Research on Android Graphics Display of HDMI Interface on TV

Can Guo, Kun Liu* and Feng Yuan
College of Oriental Application & Technology, Beijing Union University, Beijing, China
*Corresponding author: wtliukun@buu.edu.cn

***Abstract*—This paper analyzes the problems when the Android graphic system is applied to TV box through HDMI interface. Overscanning is one problem, which will distort the image, especially in the district of the image edge. The method that re-adjusts the coordinates, height and width of each layer is proposed in this paper to solve the overscanning problem. The other is the resolution problem. Android graphics display system only supports the fixed resolution, but TVs have multiple resolutions. Adjusting graphics size according to resolution size of TV can solve this problem. Experiments have done on the two methods proposed and these methods can indeed solve the problems of overscanning and resolution when Android graphics display system applied to TV through HDMI interface.**

***Index-Terms*—HDMI, Android Graphic System, Overscanning, TV, Resolution**

I. INTRODUCTION

Smart TV will realize network search, IP TV, BBTV network visual communication, video on-demand (VOD), digital music, network news, network video telephone and other application services.TV is becoming the third kind of information access terminal after the computer and mobile phone. Users can access the information they need at any time.TV will also become an intelligent device to realize cross-platform search among TV, network and program. Smart TV[1] has a full open platform, equipped with an operating system. Users can enjoy the ordinary TV content and also can install or uninstall all kinds of application software to expand and upgrade the functions of the TV. Now users have two kinds of means to realize TV intelligence: One is to buy new-style smart TV directly. The other is to buy a TV box which has full open platform, an operating system, and runs on a high integration, high performance, low power consumption computer system chip. This box connects to the ordinary TV through HDMI interface. Obviously, the price advantage is the absolute advantage of TV box, and will be the main force of smart TV in the next few years.

The TV box[2] has a full open platform and is equipped with an operating system. It runs on a chip of a computer system with high integration, high performance and low power consumption. When the TV box is connected to an ordinary TV through HDMI interface, the intelligence of ordinary TV can be realized. But there are two problems of overscanning[3] and resolution when TV box is used. In order to improve display quality, android graphic display algorithm needs to be reformed. This paper analyzes these graphics display problems in detail, and puts forward the corresponding algorithms to solve the graphics display problems. Experiments show that the improved algorithm does solve the graphics display problems.

II. THE PROBLEM OF OVERSCANNING OF TV AND ITS SOLUTION

A. Overscanning of TV

Overscanning exists in each TV. There are two main reasons for overscanning: one is from TV manufacturers. As a result of price war, the profit of TV is very small, so TV manufacturer should reduce cost as far as possible. For this, the geometrical linear circuits of TV are simplified greatly, its geometrical linear is poorer commonly, and its images' deformation is very serious. In order to alleviate the users' visual suffering caused by geometric linear difference, manufacturers deliberately set the overscanning to be larger, and the area around the image on the outermost layer with the largest distortion is pushed off the TV screen, so the users can't feel the distortion of the image. The second reason is that maintenance personnel or users themselves increase the line and field amplitude adjustment in the TV maintenance process. These adjustments make the images too large, so the overscanning is produced.

For the overscanning, the image is more or less with geometric distortion, and the more obvious it is to the edge. The manufacturers adjust the overscanning to 5-10% of the picture, so the distortion can't be seen by users. For example, the image which size is 640*480 is magnified on the TV, so the area of 600*450 in the image can be seen on the screen and the outermost layer with distortion is outside the screen.

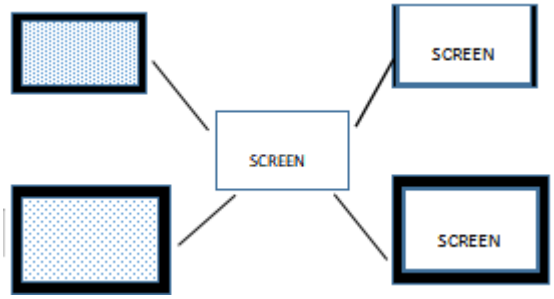

Figure 1. Original Image and Larger Image

As shown in figure 1, the image in the upper left corner is the original image to be displayed, the black

part of the edge is the overscanning image, and the middle picture represents the TV screen. The effect of the image displayed on the screen is like the image in the upper right corner, and obviously the overscanning part will also be displayed on the screen. The image in the lower left corner is the image being stretched and enlarged. The effect of this image on the screen is shown in the lower right corner. Because of the image enlargement, the overscanning part is stretched out of the screen and the user will not see this part of the image.

