Matrix-Based Android UI Development

Song Zengbin
School of Mathematical Sciences,University of Electronic Science and Technology of China
ChengDu 611731, China

Huang Jin, Wu Guangxu
School of Mathematical Sciences, University of Electronic Science and Technology of China
ChengDu 611731, China

***Abstract*—Android system is a research hotspot in recent years, so a beautiful UI become more important. Matrix-based can make android UI development more easily and quickly ,and the software also run faster than developed by openGL_ES .In the paper, We first briefly introduce two-dimensional graphics geometric transformation, then combined android.graphics.Matrix to introduce it in detail . Finally, we introduce the use of android.graphics.camera to achieve 3D effect, and also give some codes and 3D screenshots of program.**

Keywords- Android; Matrix; 3D; UI; Camera; Computer Graphics;

I. INTRODUCTION

Android is Google's Linux-based platform for developing open-source mobile operating system. In recent years, anrdoid system become more popular, according to latest statistics released this year by a research firm named Ovum, the Android application market downloads will reach 8.1 billion. Android includes operating systems, UI and applications. 3D UI are also becoming more and more heat. Generally, there are two ways in Android UI development .One way is to use openGL_ES draw on the surfaceview directly. The advantage of this approach is the development of complex 3D animation,but complex developmen and complex interaction are the disadvantages. Software also runs slowly .The other way is directly use matrix to control the image drawed on the canvas. The advantage of this approach is making development easier, having a simple way to interact , and program run quickly. But when you expect a complex UI,this way will not work.

II. COMPUTER GRAPHICS BASICS

The nature and algorithms of matrix will not be mentioned in this paper, if you have any problems ,please refer to references[3].

A. Two-dimensional Translation

First look at a single example, now we suppose point P0 (x0, y0) move to P (x, y) after the translation,where translation vector in x direction is △x,and translation vector in y direction is △y. So, the point P(x, y) coordinates is:

x = x0 + △x
y = y0 + △y
A matrix expression of above is as follows:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x0 \\ y0 \\ 1 \end{bmatrix}$$

We found that when similarly translate image ,we only need to modify the two elements in top right corner of the matrix .

B. Two-dimensional Rotation

By specifying a rotation axis and a rotation angle, it can be a rotation transformation.

Now we suppose point P0 (x0, y0) change into P (x, y) after θ degrees rotation. By using the vector, we get the following:

x0 = r cosα
y0 = r sinα
x = r cos(α-θ) = x0 cosθ+ y0 sinθ
y = r sia(α-θ) = -x0 sinθ+y0 cosθ
A matrix expression of above is as follows:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x0 \\ y0 \\ 1 \end{bmatrix}$$

If the image is rotated around a point(a,b),we should first translate the image to the point(a,b),then rotate,and then shift the rotated image back to the original coordinates.

C. Two-dimensional Scaling

Change the size of an object can use the scaling transformation. We research the scaling transformation relatived to the coordinate origin. A simple two-dimensional scaling is done by multiplying the scaling factor sx and sy with object coordinates (x0, y0).

x=x0*sx , y=y0*sy
A matrix expression above is as follows:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x0 \\ y0 \\ 1 \end{bmatrix}$$

If the image is scaled relative to other points, similar to the rotation, first translate, then scale, last shift back to the original coordinates.

III. MATRIX USED IN ANDROID

In android Matrix is composed of nine float

978-1-4673-1411-4/12/$31.00 ©2012 IEEE

elements,which is a 3*3 matrix.As shown:

$$\begin{bmatrix} cosX & -sinX & translateX \\ sinX & cosX & translateY \\ 0 & 0 & scale \end{bmatrix}$$

Explain the above matrix. sinX and cosX is cos and sin values of rotation angles. Note that the rotation angle is calculated by clockwise. translateX and translateY mean the distance translated in x and y directions. 'scale' is the scale ratio, Such as 1 means unchanging, and 2 represents half scale.

An example is followed,

```
float cosValue =(float) Math.cos(-Math.PI/6);
float sinValue = (float) Math.sin(-Math.PI/6);
mMatrix.setValues(new float[]{
cosValue, -sinValue, 100,
sinValue, cosValue, 100,
0     ,     0     ,     2});
```

What the codes do?Scaling half, rotate 30 degrees counterclockwise,then along the x-axis and y-axis shift 50 pixels respectively. The reason of translating 50 pixels instead of 100 pixels is half-scaled.

Achieved results is followed:

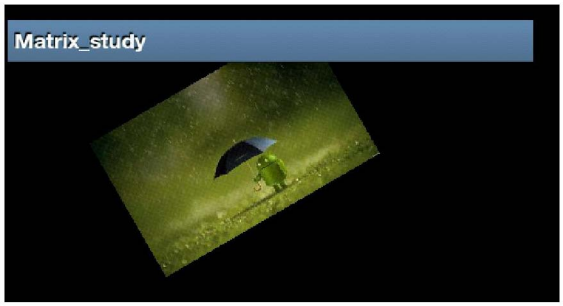

Fig.1 The matrix by direct assignment

Here the way(the direct assignment) talked about may be a bit difficult to understand, but fortunately, android provides a more convenient way to operate matrix.

