SE 3XA3: Test Plan
Dinododger

Team 39, S.R.A Squad
Shrey Mittal, mittas1
Razan Abujarad, abujarar
Zhiwen Yang, yangz18

27 October, 2017

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	Version	Notes
2017/10/27	1.0	Creation of Test Plan document
2017/12/6	1.1	Final revision (rev1)

1 General Information

1.1 Purpose

The purpose of this document to describe SRA Squad's plan to test the ~~DinoDodger module, User Interface Module and Animation Module~~ UI module, PlayScene module. The implementation was done before test cases were created such that all aspects of the implementation can be tested with the knowledge of the functionality of DinoDodger in mind.

1.2 Scope

The scope of DinoDodger is to entertain desktop users regardless of the presence of an internet connection. ~~The user interface and animation modules are integrated together in the DinoDodger module which can be ran using a desktop supporting Java 8.~~

1.3 Overview of Document

This document contains SRA Squad's test plan and description and unit test cases of the proof of concept and also comparison to the existing implementation (T-rex Runner)

2 Plan

2.1 Software Description

~~Out implementation of DinoDodger consists of 4 code modules: The character animation module, the background animation module, the user interface module and the DinoDodger module that integrates all of the modules together.~~ SRA Squad's implementation consists of the UI module which interacts with the PlayScene module and DinoDodger module that manages character and landscape selection choices. Other modules responsible for counting and updating points, creating ADTs for characters, cacti obstacles and pteradactyl obstacles interact privately with the PlayScene Module. Both character and pteradactyl use the SpriteAnimation module.

2.2 Test Team

The test team consists of the members: - Razan Abujarad - Shrey Mittal - Zhiwen Yang

2.3 Automated Testing Approach

See Sections 3 and 4 regarding automated testing where applicable.

2.4 Testing Tools

~~The primary module to be tested is the DinoDodger module which will be tested by writing JUnit test cases as well as some manual test cases. The animation and user interface modules will be tested manually.~~ Each module will be tested by introducing a temporary main routine within the module or, since all the modules are used effectively in the PlayScene module, manual testing will be done by disabling certain features to isolate what must be tested specifically.

2.5 Testing Schedule

~~The animation and user interface testing will begin with the completion of the functionalities and the DinoDodger module will be tested with the completion of the module and the integration of the other modules.~~ Click Here for a link to the Gantt File to see the testing schedule.

3 System Test Description

3.1 Tests for Functional Requirements

3.1.1 Gameplay testing

Jump Test

1. testid: JT-01

   Type: Functional/Manual

   Initial State: Character is ~~stationary~~ running

Input: ~~"Up" Arrow Key~~ Space bar

Output: Character's pixelated image ~~jumps to a certain height and returns to the ground level and starts running~~ Is transitioned along the y-axis from its initial y-position of 300 to 150, and back to 300 in 800 milliseconds

How test will be performed: Test will be performed ~~to initiate gameplay mode. The "up"~~ Within the PlayScene module once the UI module switches to PlayScene upon pressing the 'Play' button. All other features such as obstacles will be disabled. The Space key will be pressed to make the character jump.

2. testid: JT-02

Type: Functional/Dynamic/Manual

Initial State: Character is running and Obstacles are enabled

Input: ~~"Up" Arrow Key~~ Space bar

Output: Character's pixelated image ~~jumps to a certain height and returns to the ground level and resumes running~~ Transitions as intended in JT-01 and is able to overcome incoming obstacles

How test will be performed: ~~Test will be performed during gameplay mode when the character starts running. The "Up" key will be pressed as input to produce the reactionary "jump" output.~~ Same as JT-01 but with obstacles enabled.

3. testid : OCT-01

Type: Functional/Dynamic/Manual

Initial State: Character is running and Obstacles are enabled

Input: Spacebar

Output: Character falls on top of Obstacle such as a cactus or low-altitude pteradactyl and Gameplay ends. Game Over scene displayed consequently.

How test will be performed: The game will be played as normal until a low-level pteradactyl or cactus is encounted where the character will be made to jump such that it intercepts the obstacle when descending. This can be achieved by pressing the Spacebar before the optimal time to ensure a collision between the two.

4. testid: OCT-~~01~~ 02

   Type: Functional/Dynamic/Manual

   Initial State: Character is running

   Input: No input from user; program generates obstacles within the path of the character

   Output: Gameplay mode terminates once the character comes into contact with obstacle.

   How test will be performed: While the character is running during gameplay mode, obstacles are generated by the program that come in the path of the character. If not overcome, meaning if there is intersection between the pixelated images of the character and the obstacle, the gameplay mode will end.

