【eoeAndroid特刊】

第二期：图像处理篇（1）

发布版本： Ver 1.0.0(build 2009.05.20)

变更信息：

从本期开始，将本系列名称由【协同翻译】变更为【eoeAndroid特刊】，每次以专题的形式策划、组织和汇总文章，相关约定如下：

- 每期以专题的形式组织文章和资料；
- 每期历时1周或2周；
- 每期设定一个主题作为专题方向；
- 参与人员不限，要求有积极有热情，有奉献精神；
- 文章数量不限，但要求和专题方向一致；
- 每期策划主题和方案在eoeAndroid社区发布；

我们欢迎更多的朋友加入进来，一起分享，一起成长。

写在前面：

本篇简介：

本期是eoeAndroid策划的第二篇专题，主要整理和翻译在Android中进行图像处理的一些资源和文章，通过本专题内容的学习，可以掌握如何在Android上对图片编程，主要包括但不限于如下方向的内容：

- Android中支持的图片格式介绍；
- Android中图片库介绍
- 图片的显示（本地的，网络的）；
- 图片的格式转换；
- 动画效果；
- 图片特效；

本专题旨在帮助更多的人熟悉和掌握图像编程，其中收录和整理的文章是我们挑选、翻译、整理、撰写的和本专题相关的内容，由于图像处理是个非常复杂、有技术含量的领域，其衍生诸如图片动画、绘制图片等相当高级一些，本着高效灵活的原则，本期专题暂不包含诸如动画这类文章，会在后续专题中补充。

致谢：

本期专题得到如下同学的大力支持和积极响应，谢谢你们辛苦老大，谢谢你们为Android发展和普及做出的贡献。

- apcwowo
- 404
- IceskYsl

- haiyangjy
- zhoubo5262
- binbinming

活动发起地址：
协作翻译第二期：图像处理篇
http://www.eoeandroid.com/viewthread.php?tid=257&extra=page%3D1

目录导航：

第二期：图像处理篇（1）..... 1
写在前面：..... 1
1. 应用风格和主题..... 4
1.1 如何新建自定义的风格和主题：..... 4
1.2 主题..... 5
1.2.1 在manifest当中设置主题..... 5
1.2.2 在程序当中设置主题..... 6
2. Android如何绘制视图..... 7
3. Handling UI Events..... 8
3.1 Event Listeners..... 8
3.2 Event Handlers..... 9
3.3 Touch Mode..... 10
3.4 Handling Focus..... 10
4. 2D Graphics..... 12
4.1 Drawable..... 12
4.1.1 从资源图像文件中创建..... 12
4.1.2 从XML文件中创建..... 13
4.2 ShapeDrawable..... 14
4.3 NinePatchDrawable..... 15
4.4 Tween Animation..... 16
4.5 Frame Animation..... 17
5. 图片的缩放和旋转..... 19
5.1 目标：..... 19
5.2 代码示例：..... 19
5.3 展示效果..... 20
6. 3D 和 OpenGL..... 22
6.1 使用 API..... 22
6.2 附加信息..... 22
7. GLSurfaceView 介绍..... 23
7.1 GLSurfaceView介绍..... 23
7.2 关于用户的输入?..... 24
7.3 GLSurfaceView 其他例子:..... 26
7.4 选择 一个 Surface..... 26
7.5 Continuous Rendering vs. Render When Dirty..... 27
7.6 Help With Debugging..... 27
8. 其他..... 28
翻译人员..... 28
BUG提交..... 28
参加翻译..... 28
关于eoeAndroid..... 28

正文开始

1. 应用风格和主题

翻译：海阳|haiyang(http://www.haiyangjy.com)

当你设计你的程序的时候，你可以用风格和主题来统一格式化各种屏幕和UI元素。

- 风格是一个包含一种或者多种格式化属性的集合，你可以将其用为一个单位用在布局XML单个元素当中。比如，你可以定义一种风格来定义文本的字号大小和颜色，然后将其用在View元素的一个特定的实例。
- 主题是一个包含一种或者多种格式化属性的集合，你可以将其为一个单位用在应用中所有的Activity当中或者应用中的某个Activity当中。比如，你可以定义一个主题，它为window frame和panel 的前景和背景定义了一组颜色，并为菜单定义可文字的大小和颜色属性，你可以将这个主题应用在你程序当中所有的Activity里。

风格和主题都是资源。你可以用android提供的一些默认的风格和主题资源，你也可以自定义你自己的主题和风格资源。

1.1 如何新建自定义的风格和主题：

1.在res/values 目录下新建一个名叫style.xml的文件。增加一个<resources>根节点。
2.对每一个风格和主题，给<style>element增加一个全局唯一的名字，也可以选择增加一个父类属性。在后边我们可以用这个名字来应用风格，而父类属性标识了当前风格是继承于哪个风格。
3.在<style>元素内部，申明一个或者多个<item>,每一个<item>定义了一个名字属性，并且在元素内部定义了这个风格的值。
4.你可以应用在其他XML定义的资源。

风格
下边是一个申明风格的实例：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
 <style name="SpecialText" parent="@style/Text">
 <item name="android:textSize">18sp</item>
 <item name="android:textColor">#008</item>
 </style>
</resources>
```

如上所示，你可以用<item>元素来为你的风格定义一组格式化的值。在Item当中的名字的属性可以是一个字符串，一个16进制数所表示的颜色或者是其他资源的引用。

注意在<style>元素中的父类属性。这个属性让你可以能够定义一个资源，当前风格可以从这个资源当中继承到值。你可以从任何包含这个风格的资源当中继承此风格。通常上，你的资源应该一直直接或者间接地继承Android的标准风格资源。 这样的话，你就只需要定义你想改变的值。

在这个例子当中的EditText元素，演示了如何引用一个XML布局文件当中定义的风格:

```
<EditText id="@+id/text1"
 style="@style/SpecialText"
 android:layout_width="fill_parent"
```