There are four matrix operations in total: translate, rotate, scale and skew. In addition to translate, the other three operations can have a center point. Set, post and pre are three operating mode which provided by Android API for each matrix operations . The operation mode set will directly set the values of matrix ,once setted,the matrix values will be changed. The operation mode post is the current matrix multiply matrix given by parameter. To complete the necessary transformation ,you can use the post repeatedly. For example, an image rotated 30 degrees, and then shifted to point (100,100) , you can do as follows:

```
Matrix m = new Matrix();
m.postRotate(30);
  m.postTranslate(100, 100);
```

The operation mode pre is matrix given by parameter multiply the current matrix .If use pre to achieve the above example ,we can do it like this:

```
Matrix m = new Matrix();
  m.setTranslate(100, 100);
m.preRotate(30);
```

Rotate,scale and skew can be executed around a central point. If not specified, by default, they are built around point (0,0).There is an example.

```
//Will be scaled to 100 * 100
mMatrix.setScale(100f/bmp.getWidth(),100f/bmp.getHe
ight();
//Translate to point(100,100)
mMatrix.postTranslate(100, 100);
  //point(100,100)as a center, tilted in x and y-axis
mMatrix.postSkew(0.2f, 0.2f, 100, 100);
```

Achieve result is followed

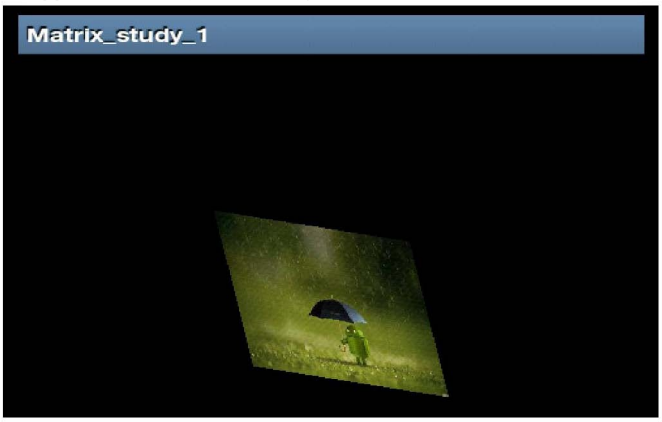

Fig.2 The matrix setted by Android api of Matrix

IV. CAMERA CLASS IN ADROID

Through the transformation matrix described above, we can achieve most of the animation. But in this way, most of them is 2D effect . If a three-dimensional animation to achieve and what to do?

Here I will introduce the Camera class which is a logical concept rather than a photographic camera. We compare phone's screen to camera window. Through this window, we see the contents of display—application's interface. If we look at the screen from different angles ,it presents a three-dimensional effect naturally. Such as a cup on the table, we see from the front it is a kind, from the rear is a sample,from above is another kind. The Camera class is used to do this. Moving forward in the Z axis of camera's perspective, the actual results is to enlarge the image. If move in the Y-axis, the picture moves up and down, and move in the X-axis, the picture moves left and right. For example: move 100pixels in the Z axis, the code is

```
mCamera.translate(0.0f, 0.0f, 100.0f);
```

The Y-axis rotation correspond to the picture in the vertical flip. The X-axis rotation correspond to the picture in the horizontal flip,and similar z-axis. For example the code of the Y-axis rotation is:

```
mCamera.rotateY(rotationAngle);
```

All transformation has done. Then you can get the matrix like this,

```
mCamera.getMatrix(imageMatrix);
```

You can use the matrix to draw the image on the canvas to achieve three-dimensional effects.

Here are a few examples of three-dimensional effect I have done.

Fig.3 Image Browser with 3D effect

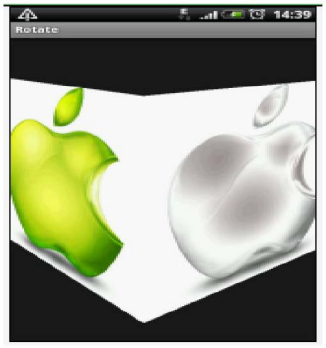

Fig.4 Another Image Browser with 3D effect

V. CONCLUSIONS

In the android, if you want to make a 3D effect, of course you can use OPNGL_ES. If you need a faster display speed, or just want a simple 3D effect, you don't forget the matrix and camera in Android.

REFERENCES

[1] Donald Hearn and M.Pauline Baker,"Computer Graphics with OpenGL", Electronics Industry,BeiJing.2003,pp. 188-245

[2] Yang Feng sheng,"Android Unleashed",China Machine Press.Beijing. 2009,pp.111-141.

[3] Gene H.Golub and Charles F.van Loan,"Matrix Computations", Posts&Telecom Press,BeiJing,2001.pp.30-108

[4] AKELE,K.and T.JERMOLUK(1998)."High-Performance Polygon Rendering", Computet Graphics,22(4),pp.239 -246.

[5] DeROSE,T.D.(1998)."Geometric Continuity,Shape Parameters,and Geometric Constructions for CatmullRom Splines", ACM Transactions on Graphics,PP.1-41.

[6] BISHOP,G.and D.M.WIEMER(1986)"Fast Phong Shading",in proceedings of SIGGRAPH'86, Compute Graphics, 20(4), pp.103-106.