5. testid: PT-01

   Type: Functional/Dynamic/Manual

   Initial State: ~~Character is stationary and initiatory jump command not yet executed~~ Play is pressed from main menu and Gameplay state entered with Character running.

   Input: ~~"Jump" commands by user as intended during normal gameplay mode~~ None

   Output: At top right corner, there is a ~~number display throughout the duration of gameplay that is auto-incremented constantly~~ numerical display of points and highscore, where points is incrementing at a rate of 200 milliseconds

How test will be performed: While the character is running during gameplay mode~~, number display denoting points accrued will be observed during game play to determine whether numbers are being incremented while the character has not encountered obstacles.~~ All other features such as obstacles are disabled. This test is to see if points does indeed increment over time.

6. testid: PT-02

   Type: Functional/Dynamic/Manual

   Initial State: ~~User re-enters gameplay mode~~ Gameplay stage with obstacles enabled

   Input: Spacebar "Jump" commands by user as intended during normal Gameplay mode~~; unlike above test, ensure the character survives for longer than the previous gameplay session to determine if highscore value increments once current point value exceeds it.~~ Mouse input to click "Play Again"

   Output: Gameplay mode terminates once the character comes into contact with obstacle. Upon collision between character and obstacle, when Game Over stage is entered, the resulting points and highscore values displayed upon the window will be observed.

   How test will be performed: To ensure the highscore is updated, it will be ensured the character ~~survives for longer than the previous gameplay session to allow more points to be accrued.~~ is able to run in the presence of obstacles Once this occurs, the character will deliberately be made to come in contact of an obstacle to end Gameplay mode. The highscore should be updated to reflect the greater number of points accrued. This test will be repeated two more times by pressing "Play Again"; the second time in such a way the character is made to collide sooner than before thus ensuring fewer accrued points. The result should not affect the first round's highscore result. The third time, the character will me made to collide after a longer time compared to the first time. As a result, the highscore should be higher as the points accrued should be higher.

7. testid: PT-03

   Type: Functional/Dynamic/Manual

   Initial State: Gameplay stage with all features enabled

   Input: Spacebar to "Jump" and Mouse input to click "Main Menu"

   Output: Game Over scene displayed when character is deliberately made to collide. Points and Highscore will be observed.

   How the test will be performed: The game will be played as normal and character will be made to deliberately collide to observe the Points and Highscore. After Game Over scene is displayed, "Main Menu" will be clicked. The Highscore value on the top right of the window will be observed to ensure it is indeed 0.

3.1.2 User Interface

8. testid: CST-01

   Type: Functional/Manual

   Initial State: User Interface appears when program is executed

   Input: Left-click input on two check boxes, one for character and one for landscape. (There are three options for each). Then clicking "Play" button

   Output: UI redirects to gameplay mode ~~with landscape and character displayed standing stationary.~~

   How test will be performed: Since there are three characters and three landscapes, there will be nine total combinations. This test will be carried out nine times with each combination to observe the result.

9. testid: CST-02

Type: Functional/Manual

Initial State: User Interface appears when program is executed

Input: Only Play button is pressed.

Output: UI redirects to gameplay mode with default landscape and character displayed standing stationary.

How test will be performed: Only the "play" button will be pressed so default combination is displayed.

3.2 Tests for Nonfunctional Requirements

~~Difficulty Test~~

3.2.1 Usability

10. testid: UT-01

Type: Dynamic/Manual

Initial State: User Interface displayed upon execution of program

Input/Condition: ~~Gameplay mode launched with initiatory jump command and then game played as intended using the jump "up" key~~ Space bar is input for jump; Mouse is cursor input to "Play Again" or access main menu

Output/Result: Game is played as intended by user

How test will be performed: Small group of test users will be asked to play the game and deliver their feedback on usability as a rating from 1-5, 1 being poor, to 5 being excellent.

3.2.2 Performance

11. testid: PFT-01

    Type: Dynamic/Manual

    Initial State: User Interface displayed upon execution of program

    Input/Condition: ~~Gameplay mode launched with initiatory jump command and then game played as intended using the jump "up" key~~ Space bar is input for jump; Mouse is cursor input to "Play Again" or access main menu

    Output/Result: Game is played as intended by user

    How test will be performed: Small group of test users will be asked to play the game and deliver their feedback on Performance as a rating from 1-5, 1 being poor, to 5 being excellent.

3.2.3 Look and Feel

12. testid: LFT-01

    Type: Dynamic/Manual

    Initial State: User Interface displayed upon execution of program

    Input/Condition: ~~Gameplay mode launched with initiatory jump command and then game played as intended using the jump "up" key~~ Space bar is input for jump; Mouse is cursor input to "Play Again" or access main menu

    Output/Result: Game is played as intended by user

    How test will be performed: Small group of test users will be asked to play the game and deliver their feedback on Look and Feel as a rating from 1-5, 1 being poor, to 5 being excellent.

