2015 International Conference on Automation, Cognitive Science, Optics, Micro Electro-Mechanical System, and Information Technology (ICACOMIT), Bandung, Indonesia, October 29–30, 2015

Mobile Game Testing: Case Study of A Puzzle Game Genre

Arlinta Christy Barus, Roy Deddy Hasiholan Tobing, Dani Novita Pratiwi, Siska Adelina Damanik, Jenny Pasaribu
Faculty of Informatics Engineering and Electro
Del Institute of Technology, Laguboti, Indonesia
Email: arlinta@del.ac.id

***Abstract*—The advancement of mobile technologies allows developers to build more sophisticated application specifically mobile game. As the result, mobile game has more complex gameplays and incorporates more complex resources, such as 2D images, 3D objects, artificial intelligence, sounds, and many more. Developers are spoilt by the vast number of available game engines that enable rapid mobile game development. However, the mobile game source codes that orchestrate all the interconnected resources are also getting large in the size and eventually leading to difficulties in tracing bugs in the codes. Meanwhile, software testing is still a growing area that, despite vary proposed techniques to identify errors in source code have been developed, yet it has to consider more studies in mobile game category. This research aims to conduct an appropriate testing on a mobile game application. We choose to test an awards-winning mobile game using an available testing tool, named Unity Test Tool. The tool is explored during this study and then applied to test the game that of puzzle genre. At the end of the study, the research finds that the game is faults/bugs-free.**

Keywords—mobile game testing; Unity tools; unit test

I. INTRODUCTION

Mobile application has specifically been designed and developed for devices that have more limited resources compared to personal computers or laptops [1]. Mobile application, that is also more well known as "app", runs on devices that are so-called smartphones or tablets which relatively have smaller screen size, processor speed, memory capacity, internal storages, and mostly have on-screen keyboards. Available apps can be used to play music, watch movies, set timers, send emails, call friends, and for various more serious purposes such as education, productivity, medical, shopping, and entertainment. In entertainment section, game is one of the most favorite categories. According to recent Quartile 1 (Q1) 2015 surveyed by Mobomarket, game is the most downloaded mobile app, which has 45.61% shares, followed by social and photography categories [2]. Moreover, the size of game industry is very large that in US alone. It is predicted that mobile game revenue will reach 3.31 billion US dollars in 2016. The number has been raised up from about 2.03 billion in 2013 [3]. The large scale of mobile game market and revenues eventually attracts more game developers. Mobile game rapid development method is used to develop and deliver ready-to-play game application in relatively short time. The method involves game engines in the process of game implementation.

Game engine allows developers to build a mobile game rapidly as the software has been already equipped with the building blocks for mobile game app, such as collision and physics resources, 3D game assets, game renderers, artificial intelligence, visual effects, sounds, and other sub systems. The game developer can tailor the available assets from the game engine, develop, and add specific resources to the games, and finally bind them by writing program codes. However, the developers need to make sure the game can be distributed, deployed in the users' gadgets and run smoothly without errors or bugs. This requires a mobile game testing process.

In general, software testing is a set of activities conducted to evaluate the quality of the product from the software requirements perspective. The software under test must provide all the already defined functionalities or capabilities, match the required workflows or algorithm, and return no errors. IEEE Standard for Software Test Documentation [4] emphasizes that testing is a process to detect the difference of developed software and the features requirements for that software. Two commonly used approaches for testing process are black-box testing and white-box testing [5]. However, the approach to test a mobile game can be different from the common software. Generally, software has functions that whenever be executed, the users can expect the results will be similar linearly. In contrast, mobile game usually comprises of interacting sub systems and continuous game loop. Hence, the results can be different for each playing session whenever the user gives inputs to the game. This situation can be noticeably experienced from game with artificial intelligence and applies physics laws to the gameplay.

