SE 3XA3: Test Report
Namcap

Team 2, VPB Game Studio
Prajvin Jalan (jalanp)
Vatsal Shukla (shuklv2)
Baltej Toor (toorbs)

December 8, 2016

Contents

List of Tables

List of Figures

This document ...

1 Functional Requirements Evaluation

1.1 Game Functionality Testing

A Robot (automated) unit testing class was implemented and used to test the mechanics of the game.

1. GFT1

   Type: Functional, Dynamic, Automated

   Initial State: Application is displaying the main menu page

   Input: Cursor clicked on Start Game button

   Expected Output: New game is started and window is changed to reflect a new game state

   Output: New game was started and window was changed to reflect a new game state

   Result: PASS

2. GFT11

   Type: Functional, Dynamic, Automated

   Initial State: Within game state

   Input: Escape button pressed

   Expected Output: Application must pause and ask user if they want to quit

Table 1: **Revision History**

Date	**Version**	**Notes**
2016-12-08	1.0	Completion of Functional Requirements Evaluation
2016-12-08	1.1	Completion of Unit Testing Section

Output: Game was paused and user was asked to quit or continue

Result: PASS

Player Movement/Collision Testing

1. GFT2

   Type: Functional, Dynamic, Automated

   Initial State: Within the game state

   Input: Arrow keys

   Expected Output: Player moves in the respective direction (if path is clear)

   Output: Player moved in the respective direction when path was clear

   Result: PASS

2. GFT3

   Type: Functional, Dynamic, Automated

   Initial State: Player comes in contact with wall

   Input: No input

   Expected Output: Player stops moving when coming in contact with the wall

   Output: Player's x and y coordinates were not changed when in contact with the wall

   Result: PASS

3. GFT4

   Type: Functional, Dynamic, Automated

   Initial State: Player comes in contact with enemy

   Input: No input

   Expected Output: If player has more than 1 life, decrement lives. If player has one life, end game.

Output: Player's life was decremented by 1 when player had more than one life. The game was ended if player was on their last life.

Result: PASS

4. GFT6

   Type: Functional, Dynamic, Automated

   Initial State: Player comes in contact with dots

   Input: Arrow keys

   Expected Output: Dot disappears after collection

   Output: Dot disappeared afer player colletcted it

   Result: PASS

5. GFT7

   Type: Functional, Dynamic, Automated

   Initial State: Player collects the big dot

   Input: Arrow keys

   Expected Output: Big dot disappears after collection

   Output: Big dot disappeared after player collected it

   Result: PASS

6. GFT8

   Type: Functional, Dynamic, Automated

   Initial State: Player collects the big dot

   Input: Arrow keys

   Expected Output: Player is able to collide with enemies

   Output: Player does not lose any lives when colliding with enemy

   Result: PASS

Enemy Movement/Collision Testing

1. GFT5

   Type: Functional, Dynamic, Automated

   Initial State: Within the game state

   Input: No input

   Expected Output: Enemies move on a valid path

   Output: Enemy does not go through barriers

   Result: PASS

2. GFT9

   Type: Functional, Dynamic, Automated

   Initial State: Player collects the big dot

   Input: No input

   Expected Output: Enemies change colour

   Output: Enemies changed colours

   Result: PASS

3. GFT14

   Type: Functional, Dynamic, Automated

   Initial State: Player collides with enemy after collection of big dot

   Input: Arrow keys

   Expected Output: Enemy is removed from game and respawned back to their original cell

   Output: Enemy is respawned back to the center of the game

   Result: PASS

Scoring Testing

1. GFT10

   Type: Functional, Dynamic, Automated
   Initial State: Player collects all dots
   Input: Arrow keys
   Expected Output: Game over screen is activated
   Output: Player's score is displayed along with the Game Over screen
   Result: PASS

2. GFT12

   Type: Functional, Dynamic, Automated
   Initial State: Player collects dot
   Input: Arrow keys
   Expected Output: The points are increased
   Output: Player's score is increased by 100
   Result: PASS

3. GFT13

   Type: Functional, Dynamic, Automated
   Initial State: Player collects big dot
   Input: Arrow keys
   Expected Output: The points are increased at twice the rate
   Output: Player's score is increased by 200
   Result: PASS

2 Nonfunctional Requirements Evaluation

2.1 Usability

2.2 Performance

2.3 etc.

3 Comparison to Existing Implementation

This section will not be appropriate for every project.

4 Unit Testing

Unit Testing for Namcap was done using Java's JUnit testing suite, and results of all tests were written and summarized to a text file. If any tests failed, the exception would be included so the development team could analyze and repair any errors. Figure 1 is an example of the text file.