```
 android:layout_height="wrap_content"
 android:text="Hello, World!" />
```

现在这个EditText组件的所表现出来的风格就为我们在上边的XML文件中所定义的那样。

1.2 主题

就像风格一样，主题依然在<style>元素里边申明，也是以同样的方式引用。不同的是你通过在Android Manifest中定义的<application>和<activity>元素将主题添加到整个程序或者某个Activity，但是主题是不能应用在某一个单独的View里。

下边是申明主题的一个例子：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
 <style name="CustomTheme">
 <item name="android:windowNoTitle">true</item>
 <item name="windowFrame">@drawable/screen_frame</item>
 <item name="windowBackground">@drawable/screen_background_white</item>
 <item name="panelForegroundColor">#FF000000</item>
 <item name="panelBackgroundColor">#FFFFFFFF</item>
 <item name="panelTextColor">?panelForegroundColor</item>
 <item name="panelTextSize">14</item>
 <item name="menuItemTextColor">?panelTextColor</item>
 <item name="menuItemTextSize">?panelTextSize</item>
 </style>
</resources>
```

注意我们用了@符号和？符号来应用资源。@符号表明了我们应用的资源是前边定义过的(或者在前一个项目中或者在Android 框架中)。问号？表明了我们引用的资源的值在当前的主题当中定义过。通过引用在<item>里边定义的名字可以做到(*panelTextColor* 用的颜色和*panelForegroundColor中定义的一样*)。这中技巧只能用在XML资源当中。

1.2.1 在manifest当中设置主题

为了在成用当中所有的Activity当中使用主题，你可以打开AndroidManifest.xml 文件，编辑<application>标签，让其包含android:theme属性，值是一个主题的名字，如下：

```
<application android:theme="@style/CustomTheme">
```

如果你只是想让你程序当中的某个Activity拥有这个主题，那么你可以修改<activity>标签。

Android中提供了几种内置的资源，有好几种主题你可以切换而不用自己写。比如你可以用对话框主题来让你的Activity看起来像一个对话框。在manifest中定义如下：

```
<activity android:theme="@android:style/Theme.Dialog">
```

如果你喜欢一个主题，但是想做一些轻微的改变，你只需要将这个主题添加为父主题。比如我们修改Theme.Dialog主题。我们来继承Theme.Dialog来生成一个新的主题。

```
<style name="CustomDialogTheme" parent="@android:style/Theme.Dialog">
```

继承了Theme.Dialog后，我们可以按照我们的要求来调整主题。我们可以修改在Theme.Dialog中定义的每个item元素的值，然后我们在Android Manifest 文件中使用CustomDialogTheme 而不是 Theme.Dialog 。

1.2.2 在程序当中设置主题

如果需要的话，你可以在Activity当中通过使用方法setTheme()来加载一个主题。注意，如果你这么做的话，你应该初始化任何View之前设置主题。比如，在调用setContentView(View) 和inflate(int, ViewGroup)方法前。这保证系统将当前主题应用在所有的UI界面。例子如下：

```
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
...
setTheme(android.R.style.Theme_Light);
setContentView(R.layout.linear_layout_3);
}
```

如果你打算在程序代码中来加载主界面的主题，那么需要注意主题当中不能包括任何系统启动这个Activity所使用的动画，这些动画将在程序启动前显示。在很多情况下，如果你想将主题应用到你的主界面，在XML中定义似乎是一个更好的办法。

关于风格和主题更加详细的信息，请参照Available Resource Types: Style and Themes.
关于默认的主题和风格资源请参照 R.style.

2. Android如何绘制视图

译者：Icesksl
原文：http://developer.android.com/guide/topics/ui/how-android-draws.html

当Activity获得焦点时，其就会被要求绘制其布局，android框架会处理具体的绘制功能，但是其布局的继承关系的根节点必须由Activity提供。

绘制的时候从模板的根节点开始，其被要求计算和绘制布局树。绘制动作会遍历布局节点树，然后渲染每个节点视图。概括来说，view组负责要求其所有的子节点完成绘制（使用draw()方法），与之相应的是，每个view负责绘制自己。因为这棵树是有序的，也就是说父节点会被早绘制，树上的后续节点会被依次绘制。

绘制动作分成两个进程：一个是测绘进程，一个是模板路径。measuring路径的实现方法是in measure(int, int)，其从上至下遍历view树。在每次递归过程中，view将其自身的参数提交进去。在measure最后，每个view都会存储其自身的参数。第二个pass在调用 layout(int, int, int, int)时运行，其也是自上向下遍历view树。在这个pass中，每个都几点负责将按照其子节点按照上面通过measure获取的参数，定位在布局中。

Android的框架只绘制有效区中的视图，并同时处理视图的背景。你也可以使用invalidate()方法强制绘制。

当一个视图的measure()方法返回时，其 getMeasuredWidth() 和getMeasuredHeight()方法的值就已经被设定了，后续的子节点视图也同样遵守这个原则。一个View的measured宽和measured的高的值受其父节点视图约束。如此可以保证在度量结束后，所有的父节点都可以接受其所有的子节点的参数。一个父视图的measure()方法可以被子节点多次调用。例如，父视图开始会对每个子视图首先不指定尺寸调用该方法获取每个字视图的尺寸，最后再用获取到的尺寸的和去判断是否超过或者小于父视图（例如子视图不指定自己需要多少的空白，那么父视图将会接管这个事）。

measure pass使用两个类来传递尺寸，View使用View.MeasureSpec 类告诉其父视图自己的尺寸以及位置，最基本的LayoutParams类只用来描述视图的宽和高，对每个尺寸，可以有下面三种方式指定。

- 一个具体的数字
- FILL_PARENT, 这个意思是当前View要和其父视图同样大小（减去padding的值）
- WRAP_CONTENT, 这个意思是当前视图只需要包围其内容即可（包含padding）。

要初始化一个布局的时候，需要调用requestLayout()方法，该方法被调用的典型场景是一个View觉得其自身不在适应当前规定的范围的时候。

LayoutParams针对ViewGroup中不同的子类提供了不同的子类，例如RelativeLayout拥有其自身的LayoutParams子类，其中包含对其子视图水平和竖直居中。

