|  |
| --- |
| 2  连续改变颜色和位置的字符串显示 |

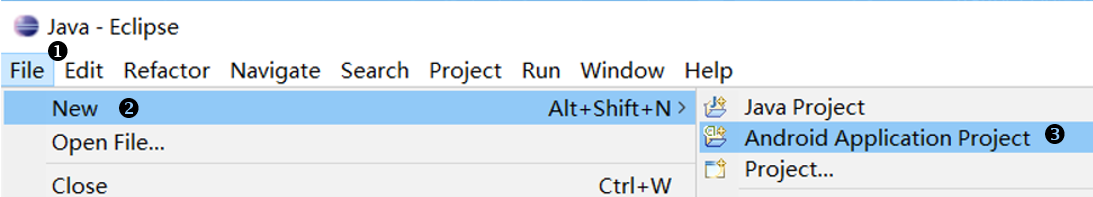
|  |
| --- |
| 2-1 程序实现 |

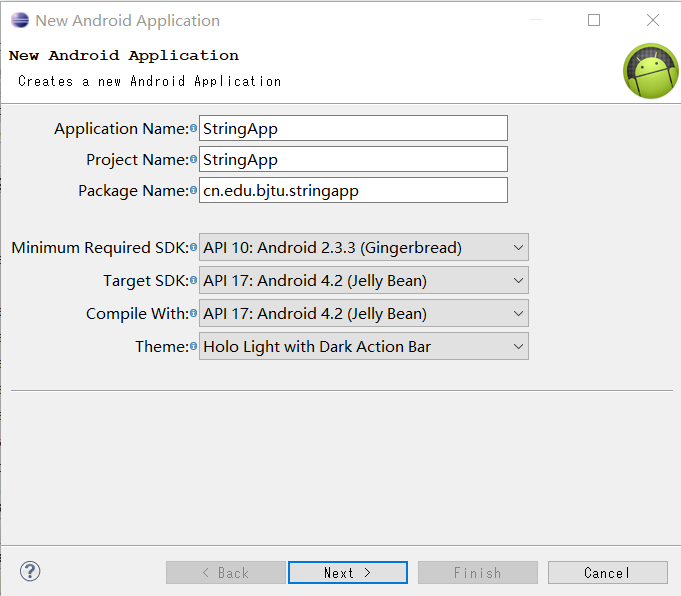
Activity通过视图显示界面，视图通过cavas显示对象，包括字符串。本应用连续改变显示字符串的颜色和位置，实现字符串的一种艺术显示效果。

