

Project, Phase 3: Testing

Colton Blackwell, Jonathan Bryan, Ridham Sharma, Sophia Don Tranho

*Jacoco was used for evaluating the branch/condition coverage of the tests.

*Jacoco coverage images are a result of the class in isolation and not the sum of all test cases in every '...Test'.java file

AppTest.java

Info:

- System tests:
 - First test needed as they ensure Game class and Main run properly.
 - testAppBuild()
 - Try/Catch on main to check if it gets initialised correctly
 - testGameStates()
 - Testing game state based on .keyPressed() user input
 - General test to ensure the app builds correctly
 - Test to ensure all basic game states are loading/running correctly

Element	Missed Instructions	Cov.	Missed Branches	Cov.
com.mycompany.app		68%		34%
Total	1,358 of 4,283	68%	274 of 419	34%

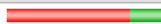
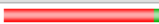
Element	Missed Instructions	Cov.	Missed Branches	Cov.
CollisionHandler		37%		8%
Player		48%		17%
Game		71%		70%
Raccoon		60%		19%
KeyHandler		30%		20%
Board		91%		82%
Entity		88%		66%
PopUp		94%		66%
Main		90%		n/a
PathFinder		99%		87%
Cell		100%		n/a
Node		100%		n/a
Total	1,358 of 4,283	68%	274 of 419	34%



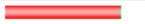

















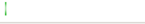
*(Jacoco) AppTest coverage

BoardTest.java

- Unit Tests/Functional testing:
 - AssertFishLoaded()
 - To ensure there are regular rewards (orange koi fish) spawned in the maze
 - AssertObjectRemoval()
 - To ensure that the object is properly removed and that there is a blank cell in its place

- AssertFishRemoval()
 - Testing feature which removes the fish from the board and updates the fish count.
- AssertCellLoadedI()
 - Testing when user encounters regular reward, point goes up by one
- AssertOpenEnd()
 - Testing feature of opening end block when user is allowed to pass to next level.




Element	Missed Instructions	Cov.	Missed Branches	Cov.
com.mycompany.app		22%		6%
Total	3,295 of 4,264	22%	392 of 419	6%


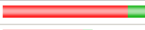












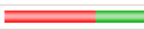

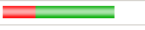
















Element	Missed Instructions	Cov.	Missed Branches	Cov.
Game		13%		0%
CollisionHandler		1%		0%
Player		11%		0%
Raccoon		10%		0%
PathFinder		14%		8%
KeyHandler		3%		0%
Board		70%		41%
PopUp		28%		33%
Entity		73%		58%
Main		0%		n/a
Cell		100%		n/a
Node		100%		n/a
Total	3,295 of 4,264	22%	392 of 419	6%

(Jacoco) BoardTest coverage

CollisionHandlerTest.java

- Unit Test:
 - testCollisionAllDirection()
 - Tests the collision handling for all four movement directions (up, down, right, left)
 - assertEquals(game.p1.y, y)
 - Checks if the entity's y position remains unchanged after attempting to move in the up and down directions
 - assertEquals(game.p1.x, x)
 - Checks if the entity's x position remains unchanged after attempting to move in the right and left directions


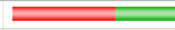
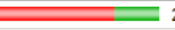
Element	Missed Instructions	Cov.	Missed Branches	Cov.
 com.mycompany.app		36%		23%
Total	2,722 of 4,264	36%	321 of 419	23%


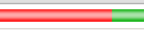

















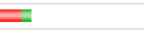













Element	Missed Instructions	Cov.	Missed Branches	Cov.
 Game		13%		0%
 Raccoon		10%		0%
 PathFinder		14%		8%
 Player		41%		68%
 KeyHandler		3%		0%
 CollisionHandler		68%		37%
 Board		70%		41%
 PopUp		28%		33%
 Entity		73%		58%
 Main		0%		n/a
 Cell		100%		n/a
 Node		100%		n/a
Total	2,722 of 4,264	36%	321 of 419	23%

(JaCoco) CollisionHandlerTest coverage

KeyHandlerTest.java

- Functional unit testing
 - testPlayerInput()
 - Test case for WASD keys and player's reaction to that movement (Ex. up, down, etc.)
 - testKeyRelease()
 - Testing whether input in game.keyHandle.keyReleased(input) accidentally triggers the position of the user.