MeasureSpecs用来将需求放到。一个MeasureSpec可以是下面三个中的一个方法。

    - UNSPECIFIED: 不指定大小，这个标识标识父视图不指定其值，而是让子视图按照自身的需要设定，比如在LinearLayout中，父视图可以使用高度为UNSPECIFIED，宽度为EXACTLY 240来调用子视图的measure() 方法，获取其子视图在240宽的情况下需要多少高。
- EXACTLY: 明确指定大小，子视图只能使用其父视图指定的大小。
- AT_MOST: 这个用来指定其子视图的最大值，子视图必须遵循这个限制。

3. Handling UI Events

译者：zhoubo5262
原文：http://developer.android.com/guide/topics/ui/ui-events.html

在Android平台上，捕获用户在界面上的触发事件有很多种方法，View类就提供这些方法。你在使用各种View视图来布局界面时，会发现几个公用的回调方法来捕捉有用的UI触发事件，
当事件在某个View对象上被触发时，这些方法会被系统框架通过这个对象所调用，例如：当一个View(如一个Button)被点击，onTouchEvent()方法会在该对象上被调用，所以，为了捕获和处理事件，必须去继承某个类，并重载这些方法，以便自己定义具体的处理逻辑，显然，你更容易明白，为什么在你使用View类时会嵌套带有这些回调方法的接口类，这些接口称为event listeners，它是你去获取UI交互事件的工具
在你继承View类，以便建立一个自定义组，也许你想继承Button ， 你会更普遍使用事件监听来捕捉用户的互动，在种情况下，你可以使用类的event handlers.来预定义事件的处理方法。

3.1 Event Listeners

View类里的event listener是一个带有回调方法的接口，当UI里的组建是被用户触发时，这些方法会被系统框架所调用
onClick()
来自View.OnClickListener 它会被调用当点击这个Item(在触摸模式),或者当光标聚集在这个Item上时按下"确认"键 ，导航键，或者轨迹球。
onLongClick()
来自View.OnLongClickListener. 它会被调用当长按这个Item(在触摸模式),或者当光标聚集在这个Item上时长按"确认"键 ，导航键，或者轨迹球。
onFocusChange()
来自View.OnFocusChangeListener 它会被调用当光标移到或离开这个Item，
onKey()
来自View.OnKeyListener..它会被调用，当光标移到这个Item，按下和释放一个按键的时候
onTouch()
来自View.OnTouchListener. 它会被调用 ，在这个Item的范围内点触的时候
onCreateContextMenu()
来自View.OnCreateContextMenuListener. 它会被调用， 当上下文菜单被建立时(由于持续的"长按") 见讨论Creating Menus更多的信息。
这些方法和嵌套接口类都是一一对应的，如果确定其中一种方法处理你的互动事件，你需要在Activity中实现这个带有这个方法的接口，并把它作为匿名类，然后，通过实例的View.set...Listener() 方法来设置监听器(例如，调用setOnClickListener()，来设置OnClickListener做为监听器)

以下的例子展示了怎样去给Button设置一个监听器

```
// Create an anonymous implementation of OnClickListener
private OnClickListener mCorkyListener = new OnClickListener() {
    public void onClick(View v) {
      // do something when the button is clicked
    }
};

protected void onCreate(Bundle savedValues) {
```

```
    ...
    // Capture our button from layout
    Button button = (Button)findViewById(R.id.corky);
    // Register the onClick listener with the implementation above
    button.setOnClickListener(mCorkyListener);
    ...
}
```

下面这个列子我们会发现用Activity去实现OnClickListener接口，并作为它的一部分，会更方便，而不必去加载额外的类和对象

```
    protected void onCreate(Bundle savedValues) {
        ...
        Button button = (Button)findViewById(R.id.corky);
        button.setOnClickListener(this);
    }

    // Implement the OnClickListener callback
    public void onClick(View v) {
      // do something when the button is clicked
    }
    ...
}
```

这里注意一下，以上的例子可以看出onClick()是没有返回值的，但是有些事件处理方法是必须带返回值，它取决于的具体的事件，有些那么做的原因，看下面的例子:

onLongClick()

它返回的布尔值表明你已经完成了这个事件的处理，还是应该把它继续传下去。返回true表明已经处理完成并且停止了传递，如果返回为false表明事件还没有完成，或者它还需要继续被传递给其他的监听器

onKey()

它返回的布尔值表明你已经完成了这个事件的处理，还是应该把它继续传下去。返回true表明已经处理完成并且停止了传递，如果返回为false表明事件还没有完成，或者它还需要继续被传递给其他的监听器

onTouch()

它返回的布尔值表明你是否已经完成了这次事件的行动，重要的是后面可能还有很多后续的行动，这样，如果你返回false，表明在接到下一次的后续行动中，你还没有完成之前行为也没有意向去处理随后的行动，因此，在这个事件的后续行动中将不会再被调用。 如fingure手势，或最终行动事件

记住：我们所关注的事件肯定是发生在高亮聚集的焦点，它从总视图（顶级的）被一级一级的向下传递，直到我们想要关注的组件，当焦点聚集在这个视图（或视图中的子视图）时 ，你能够使用dispatchKeyEvent() 作为一种代替方法，来捕获在视图上的按键事件，你还可以使用onKeyDown()和onKeyUp().来捕获所有事件内的交互活动

笔记：在 Android 框架中会调用event handlers先处理事件，然后会适当的传递给二级默认的预定义handlers中;因此 如果返回true，将会停止这个事件的传递，View中默认事件处理方法的回调也会被阻止。

因此，当你返回true肯定表明你是要终止这个事件的延续。

3.2 Event Handlers

如果您建立一个继承于View自定义组件,然后您可以定义一些回调方法用作默认的事件处理程序,。该文件中关于Building Custom Components，您会学习一些共用的回调方法用于事件处理，其中包括：

- onKeyDown(int, KeyEvent) - -当一个按键被按下时被调用
- onKeyUp(int, KeyEvent)-当一个按键弹起时被调用
- onTrackballEvent(MotionEvent)- 轨迹球被触动时调用
- onTouchEvent(MotionEvent) - 当试图得到或失去高亮时.