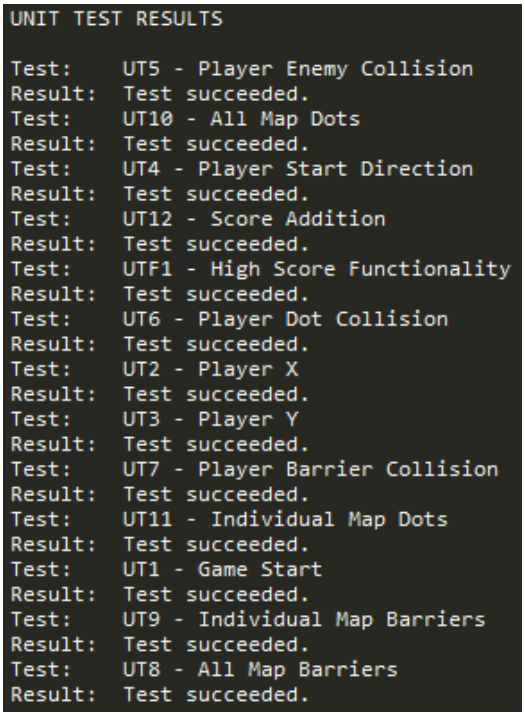

Figure 1: Unit Test Results

Test Cases

1. UT1

   Type: Unit, Static, Automated

   Initial State: Application is displaying the main menu page

   Input: Start Game button action is performed

   Expected Output: New game is started and window is changed to reflect a new game state

   Output: New window (game board) successfully opened

   JUnit Test Result: PASS

2. UT2

   Type: Unit, Static, Automated

Initial State: Player is in starting position at the start of the game

Input: Current X accessor method for the player

Expected Output: 200 (start X position)

Output: 200

JUnit Test Result: PASS

3. UT3

   Type: Unit, Static, Automated

   Initial State: Player is in starting position at the start of the game

   Input: Current Y accessor method for the player

   Expected Output: 300 (start Y position)

   Output: 300

   JUnit Test Result: PASS

4. UT4

   Type: Unit, Static, Automated

   Initial State: Player is in starting position at the start of the game

   Input: Current direction of the player

   Expected Output: 'R' (player starting direction)

   Output: 'R'

   JUnit Test Result: PASS

5. UT5

   Type: Unit, Static, Automated

   Initial State: Player is in starting position at the start of the game

   Input: PlayerX, PlayerY, EnemyX, EnemyY (200,300,185,300)

   Expected Output: Player lives decremented (player to enemy collision succeeded)

   Output: 2 (player lives left)

JUnit Test Result: PASS

6. UT6

   Type: Unit, Static, Automated

   Initial State: Player is in starting position at the start of the game, all dots are on map

   Input: PlayerX, PlayerY ([180,300],[20,180],[20,180])

   Expected Output: Score increases the first two times, but not the last

   Output: ([score increases to 100, no dot],[score increases to 200, no dot],[score remains at 200, no dot])

   Junit Test Result: PASS

7. UT7

   Type: Unit, Static, Automated

   Initial State: Player is in starting position at the start of the game

   Input: X and Y positions around the player (Player Positions tested (X,Y): [200,300],[20,20],[100,180])

   Expected Output: 2 barriers around the first position, 2 barriers around the second position, 0 barriers around the third position - true and false values

   Output: ([false,false,true,true],[true,false,true,false],[false,false,false,false])

   Junit Test Result: PASS

8. UT8

   Type: Unit, Static, Automated

   Initial State: Board is created with only barrier and dot entities

   Input: X and Y positions for all barrier locations

   Expected Output: True for all barrier locations (manually stated in JUnit class)

   Output: True for all barrier locations

   JUnit Test Result: PASS

9. UT9

   Type: Unit, Static, Automated

   Initial State: Board is created with only barrier and dot entities

   Input: X and Y positions for a location without a barrier, and an update to that location (to create a barrier) [10,15]

   Expected Output: True, a barrier exists for that location [10,15]

   Output: True

   JUnit Test Result: PASS

10. UT10

    Type: Unit, Static, Automated

    Initial State: Board is created with only barrier and dot entities

    Input: X and Y positions for dot locations

    Expected Output: True for all dot locations (manually stated in JUnit class)

    Output: True for all dot locations (true that they are all 1)

    JUnit Test Result: PASS

11. UT11

    Type: Unit, Static, Automated

    Initial State: Board is created with only barrier and dot entities

    Input: X and Y positions for a location without a dot, and an update to that location (to create a dot) [0,0]

    Expected Output: True, a dot exists for that location [0,0]

    Output: True

    JUnit Test Result: PASS

12. UT12

Type: Unit, Static, Automated

Initial State: Board is created with game entities, player and score objects are created

Input: Score values to increase by ([1000],[1313232],[0])

Expected Output: Updated score value after each addition ([1000],[1314232],[1314232])

Output: Score updated successfully ([1000],[1314232],[1314232])

JUnit Test Result: PASS

13. UTF1

Type: Unit, Dynamic, Automated

Initial State: Application is in gameplay state

Input: Score addition (16000); High score update; Score addition (4000); High score read from file

Expected Output: High score not affected by score addition within game (should be read as 16000)

Output: High score remains as updated (16000)

JUnit Test Result: PASS

5 Changes Due to Testing

6 Automated Testing

7 Trace to Requirements

8 Trace to Modules

9 Code Coverage Metrics