Element	Missed Instructions	Cov.	Missed Branches	Cov.
 com.mycompany.app		38%		27%
Total	2,609 of 4,264	38%	305 of 419	27%



Element	Missed Instructions	Cov.	Missed Branches	Cov.
 Game		21%		25%
 Raccoon		10%		0%
 PathFinder		14%		8%
 Player		41%		68%
 KeyHandler		23%		12%
 CollisionHandler		68%		37%
 Board		70%		41%
 PopUp		28%		33%
 Entity		73%		58%
 Main		0%		n/a
 Cell		100%		n/a
 Node		100%		n/a
Total	2,609 of 4,264	38%	305 of 419	27%

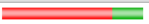
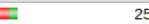




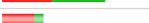














(JaCoco) KeyHandlerTest coverage

PlayerTest.java

- Functional Testing
 - AssertPlayerDirection() => Tests if the player's direction is being updated correctly based on user input

- AssertRewardCollection() => Tests cell 3 as regular reward in .collectReward()
- AssertPenaltyCollection() => Tests cell 4 as penalty in .collectPenalty()
- AssertReset() => Check whether time is set back to 0 when game is reset
- Structural Testing
 - AssertHide() => Checks if the player.hideState works and user is in a hideable cell
 - AssertSpriteChange() => Goes through a sequence of sprite counter values and checks whether the player's spriteNum updates as expected



Element	Missed Instructions	Cov.	Missed Branches	Cov.
com.mycompany.app		40%		28%
Total	2,551 of 4,264	40%	301 of 419	28%

Element	Missed Instructions	Cov.	Missed Branches	Cov.
Game		21%		25%
Raccoon		10%		0%
PathFinder		14%		8%
Player		51%		76%
KeyHandler		23%		12%
CollisionHandler		68%		37%
Board		70%		41%
PopUp		28%		33%
Entity		73%		58%
Main		0%		n/a
Cell		100%		n/a
Node		100%		n/a
Total	2,551 of 4,264	40%	301 of 419	28%

(JaCoco) PlayerTest coverage

RaccoonTest.java

- AssertRaccoonLocation()
 - Testing .setLocation() and asserting whether the position is equal to the cellSize*x/y position.
- AssertRaccoonDirection()
 - Checks behaviour of the racoon entity by undergoing different direction changes and asserts if the racoon's final position is different from its initial position
- AssertRaccoonSearchPath()
 - Testing the path finding algorithm of the raccoon and checking that the raccoon's movements are based on the map environment.

Element	Missed Instructions	Cov.	Missed Branches	Cov.
com.mycompany.app		56%		44%
Total	1,850 of 4,264	56%	233 of 419	44%

Element	Missed Instructions	Cov.	Missed Branches	Cov.
Game		21%		25%
Player		51%		76%
KeyHandler		23%		12%
CollisionHandler		68%		37%
Board		70%		41%
Raccoon		68%		44%
PopUp		28%		33%
Entity		73%		58%
Main		0%		n/a
PathFinder		99%		87%
Cell		100%		n/a
Node		100%		n/a
Total	1,850 of 4,264	56%	233 of 419	44%

(JaCoco) *RaccoonTest* coverage

Findings

After doing some playtesting of the game, we noticed that the player's hitbox felt wider than it should have been. You would often die to the raccoon after getting stuck on the corner of a block in tight sections of the maps. This was a simple fix, since all we had to do is make the size of the player hitbox a bit thinner.

The other issue that came up was after beating the second level. Upon reaching the exit, the game would freeze instead of displaying the "You Win" screen and giving the option to move to level 3. I suspected that it was due to the end of level two being on the top of the screen, so I dug into the code to find the bug. The bug ended up being caused by an `IndexOutOfBoundsException` exception, when the player was trying to move to a tile that did not exist. To fix this, I changed which edge of the player's sprite was being used to detect where it was.

Before:

```
public boolean checkGameWin(int left, int right, int top, int bottom) {
    // if the next cell is the last cell/end cell, the game is won
    if (left == gamepanel.boardCol || right == gamepanel.boardCol || top == gamepanel.boardRow || bottom == gamepanel.boardRow) {
        gamepanel.keyHandle.gameState = 5;
        return true;
    }
    return false;
}
```

After:

```
public boolean checkGameWin(int left, int right, int top, int bottom) {
    // if the next cell is the last cell/end cell, the game is won
    if (right == gamepanel.boardCol || bottom == gamepanel.boardRow || right == 0 || bottom == 0) {
        gamepanel.keyHandle.gameState = 5;
        return true;
    }
    return false;
}
```

This simple change fixed the bug and allowed for us to have an exit on any side of the map.