还有其他一些方法，这不属于View类，但可以直接影响到你处理事件的方式，所以在布局内管理更复杂的事件们可以考虑到这些方法：

- Activity.dispatchTouchEvent(MotionEvent)它是让你的Activity 去拦截所有的事件，在它们被传送到具体窗口之前
- ViewGroup.onInterceptTouchEvent(MotionEvent)这个方法让ViewGroup去查看事件，并由他来传递到具体的子窗口
- ViewParent.requestDisallowInterceptTouchEvent(boolean)子窗口通过这个方法告诉父窗口，不容许父窗口使用onInterceptTouchEvent(MotionEvent). 去拦截触摸事件，

3.3 Touch Mode

当用户在使用方向键或轨迹球浏览用户界面时，有必要给于一个焦点在可操作的组件上(如一个Button)，使用户可以看到它将接受输入命令。如果设备有触摸功能，那么，当用户与界面的交互就不再需要有一个高亮在组件上，或一个焦点在view上，因此，模式的互动名为"触摸模式"

对于一个触摸设备，一旦有用户接触屏幕时，该设备将进入触摸模式.在点触某个View后，只有的它的方法isFocusableInTouchMode()返回为真时，才会有聚集焦点，如文本编辑工具。其他的界面只可以点触，但不会聚集焦点（高亮），如button 被点触时就不会聚集焦点，当它被按下时只会调用on-click监听器的回调方法

任何时候用户接触方向键或者滚动轨迹球时，该设备将退出触摸模式，并聚集焦点，用户可以恢复与用户界面的键盘交互，而不必在屏幕上

触摸模式的状态是由整个系统来维持的(all windows and activities),要查询目前所处的状态,你可以调用isInTouchMode()方法来获得，看看设备目前是否处于触摸模式。

3.4 Handling Focus

系统框架将处理日常的焦点移动来响应用户的输入，它包刮改变焦点（当界面是被移除，隐藏，或者作为一个新的View变为可用状态），通过isFocusable()这个方法我们可以知道view是否具有接受焦点的资格，也可以通过setFocusable().来设置view接受焦点的资格,对应在触摸模式下，你可以调用isFocusableInTouchMode().来获知是否有焦点来响应点触，也可以通过setFocusableInTouchMode().来设置是否有焦点来响应点触的资格

系统框架控制焦点移动到另一个组建的算法是在某一方向上邻近的组件，在极个别情况下，默认的算法可能不符合开发者的预想要求，在这种情况下，你可以覆写下列XML属性的布局文件： nextFocusDown ， nextFocusLeft ， nextFocusRight ，和nextFocusUp 设置他们的值来明确焦点从当前界面移动下个界面的Id

例如：

```
<LinearLayout
    android:orientation="vertical"
    ... >
  <Button android:id="@+id/top"
          android:nextFocusUp="@+id/bottom"
```

```
            ... />
    <Button android:id="@+id/bottom"
            android:nextFocusDown="@+id/top"
            ... />
</LinearLayout>
```

一般来说，在这个垂直布局，浏览的焦点会从第一个按钮开始，不会是从第二个或者其他的，现在top Buttont已经通过nextFocusUp (反之亦然)确定了bottom

通常如果你想宣布用户界面具有焦点的资格 (如果这个界面在传统上是没有的)，可以在xml布局里去加上的android:focusable的属性，并设置它的值，您也可以宣布在触摸模式下具有焦点的资格，同样也只在xml里添android:focusableInTouchMode.的属性，并设置它的值

当用户请求在某个界面聚集焦点时，会调用requestFocus().这个方法

监听到焦点活动(获得焦点或失去焦点都会被通知)，会调用onFocusChange(),这个方法，这也是上节所讨论的Event Listeners节段

4. 2D Graphics

译者：apcwowo
原文：http://android-developers.blogspot.com/2009/04/introducing-home-screen-widgets-and.html

Androiid 为我们提供了一个用来绘制图片与动画2D的图像库，这两个包分别是 android.graphics.drawable 和 android.view.animation ,在这两个包中可以找到相同的类去呈现绘图与动画的两个不同面。 在这个文档中将介绍如何在你的Android应用程序中使用这个库。我们将讨论基础类Drawable对象如何绘图，如何使用一对Drawable的子类，还有如何去创建图片和动画

4.1 Drawable

Drawable是一个通用的抽象类，它的目的是告诉你什么东西是可以画的。你会发现基于Drawable类扩展出各种绘图的类包括：BitmapDrawable ShapeDrawable PictureDrawable LayerDrawable,当然你可以继承它来创建你自己的绘图类.

有三种方法可以定义和实例化一个Drawable,保存一个图片到你工程资源中，使用XML文件来描述Drawable属性或者用一个正常的类去构造。下面我们将讨论两种技术（对一个有开发经验的开发者来说构造并不是最新的技术）

4.1.1 从资源图像文件中创建

一个比较简单的方法是添加一个图片到你的程序中，然后通过资源文件引用这个文件，支持的文件类型有PNG(首选的) JPG（可接受的）GIF（不建议），显然这种对于显示应用程序的图标跟来说是首选的方法，也可以用来显示LOGO，其余的图片可以用在例如游戏中。

把一个图片资源，添加你的文件到你工程中res/drawable/目录中去，从这里，你就可以引用它到你的代码或你的XML布局中，另一种方法，引用它也可以用资源编号，比如你选择一个文件只要去掉后缀就可以了（例如：my_image.png 引用它是就是my_image）

下面的代码片段示范如何去建立一个ImageView,把drawable资源中的图片添加到这个布局中

```
LinearLayout mLinearLayout;

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Create a LinearLayout in which to add the ImageView
    mLinearLayout = new LinearLayout(this);

    // Instantiate an ImageView and define its properties
    ImageView i = new ImageView(this);
    i.setImageResource(R.drawable.my_image);
    i.setAdjustViewBounds(true); // set the ImageView bounds to match the
Drawable's dimensions
    i.setLayoutParams(new Gallery.LayoutParams(LayoutParams.WRAP_CONTENT,
LayoutParams.WRAP_CONTENT));
```

```
    // Add the ImageView to the layout and set the layout as the content
view
    mLinearLayout.addView(i);
    setContentView(mLinearLayout);
}
```

在其它情况下，你可能需要用Drawable对象来处理图片资源，如果要这么做，你可以跟下面的代码一样去做。

```
Resources res = mContext.getResources();
Drawable myImage = res.getDrawable(R.drawable.my_image);
```

