Automatic Functional Testing of Unity 3D Game on Android Platform

Huixian HU [1,a], Lu LU [2,b]

[1,2] Department of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China

[a]Katherinehhx@foxmail.com, [b] lul@scut.edu.cn

Keywords: Android, Functional testing, Unity, Software testing, Game engine

Abstract. As Android games plays a big role in humans' daily amuse activity, the rapid growth of mobile games has led to the constant development and update which have resulted to the heavy loan of testing. Unity is one of the most popular game engines that establish a more effective rendering of game scene. In this paper, we put forward a feasible solution to execute a whole procedure of automatic functional testing of Unity game on Android platform. The solution is exploring the coordinate systems of the Android platform and Unity to address the component-based testing with less manual cost.

Introduction

A fast-growing demand of mobile games for Android devices caters to the need of recreation. To efficiently develop mobile games is bound to propose supplement of game engine designed for the creation and development of video games [1]. The continual development and update of mobile Android games will lead to the heavy loan of software testing. Nevertheless, mobile game testing is distinct from traditional mobile application testing since the tested objects are usually end products in virtue of the corporations between Android SDK and 3D game engine [2].

This paper designs an auxiliary tool in the initial stage of the Android game functional testing developed With Unity. The following paper is organized like this: we demonstrate the latest research on the Android testing theory and the mobile game development in the next part. In the third part, we will present the automatic functional testing on the mobile game constructed by the Unity Android platform. Later in the fourth part, a simulation will be conducted to show that the solution put forward in the previous part is valid. After that, the paper will discuss on the result of the experiment raised before and testify if the testing method can serve as a complementary tool in the functional testing in the fifth part. In the end of the paper, we will conclude our work on the issue and raise new problem in the future research.

Background

To address the challenge of the automatic functional testing on Android game developed with the help of Unity engine, we should review on the working mechanism of creating a game in this circumstance.

Game engine's rendering module is critical to the immersive experience brought form the three-dimensional graphics of a mobile game [3]. It provides powerful and abundant support of post processing effects and particle materials. To create vivid interaction in animation, it as well extends physical and mathematic logic related models which visibly make it easier and more efficient to develop a game that behaves physically realistic.

To minimize the coupling degree among distinct threads in the Java program, encapsulation can be a superb practice in the design of testing method. We introduce testing JAR file under the same directory of the game Android project, which can accept the required parameters and protect the encapsulation.

Test automation is achieved through composite methods of the following aspects [4]: test suits generation, awareness of context [5], recording and replaying. GUI testing is being widely used in the mobile application testing [6]. Generating test cases before testing is most time consuming and generally demands great manual work without test automation. Hence, it comes first to ponder over the ways and means of adaption of Android games by Unity. What's more, the testing technique must be sensible and smart of what's going on while the game runs.

Unity Android Game Testing Technique

To accomplish the automation of software functional testing, there are three inevitable steps to accomplish : Record the test scripts, execute the recorded scripts and replay the test procedures. To realize the purpose of capturing the operation on both the controls of Android and Unity3D game engine, the recording module are supposed to figure out what controls are triggered and trace the operating sequence while the testers are exploring the game applications.

