퀄컴 증강 현실 SDK 를 위한 시뮬레이터[1]

친 통 탄*, 강대기 (교신저자)**
*말레이지아 멀티미디어대학교 정보기술학과
**동서대학교 컴퓨터공학전공
e-mail : johnny.tan.shinto88@gmail.com, dkkang@dongseo.ac.kr

Simulator for Qualcomm Augmented Reality SDK

Chin Tong Tan*, Dae-Ki Kang (corresponding author)**
*Faculty of Information Technology, Multimedia University, Malaysia
**Division of Computer and Information Engineering, Dongseo University, South Korea

Abstract

A simulator in Android is developed for Qualcomm Augmented Reality (QCAR) software development kit (SDK). The main purpose is to replace the testing of application on actual Android device. Bugs in the application can be found easily when testing is done in the simulator (with support of Android emulator) before the testing on real device. The simulator does not require a camera in testing augmented reality application. Works included study on QCAR SDK's behavior is done to ensure that the simulator performs similar to the SDK. Description on how would the simulator works with QCAR SDK is included in the paper.

1. Introduction

An augmented reality (AR) project is received lately and study on related field is done. Platform of the application is decided to be Android and the software development kit to be used will be QCAR SDK by Qualcomm[3]. Since Android platform is chose, mobile devices will be targeted area for this application.

QCAR SDK works as a marker detector to detect the existence of marker in the camera view. Once the marker is identified, the center of the marker will be labeled as the origin for object modeling purpose. In modeling, Open Graphics Library for Embedded Systems (OpenGL ES) is written in Java Native Interface (JNI) for QCAR SDK.

Here comes the problems, for augmented reality, computer vision is used to detect the environment or marker in the view. A camera is always needed to detect the marker so that object rendering on the marker could start. When testing is needed, camera is necessary and Android emulator does not provide this function. Installation of camera on Android emulator and the computer with a camera as vision is required as well.

While testing is done in Android emulator with a static camera, moving of the marker is required to check the effect of augmented reality from different view. Since the camera is static, moving the marker around the camera view is essential. For a mobile augmented reality application, altering the orientation of the camera is expected instead of altering the marker. Although they made no different for a computer, but this does affect the experience of user while using the augmented reality application especially for game. The same effect does go to the programmers while developing the application.

The other difficulty for testing on real device is the detection of the marker. The camera has to be placed within an acceptable range of distance from the marker in order to spot the features on the marker. It can neither be placed too near as lack of focus function of some camera, nor too far as the features would not be noticed. With this limitation, there is no way to view a rendered object too close or too far, the tracker's features would not be even detected to start the rendering.

With these problems existed in learning phase during the development of augmented reality project, a solution to solve these problems is considered, that is building a simulator. Running the augmented reality application on the simulator will reflect similar experience of running the application on an actual device with the camera on. The main purpose of this simulator is to make the job of programmer simpler while testing an augmented reality application.

2. Methodology

The simulator is developed in Android platform using Java. Minimum supported version of Android for QCAR is version 2.1, but the version 2.2 is chosen for development. Even though the simulator is built in Android, it is not meant to run in an actual Android device. Only actual mobile device with physical keyboard attached on it is usable, because keyboard is one of the requirements to control the simulator.

There is a reason to select Android version 2.2 instead of version 2.1, only Android version 2.2 or later can use OpenGL ES 2.0 without any support from Android Native Development Kit (NDK) [1]. In point of fact, Android version 2.0 has started support OpenGL ES 2.0 however they must be written in the JNI and these codes has to be built by Android NDK. JNI and Android NDK are not part of the

[1] 본 연구는 지식경제부의 지원을 받는 동서대학교 유비쿼터스 지역혁신센터의 연구 결과로 수행되었습니다.

methodologies used in the simulator development. Since JNI is not used, OpenGL will be coded in Java platform instead of JNI.

OpenGL ES Tutorials for Android [2] is studied as well to understand the simple way to use OpenGL ES for Android. Methods learnt are reused in the development of the simulator.

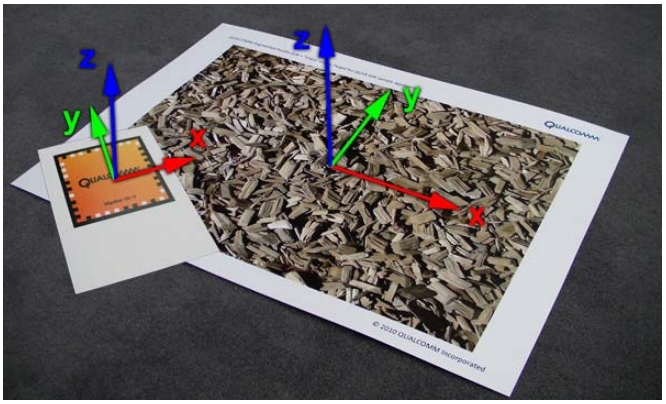
QCAR SDK version 0.10.0 BETA 2 is the tool used for augmented reality application development. QCAR SDK uses JNI in object rendering. QCAR SDK can track up to five single markers at a time. In this simulator, only tracking of one marker at a time is used.

Sample project from QCAR SDK named ImageTargets is studied. Development of the simulator is based on this sample project. Since the simulator is working for this sample project, it will be working for any other project that uses single markers tracking.

3. Simulator Description

3.1 Coordinate System

The coordinate system used in the simulator will be the same as QCAR SDK (as illustrated in figure 1). In QCAR, the marker will be automatically set as the xy-plane and the camera will be place at +z axis (blue line), looking (pointing) at the marker. The absolute z-axis direction the camera pointing at will always be in negative.