注意：保持每个资源类型的一至，可以保证你项目状态的一致性，就不用担心有许多不同类型的对象来实例化它。例如：如果使用相同的图像资源来实例化两个Drawable对象。然后修改一个Drawables的属性（例如alpha），然后不幸得是这个效果也会出现在另一个对象上去。所以当处理同一个资源的多个实例对象时，不是直接转换为Drawable，而是应该执行tween animation

下面带XML 说明一下如何添加一个资源到ImageView(为了好玩染了下红色)

```
<ImageView
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:tint="#55ff0000"
  android:src="@drawable/my_image"/>
```

更多的工程资源信息，可以阅读Resources and Assets

4.1.2 从XML文件中创建

到如今，你应该比较熟悉按Android的原则去开发一个用户接口，因此，你也应该理解了定义一个XML文件对于对象的作用与灵活的重要性。这个理念无数次用于Drawables

如果你想创建一个Drawable对象，而这个对象并不依赖于变量或用户的交换，把它定义到XML中去应该是一个不错的方法。即使你期望在你的应用程序中改变其属性来增加用户体验。你应该考虑把对象放入XML中，因为你可以随时修改其属性。

当你在你的XML中定义了一个Drawable,保存这个XML文件到你工程目录下res/drawable目录中，然后 通过调用Resource.getDrawable()来检索并实例化，传递给它XML文件中的资源ID号。

任何Drawable的子类都支持inflate这个方法，这个方法会通过XML来实例化你的程序。任何Drawable都支持XML的扩展来利用特殊的XML属性来帮助定义对象的属性，可以查看任何Drawable子类文档来看如何定义XML文件

下面的XML定义了TransitionDrawable:

```
<transition xmlns:android="http://schemas.android.com/apk/res/android">
  <item android:drawable="@drawable/image_expand">
  <item android:drawable="@drawable/image_collapse">
</transition>
```

保存上面的文件为res/drawable/expand_collapse.xml，下面的代码去实例化一个TransitionDrawable,并把它设置为ImageView的内容

```
Resources res = mContext.getResources();
TransitionDrawable transition = (TransitionDrawable)
res.getDrawable(R.drawable.expand_collapse);
ImageView image = (ImageView) findViewById(R.id.toggle_image);
image.setImageDrawable(transition);
```

接着，这个函数可以让它允许1分钟。

```
transition.startTransition(1000);
```

参考Drawable类列表可以得到更多所支持的XML属性。

4.2 ShapeDrawable

当你想去画一些动态的二维图片，一个ShapeDrawable对象可能会对你有很大的帮助。通过ShapeDrawable，你可以通过编程画出任何你想到的图像与样式
ShapeDrawable继承了Drawable,所以你可以调用Drawable里有的函数，比如视图的背景，通过setBackgroundDrawable()设置。当然，你可以在自定义的视图布局中画你的图形，因为ShapeDrawable有自己的draw()方法。你可以创建一个视图的子类去画ShapeDrawable在View.OnDraw()方法期间。这是一个基本的视图扩展类去画ShapeDrawable

```
public class CustomDrawableView extends View {
    private ShapeDrawable mDrawable;

    public CustomDrawableView(Context context) {
        super(context);

        int x = 10;
        int y = 10;
        int width = 300;
        int height = 50;

        mDrawable = new ShapeDrawable(new OvalShape());
        mDrawable.getPaint().setColor(0xff74AC23);
        mDrawable.setBounds(x, y, x + width, y + height);
    }

    protected void onDraw(Canvas canvas) {
        mDrawable.draw(canvas);
    }
}
```

在这个构造函数中，ShapeDrawable是定义了个ovalShape类型,然后又设置了颜色与边界，如果你不去设置边界，这个图形是不会画出来的，但是，如果你不设置颜色，它默认为黑色。
通过视图的自定义，你可以通过各种方法画出你喜欢的东东来，下面是个简单的例子。

```
CustomDrawableView mCustomDrawableView;

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    mCustomDrawableView = new CustomDrawableView(this);

    setContentView(mCustomDrawableView);
}
```

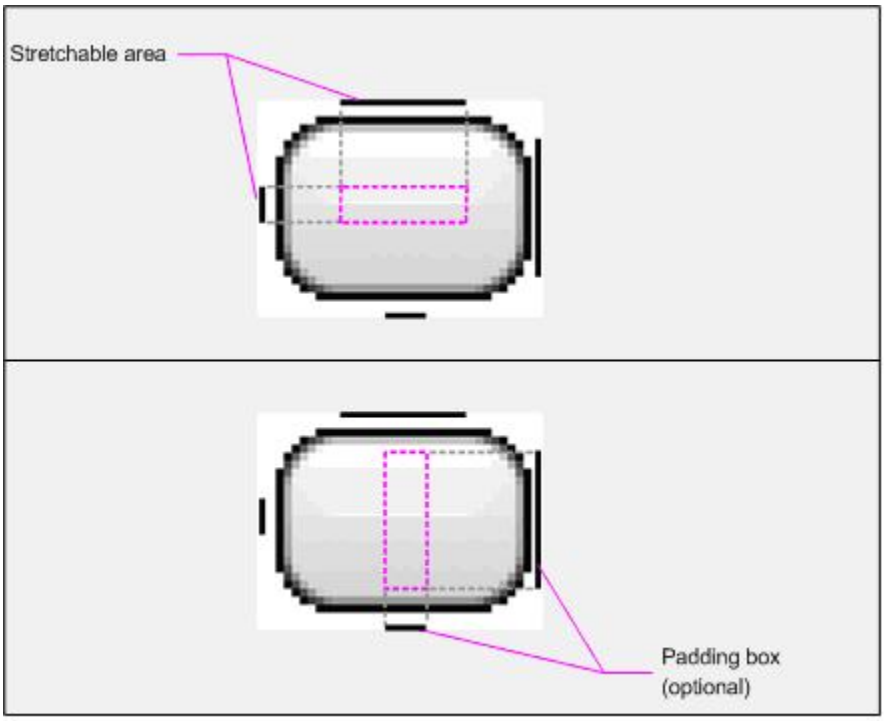
如果你想用XML文件配置来取代原有布局来画自定义的drawable,于是你自定义的Drawable类必须重载view(Context,AttributeSet)构造函数。下面是范例：

```
<com.example.shapedrawable.CustomDrawableView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
/>
```