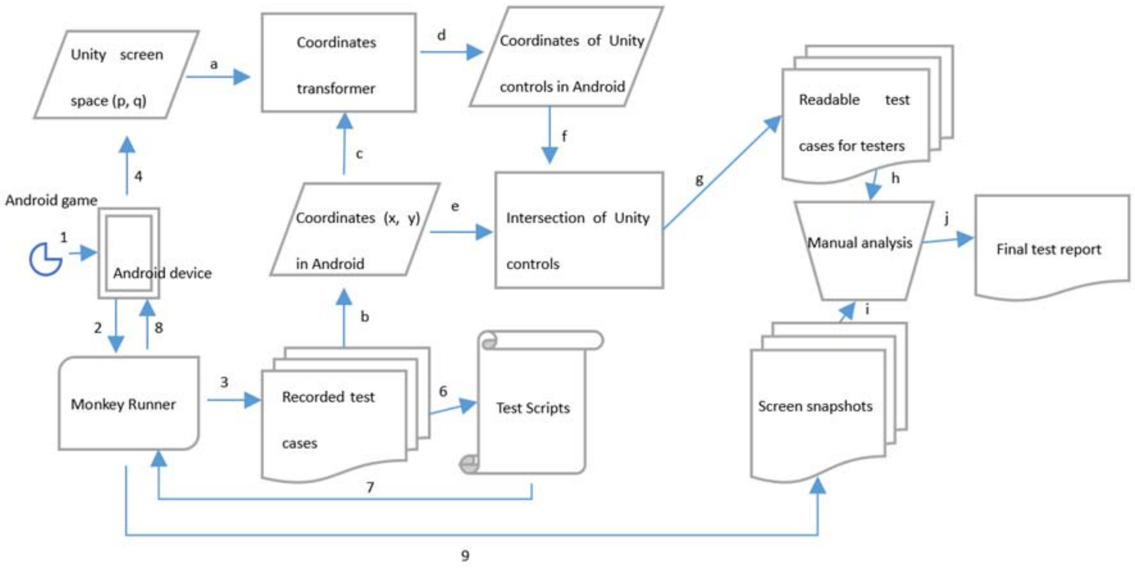

Fig. 1. Workflows of test method

The testing procedure lists and executes in Figure 1.

0)	Add scripts to the test broadcaster listener in Android to record clicks in the Unity and monitor the states of the controls or components added to the game project.
1)	Launch Monkey Runner [7] prepared to record testing procedure.
2)	Run the Unity game on a smart phone or virtual device, execute testing cases according to the functional requirements.
3)	Search the path of the file system and get the files giving the coordinates of button components in screen space.
4)	Map the Unity coordinates into Android screen coordinates by executing location transformation to extract operations related to controls of Unity3D.
5)	Interpret and analyze the testing result into readable testing report

To minimize the influence brought by the testing modules, the capture module is bound to the test object by adding scripts to the test broadcaster to Android, which records click in the Unity. One thing to note is that Unity has its own screen space [8], in other words, coordinate system. The purpose of designing a broadcaster is acting as a listener to monitor the states of the controls or components which are added to the game project.

After the click information which track the coordinates of position represented in screen space in Unity is collected by the test monitor, it's encapsulated together into the Android project exported by the engine. When running the Unity game on a smart phone or virtual device, search the path of the file system and get the click.txt file and button.txt file. The button.txt file gives the coordinates of button components in screen space, too.

As the Monkey Runner is recording the touch event of the tested application, the mapping function is responsible for interpreting the attached file of delivered click position in Unity. Track after the position of the finger touching the screen of smartphone or the mouse interacting with Android virtual device. Collect both two kinds of coordinates for the latter mapping processing. The construction of mapping coordinates starts with the following definition:

The coordinates of Unity defined as U, and A for Android. Each pair of coordinates in Unity is presented as (P, Q) where $P = \{p_1, p_2, p_3, \ldots, p_n\}$ and $Q = \{q_1, q_2, q_3, \ldots, q_n\}$. Similarly, let Android coordinates presented as (X, Y) where $X = \{x_1, x_2, x_3, \ldots, x_n\}$ and $Y = \{y_1, y_2, y_3, \ldots, y_n\}$. After matrix projection, p_n and q_n should be in form with x_n and y_n. That is,

$$x_n = a_{11}p_n + a_{12}q_n + a_{13}. \tag{1}$$
$$y_n = a_{21}p_n + a_{22}q_n + a_{23}. \tag{2}$$

Where $a_{11}, a_{12}, a_{13}, a_{21}, a_{22}, a_{23}$ separately represents constant numbers to project Unity coordinates into Android coordinates.