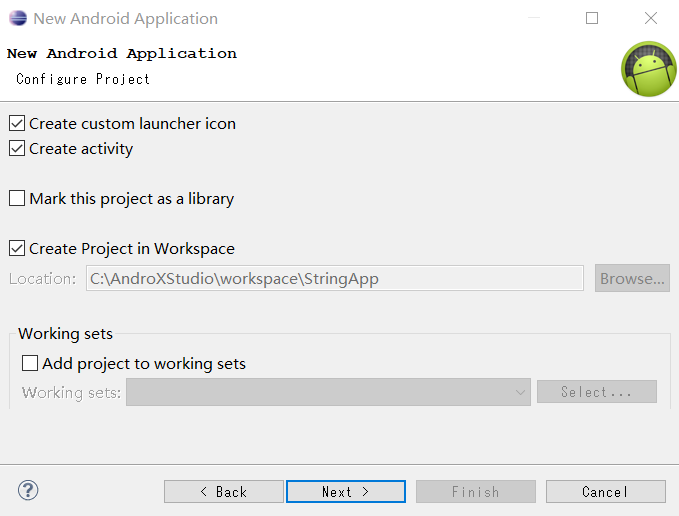
①

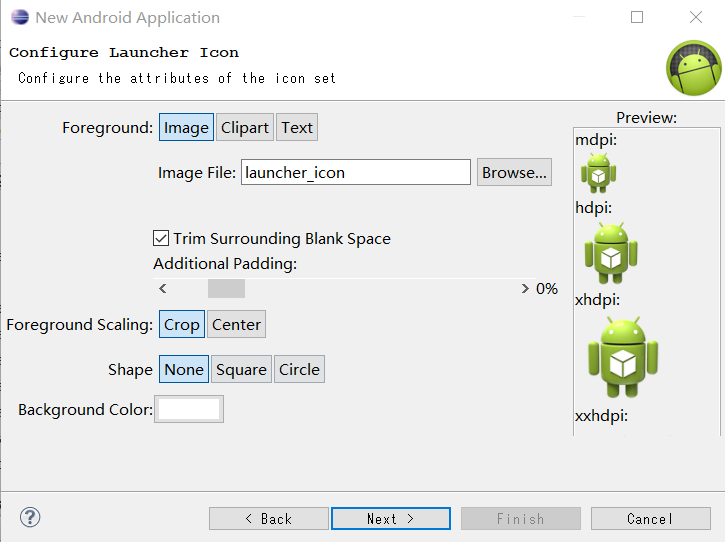
⮊ 项目创建

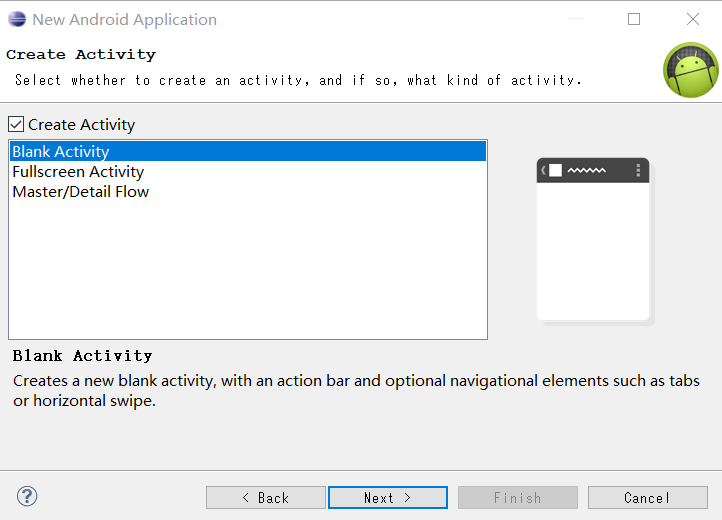
编辑显示字符串的应用程序，创建项目如下所示。

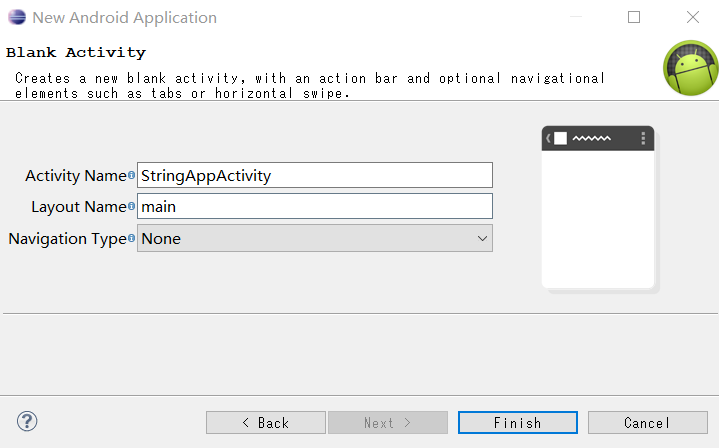












⮊ 创建自定义视图类

我们首先需要在系统已经生成的StringAppActivity.java基础上，创建显示字符串的自定义视图类。

1. 选择新创建

在Package Expore中，选择src文件夹下包名cn.edu.bjtu.stringapp，鼠标右键点击“new”项目栏，弹出菜单后，在选择 [Class]项。

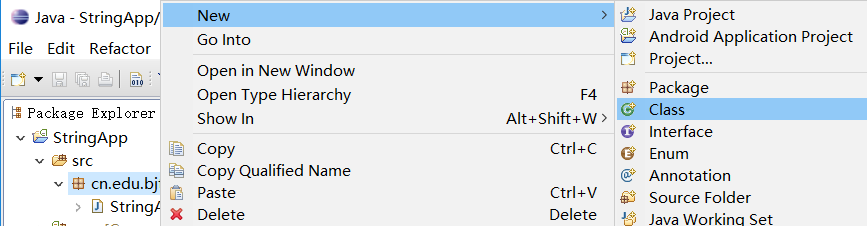


图 3-1-1 | 新建一个类

1. 新建类

“新 Java 类”对话框弹出后，在“名称”列表项中输入[StringView]，“超类”列表项中输入字符串[android.view.View]，然后按下 [完成]按钮完成创建StringView.java文件。