ShapeDrawable类（像很多其他Drawable类型在android.graphics.drawable包）允许你定义drawable公共方法的各种属性。有些属性你可以需要调整，包括，透明度，颜色过滤，不透明度，颜色。

4.3 NinePatchDrawable

NinePatchDrawable 绘画的是一个可以伸缩的位图图像，Android会自动调整大小来容纳显示的内容。一个例子就是NinePatch为背景，使用标准的Android按钮，按钮必须伸缩来容纳长度变化的字符 NinePatchDrawable是一个标准的PNG图像，它包括额外的1个像素的边界，你必须保存它后缀为.9.png,并且保持到工程的res/drawable目录中。
这个边界是用来确定图像的可伸缩和静态区域。你可以在左边和上边的线上画一个或多个黑色的1个像素指出可伸缩的部分（你可以需要很多可伸缩部分），它的相对位置在可伸缩部分相同，所以大的部分总是很大的。
你还有可以在图像的右边和下边画一条可选的drawable区域（有效的，内边距线）。如果你的视图对象设置NinePath为背景然后指定特殊的视图字体，它将自行伸缩使所有的文本来适应根据右线与底部线设计好的区域（如果有的话），当然内边距线不包括其中，Android可以使用左边的线与上面的线来定义一个drawable区域。
我们来澄清一下这两条不同的线，左边跟顶部的线来定义哪些图像的像素允许在伸缩时被复制。底部与右边的线用来定义一个相对位置内的图像，视图的内容就放入其中。
下面是一个例子用NinePatch 文件来定义个按钮



NinePath通过左边的线与顶部的线定义一个可伸缩的区域而底部的线与右边的线可以定义个可画区域，在上面的图片中，灰色点的线定义了个图像的区域为了图像的伸缩，图像底部的粉红色的矩形框定义了个视图内向允许呈现的区域，如果内容不适合这个区别，它们将会将图像拉伸。
Draw9-path工具提供了一个非常简单的方法去创建你的NinePath图像，. 利用WYSIWY图形编辑器，如果你定义的区域可伸缩区域有超出图纸范围的危险它甚至会提出警告。
下面是一个简单的XML来演示如何在一对按钮上添加NinePatch(NinePath图片保存在res/drawable/my_button_background.9.png

```
<Button id="@+id/tiny"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerInParent="true"
        android:text="Tiny"
        android:textSize="8sp"
        android:background="@drawable/my_button_background"/>

<Button id="@+id/big"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerInParent="true"
        android:text="Biiiiiiig text!"
        android:textSize="30sp"
        android:background="@drawable/my_button_background"/>
```

需要注意的是宽度与高度需要设置为wrap_content,使按钮适合文字。
下面是两个按钮基于XML跟NinePath图片的呈现，随着按钮通过文字大小的变化，背景图像也会去适应。

4.4 Tween Animation

一个tween动画将对视图对象中的内容进行一系列简单的转换（位置，大小，旋转，透明性）。如果你有一个文本视图对象，你可以移动它，旋转它，让它变大或让它变小，如果文字下面还有背景图像，背景图像也会随着文件进行转换。Animation package 提供了所有的类来供tween 动画使用 。
Tween 动画定义了一个动画指令的队列，定义可以使用XML也可以再Android的代码中，像定义布局一样，我们建议使用XML来定义，因为它具备的阅读性，重用性，可以交换性大大超过了硬编码。在下面的例子中，我们使用XML（参考AnimationSet 类或者其它的动画类来学习如何在代码中定义）
动画的指令定义了你想要发生什么样的转换，当他们发生了，应该执行多长时间，转换可以是连续的也可以使同时的。例如，你让文本内容从左边移动到右边，然后旋转180度，或者在移动的过程中同时旋转，没个转换需要设置一些特殊的参数（开始和结束的大小尺寸的大小变化，开始和结束的旋转角度等等，也可以设置些基本的参数（例如，开始时间与周期），如果让几个转换同时发生，可以给它们设置相同的开始时间，如果按序列的话，计算开始时间加上其周期。
动画的XML文件还是在你工程中res/anim目录，这个文件必须包含一个根元素，可以使<alpha> <scale> <translate> <rotate>
插值元素或者是把上面的元素都放入<set>元素组中，默认情况下，所以的动画指令都是同时发生的，为了让他们按序列发生，需要设置一个特殊的属性startOffset,下面的例子会演示。

下面的ApiDemos XML文件定义了视图对象的伸缩功能，同时发生旋转

```
<set android:shareInterpolator="false">
   <scale
          android:interpolator="@android:anim/
accelerate_decelerate_interpolator"
          android:fromXScale="1.0"
          android:toXScale="1.4"
```

```
          android:fromYScale="1.0"
          android:toYScale="0.6"
          android:pivotX="50%"
          android:pivotY="50%"
          android:fillAfter="false"
          android:duration="700" />
   <set android:interpolator="@android:anim/decelerate_interpolator">
      <scale
             android:fromXScale="1.4"
             android:toXScale="0.0"
             android:fromYScale="0.6"
             android:toYScale="0.0"
             android:pivotX="50%"
             android:pivotY="50%"
             android:startOffset="700"
             android:duration="400"
             android:fillBefore="false" />
      <rotate
             android:fromDegrees="0"
             android:toDegrees="-45"
             android:toYScale="0.0"
             android:pivotX="50%"
             android:pivotY="50%"
             android:startOffset="700"
             android:duration="400" />
   </set>
 </set>
```

屏幕坐标（本例中没有使用）是（0,0）在左上角，并向右增加下去。
还有许多值，像pivotX指定对象是相对自己还是父类。必须使用正确的格式来设置（”50” 是相对于父类50% “50%”相对于自己%50）
您可以决定如何转变在一段时间内适用于指定的插值，Android包括了几个插值子类指定不同的速度曲线，例如：AccelerateInterpolator 告诉了开始比较慢，速度慢慢的变快，每个都有一个属性值，可以用于在XML中 。
保存hyperspace_jump.xml到你工程中的res/anim目录，下面的JAVA代码将引用它来布局一个ImageView对象。

```
ImageView spaceshipImage = (ImageView) findViewById(R.id.spaceshipImage);
Animation hyperspaceJumpAnimation = AnimationUtils.loadAnimation(this,
R.anim.hyperspace_jump);
spaceshipImage.startAnimation(hyperspaceJumpAnimation);
```