In this research, the authors conduct a series of activities for mobile game testing. The testing process uses a both nationally and internationally awarded mobile game, XYZ, as a case study. The game was developed by the authors' university business unit and of the puzzle genre. As the game was built using Unity game engine [6], the study utilizes the testing tools equipped in the engine, named Unity Test Tools. The testing package has three testing components, which are Assertion Explorer, Unit Test Runner, and Integration Test Runner [7]. This research focuses on the mobile game unit testing and makes use of Unit Test Runner of Unity Test Tools. The reason is that XYZ source code has never been

978-1-4673-7408-8/15/$31.00©2015 IEEE

tested in a proper manner prior to its participation to the competitions. Unit testing will reveal if there exist errors in structural level of the code. To achieve the results, the research is conducted systematically by:

- Gathering information activities for the testing process references. The objective of this phase is to have better comprehension on testing process in general, mobile game testing, and the testing approaches and tools. The collected information is in form of journal or conference papers, and articles from the Internet.
- Exploring activities to familiarize the authors Unity game engine, Unity Test Tools, and the case study subject. During this step, the authors also conduct testing on a small scale mobile game.
- Testing hands-on for the case study subject. This third step applies collected knowledge from the previous steps.

The following section gives the literature study related to this research. Section III presents detail experiments conducted in this study. Results of the experiment along with some discussion are presented in Section IV. Last section gives the conclusion of the study.

II. Literature Review And Tools Exploration

A. Mobile Game

Game is different from software in general. Game comprises of many interacting sub systems which run continuously during the game is played by users. It is called game loop. Meanwhile, mobile game can be defined as "embedded, downloaded, or networked games conducted in handheld devices such as mobile phones, portable consoles, and PDAs" [8]. The definition itself implies the game characteristics. The examples of mobile games are Angry Bird, Candy Crush Saga, Clash of Clans, and Teka Teki Saku. The games can be deployed in Android and iOS smartphones platforms. To develop a mobile game rapidly, developers may use game engines. According to Gregory [9], game engine refers to "software that is extensible and can be used as the foundation for many different games without major modification". The typical game engine architecture is shown by Fig. 1.

There are several techniques that can be applied for game testing, such as [10]:

1) Combinatorial Testing. The technique aims to discover defects by using small test sets. In this technique, the individual elements are included in the test, and then create homogenous and heterogeneous testing types.

2) Test Flow Diagrams. In this technique, the game behaviors from the users' perspective are modelled into graphical representation. The testing process by exercising and exploring the possible paths is depicted in the model that comprises of flow, events, actions, states, primitives, and terminators elements.

3) Cleanroom Testing. The testing process in this approach is conducted by testing the game based on the assumptions of how the players will play the game. The test cases are generated based on the data of users' tendencies.

4) *Test Trees.* In this testing technique, tree structures of the game features are documented and decomposed into test cases.

5) Play Testing and Adhoc Testing. These are less structured test ing techniques by having the targeted players of the game are recruited, play the game and give feedbacks, mostly on the topic of game balance and difficulty. The test method also allows the testers to intuitively explore the game [11]. Several play testing services are available at the Internet, such as Playtest Cloud [12] that applies crowdsourcing method in the process or Playtextix [13] which has setup the test implementation stages.

Meanwhile, the types of testing methods specifically for mobile game testing that have been developed are: functional testing, compatibility testing, performance testing, localization testing, regression testing, and load testing [14].

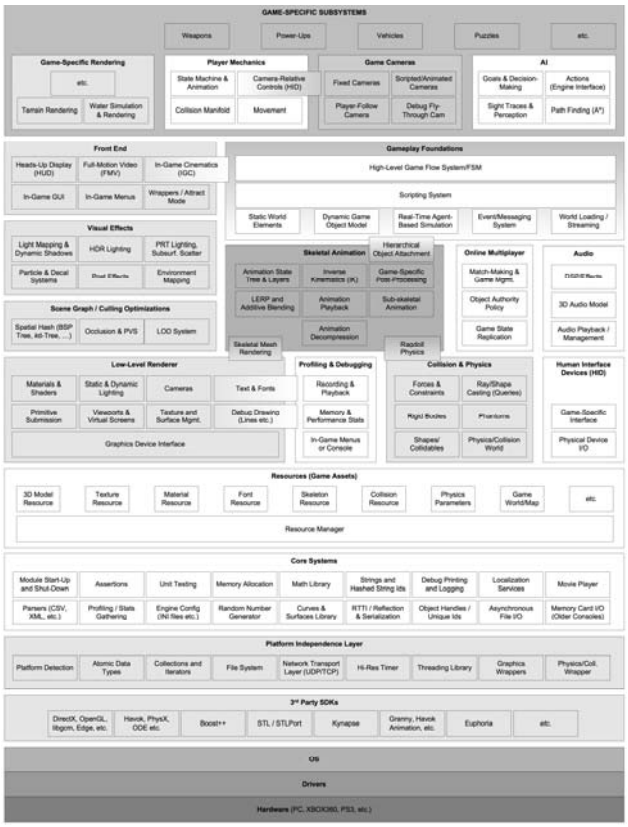

Fig. 1. Typical Game Engine Architecture [9].