3.2.4 Learnability

13. testid: LT-01

    Type: Dynamic/Manual

    Initial State: User Interface displayed upon execution of program

    Input/Condition: Space bar is input for jump; Mouse is cursor input to "Play Again" or access main menu

    Output/Result: Game is played as intended by user

    How test will be performed: Small group of test users will be asked to play the game and deliver their feedback on Learnability as a rating from 1-5, 1 being poor, to 5 being excellent.

3.3 Traceability Between Test Cases and Requirements

Traceability between test cases and requirements will be done by noting the existing functional and non-functional requirements agreed upon in the SRS document. Whatever non-functional requirements are applicable for this game will be assessed using individual opinions on how the game was implemented in terms of look and feel, performance and usability. Functional requirements wherever possible will be tested using JUnit, otherwise will be tested ~~using TestFx or~~ manually through observation.

4 Tests for Proof of Concept

For this project, it was agreed that the most challenging parts would be to ~~animate the character, animate the background, and create a UI~~ Animate the character and animate the obstacles and release them at certain time intervals, and implement a point counter which would auto-increment points, update and reset the highscore when appropriate.. Referenced in tests for

functional requirements are tests that include: JT-01, JT-02, CST-01, CST-02, OCT-01, OCT-02, PT-01, PT-02, PT-03 which cover these. Upon testing the cases mentioned, the Proof of Concept can be met ~~as they involve the animation of the character, a jump command, background animation which will contain the obstacles to be encountered, and a UI to select character and background from.~~

5 Comparison to Existing Implementation

The existing implementation of this project lacks a User Interface and different options to choose for landscapes and characters, thus the only comparable grounds are the animation of the character and background, jump commands, obstacle interceptions and point value increments. Based on comparable traits, DinoDodger compares well.
soutAs of now, the progress of DinoDodger in these areas is reasonable. If comparison to the original product was a possible test case, requirements including the animation of the background, animation of the character have been met.

6 Unit Testing Plan

~~Wherever possible, if a value is quantifiable and comparable, JUnit testing will be used through assertion commands. An example of this is verifying that the difficulty requirement through the increasing speed of the background animation is met as the speed is incremented and is comparable. TestFx will be used to test the functionality of the UI and its buttons and checkboxes to determine if correct output is produced. In this case, if a certain combination of character and background are chosen, boolean values of what has been chosen can be stored and assessed using event handling test cases. For any cases that cannot be quantified, independent manual assessment will be required to determine whether the requirement has been met.~~ JUnit testing is considered for DT-01. A form of manual Unit testing is also planned for JT-01 and PT-01 where the obstacle features will be disabled to only test a particular feature during runtime.

6.0.1 Difficulty

Difficulty test

- testid: DT-01

  Type: Dynamic/Manual/Unit

  Initial State: User Interface displayed upon execution of program

  Input/Condition: Gameplay mode launched ~~with initiatory jump command~~ when "Play" is clicked from main menu and then game played as intended using the jump ~~"up" key~~ Space bar command

  Output/Result: Character jumping over obstacles as game intends; game progressively gets difficult with ~~speed~~ the increasing speed of the obstacles.

  How test will be performed: Initial speed be stored in a variable. Using initial time and a designated end time(s), the updated speeds of the pteradactyl and cacti will be compared to the initial speed to ensure the updated speed is greater. This procedure can be asserted using JUnit.

6.1 Unit testing of internal functions

~~Unit testing of internal functions cannot be done individually since the functions control a UI animation output. In order to test important functions like selection character and background, jump command and function to determine whether the character comes into contact with a obstacle, the existing functional requirements in section 3 can satisfy these.~~ Unit testing in practice can only be done effectively by disabiling certain modules within the PlayScene module. Like in JT-01 where obstacles are disabled, similar methods will also be used in PT-01 and DT-01.

6.2 Unit testing of output files

Not applicable

7 Appendix

7.1 Symbolic Parameters

none

7.2 Survey Questions

Usability: How would you rate the usability level of this game? Select 1=lowest, 5=highest. 1 2 3 4 5

Learnability: How would you rate the learnability level of this game? Select 1=lowest, 5=highest. 1 2 3 4 5

Performance: How would you rate the performance level of this game? Select 1=lowest, 5=highest. 1 2 3 4 5

Look and Feel: How would you rate the look and feel of this game? Select 1=lowest, 5=highest. 1 2 3 4 5