作为替代startAnimation(),你可以通过Animation.setStartTime来定义开始时间，通过view.setAnimationl来指定这个动画
关于XML的参数 属性 有效的标记，可以参见Available Resources中动画的讨论。
注意：你的动画不顾后果的移动或调整大小，视图不会去适应你的动画而去自动调整，即使如此，动画将会超出视图的范围，但不会被截断，然而，截断发生在你的动画如果超出了父类的视图。

4.5 Frame Animation

创建一个连续不同的图片的序列是传统动画的场景，按照指令播放，就像滚动的电影，在Android中基础类AnimationDrawable就是专门来负责帧动画。

虽然你可以在代码中定义帧动画，可以使用AnimationDrawable类的API.，它是非常简单通过XML文件列出动画中的所有帧，像上面的动画tween，这种类别动画的XML文件放入工程中的res/anim目录。既然这样，指令按照周期去执行每帧动画。
在XML文件包含一个<animation-list>根节点元素和好几个子节点<item>来定义每帧。一个资源分别定义了帧的名字与帧的持续时间。，下面为范例：

```
<animation-list xmlns:android="http://schemas.android.com/apk/res/
android"
    android:oneshot="true">
    <item android:drawable="@drawable/rocket_thrust1"
android:duration="200" />
    <item android:drawable="@drawable/rocket_thrust2"
android:duration="200" />
    <item android:drawable="@drawable/rocket_thrust3"
android:duration="200" />
</animation-list>
```

这个动画播放三个帧动画，通过设置android:oneshot属性为true,它将会在最后一帧停下来，如果设置为false这个动画将循环播放。这个文件保存到工程目录res/anim目录下为rocket_thrust.xml,你也可以添加一个背景图片到视图中，然后开始播放。下面为范例：

```
AnimationDrawable rocketAnimation;

public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);

  ImageView rocketImage = (ImageView) findViewById(R.id.rocket_image);
  rocketImage.setBackgroundResource(R.anim.rocket_thrust);
  rocketAnimation = (AnimationDrawable) rocketImage.getBackground();
}

public boolean onTouchEvent(MotionEvent event) {
  if (event.getAction() == MotionEvent.ACTION_DOWN) {
    rocketAnimation.start();
    return true;
  }
  return super.onTouchEvent(event);
}
```

一个比较需要特别注意的是，在AnimationDrawable调用onCreate()过程中不能调用start()，这是因为AnimationDrawable不能在不完全的窗口上运行，如果你想立即播放动画，没有必要的交互，你可以再onWindowFocusChanged()方法中调用它。这样它将成为窗口焦点

5. 图片的缩放和旋转

作者：IceskYsl(http://iceskysl.1sters.com/)

5.1 目标：

本文将讲述如何如何在Android中使用Matrix实现图片的缩放和旋转，通过本文学习，你将学会如何通过Matrix操作图像。

5.2 代码示例：

直接上代码了，我在代码中附带了详细的解释，代码如下：

```
package com.eoeandroid.demo.testcode;

import android.app.Activity;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Matrix;
import android.graphics.drawable.BitmapDrawable;
import android.os.Bundle;
import android.view.ViewGroup.LayoutParams;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.ImageView.ScaleType;

public class bitmaptest extends Activity {
public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setTitle("eoeAndroid教程: 缩放和旋转图片 -by:IceskYsl");
        LinearLayout linLayout = new LinearLayout(this);

        // 加载需要操作的图片，这里是eoeAndroid的logo图片
        Bitmap bitmapOrg = BitmapFactory.decodeResource(getResources(),
               R.drawable.eoe_android);

        //获取这个图片的宽和高
        int width = bitmapOrg.getWidth();
        int height = bitmapOrg.getHeight();

        //定义预转换成的图片的宽度和高度
        int newWidth = 200;
        int newHeight = 200;

        //计算缩放率，新尺寸除原始尺寸
        float scaleWidth = ((float) newWidth) / width;
        float scaleHeight = ((float) newHeight) / height;

        // 创建操作图片用的matrix对象
        Matrix matrix = new Matrix();
```

```
        // 缩放图片动作
        matrix.postScale(scaleWidth, scaleHeight);

        //旋转图片 动作
        matrix.postRotate(45);

        // 创建新的图片
        Bitmap resizedBitmap = Bitmap.createBitmap(bitmapOrg, 0, 0,
                          width, height, matrix, true);

        //将上面创建的Bitmap转换成Drawable对象，使得其可以使用在ImageView, ImageButton中
        BitmapDrawable bmd = new BitmapDrawable(resizedBitmap);

        //创建一个ImageView
        ImageView imageView = new ImageView(this);

        // 设置ImageView的图片为上面转换的图片
        imageView.setImageDrawable(bmd);

        //将图片居中显示
        imageView.setScaleType(ScaleType.CENTER);

        //将ImageView添加到布局模板中
        linLayout.addView(imageView,
          new LinearLayout.LayoutParams(
                      LayoutParams.FILL_PARENT, LayoutParams.FILL_PARENT
                )
        );

        // 设置为本activity的模板
        setContentView(linLayout);
    }
}
```

这里没用定义XML布局模板，而是直接在代码中生成了需要的模板和视图组件，你可以可以定义XML模板，其他原理是一样的。

5.3 展示效果

运行项目，其显示效果如下(图片被我拉伸了 :( )：

6. 3D 和 OpenGL

译者：404
原文：http://developer.android.com/guide/topics/graphics/opengl.html

通过OpenGL API，Android包含对高性能3D图形的支持 — 具体而言，是指OpenGL ES API 。OpenGL ES 是根据 OpenGL 规范进行裁剪后定制而形成的一套标准，针对嵌入式设备而设计。 各版本的 OpenGL ES 都广泛地向主要的OpenGL 标准看齐。目前，Android 支持 OpenGL ES 1.0，即相当于 OpenGL 1.3 。换言之，假设你的应用程序能够在桌面系统上结合OpenGL 1.3运行，那么它亦将能够运行在Android上。

Android 提供的这一API类似于 J2ME 中的OpenGL ES API（JSR 239） 。然而，有可能并非完全一致；因此，对一些偏差要多加留意。

6.1 使用 API

这里给出一个比较抽象的使用API的步骤：

1. 编写一个自定义的 View 子类。
2. 取得一个OpenGLContext句柄，以便使用 OpenGL 提供的那些功能。
3. 在View 类的 onDraw() 方法里，取得一个指向GL对象的句柄，并调用它所提供的一些方法来执行GL操作。

关于这方面的一个例子（基于经典的 GL ColorCube），可以看看如何结合线程一并使用com.android.samples.graphics.GLSurfaceViewActivity.java.