B. SurfaceFlinger

The basic structure of SurfaceFlinger[4] is shown in figure 2. The management objects of SurfaceFlinger include the followings: ISurface, ISurfaceComposer, mLayerMap, layersSortedByZ, FrameBufferNativeWindow, OpenGL ES[5], FrameBuffer[6], Display Controller. In SurfaceFlinger, each Surface corresponds to a separate graphics buffer in the system. ISurface and ISurfaceComposer are the interface instances which the client calls the SurfaceFlinger. mLayerMap is the management object that the Surface on the server side manages. layersSortedByZ is a Layer array that is sorted by the Surface's z-axis from the front to the back. FrameBufferNativeWindow is responsible for graphical display buffer output management. OpenGL ES[5] is a graphics library which responsible for 3D graphics calculation, image synthesis, etc. FrameBuffer[6] is a platform-specific graphic buffer. Display Controller is a platform-dependent graphics buffer controller that outputs the contents of the graphics buffer in a acceptable sequence to the external graphics devices. Surface is a description of this buffer, and it can be directly drawn on it by the method it provides. SurfaceFlinger's main functions are as followings:

●In response to the client's request, it creates a Layer connection to the Surface of the client.

●It receives client requests and modifies Layer properties, such as size, z-axis order, transparency and etc.

●It maintains the z-axis sequence of Layer and crop the final output of Layer.

●It displays the contents of Layers on the screen.

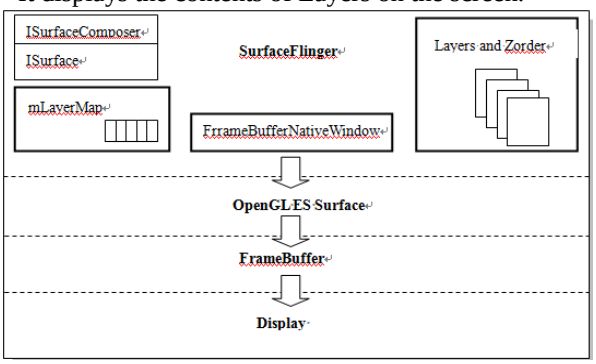

Figure 2. The framework of SurfaceFlinge

C. Trimming the Layer

Before SurfaceFlinger draws the Layer, it is necessary to calculate the visible area and the covered area of each Layer according to the z-axis sequence, and cut out the output range. The calculation principle is as followings:

1) It gets the initial visible area with its own coordinates, height and width.

2) It trims the area covered by the above window.

3) It draws the intersection area between the visible area and the area that needs to be refreshed of each Layer on the main screen from the deepest Layer of the z-axis.

As shown in figure 3, the figure in the first row of rectangle is the assumed case, the brown area is the screen, and blue areas are the layer to be displayed. The processing is shown in the three rectangles in the second row. The first step is removing the layer that does not cover the screen which is shown on the left image. Then it removes layers that are not on the screen which is shown on the middle image. The finally step is to divide the visible layer into rectangles which is shown on the right image.

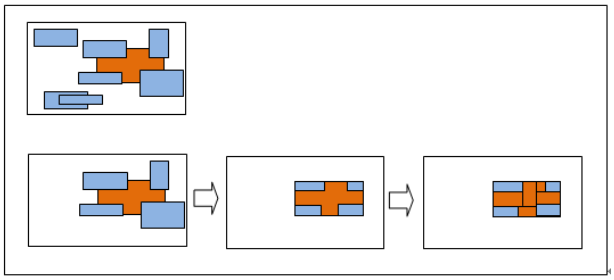

Figure 3. Trimming the Layer

D. Avoiding the overscanning of TV

For avoiding overscanning of TV, we take the screen as origin and reduce the image to a certain scale before outpuing it to the FrameBuffer, and then display the reduced image by controller. This scale value can be placed in the settings of Android as a controlling scanning parameter, and users can adjust this parameter according to the overscanning status of their TV.

