x 100% = 73%

updateInteraction:
Total branches = 7
Branches covered = 6
Branch Coverage = (6/7) x 100% = 85%

debugBoardOutput:
Total branches = 1
Branches covered = 1
Branch Coverage = (1/1) x 100% = 100%

updateMovables:
Total branches = 2
Branches covered = 2
Branch Coverage = (2/2) x 100% = 100%

initControls:
Total branches = 1
Branches covered = 1
Branch Coverage = (1/1) x 100% = 100%

initDebugEntities:
Total branches = 2
Branches covered = 9
Branch Coverage = (2/9) x 100% = 22%

cellArrayCreationTest()
Total branches = 11
Branches covered = 8
Branch coverage = (8/11) x 100 = 72%

testRandom()
Total branches = 11
Branches covered = 11
Branch coverage = (11/11) x 100 = 100%

testObstacle()
Total branches = 7
Branches covered = 5
Branch coverage = (8/11) x 100 = 71%

testManualCase()
Total branches = 11
Branches covered = 11
Branch coverage = (11/11) x 100 = 100%

testOneStep()
Total branches = 8
Branches covered = 8
Branch coverage = (8/8) x 100 = 100%

Total Branch Coverage = 85%

Features or code segments that are not covered and why:

- DrawImage and doDrawing functions are not covered as they require graphics objects passed through a parameter to be pre-existing.
- The problem is these functions are not called by us. We only override these functions that are used in the super implementation.
- This means we cannot test with graphics since we do not control the graphics object.
- A possible solution is to create a mock graphic object inside the test function to avoid using a parameter.
- However, as we never used mock objects in the real game, this test could be inaccurate. In addition, these codes are from super implementations of imported libraries and are already optimized.

Test Findings

Learnings from writing and running tests:

1. We learned to see how the methods run in connection with the different classes.
2. We saw that the different classes are in sync with each other.
3. We wrote tests that used hardcoded values which lets us learn the logic of code from the foundation level.

Changes to the production code during the testing phase:

1. The game didn't have certain validations for bonus rewards, and penalties, and how they updated the score. So corrections were made to that.
2. A new feature was added where Bonus Rewards show up always, but after some time they go to a random position.
3. A new start screen has been added.
4. Changed Input.txt file to add more movables and interactables. Number of Enemies increased from 1 to 3, Number of Bonus Rewards changed from 1 to 3, Number of Penalties changed from 2 to 4.
5. Change interactable score points and deductions. Penalty changed from 1 to 3, Bonus Reward changed from 1 to 5.
6.

Reveal and fix any bugs and/or improve the quality of code in general:

1. Made corrections to how the punishment updates the score.
2. Made corrections to how the BonusRewards updates the score.
    - In Phase 2, we were able to increment the score of bonus reward but the changes were not displayed on board.
3. Switched to lose state when the score was below 0.
4. There are different tests with a diverse range of input values so that the tests cover a wide area. This helps in improving the quality of the code.
5. A few new methods were added so that we don't have to repeat numerous lines of code at different places. So it's a little bit of refactoring the code so that it becomes a little better for making changes in the future and easier for other developers or testers to understand.

Discuss your more important findings:

- The updates in the scores were giving problems.
- The AI enemies concept was very interesting to learn for us.
- We worked a little bit on stashes on Git which was new.
- File paths for the images and animations had to be updated. Different IDEs compile differently and it's difficult for everyone to be consistent.