For this testing research, XYZ mobile game is used as the case study subject. XYZ is an award-winning mobile game developed by the authors' university business unit. The puzzle game has never been tested in a formal way by the developers. Hence, this research is to help the developers to find faults in the source code in a good manner. This game has gameplay in which the players are expected to finish more than 80 stages and aim to get highest score. In each stage, the players can win

the game by removing trash from a lake. The actor of the game is a fish from local species that has a bubble gun to lift garbage from the lakebed. In the attempt to clean the lake, the players can face challenge from the obstacles set in each game stage. The snapshot of the game is available in Fig. 2. Additional information about XYZ mobile game is:

- The mobile game is exported to Android platform with 2.3.1 (API 9) as the minimum version. The file size is approximately 38.21 MB.
- The latest version of the game is 1.3.0.

Fig. 2. XYZ mobile game.

B. *Unity Game Engine and Unity Test Tools*

In this research, the authors have tools exploration to get more knowledge on Unity game engine and its test tools. The knowledge is used to develop test case and apply testing methods that are supported by the testing tools. The process of tools exploration can be seen in Fig. 3.

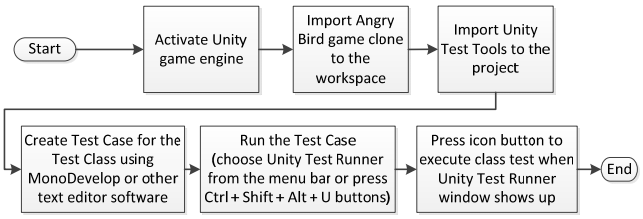

Fig. 3. Tools exploration workflow.

Unity is a commercial game engine used to develop game for web, desktop and mobile platforms. Developers use Unity to work on the gameplay. Meanwhile for writing the game source code, Unity has editor such as MonoDevelop. During the development process, the developers write the code without concern on the platform types. After the code is written, the developers can export the game into particular platform [15]. Unity has packages that can be downloaded to enhance its capabilities. One of the game engine packages that can be used for testing purpose is Unity Test Tools [16]. The tools can be used to have combinatorial testing and have three testing components that can be used to discover mobile game bugs, which are:

1) Integration Test Framework. The component allows the users to automate the assets verification and test the assets behavior or the interaction between the assets.

2) Assertion Component. This visual component is used to confirm the states of various game objects.

3) Unit Test Runner. The component integrates NUnit framework with the editor and le the developers to execute unit testing in Unity. The component provides test runner for execute test case and report the results. The following Fig. 4 shows Unity Test Tools with Unit Test Runner component.

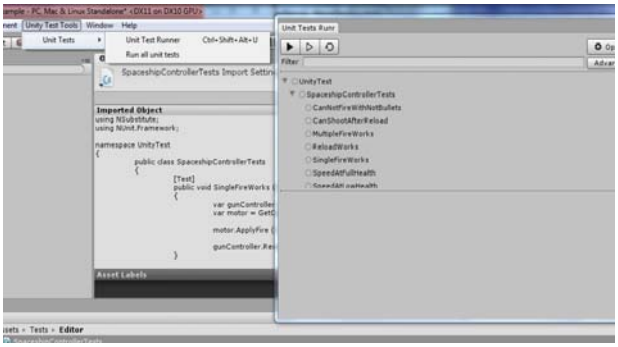

Fig. 4. Unity Test Tools.

III. THE EXPERIMENTS

Before carrying out the study, the tool exploration has been performed by having Angry Bird game clone source codes downloaded from the Internet [17] and makes it as the test subject. The testing process must follow rules which are:

1) Test class should use [TextFixture] tag.
2) Each test method has to have [Test] tag.
3) Each test has to contain three components:

a) Arrange: Variable initialization for tested data.

b) Act: Invoke the code from the tested game.

c) Assert. Comparison method for evaluating whether the tested function is accepted or not.

