Advanced Science and Technology Letters
Vol.51 (CES-CUBE 2014), pp.46-51
http://dx.doi.org/10.14257/astl.2014.51.11

Android App for Smooth Multimedia Recording Service via the Frame Buffer

Sang-Min Seo[1], Hyeon seok Oh[1], Yoon-Ho Choi[2]

[1] Department of Computer Science, Kyonggi University,
443-760, Suwon-Si, Gyeonggi-Do, Korea
drone1256@naver.com
[2] Department of Convergence Security, Kyonggi University,
443-760, Suwon-Si, Gyeonggi-Do, Korea
ychoi@kyonggi.ac.kr

Abstract. Existing Android screen recording applications (hereafter, app) are dependent on the codec of the recording function. In this paper, we propose a new app that records the multimedia file at high speed for smooth multimedia recording service. Through a test, it is shown that the proposed app can record images by as much as over 20 frames per second without changing the encoding method.

Keywords: mobile device, android app, jni, frame buffer, recording service

1 Introduction

In recent years, various mobile operating systems have been developed, and various mobile devices including smart phone have been equipped with computing performance, the capacity of memory devices, and the I/O speed that are similar to those of general PCs. Mobile applications (hereafter, apps) that process sounds and images in mobile devices largely consist of the image recording module and the streaming module that transmits the output through streaming via the networks. In this paper, we propose a new app for recording images at a high speed.

To overcome the limitation that the quality of images degrades at the point of storing images, the proposed app, called the mobile screen video recorder using frame buffer (FB-MSVR), collects screen images from the Android mobile device by accessing the frame buffer (fb) via the java native interface (jni). Specifically, the screen information is collected by using the method of collecting images stored in the fb by using the Dalvik jni in the following order: JAVA -> jni -> C/C++ source code -> frame buffer within the Linux Kernel. Contribution of this paper can be summarized as follows.

- **Quick recording function**: By using the H.264 or MPEG-4 codecs that have been embedded from the lower versions of Android, the FB-MSVR can record the screens of Android mobile devices by using the jni in the Android frame buffer at a speed of 23~24 frames per second (fps) without being dependent on the Android version and codec.

ISSN: 2287-1233 ASTL

Table 1. Specifications of existing apps and the proposed app for recording video

Name	Recording		
	quality (fps)	Codec	Memory
screen cast video recorder [1]	Not operating normally		8.9MB
Screen video recorder [2]	24	MPEG-4, No `audio support	39MB
Z-screenrecorder [3]	Not operating normally		28MB
afreecaTV [4]	Operating after being connected with the desktop		
mobizen [5]	Operating after being connected with the desktop		
FB-MSVR	24	MPEG-4 AC-3	5.2MB

This paper is organized in the following way. In section 2, we overview the characteristics of existing screen recording programs. Section 3 shows operational sequence diagrams of the proposed app. In section 4, graphical user interface of the proposed app is shown. Finally, we conclude this paper in section 5.

2 Related work

In Table 1, we summarize the performance of the existing scree recording apps, which is currently used commercial in Android mobile devices.

Among existing apps, only the screen video recorder [2] recorded the screen at an average speed of 24 fps by using the MPEG-4. However, it provided neither audio nor streaming services. The screen cast video recorder [1] and the Z-screenrecorder [3] did not properly record. Also, they had the disadvantage of not providing streaming services. In the case of AfreecaTV [4] that is most widely used for real-time streaming service, both recording and streaming functions work only in connection with a separate desktop. Also, the streaming function of AfreecaTV [4] was supported by using only H.264 codec. In the same way that AfreecaTV provides both recording and streaming functions [4], the recording function of mobizen [5] was also provided after being connected with a separate desktop.

3 Overall operation of the proposed app

3.1 Characteristics of the proposed app

The proposed app records the audio and the screen of Android smart phones by using of the codecs that are basically supported by Android. Thus, it is possible to share smart phone screens and audios with other users through the streaming function. In addition, because the proposed app uses the flv file format for streaming service, it is

possible to view the recorded images and the audios on various devices.

The proposed app used FFMPEG open source libraries [6]. FFMPEG open source libraries supports 371 audio and video codecs including MPEG, H.263, H.264, and ACC, support various forms of encoding and decoding for images, and provides the hosting of the MPlayer project server. Therefore, FFMPEG open source libraries were used for encoding during recording and streaming. Encoding used the MPEG-4 and ACC codecs that are widely used in the recording function, and used the flv for the streaming function. The flv supported by Adobe Systems can maintain the quality of images while reducing usage. Therefore, it can take into account the amount of data consumption according to transmission via the wireless network.

3.2 Overall operation for recording screen information

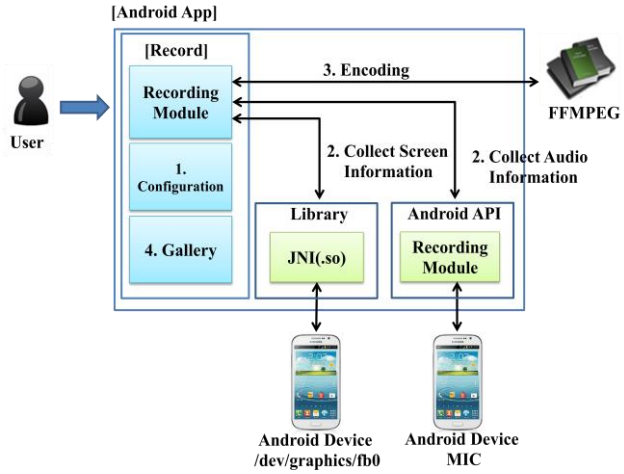

Fig. 1. Event flows for recording screen of mobile device

For recording screen of Android mobile device, the FB-MSVR stores the current screen information on mobile device into the SD card after encoding it in the video file format. As shown in Figure 1, the recording function operates in the following way.