When SurfaceFlinger generates a graphics, it calculates the coordinate, height, width and z-axis sequence of each layer and render each image one by one[7]. Therefore, it is necessary to adjust these information according to overscanning parameters when calculating trimming areas. Overscanning only affects the size and position of each layer, and has no affect on the z-axis, so we just resize the coordinates, height, and width. The entire graphics system of Android is based on OpenGL ES and the origin is located at the lower left corner of the screen rather than the center of the screen[8]. In order to achieve the effect of shrinking according to the center of the screen, coordinate is required to be transformed.

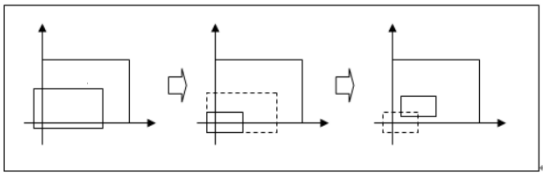

Figure 4. Layer size and coordinate transformation

As shown in the far left picture of figure 4, part of the layer will not be displayed on the screen. The scaling

process is shown in the middle picture. The layer is scaled with the lower-left corner as the vertex. The translation process is shown in the right picture. The X and Y coordinates of the scaled picture are translated, so that the areas that cannot be displayed on the screen can be displayed normally now. Each part of the image is processed in the same way.

The algorithm formula is as followings:

Float $\text{sfx} = (left/100.0 + right/100.0)/2.0$;

Float $\text{sfy} = (top/100.0 + bottom/100.0)/2.0$;

Float $\text{tfx} = (1 - left/100.0) * width * sfx/2.0$;

$\text{tfy} = (1 - bottom/100.0) * height * sfy/2.0$;

glTranslatef(tfx, tfy, 0); glScalef(sfx, sfy, 1);

After coordinate is transformed, recalculation is also required in Open GL trimming.

glScissor(r.left * sfx + tfx, sy + tfy, r.width(), r.height());

E. Experiments on the avoiding overscanning method

Figure 5 shows the display effect of the original image. It is found that part of the images on the four sides are not displayed on the TV screen.

Figure 5. Original image

Figure 6 shows the display effect after avoiding overscanning. The four corners and four sides of the image can be seen normally. The experiments proves that the method proposed in this paper does deal with the overscanning problem.

Figure 6. Image after avoiding overscanning

III. THE PROBLEM OF MULTI RESOLUTION

A. Multi resolution

Most CRT digital HDTVS on the market actually have the display resolution of 640*480, while HDTV LCD's resolution is generally 1280*720, and also some of them have the resolutions of 1920*1080, 1024*768, 720*576, 1280*1024 and etc. Android is currently a mobile operating system that only supports fixed-resolution displays, and it can not support multi resolution TV[9]. An improved algorithm is proposed in this paper to make the same image display effect at different resolutions. The image display effects at resolutions 640*480 and 1920*1080 are the same as that at resolutions 1280*720. So a TV box can be used to any TV with different resolutions.

B. Method for multi resolution

The real resolution is the resolution of the TV. The Android graphics system will draw image according to the resolution. All system resources are loaded according to the real resolution and then display on the screen. First of all, graphics layout is inconsistent. Due to the layout of each resolution is not the same, the size of the display is also different. Second, the higher the resolution, the smaller the font size. When it is about 3 meters away from the TV, it is almost impossible to see clearly, so it is not suitable for the market demand. Third, the higher the resolution, the less smooth the system is. Fourth, switching the resolution will restart Android. [10]. When Android starts, the Window layout system will be initialized according to the current resolution. If the resolution changes, the Window layout system will be re-initialized, which will restart the entire Android system and all applications opened by users will be closed. Lastly, software compatibility is poor. Currently, a variety of Android applications on the market are basically based on the development of small resolution mobile phone screen, in the case of large resolution, these applications either can not be installed, or display disorder[11].