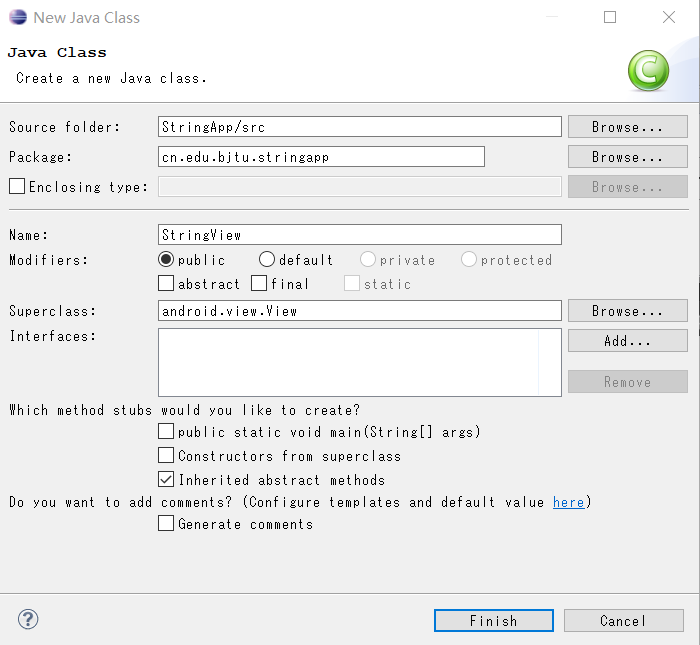


图 3-1-2 | 创建StringView 类

⮊ 运行结果

编辑程序的执行结果如下所示。如果程序运行界面中会显示一个由字符串组成的，从红色到蓝色变化的正弦波形。



图 3-1-4 | 字符串显示例子程序

|  |
| --- |
| 2-2 源码 |

⮊ StringAppActivity.java

StringAppActivity 类是显示本程序主界面的类。

**注释掉//import** android.view.Menu;

//@Override

//**public** **boolean** **onCreateOptionsMenu**(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

//getMenuInflater().inflate(R.menu.string\_app, menu);

//**return** **true**;

//}

**把**protected void onCreate(修改为：

public void onCreate(

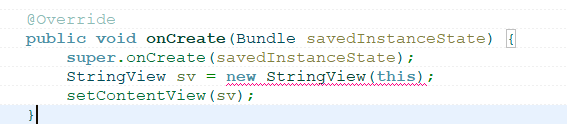
在public void onCreate(中的super.onCreate(savedInstanceState)语句后，添加：

StringView sv = new StringView(this);

把setContentView(R.layout.main)修改为：setContentView(sv);

|  |
| --- |
| 001: package cn.edu.bjtu.stringapp;  002:  003: import android.app.Activity;  004: import android.os.Bundle;  005:  006: public class StringAppActivity extends Activity {  007: /\*\* Called when the activity is first created. \*/  008: @Override  009: public void onCreate(Bundle savedInstanceState) {  010: super.onCreate(savedInstanceState);  011:  012: StringView sv = new StringView(this);  013: setContentView(sv);  014: }  015: } |

列表 3-1-1 | StringAppActivity.java



在程序第12行，会提示StringView 构造函数未定义。所以，需要定义StringView.java

⮊ StringView.java

StringView是显示字符串结果的自定义视图类。完善public class StringView extends View {}如下：

**private** **final** **int** NUM = 100;

**private** **final** **double** TWOPI = 2 \* 3.14159;

**private** Point[] point;

**private** **int** cxView;

**private** **int** cyView;

**public** **StringView**(Context context) {

**super**(context);

point = **new** Point[NUM];

setBackgroundColor(Color.WHITE);

}

@Override

**protected** **void** **onDraw**(Canvas canvas) {

Paint paint = **new** Paint();

cxView = getWidth();

cyView = getHeight();

paint.setAntiAlias(**true**);

DisplayString(canvas, paint);

**super**.onDraw(canvas);

}

**private** **void** **DisplayString**(Canvas canvas, Paint paint) {

String str = "2018北京交通大学";

calcPos();

paint.setTextSize(20);

**for**(**int** i = 0; i != point.length; ++i) {

paint.setColor(Color.rgb(200-(i\* 2), 0, 50+(i\*2)));

canvas.drawText(str, point[i].x, point[i].y, paint);

}

}

**private** **void** **calcPos**() {

**for**(**int** i = 0; i != NUM; ++i) {

point[i] = **new** Point();