1. The configuration module set the number of frames per second (fps) and the name of encoded file for recording the image.
2. Call the library that brings the screen information of fb0, which is connected with the recording module by the jni. By calling the "/dev/graphics/fb0" device driver of the Android device, the library collects screen information. Also, sound information is collected via the recording module by using the Android API.
3. After encoding the collected image and audio information in the video file format by using MPEG-4 through FFMPEG open source libraries,

the recording module stores it in the SD card with the given name, which is set by the configuration module.

4. Play the video file through the gallery.

3.3 Sequence diagram for recording screen information

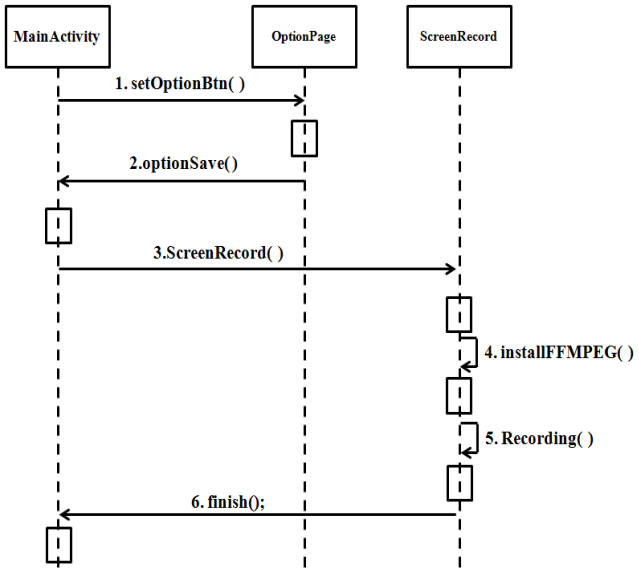

Fig. 2. Sequence diagram for recording screen information

In Figure 2, we show how to record screen of the Android mobile device. The operation of details is as follows.

1. After moving to the OptionPage by calling setOptionBtn(), set the environment for recording the screen.
2. Save values for environment setting by calling optionSave().
3. By calling ScreenRecord(), the proposed app operates in the background mode.
4. FFMPEG open source libraries are installed by calling installFFMPEG().
5. Screen information in the fb0 is collected by calling Recording(), and sound information is also collected by using the device's microphone via the basic Android image API. Such collected information are encoded in the format of MPEG-4 through FFMPEG open source libraries and are stored at the SD card.
6. After finish() is called, the recording service that has been operating in the background is terminated and completed.

4 Performance evaluation

After organizing the experimental environment in the following way, we measure the

speed for obtaining screen information of the mobile device by using the existing API and the jni. For the client side, we used the Android smart phone and for the server side, we used the following servers.

- **Client side**
  - **Android device**: Galaxy Note 2
  - **Development program**: eclipse juno 64bit
  - **Android version**: Jelly Bean 4.1.2
  - **Android NDK**: android-ndk-r9d-windows-x86_64bit
  - **Tool for checking codec information (the length of images, the type of codec)**: mediainfo [7]
- **Server side**
  - **Server OS**: Windows 7 64bit Ultimate
  - **Web Server**: Tomcat 7.0
  - **Streaming Server**: Red5 1.0.1

After measuring the number of frames for 10 second respectively, the average number of frames obtained per second was quantified.

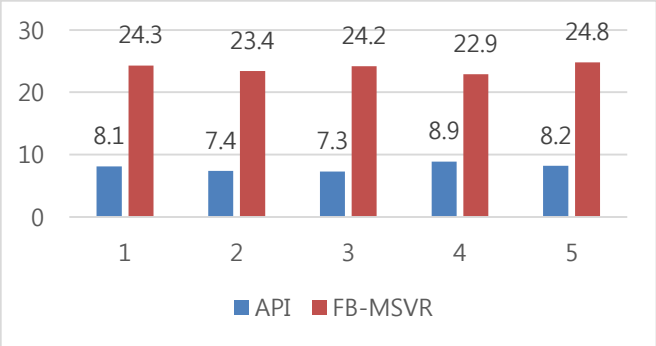

Fig. 3. Number of recorded frames

In Fig 3, it is shown that the average fps of the Android API was 7.8 fps and that of the FB-MSVR was 23.6 fps. That is, when screen information is obtained via the Android API, the fps was not fully met to support the real-time recording of screen information on the Android mobile device, but when obtaining screen information in fb0 via the FB-MSVR, the fps was over 20 frames per second, thereby exhibiting smooth images.

5 Conclusion

In this paper, we proposed the method of proving the streaming service of recorded screen information by using flash video files and verified its performance through testing. Our experiment confirmed that the proposed app can record images by as much as 20 fps in an average.

References

1. "Screen Cast video recorder,"
https://play.google.com/store/apps/details?id=com.ms.screencastfree
2. "Screen video recorder v3.0,"
https://play.google.com/store/apps/details?id=com.cocoapps.screenrecorder
3. "Z-screenrecorder,"
https://play.google.com/store/apps/details?id=com.zausan.zscreenrecorder
4. "AfreecaTV," https://play.google.com/store/apps/details?id=kr.co.nowcom.mobile.afreeca
5. "Mobizen," http://www.mobizen.com/
6. “Red 5,” http://www.red5.org/
7. “Mediainfo,” http://mediaarea.net/ko/MediaInfo

