Display technique of mobile video monitor on android

Yongxin Zhuang[a], Yang Li[b]

School of Information Engineering Guangdong University of Technology in Guangzhou, China

[a]727730919@qq.com , [b]Lyang@gdut.edu.cn

Abstarct

As the popularization of smart mobile device and the convenience it bring, the video monitor platform has changed to mobile client application. This paper develops the video monitor application based on Android. Through the research of traditional display method about network video data, we put forward a new bottom image processing method based on OpenGL ES. Accordion to the analysis and comparison of three methods, we prove that the method based on OpenGL ES is an optimized display method of mobile video monitor device.

Keywords: android; mobile client; video monitor; display method

Introduction

To solve the problem that traditional video monitor depends on wire and it can't be watched anywhere and anytime, developing network video monitor application on mobile device such as smart phone and tablet computer has practical significance. Nowadays, the Android OS which has the largest mounts of users is main platform of network video monitor. Developing a network video monitor client not only includes decoding of video data, adaptive control of video transmission, but also display module of video data. The display module show the video data in front of user, so the support of different frame rate of network video depends on the display method and the display module also directly affect the fluency and sharpness when user watch video. Through research on traditional display method based on ImageView and SurfaceView,this paper puts forward a display technique based on bottom image processing engine OpenGL ES and compares with each other on video fluency and resource occupancy. Through experiment data, we obtain a best method which has excellent display quality and performance.

1 Android graphics system

The graphics system in Android SDK includes two parts. The first part is the software package: android view, android widget, android graphics which based

on SGL(Skia Graphics Library).The other part is android.opengl which based on OpenGL ES. The view, widget and graphics based on SGL are applied on 2D drawing of applications. The OpenGL ES is subset of OpenGL and applied on 3D drawing, but its 2D drawing capacity is also powerful and support hardware acceleration. The surface is the cache of screen and store the drawing data. Skia can draw 2D image on surface and OpenGL ES can also draw 2D/3D image on surface[1].The graphics system as show in *Fig.1*.

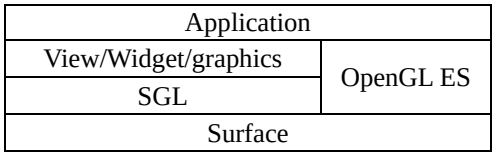

Fig.1 The components of android graphics system

Actually, the interface we see on android application belongs to View class. The View class provide the basic component which take charge of refreshing screen and dealing drawing event for user interface. As *Fig.2* show, ImageView and SurfaceView are subclass of View and they both has drawing capacity. View draws image in main UI thread, so if the drawing time is too long, then refreshing screen will block the main thread and result in the long-time non-response of program. If the main UI thread hasn't respond user's operation for more than 5 seconds，android platform will notice user whether to close the application[2].The difference from View is that ImageView and SurfaceView can create a new thread and draw image in it which avoid the non-response of program when the amount of data is too large and long-time drawing result in block of main thread. Traditional video module use ImageView and SurfaceView and this paper put forward a new method OpenGL ES based on the these two method which uses GLSurfaceView as top component。

