public BitmapShader(Bitmap bitmap,Shader.TileMode tileX,Shader.TileMode tileY)

调用这个方法来产生一个画有一个位图的渲染器（Shader）。

bitmap 在渲染器内使用的位图

tileX The tiling mode for x to draw the bitmap in. 在位图上X方向花砖模式

tileY The tiling mode for y to draw the bitmap in. 在位图上Y方向花砖模式

TileMode：（一共有三种）

CLAMP ：如果渲染器超出原始边界范围，会复制范围内边缘染色。

REPEAT ：横向和纵向的重复渲染器图片，平铺。

MIRROR ：横向和纵向的重复渲染器图片，这个和REPEAT 重复方式不一样，他是以镜像方式平铺。

还是不太明白？那看一下效果图吧！



1

package xiaosi.BitmapShader;

import android.app.Activity;

import android.os.Bundle;

public class BitmapShaderActivity extends Activity {

/\*\* Called when the activity is first created. \*/

private BitmapShaders bitmapShaders = null;

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

bitmapShaders = new BitmapShaders(this);

setContentView(bitmapShaders);

}

}

2

package xiaosi.BitmapShader;

public class BitmapShaders extends View

{

private BitmapShader bitmapShader = null;

private Bitmap bitmap = null;

private Paint paint = null;

private ShapeDrawable shapeDrawable = null;

private int BitmapWidth = 0;

private int BitmapHeight = 0;

public BitmapShaders(Context context)

{

super(context);

//得到图像

bitmap = ((BitmapDrawable) getResources().getDrawable(R.drawable.h)).getBitmap();

BitmapWidth = bitmap.getWidth();

BitmapHeight = bitmap.getHeight();

//构造渲染器BitmapShader

bitmapShader = new BitmapShader(bitmap,Shader.TileMode.MIRROR,Shader.TileMode.REPEAT);

}

@Override

protected void onDraw(Canvas canvas)

{

super.onDraw(canvas);

//将图片裁剪为椭圆形

//构建ShapeDrawable对象并定义形状为椭圆

shapeDrawable = new ShapeDrawable(new OvalShape());

//得到画笔并设置渲染器

shapeDrawable.getPaint().setShader(bitmapShader);

//设置显示区域

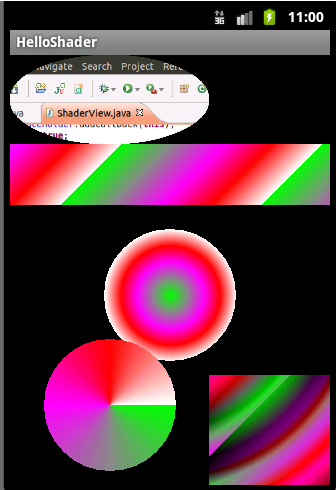
shapeDrawable.setBounds(20, 20,BitmapWidth-60,BitmapHeight-60);

//绘制shapeDrawable

shapeDrawable.draw(canvas);

}

}



package com.ldj.helloshader;

import android.content.Context;

import android.graphics.\*;

import android.graphics.Shader.TileMode;

import android.graphics.drawable.BitmapDrawable;

import android.graphics.drawable.ShapeDrawable;

import android.graphics.drawable.shapes.OvalShape;

import android.view.\*;

public class ShaderView extends SurfaceView implements SurfaceHolder.Callback,Runnable{

//声明渐变的颜色数组

private int[] color =

new int[]{Color.GREEN,Color.GRAY,Color.MAGENTA,Color.RED,Color.WHITE};

private boolean loop = false;

private SurfaceHolder surfaceHolder;

private Bitmap bitPic = null;

int bitPicWidth = 0;

int bitPicHeight = 0;

//声明一个图片渲染

BitmapShader bitmapShader = null;

//声明一个线性渐变

LinearGradient linearGradient = null;

//声明一个环形渐变

RadialGradient radialGradient = null;

//声明一个扫描渐变

//-围绕一个中心点扫描渐变就像电影里那种雷达扫描

SweepGradient sweepGradient = null;

//声明一个组合渲染

ComposeShader composeShader = null;

//定义画笔

Paint paint = null;

//利用这个类也可以实现绘制图像的功能

ShapeDrawable shapeDrawable = null;

public ShaderView(Context context) {

super(context);

surfaceHolder = this.getHolder();

//增加回调

surfaceHolder.addCallback(this);

loop = true;

paint = new Paint();

//获取图像资源

bitPic =

((BitmapDrawable)this.getResources().getDrawable(R.drawable.screenshot))

.getBitmap();

//将图片的长和高的值赋给变量

bitPicWidth = bitPic.getWidth();

bitPicHeight = bitPic.getHeight();

/\*

\* ~~~BitmapShader(Bitmap,TileMode,TileMode)~~~

\*/

bitmapShader = new BitmapShader(bitPic, TileMode.REPEAT, TileMode.MIRROR);

/\*

\* ~~~LinearGradient(x0,y0,x1,y1,int[Color],float[],TileMode)~~~

\*/

linearGradient = new LinearGradient(0,0,100,100,color,null,TileMode.REPEAT);

/\*

\* ~~~RadialGradient~~~

\*/

radialGradient = new RadialGradient(160,240,66,color,null,TileMode.MIRROR);

/\*

\* ~~~SweepGradient~~~

\*/

sweepGradient = new SweepGradient(100,350,color,null);

//~~~ComposeShader（shaderA,shaderB,Mode）~~~

//组合线性和环形两种渐变,当然其他的也可以的

composeShader

= new ComposeShader(linearGradient,radialGradient,PorterDuff.Mode.DARKEN);

}

@Override

public void run() {

while(loop) {

draw();

try {

Thread.sleep(100);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

@Override

public void surfaceChanged(SurfaceHolder arg0, int arg1, int arg2, int arg3) {

}

@Override

public void surfaceCreated(SurfaceHolder arg0) {

new Thread(this).start();

}

@Override

public void surfaceDestroyed(SurfaceHolder arg0) {

loop = false;

}

private void draw() {

Canvas canvas = surfaceHolder.lockCanvas();

/\*

\* ~~~BitmapShader

\*/

//构造形状为椭圆的shapeDrawable对象

shapeDrawable = new ShapeDrawable(new OvalShape());

//设置显示的图片

shapeDrawable.getPaint().setShader(bitmapShader);

//设置显示的长和高

shapeDrawable.setBounds(0, 0, bitPicWidth, bitPicHeight);

//绘制图像

shapeDrawable.draw(canvas);

//~~~LinearGradient~~~

//设置画笔的渲染类型

paint.setShader(linearGradient);

canvas.drawRect(0, bitPicHeight, 320, 150, paint);

//~~~RadialGradient~~~

paint.setShader(radialGradient);

canvas.drawCircle(160, 240, 66, paint);

//~~~SweepGradient

paint.setShader(sweepGradient);

canvas.drawCircle(100, 350, 66, paint);

//~~~ComposeShader~~~

paint.setShader(composeShader);

canvas.drawRect(bitPicWidth, 320, 320, 480, paint);

surfaceHolder.unlockCanvasAndPost(canvas);

}

}

总结：

配色弄的不是很好看，大家可别见怪，大体来说渲染图像或图形三步：

首先是声明渲染或渐变类。

然后将画笔setShader为声明的类。

最后绘制的时候用此画笔即可。