(Figure 1) Coordinate Systems [3]

For OpenGL, the camera is looking at –z axis and placed at the origin. Moving the camera towards +z axis without changing the orientation will make the camera view similar to the initial view of QCAR SDK.

Other coordinate system would be the same as OpenGL. Both the simulator and QCAR SDK are running in OpenGL also, meanwhile the standard of coordinate system would not be different.

3.2 Modeling Method

Model to be rendered are stored in few different data formats. They are three arrays of vertices, indices and texture coordinates. For vertices and indices, the same order is used for rendering in both simulator and QCAR SDK. For texture coordinates, the way of mapping texture onto an object for Java and JNI is different. In JNI, bottom left corner will be the starting coordinate of UV mapping and Java will be top left corner. Coordinates for y-axis is inverted before mapping of texture is done.

Every objects modeled in the simulator can be reused in QCAR JNI with the same list of vertices, indices and texture coordinates. Only modification on syntax and changes in way of calling need to be done when it is switching from one platform to another.

The marker for the simulator will be rendered to virtually show that the marker is detected like an actual augmented reality application. The marker will be rendered as a plane at the xy-plane. Texture of the marker is mapped on the same plane also. In figure 2, stone texture is mapped onto the xy-plane and a sample object, which is a dice rendered on top of it.

(Figure 2) Rendered Marker with Object

As recommendation, objects should be modeled in front of the xy-plane (the marker). Object modeled behind the xy-plane is not suggested as the object or part of the object that is placed behind the marker would not be hidden in QCAR SDK. In QCAR SDK, object rendered in the area of –z axis will be rendered to be further in QCAR SDK. In the physical world, everything is supposed to be placed on top or in front of the surface. If object is placed underneath the marker, it should be hidden as the marker is blocking the object physically.

3.3 Interaction Design

As explained in methodology, keyboard is part of the tools needed so it will be the interaction tool for the simulator. Most of the smartphones designed with a wider touch display without any physical keyboard. Virtual keyboard is designed to replace the actual keyboard. The size of the virtual keyboard will be around a quarter of the whole display or more. When showing the virtual keyboard, some of the significant view might be blocked. Making the keyboard display temporary is not the case as the keyboard is needed frequently in testing phase.

The other purpose of not using the virtual keyboard is that the whole touch display is meant for user interaction to the augmented reality application, not the simulator. Virtual

keyboard on screen will block developer from fully control the touch screen with the application and this will reduce the experience of developer while testing the application. This is not how the simulator should act. The external keyboard provided in the Android emulator is used to replace the virtual keyboard.

What is the purpose of the keyboard since it is not essential in the actual Android device? It is meant to control the camera in OpenGL. Buttons on the keyboard is used to replace the direct movement of the camera in the actual mobile device.

There are total of ten keys to control the camera. Four of the ten buttons is for transformation of the camera such as moving the camera up, down, left or right. Then the other four buttons is for rotation. This is just the same as moving from left side to the right side or even from top to bottom while the marker is always looked at in the camera movement. There is one limitation set for camera rotation. Some of the orientations are restricted because these orientations are not attainable with ordinary movement of camera. The last two buttons are for zooming. Zoom in and zoom out is the same as moving closer to the marker or further from it.

For the user interaction with the augmented reality application through touch screen, developers will need to design it on their own. The touch screen is fully reserved for the augmented reality application only. Controlling the application on the actual device will be the same as designed in the simulator. No major changes are needed as they the interaction part should be both coded in Java.

3.4 Comparison with QCAR SDK

The simulator developed is not exactly the same as the QCAR SDK. One of it is the acceptance range for QCAR SDK to detect the tracker. The simulator does not have this problem because it does not need to locate any tracker. Every model can be viewed even it is closed to the tracker. Figure 3 shows that the dice (rendered object) can be seen closely. In figure 4, it is the closest range the camera can be placed from the tracker. If the camera and the tracker are placed any closer to each other, features of the tracker would not be noticeable and rendering would not be started.

(Figure 3) Model View in Simulator

(Figure 4) Model View through QCAR SDK

While rendering is done in the simulator with Android emulator, the emulator will sometimes have trouble in rendering few points that made the objects look unusual. To avoid this problem, testing on actual mobile device is recommended as the mobile device does not have this error.

4. Conclusion and Future Work

The simulator for augmented reality application is developed to make the job of developers easier while building augmented reality application. Simulator is designed for QCAR SDK platform. Studies of QCAR SDK led the simulator behave more like the actual QCAR SDK. Testing on augmented reality application in the simulator is easier compare to testing on actual mobile device, because the restriction in the simulator is less. Augmented reality application designed and tested in the simulator can be transferred to QCAR SDK effortlessly.

As for the future enhancement, the simulator can be modified to be usable in a mobile device as an alternative to run in Android emulator that has some weaknesses. It is unquestionable that improvement of the simulator will be needed throughout the development of the augmented reality application.

References

[1] Android Developer. (2009, September). What is the NDK? Retrieved March 24th, 2011, from Android Developers: http://developer.android.com/sdk/ndk/overview.html

[2] Bergman P.E. (2010, February). OpenGL ES Tutorial for Android. Retrieved March 22nd, 2011, from Jayway: http://blog.jayway.com/author/pererikbergman/

[3] Qualcomm Incorporate. (2011). Augmented Reality. Retrieved March 24th, 2011, from QDevNet: http://developer.qualcomm.com/dev/augmented-reality