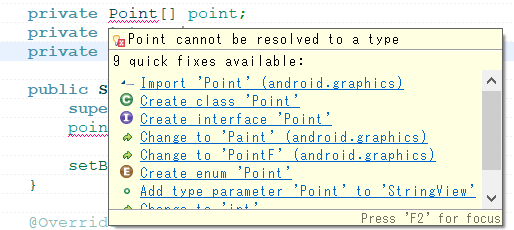
point[i].x = i \* cxView / NUM;

point[i].y = (**int**) (cyView / 2 \* ( 1 - Math.*sin*(TWOPI \* i / NUM)));

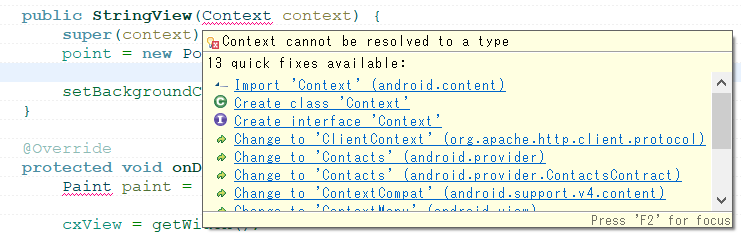
}

}

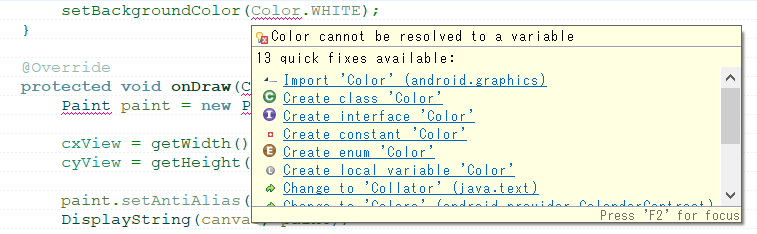
}



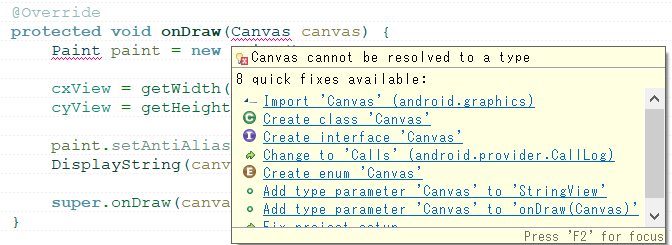
导入(Import)：point类



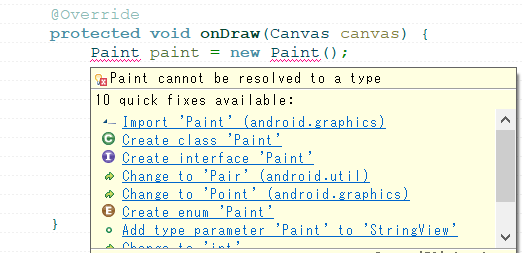
导入(Import) Context类



导入(Import)color类



导入(Import)canvas类



导入(Import)paint类

|  |
| --- |
| 001: package cn.edu.bjtu.stringapp;  002:  008: import android.view.View;  009:  010: public class StringView extends View {  011: private final int NUM = 100;  012: private final double TWOPI = 2 \* 3.14159;  013:  014: private Point[] point;  015: private int cxView;  016: private int cyView;  017:  018: public StringView(Context context) {  019: super(context);  020: point = new Point[NUM];  021:  022: setBackgroundColor(Color.WHITE);  023: }  024:  025: @Override  026: protected void onDraw(Canvas canvas) {  027: Paint paint = new Paint();  028:  029: cxView = getWidth();  030: cyView = getHeight();  031:  032: paint.setAntiAlias(true);  033: DisplayString(canvas, paint);  034:  035: super.onDraw(canvas);  036: }  037:  038: private void DisplayString(Canvas canvas, Paint paint) {  039: String str = "2018北京交通大学";  040:  041: calcPos();  042:  043: paint.setTextSize(20);  044: for(int i = 0; i != point.length; ++i) {  045: paint.setColor(Color.rgb(200-(i\* 2), 0, 50+(i\*2)));  046:  047: canvas.drawText(str, point[i].x, point[i].y, paint);  048: }  049: }  050:  051: private void calcPos() {  052: for(int i = 0; i != NUM; ++i) {  053: point[i] = new Point();  054: point[i].x = i \* cxView / NUM;  055: point[i].y = (int) (cyView / 2 \* ( 1 - Math.sin(TWOPI \* i / NUM)));  056: }  057: }  058: } |