The code in Fig.5 is the example of Angry Bird Clone source code that has been modified using the abovementioned rules in order to generate test case. After applying the rules to the source code, the next steps are:

- Instantiate the tested `Pig` class.
- Initialize the `Health` variable value to be `160f`.
- Call the tested method, which is `Start` method.
- Use `Assert.AreEqual` function to check if the expected result equals the real result, which is `130f` (`160f - 30f`).

```
using System;
using System.Collections.Generic;
using System.Threading;
using NUnit.Framework;
using UnityEngine;

namespace UnityTest
{
```

```
    [TestFixture]
    internal class SampleTests
    {
        [Test]
            [Category("Success")]
            public void SuccessTest()
            {
                    var pig = new Pig ();
                    pig.Health = 160f;
                    pig.Start ();
                    Assert.AreEqual(130f,
pig.Health);
            }
     }
}
```

Fig. 5. A part of Modified Angry Bird Clone Source Code.

The environment for testing process is as follows:

1) Hardware specification

a) Processor: Intel Pentium 4 Core i3.

b) Memory: 4.00 GB RAM.

c) Hard disk: 500 GB.

2) Software specification

a) Operating system: Windows 7 Professional.

b) Tools: Unity, Unity Test Tools, MonoDevelop.

c) Programming Language: C#.

IV. Results And Discussions

XYZ mobile game building blocks has several components, which are: (a) `Rigidbody`, to manage the gravitation attributes of an object, (b) `Box/Sphere Colider`, to create the physical surface of the game objects, and (c) `Script`, to define the methods of each object. For the testing process, the following program classes are set up to be the test class:

1) `GelembungController`. The class is for creating graphical effects, managing the physical behavior of the actor's weapon. The class has 21 methods.

2) `PoraController`. The class is for creating graphical effects, managing the physical behavior of the game actor. The class has 12 methods.

3) `PenghalangController`. The class is for creating graphical effects, managing the physical behavior of the game obstacles. The class has 7 methods.

4) `SampahController`. The class is for creating graphical effects, managing the physical behavior of the lake trashes. The class has 10 methods.

After reviewing each class and method, there are six methods prepared as the test cases. The criteria for selecting the methods is based on the fact that other methods interact with game engines sub systems to render images, sounds and provide artificial intelligence. Hence, the methods are not easily tested using the Unit Test Tools because the dependency with other game resources and results domain is very large. The process of testing the test cases is shown in Fig. 6. From the testing results, the authors can conclude that the methods are bugs free (marked by the green checkmark).

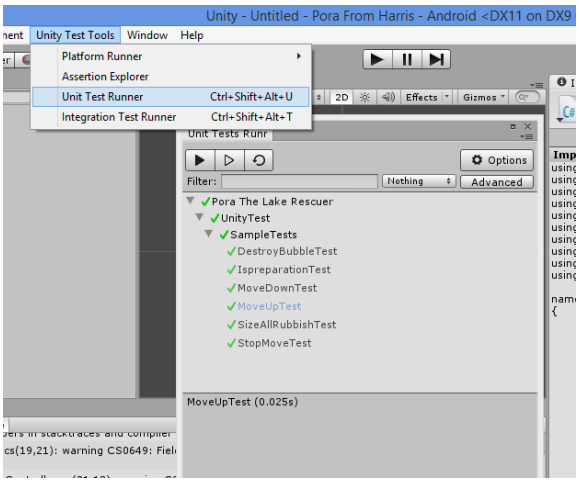

Fig. 6. Test cases run on Unity Test Tools.

During the testing process, the authors also tried to test whether the test case is reliable by entering inputs that do not match for the required condition. We found out that the Unity Test Runner shows the tested methods are error or not bugs free. The depiction can be seen in Fig. 7 (marked by red circle).