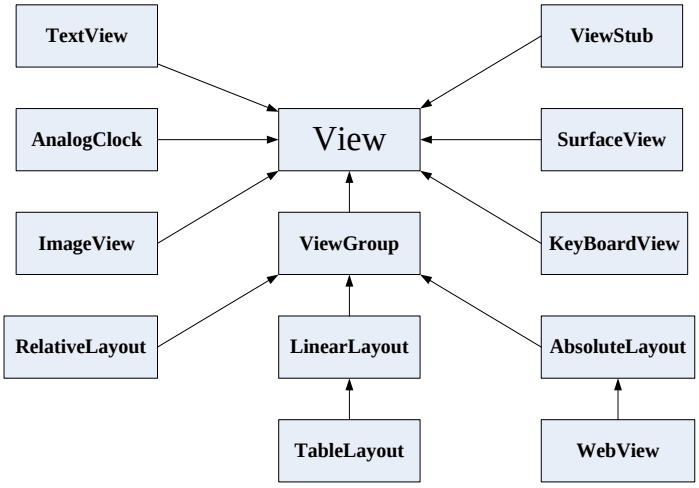

Fig.2 The View control and its child controls

2 Display technique of video data on android

2.1 Display method of ImageView

ImageView is a control which display image on android and support RGB image, so setting the data of ImageView every period can generate dynamic video. We can decode the H.264 data which obtain in the internet through RTP protocol as RGB data[3] and call the *decodeByteArray()* method from android.graphics.BitmapFactory to parse RGB data as Bitmap object which can be recognized by android[4] . The *setImageBitmap(Bitmap bm)* method of ImageView can directly draw bitmap. But through analysis of source code,we know that the method will call the father class-View class's *invalidate()* method, so that it can only call *setImageBitmap(Bitmap bm)* method in main UI thread and it will block thread easily. So Creating a class to inherit ImageView is necessary. We create a thread named postRunnable in the new class and call the *postInvalidate()* method in the thread's *run()* method. The *postInvalidate()* method will call *onDraw(Canvas canvas)* method automatically, so that we only rewrite this method and fresh screen. After this method, call Thread .sleep(long time) to let the current thread sleep. The sleeping time depends on frame rate of video stream, and *time=1000ms/frame rate*.Start thread postRunnable at last of the class's constructor[6] .

2.2 Double buffering display method of SurfaceView

The difference from ImageView is that SurfaceView has double buffering mechanism. This means SufaceView has two buffer: front buffer and back buffer. These two buffers are displayed on screen alternately[7].The advantage of double buffers is that it can avoid the screen flicker when the current image hasn't been drawn and screen is forced to draw another image in the large frame rate video image.

In the first we should create a class to inherit from SurfaceHolder.Callback interface of SurfaceView[8].Because SurfaceView also inherit from View, we need to rewrite the *onDraw()* method. Utilize the *decodeByteArray()* method of BitmapFactory to parse data as Bitmap object. This is the same as the method in 3.1.SurfaceView has to run outside the main UI thread, so we should create a drawing thread named onDrawRunnable.The mission of onDrawRunnable is parsing and drawing image. Drawing image means it write the data to back buffer and the double buffers are put to screen alternately. The data in buffer need to be saved in a memory named Surface and displayed on screen through a synthesizer named SurfaceFlinger. Although Surface store the image data of current screen ,we can't access it directly. We can obtain SurfaceHolder through the *getHolder*() method of SurfaceView and call the *lockCanvas*() method of SurfaceHolder to get Canvas.Then we can use the *drawBitmap*() method of Canvas to modify Surface.

the Fig.3 is the process of drawing Surface by Canvas[7].Because the data in memory is shared with every thread, Surface is shared in these two thread. To avoid that main UI thread also operate Surface when our thread draw image,we need to lock Surface before Drawing. After calling *lockCanvas*() method, we can

obtain Canvas object and at the meantime, SurfaceView will gain a synchrolock of Surface. The synchrolock won't be release until call *unlockCanvasAndPost*() method which will switch front buffer and back buffer and at last submit to Surface.

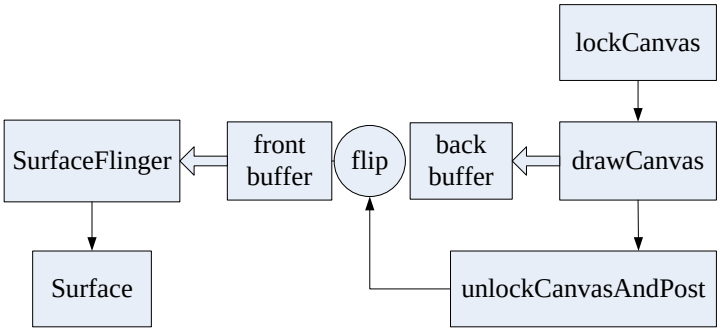

Fig.3 The principle of Canvas drawing

3 Video display technique based on OpenGL ES

Altough the ImageView method above is easy to be implemented, it doesn't use double buffering mechanism and lose frame. Although the SurfaceView method has double buffer and avoid losing frame, its image processing performance is not enough. Therefore, this paper put forward a video display technique based on OpenGL ES.