列表 3-1-2 | StringView.java

需要导入的类

003: import android.content.Context;

004: import android.graphics.Canvas;

005: import android.graphics.Color;

006: import android.graphics.Paint;

007: import android.graphics.Point;

|  |
| --- |
| 2-3 实现原理说明 |

|  |
| --- |
| ⮊ StringAppActivity.java |

1. 需导入的类

如果新建项目，自定义activity自动导入显示中必要的Activity和 Bundle类信息。Activity是由avtivity管理器管理程序生命周期并在画面显示中被使用，以及Bundle显示程序内部属性。

|  |
| --- |
| 003: import android.app.Activity;  004: import android.os.Bundle; |

1. 定义初始启动的操作

程序初始启动会调用onCreate()方法。

|  |
| --- |
| 008: @Override  009: public void onCreate(Bundle savedInstanceState) {  …  014: } |

创建自定义视图StringView之后，把引用赋值给变量sv。StringView是从View视图继承而来的，所有的View视图必须由被称作为程序上下文信息的Context保管，这个是为了区分程序中view的所属。由于Context包含在activity中，传递this参数，实现加载本程序的Context上下文。

|  |
| --- |
| 012: StringView sv = new StringView(this); |

调用传入Sv参数的 setContentView()方法，通过显示程序界面的视图设置StringView。

|  |
| --- |
| 013: setContentView(sv); |

Sv局部变量因为没有外部访问，所以可以简写如下所示。

|  |
| --- |
| setContentView(new StringView(this)); |

|  |
| --- |
| ⮊ StringView.java |

1. 需导入的类

自定义视图中显示字符串，需导入Canvas, Color, Paint类信息。视图通过Canvas画布显示对象，通过Paint显示相关对象显示属性。Color设置颜色信息。比如，Canvas画布, Paint为笔, Color就好比是颜料。

|  |
| --- |
| 004: import android.graphics.Canvas;  005: import android.graphics.Color;  006: import android.graphics.Paint; |

1. 声明成员变量

为了使用通过字符串输出坐标显示正弦，我们要声明表示正弦上的点数NUM和表示正弦波取值范围的TWOPI常量。取值范围设置为2 \* π(3.14159...)，那么整个屏幕显示的就是一个正弦波。

|  |
| --- |
| 011: private final int NUM = 100;  012: private final double TWOPI = 2 \* 3.14159; |

1. 声明成员变量

声明Point类型数组point和存储当前视图的高度和宽度的cxView 和 cyView变量。

|  |
| --- |
| 014: private Point[] point;  015: private int cxView;  016: private int cyView; |

1. 定义构造函数的操作

通过new新建StringView对象的时候，定义被调用的构造函数。

|  |
| --- |
| 018: public StringView(Context context) {  …  023: } |

如果调用构造函数，分配创建NUM个数的Point类型数组point。

|  |
| --- |
| 020: point = new Point[NUM]; |

我们通过Color.WHITE参数调用setBackgroundColor()方法，把视图背景设置为白色。

|  |
| --- |
| 022: setBackgroundColor(Color.WHITE); |

1. 定义更新操作

绘制新视图的时候，程序会再次调用onDraw()方法，并把绘制的Canvas画布引用传递过来。一旦调用onDraw()方法，我们只要对Canvas对象执行绘制操作即可。通过引用用户不能直接调用onDraw()方法，所以想要请求更新自定义视图时，就需调用invalidate()。

|  |
| --- |
| 025: @Override  026: protected void onDraw(Canvas canvas) {  …  036: } |

创建指定绘制对象背景或与全景有关属性的Paint对象，保存到局域变量paint中。

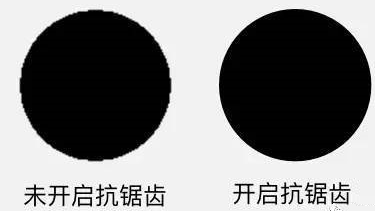
|  |
| --- |
| 027: Paint paint = new Paint(); |