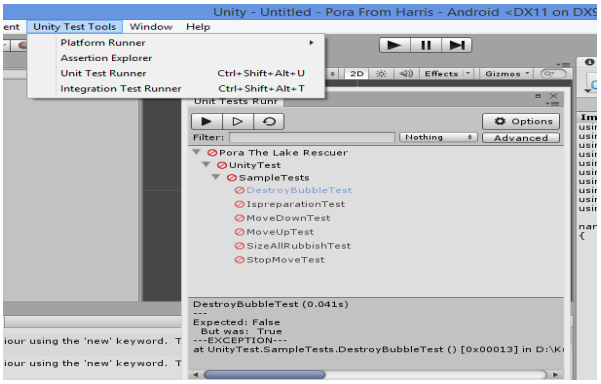

Fig. 7. Test cases run on Unity Test Tools With Errors Condition.

V. Conclusion

In this research, a mobile game testing is conducted using Unity Test Tools on XZY mobile game. In the process, it is found that not all methods in the source code can be easily tested. The nature of mobile game software that consists of interacting sub systems makes several methods cannot be separated for unit testing purpose. For future work, we suggest to study further about mobile game testing as there have not been many research works conducted on mobile game testing yet. There are still many aspects of mobile game testing that can be investigated further because of the unique properties of mobile game application compared to other common software. As this study has been merely explored unit testing of mobile application which is included as a functional testing method, other testing methods can be further explored in the future,

such as compatibility testing, performance testing, localization testing, regression testing, and load testing.

REFERENCES

[1] I. Wigmore, ”Mobile app.” Internet: http://whatis.techtarget.com/definition/mobile-app. Dec. 2013, [Jun. 1st, 2015]

[2] MoboMarket, “Q1 2015 Indonesia mobile data report based on MoboMarket users data research,” Internet: http://www.slideshare.net/BaiduIndonesia/q1-2015-indonesia-mobile-data-report-based-on-mobomarket-users-data-research. Apr. 13, 2015 [July 2nd, 2015].

[3] Statista, “Mobile gaming revenue in the United States from 2013 to 2016 (in billion U.S. dollars),” Internet: http://www.statista.com/statistics/269831/mobile-gaming-revenue-in-the-united-states/. [Jul. 2nd, 2015].

[4] IEEE, “829-1983 - IEEE standard for software test documentation,” Available: http://faculty.ksu.edu.sa/mohamedbatouche/SWE%20434/IEEE%20Std%20829%20-%201998.pdf. [Jun. 2nd, 2015].

[5] G. J. Myers, “The art of software testing 2nd edition”, New York: John Wiley & Sons, 2004.

[6] Unity, “Unity game engine,” Internet: https://unity3d.com. [Jun. 22nd, 2015].

[7] Unity, “Unity Test Tools,” Internet: https://unity3d.com/learn/tutorials/modules/beginner/live-training-archive/test-tools. Jul. 24, 2015 [Jun. 1st, 2015].

[8] E. J. Jeong, D. J. Kim, "Definitions, key characteristics, and generations of mobile games," 2009. Available: http://www.irma-international.org/viewtitle/26508. [May 11th, 2015].

[9] J. Gregory, “Game engine architecture,” Massachusetts: Wellesley, 2009.

[10] R. B. Charles, P. Schultz, T. Langdell, “Game testing all in one,” Boston: Thomson Course Technology, 2005.

[11] J. Gibson, Introduction to Game Design, Prototyping, and Development, Addison-Wesley, 2015.

[12] Playtest Cloud, “Mobile game usability testing,” Internet: https://www.playtestcloud.com. [Jun. 1st, 2015].

[13] Playtestix, “Playtestix,” Available: http://playtestix.com/files/2e83d55a81eaa64bc71379f802853d79.pdf. [Jun. 1st, 2015].

[14] V. Helppi, “Mobile game testing – part #1: the importance and difference from app testing”, 2014. Internet: http://testdroid.com/tech/mobile-game-testing-the-importance-and-difference-from-app-testing. [Jun. 9th, 2015].

[15] Unity, “Unity,” Internet: https://unity3d.com/unity/multiplatform. [May 11th, 2015].

[16] Unity, “Unity Test Tools,” Internet: https://www.assetstore.unity3d.com/en/#!/content/13802. Dec. 18, 2013 [May 11th, 2015].

[17] Github, “Angry Bird clone,” Internet: https://github.com/dgkanatsios/AngryBirdClone. Jun. 19, 2015 [May 12th, 2015].