Consult into the attach file to compare with the different coordinates system, use the matrix projection [9] to map the Unity space screen into the Android screen coordinates. We can transform Eq.1 and Eq.2 into the form of matrix as below:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p \\ q \\ 1 \end{bmatrix}. \tag{3}$$

To compute $a_{11}, a_{12}, a_{13}, a_{21}, a_{22}, a_{23}$ we make use of both the two known coordinates' data pairs. Let matrix X denote the unknown vector, we have A = X·U. That is,

$$\begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_1 & p_2 & p_3 \\ q_1 & q_2 & q_3 \\ 1 & 1 & 1 \end{bmatrix}. \tag{4}$$

Where (x_n, y_n) and (p_n, q_n) separately represents coordinates in Android screen and Unity screen. According to definition of inversion of matrix computing we have A·inv(U) = X.

After applying the results of the fourth step, we get the transformed coordinates of Unity controls in Android screen. By locating the click operations, if the current position hit any of the target controls in Unity engine, a reprocessing can be carried to interpret the coordinates into controls in Unity. Hence, we can judge if the current position interest with the controls of Unity. By use of automation android test tools, testing engineers can analyse the screen shots together with the modified files recording the Unity controls to revels errors in Unity game engine.

Experiment and Result

In this section, we conduct a simulated experiment to verify if the automatic functional testing method on Android is feasible. The tested game is a simplified demo developed under Unity game engine. The main controls in this game are the four buttons controlling the trail of the small ball. The four upper buttons are standard button controls in Android platform. To visualize the testing processes, texts in black display the controls being clicked.

According to the test result, the testing platform can serve as supplementary module to MonkeyRunner to complete an automatic workflow on Android game applications developed under Unity3D engine. Table 1 lists the coordinates of test cases in the tested application.

Tab. 1 Coordinates of Unity Buttons

Column	Matrix A	Matrix U	Matrix X
Test case 1	$\begin{bmatrix} 391 & 133 & 378 \\ 746 & 885 & 890 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 391 & 133 & 378 \\ 534 & 395 & 390 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 1280 \\ 0 & 0 & 1 \end{bmatrix}$
Test case 2	$\begin{bmatrix} 94 & 222 & 193 \\ 864 & 872 & 901 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 94 & 222 & 193 \\ 416 & 408 & 379 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 1280 \\ 0 & 0 & 1 \end{bmatrix}$
Test case 3	$\begin{bmatrix} 378 & 598 & 423 \\ 733 & 896 & 877 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 378 & 598 & 423 \\ 547 & 384 & 403 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 1280 \\ 0 & 0 & 1 \end{bmatrix}$

To ease the heavy burden of test engineers, the automatic test solution proposed in this paper optimized the existing test procedure by interpreting the numeric coordinates into readable text as labels of Unity controls. The broadcaster of Unity controls can insert the operation to Android application concurrently with the recording thread of MonkeyRunner. The optimized test script also includes the instructions that will automatic run in the execution of test scripts. The following is one of the original test cases recorded.

By applying the projection method proposed in the previous section, the test cases recorded by MonkeyRunner can be modified and clearly indicate the components of Unity. In this example, test cases figure out all the buttons related to Unity, which reveal errors introduced by the game engine.

After the improvement to the recorded script, the execution module sends simulated instructions to the virtual devices based on the sequence of recorded operations. The snapshots of the execution result are shown as Figure 2.

```
UnityButton Button Up
WAIT {'seconds' 3.0 }
TOUCH {'x' 571 'y' 808 'type' 'downAndUp' }
TOUCH {'x' 135 'y' 1162 'type' 'downAndUp' }
TOUCH {'x' 108 'y' 757 'type' 'downAndUp' }
TOUCH {'x' 189 'y' 549 'type' 'downAndUp' }
TOUCH {'x' 623 'y' 533 'type' 'downAndUp' }
TOUCH {'x' 632 'y' 1080 'type' 'downAndUp' }
TOUCH {'x' 519 'y' 1178 'type' 'downAndUp' }
UnityButton Button Right
```

There is still a long way to go, improve the efficiency, protect the security of game engine and intelligently analyze the testing result report and locate the bug.