调用getWidth()和 getHeight()方法获得View的宽带和高度保存到cxView和 cyView变量中。

|  |
| --- |
| 029: cxView = getWidth();  030: cyView = getHeight(); |

传递参数true，调用setAntiAlias()方法打开Canvas的抗锯齿。如果打开抗锯齿，那么Canvas画布在显示绘制对象时，会把轮廓处理得更光滑。

|  |
| --- |
| 032: paint.setAntiAlias(true); |



使用带canvas和paint参数的自定义方法DisplayString(),显示字符串。

|  |
| --- |
| 033: DisplayString(canvas, paint); |

1. 定义DisplayString() 自定义方法操作

调用DispalyString()方法，使用 canvas画布和 paint显示字符串。

|  |
| --- |
| 038: private void DisplayString(Canvas canvas, Paint paint) {  …  049: } |

创建画面中显示的字符串后，String类型变量str为所显示的字符串。

|  |
| --- |
| 039: String str = "2018北京交通大学"; |

通过调用calcPos()方法，计算字符串所显示的起始坐标。

|  |
| --- |
| 041: caclPos(); |

调用setTextSize()方法指定要显示字符串的大小后，进入for循环，由calcPos()方法计算显示字符串的坐标。

|  |
| --- |
| 043: paint.setTextSize(20);  044: for(int i = 0; i != point.length; ++i) {  …  048: } |

通过调用setColor()方法，使得在反复显示字体的同时，使字体颜色发生变化，Color.rgb()按照0~255红绿蓝三色比例，创建出24位24bit(8 x 8 x 8) 颜色(224 = 16777216)。比如，rgb(255, 0, 0)方法实现红色，rgb(255, 255, 255)为白色，rgb(0, 0, 0)调制成黑色。

|  |
| --- |
| 045: paint.setColor(Color.rgb(200-(i\* 2), 0, 50+(i\*2))); |

在显示指定起始坐标上的字符串时使用drawText()方法。该方法有三个参数，第一个参数表示显示字符串的引用，第二个参数和第三个参数表示起始坐标。最后一个参数表示在显示字体时作用的Paint引用。

|  |
| --- |
| 047: canvas.drawText(str, point[i].x, point[i].y, paint); |

1. 定义calc() 自定义方法操作

调用calcPos()方法，计算NUM个正弦坐标，把结果保存到成员变量point中。

|  |
| --- |
| 051: private void calcPos() { …  …  057: } |

使用For循环，计算出NUM个正弦波上的点坐标。

|  |
| --- |
| 052: for(int i = 0; i != NUM; ++i) {  …  056: } |

创建 Point类型对象，分配到point[i]数组中，式中i表示显示波形的点，该点由x和y坐标决定。

|  |
| --- |
| 053: point[i] = new Point(); |

通过公式i \* cxView / NUM把结果保存到x坐标，y坐标通过(cyView / 2 \* ( 1 - Math.sin(TWOPI \* i / NUM)))数学式计算分配得到。

|  |
| --- |
| 053: point[i] = new Point();  054: point[i].x = i \* cxView / NUM;  055: point[i].y = (int) (cyView / 2 \* ( 1 - Math.sin(TWOPI \* i / NUM))); |

|  |
| --- |
| ⮊ Color 常量 |

显示颜色，可以使用如下表中定义的表示某种颜色的常量值。如果想使用其他任意颜色需要调用Color.rgb()方法。

|  |  |  |  |
| --- | --- | --- | --- |
| **项目** | **颜色** | **项目** | **颜色** |
| BLACK | 黑色 | GREEN | 绿色 |
| DKGRAY | 深灰色 | BLUE | 蓝色 |
| GRAY | 灰色 | YELLOW | 黄色 |
| LTGRAY | 浅灰色 | CYAN | 天蓝色 |
| WHITE | 白色 | MAGENTA | 粉红色 |
| RED | 红色 | TRANSPARENT | 背景透明 |

表 3-1-3 | 颜色常量

|  |
| --- |
| 2-4 要点整理 |

* Activity界面通过View视图显示，view视图的显示区域表示为Canvas。我们可以通过Paint实现在Canvas中显示字符串。
* 如果想自定义view，那么我们需要定义从View继承的类。
* View在需要进行更新时，被调用覆盖 onDraw()函数，我们可以在这个地方叙述要显示的内容。
* Paint的setAntiAlias()方法设置抗锯齿，setColor()方法指定全景色，setTextSize()方法指定字符串的字体大小。
* Canvas的 drawText()方法在被指定的位置使用 Paint信息输出字符串。
* String是定义字符串类，Integer定义整形对象，通过toString()方法可以把保存的值转换成字符串类型。

|  |
| --- |
|  |