False resolution is a fixed resolution(such as 1280*720) regardless of the TV resolution. All system resources are loaded in accordance with the fixed resolution. It enlarges or shrinks the images according to the real resolution and then output the image through the graphical system. A parameter named zoom parameter is used to adjust resolution.

Calculation algorithm of zoom parameter:

float sfx = TV resolution width/fixed resolution width;

float sfy = TV resolution/fixed resolution;

glScalef (sfx, sfy, 1);

OpenGL layer clipping recalculation:

glScissor (r.lft * sfx, sy * sf7, r.width() * sfx, r.height() * sfy);

C. Experiments on multi resolution

When the resolution is 1280*720, the image effectis shown in figure 7; when the resolution is 1920*1080, the image effect is shown in figure 8. The image display effects is obviously different under different resolutions. After the processing of the multi resolution algorithm proposed in this paper, when the resolution is 1280*720 and 1920*1080, the display will be the same as figure 7. Experimental results show that this algorithm has the same display effect for different resolutions.

Figure 7. Resolution of 1280*720

Figure 8. Resolution of 1920*1080

IV. CONCLUSION

This paper analyzes the problems of overscanningand multi resolution when Android graphics display system is applied to TV box. A controlling scanning parameter and zoom parameter are used to solve these problems. The false resolution algorithm perfectly solves the defects of true resolution. The method for avoiding overscanning greatly enhances the compatibility of TV box products. These improved algorithms remove the biggest obstacles encountered by android-based TV box in productization.

ACKNOWLEDGMENT

This work was supported in part by a grant from Education and Teaching Research and Reform Project of Beijing Union University in 2018.

REFERENCES

[1] Smart TV, https://baike.sogou.com/v63659681.htm.
[2] TV box, https://baike.sogou.com/v63171977.htm.
[3] Overscanning, https://baike.sogou.com/v58384491.htm.
[4] SurfaceFlinger,https://wenku.baidu.com/view/6cf7c912f18583d0496459c6.html.
[5] OpenGL ES, https://baike.sogou.com/v8783161.htm.
[6] FrameBuffer, https://baike.sogou.com/v64080557.htm.
[7] Render, https://baike.sogou.com/v64947147.htm.
[8] [8] Xin Duan, “An Internet television end-to-end fault demarcation system,” Information & Communications,vol. 7, pp.227-228, 2018.
[9] Liangfu Zhao, Rui Fu, Yachao An, Hongguo Qiu, “The design and realization of one test platform towards TVOS,”Video Engineering, vol. 7, pp.43-46, 2018.
[10] Dingjing Zhang, Ying Wang, Zheng Li, Wei Bai, Delin Chen, “Analysis of Package Model for NGB TVOS Java Application Framework LayerAPI,” Video Engineering, vol. 13,pp.114-117, 2015.
[11] Jian Dong, Yuan Zhang, Min Yang, “Utilizing Hardware Acceleration Layer to Optimize Android Display System,” Journal of Chinese Computer Systems, vol. 7, pp.1546-1550, 2012.
[12] Zhili Pan, Haibing Yin, Cheng Yang, “Design and Implementation of Intelligent Terminal for Receiving Broadcast Digital Television Programs,” Journal of China University of Metrology, vol. 3, pp.323-327, 2014.

Can Guo was born on January 1, 1996 in hebei province. Graduate student of Beijing union university. Her research interest is computer network.

Kun Liu（Corresponding Author） was born on February 5, 1980**.** Doctor of Computer Architecture, Associate Professor, Master’s tutor. Graduated from the JiLin University in 2016. Worked in Beijing Union University. Her research interests include cloud computing and big data.

Feng Yuan was born on January 10, 1993 in NeiMengGu province.Graduatestudent of Beijing union university.His research interest is computer network.