OpenGL ES(OpenGL for Embedded Systems) is designed for mobile phone, PDA and game device. This paper uses C/C++ to process image by NDK(Native Development Kit).Through this method, we can implement all the graphic operation and image processing based on OpenGL ES in C/C++ code, and use NDK to compile the code into dynamically linked library(.so file) and package it into Apk file of application. Java code of application can load this dynamically linked library by JNI(Java Native Interface) and call relative method of GLSurfaceView control to display image. It will improve the efficiency of drawing and running speed of application[9].Although this method is complex than the two methods above, its performance is best among these three methods. At the meantime, OpenGL ES support the hardware acceleration which most devices are equipped with.

The following is rough process of this method:

(1) To use OpenGL ES 2.0, should add the following code in AndroidMannifest.xml

<uses-feature android:glEsVersion="0x00020000"/>

(2) Edit the C++ code about OpenGL ES and wrap it as interface of JNI. We use the texture technique of OpenGL ES to draw video image mainly, so need to call *glBindTexture*() method to bind texture and call *glTexImage2D*() method to transform video data into texture. Then call *glDrawArrays*() to draw image at last.

(3) Use Android NDK to compile C++ code into dynamically linked library,

and load this library in application.

(4) Create a class to inherit GLSurfaceView and initialize it including setuping ContextFactory and ConfigChooser and get the instance of EGL 2D.

setEGLContextFactory(*new ContextFactory*());

egl = (*EGL10*)*EGLContext.getEGL*();

setEGLConfigChooser(*translucent* ? *new ConfigChooser(8, 8, 8, 8, depth, stencil*) :

new ConfigChooser(5, 6, 5, 0, depth, stencil));

(5) Register the Renderer of GLSurfaceView.Renderer takes charge of calling API of OpenGL to render frame.To inherite Renderer, we should implement its three methods[10]:

*onDrawFrame(GL10 gl):*Every time draw GLSurfaceView will call this method, so that the drawing operation based on OpenGL ES through JNI was put here and then the screen will display video.

*onSurfaceChanged(GL10 gl, final int width, final int height):*It is a callback function when resize the video or the orientation of screen is changed. So that the zooming operation of video size was put here.

onSurfaceCreated(GL10 gl, EGLConfig config):It is called when creating GLSurfaceView, so that the initial operation of OpenGL was put here.

4 Experimental results and analysis

The testing devices: HTC G17 mobile phone (Qualcomn MSM8260 dual-core processor, Android 4.03, GPU acceleration support), video stream server based on Live555, webcam.

4.1 Fluency of image

Set the size of video image as 640*320 and frame rate as 30fps.Through testing, we find that the play speed of ImageView is faster than SurfaceView and OpenGL ES obviously.We add the drawing flag into drawing function of these three method and the result shows that ImageView has serious phenomenon of losing frame. As show in Fig.4, the curve of ImageView has many zero points which respect the serial number of lost frames. But SurfaceView and OpenGL ES both have no lost frames. The result shows that the reason ImageView's playing speed is faster than SurfaceView and OpenGL ES is that it doesn't draw every frame. ImageView draw another Image only after finishing displaying the last image. It will miss several drawing periods when waits for the finishing displaying the last image and then results in losing frame. Therefore, SurfaceView and OpenGL ES are more fluent than ImageView and we will feel incoherent about the displaying of ImageView. The GLSurfaceView which OpenGL ES uses inherits from SurfaceView , so it doesn't lose frame. Because the drawing operation of OpenGL ES is implemented using C++ code and it use GPU acceleration, the processing speed of OpenGL ES is faster than SurfaceView and also bring the more fluency.

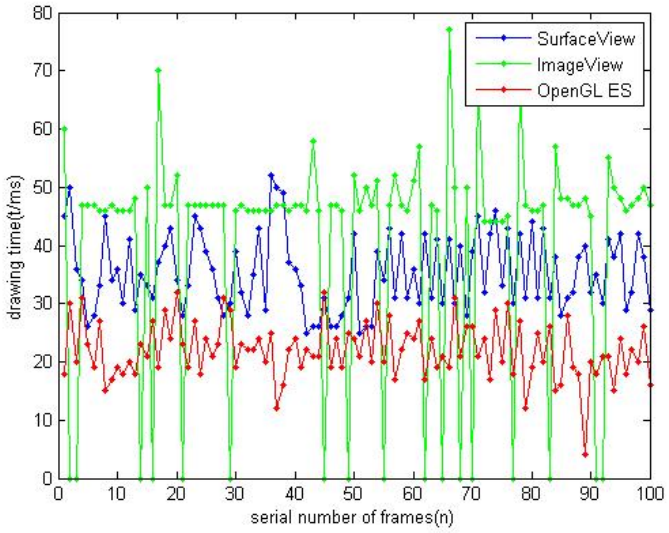

Fig.4 The drawing curves of three methods

As show in Table Ⅰ, when ignore the losing frame, we calculate the average drawing time of ImageView is 50.3ms which can't reach the image quality of 30 fps obviously. The average drawing time of SurfaceView is 38.9ms which is less than ImageView. The speed of OpenGL ES is fastest and its average drawing time is 21.9ms.Therefore, using OpenGL ES and NDK is the best displaying method to ensure image quality.