Conclusion

This paper firstly introduces the current situations of the development of Android mobiles games with the help of Unity3D. To find out a workable method of the target problems, we focus on the message delivery formats of Android and the invoking of components in Unity3D. Based on the mechanism of the three aspects of game architecture, namely, data, logic and rendering, we tentatively suggest a flow of processing the automatic method of revealing the errors occurs while the game runs. We then conducted a simulation experiment to verify the effect of the automatic functional testing and analyzed the result of generating reasonable test cases and the revealed bugs.

The main contributions of this paper can be summarized as below:

- Put forward an innovative research issue of the automatic testing on Android game based on Unity3D platform.
- Collating of references of architecture of games and mechanisms adopted in the implementations of mobile games developed with game engine.

- Proposed a feasible solution addressing the challenge of automating the functional testing of mobile games, utilizing the graph theory flexibly.

Meanwhile, the coupling degree of the path finding should be reduced so that the sequences of scene can be greatly decreased. What's more, the closeness to practice is not satisfied since we didn't simulate the situation where the game runs concurrently, that is, whether and how the performance of the mobile devices affect the functional behavior are unknown. Another existing problem to be addressed is the reusability of the test suit. The future research on the issue of mobile game testing can be study in depth in the several directions:

- Learn from other mature framework applied by Robotium or UnityTestTools to propose a feasible open framework of automatic testing on mobile game functions.
- Combine the relevant content in the field of AI (Artificial Intelligent) to generate the method of training the testing script to control the traverse path while executing the testing tasks.
- Work on the testing task allocation algorithm in depth to dispatch loan on different devices running the game scripts.
- Improve the reusability of the test suit and cut down the cost of maintenance.

The intrinsically intention is to facilitate the social development by reducing the manual input in the manufacturing of various industries. We are doing our utmost to enhance the closeness of theory and practice. With test infrastructures, one symptom of high maintenance costs is the establishment of a "tools team" to maintain the infrastructure.

Acknowledgements

Project supported by the National Nature Science Foundation of China (No. 61370103), Guangdong Province Science Application Major Fund (No. 2015B010107001), Guangzhou Produce & Research Fund (201508010057) and the Open Fund of State Key Laboratory of Wuhan University Software Engineering.

References

[1] Anderson, Eike Falk, et al. "The case for research in game engine architecture." Proceedings of the 2008 Conference on Future Play: Research, Play, Share. ACM, 2008.

[2] Schwabe G, Göth C. Mobile Learning with a Mobile Game: Design and Motivational Effects. Journal of Computer Assisted Learning, 2005, 21(3):204-216.

[3] Kurome H, Noda S, Hayasaka S, et al. Rendering Geographic Datasets with 3D Game Engine – Dealing with Compatibility Issues. Japanese Journal of Ophthalmology, 1993, 37(2):143-7.

[4] Muccini, Henry, Antonio Di Francesco, and Patrizio Esposito. "Software testing of mobile applications: Challenges and future research directions." Automation of Software Test (AST), 2012 7th International Workshop on. IEEE, 2012.

[5] Schilit, Bill, Norman Adams, and Roy Want. "Context-aware computing applications." Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on. IEEE, 1994.

[6] Amalfitano, Domenico, Anna Rita Fasolino, and Porfirio Tramontana. "A gui crawling-based technique for android mobile application testing." Software Testing, Verification and Validation Workshops (ICSTW), 2011 IEEE Fourth International Conference on. IEEE, 2011.

[7] Information on http://developer.android.com/guide/developing/tools/monkeyrunner

[8] Blackman, Sue. Beginning 3D Game Development with Unity: All-in-one, multi-platform game development. Apress, 2011.

[9] Gentle, James E. Matrix Transformations and Factorizations. Matrix Algebra: Theory, Computations, and Applications in Statistics. Springer. 2007. ISBN 9780387708737.