Table Ⅰ Drawing time of three methods

Displaying method	***Drawing time***
ImageView	50.3ms
SurfaceView	38.9ms
OpenGL ES	21.9ms

4.2 Occupancy rate of Resource

Android platform support multitask, so the resource of CPU and RAM is very important and related to the smooth running of system or whether the background service should be distroyed. The occupancy rates of CPU and RAM are shown in TableⅡ. ImageView's occupancy rates of CPU and RAM are highest in these three methods. Its occupancy rate of CPU is 75% and mobile phone is running unsmoothly. SurfaceView's occupancy rates of CPU and RAM are in the middle of three methods and these two rates are balance. OpenGL ES method has the lowest rate of CPU occupancy and its occupancy rate of RAM is just a little higher than other methods. In consideration of the situation that current mobile phone has a sufficient RAM resource relative to CPU resource. So the displaying method using OpenGL ES is the best method of three methods in occupancy rate of resource.

Table Ⅱ Occupancy rates of CPU and RAM

Displaying method	*CPU rate*	*RAM rate*
ImageView	75%	3.7%
SurfaceView	52%	3.3%
OpenGL ES	40%	5.6%

5 Conclusion

The result of experiment and analysis show that although the ImageView method is easy to be realized, but it has a serious phenomenon of losing frame and because of no using GPU to render, it occupies a higher CPU rate. For its synchrolock of canvas, the SurfaceView method doesn't lose frame. It has no screen flicker when play video with high frame rate because of its double buffering mechanism. But it also occupies a high CPU rate and its fluency of video image isn't well. This paper puts forward a displaying technique based on OpenGL ES. This method is best in video fluency, losing frame and Occupancy rate of Resource. This is because the OpenGL ES support hardware acceleration and its render code is realized by C++ code. So the OpenGL ES method brings a high running efficiency. Most image process are realized in C++ language so when android application need to process video image, we can use OpenGL ES and NDK to remove the operation code in C++ to our application easily without rewriting Java code. This reduces the developing period of application. Nowadays, The image decoding and adaptive transmission are mature in android network video monitor application. If the application utilize the optimal method this paper puts forward, then it will totally improve the performance of displaying.

References

[1] Bingfa Ye, Xiaohua Meng. Analysis and transplant of graphic system on android [J]. Telecommunications Science, 2010, 26(2): 64-69.
[2] The difference between View and SurfaceView on displaying system of android [EB/OL]. [2012-8]. http://www.2cto.com/kf/201208/147404.html
[3] Jing Wu. Application software design for webcam monitor system in android mobile platform[D]. Shanghai: Shanghai Jiao Tong University, 2012.
[4] Android Developers [K]. http://developer.android.com/guide/components/index.html
[5] Changgang Yang. Analyse Android system deeply [M]. Beijing: Publishing House Of Electronics Industry, 2011.
[6] Yafeng Wu, YaGuang Su. 3D game development on Android[M]. Beijing: Publishing House Of Electronics Industry, 2012.
[7] Haoran Cui. The design and realization of mobile game based on android platform [D]. Xi'an: Xi`an University of Science and Technology, 2012.
[8] Ya'nan Zhang, Lu Yang.Remote video surveillance system based on Android mobile phone [J]. Journal of computer applications, 2013, 33(S1) : 283-286.

[9] Youlu Wang, Daiping Li. Graphics development of android system based on NDK mode [J]. Computer Systems & Applications, 2012, 21(12) : 56-59.
[10] Vladimir Silva. Pro Android Games[M].
[11] Aaftab Munshi,Dan Ginsburg.OpenGL ES 2.0 Programming guide[M].America：Addison-Wesley Educational Publishers Inc,2008.