如何使用 OpenGL 来实际地编写 3D 应用程序已超出本篇内容的范畴，在此不作介绍；读者朋友也可以将此作为余下的练习。

6.2 附加信息

您可以从这里查阅关于OpenGL ES 的一些信息 http://www.khronos.org/opengles/.
欲了解OpenGL ES 1.0 的详细信息，参考 http://www.khronos.org/opengles/1_X/.
还可以查阅这篇针对 Android 的文档：OpenGL ES implementations.
最后，请注意，虽然Android包含一些针对OpenGL ES 1.1的基本支持，但是这些支持**尚未完善**，而且当前并不可靠。

7. GLSurfaceView 介绍

7.1 GLSurfaceView介绍

GLSurfaceView 是在Android 1.5 API 中新增加的一个类, GLSurfaceView 可以让你在书写OpenGL ES应用程序中更容易实现以下方面:

- . 提供一个glue 代码用于连接OpenGL ES 与 View System .
- . 提供一个 glue 代码可以编写OpenGL ES代码在 Activity 的生命周期中.
- . 可以很方便的选择相近的pixel帧缓冲格式.
- . 可以分别创建和管理渲染线程用于开启平滑动画.
- . 可以更容易的使用调试工具 输出OpenGL ES API的调用和错误检查.

GLSurfaceView 是一个很的好方法在使用OpenGL ES去渲染部分或者全部应用,它将是2D 和 3D 动作游戏选择，可以用于二维或三维数据可视化应用程序当中，例如谷歌地图街景。

简单的GLSurfaceView应用程序示例:

```
package com.example.android.apis.graphics;

import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.opengles.GL10;

import android.app.Activity;
import android.opengl.GLSurfaceView;
import android.os.Bundle;

public class ClearActivity extends Activity {
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
mGLView = new GLSurfaceView(this);
mGLView.setRenderer(new ClearRenderer());
setContentView(mGLView);
}

@Override
protected void onPause() {
super.onPause();
mGLView.onPause();
}

@Override
protected void onResume() {
super.onResume();
mGLView.onResume();
}

private GLSurfaceView mGLView;
}
```

```
class ClearRenderer implements GLSurfaceView.Renderer {
public void onSurfaceCreated(GL10 gl, EGLConfig config) {
// Do nothing special.
}

public void onSurfaceChanged(GL10 gl, int w, int h) {
gl.glViewport(0, 0, w, h);
}

public void onDrawFrame(GL10 gl) {
gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
}
}
```

这段程序不会这样去做:在每一帧中把屏幕清空为黑色. 但它是一个完整的OpenGL应用程序在实现android Activity 的整个生命周期中.随着activity的暂停而暂停渲染，在activity 恢复时开始恢复渲染. 可以使用此应用程序为基础，非交互式演示项目, 只是学要添加更多的OpenGL调用,在ClearRenderer.onDrawFrame方法里.注意，你甚至不需要GLSurfaceView子类.

GLSurfaceView.Renderer 接口需要实现三个方法:
onSurfaceCreated()方法被调用在开始渲染的时候. OpenGL ES绘图上下文时都会被重建(当activity暂停和恢复的时候，绘图的上下文也通常会随之丢失和重建) OnSurfaceCreated()方法主要用于创建持久的OpenGL 资源，类似于textures一样.

onSurfaceChanged()方法在surface大小尺寸改变的时候被调用, 它主要设置你的OpenGL 的观察点.你也可以在这里设置一个不会被移动的固定camera.

onDrawFrame()方法会在每帧中被调用,用于描述一个时时绘制的场景.你还可以通过调用glClear方法去清空帧缓冲，接着通过其它OpenGL ES 调用去绘制目前的场景.

7.2 关于用户的输入?

如果你想创建一个可交互的应用(类似于游戏一样), 你通常会定义一个GLSurfaceView的子类.这样你可以很荣誉获得输入事件, 这里有一个较长的例子显示如何做的:

```
package com.google.android.ClearTest;

import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.opengles.GL10;

import android.app.Activity;
import android.content.Context;
import android.opengl.GLSurfaceView;
import android.os.Bundle;
import android.view.MotionEvent;

public class ClearActivity extends Activity {
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
mGLView = new ClearGLSurfaceView(this);
setContentView(mGLView);
}

@Override
protected void onPause() {
super.onPause();
mGLView.onPause();
}

@Override
protected void onResume() {
super.onResume();
mGLView.onResume();
}

private GLSurfaceView mGLView;
}

class ClearGLSurfaceView extends GLSurfaceView {
public ClearGLSurfaceView(Context context) {
super(context);
mRenderer = new ClearRenderer();
setRenderer(mRenderer);
}

public boolean onTouchEvent(final MotionEvent event) {
queueEvent(new Runnable(){
public void run() {
mRenderer.setColor(event.getX() / getWidth(),
event.getY() / getHeight(), 1.0f);
}});
return true;
}

ClearRenderer mRenderer;
}

class ClearRenderer implements GLSurfaceView.Renderer {
public void onSurfaceCreated(GL10 gl, EGLConfig config) {
// Do nothing special.
}

public void onSurfaceChanged(GL10 gl, int w, int h) {
gl.glViewport(0, 0, w, h);
}

public void onDrawFrame(GL10 gl) {
```

```
gl.glClearColor(mRed, mGreen, mBlue, 1.0f);
gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
}

public void setColor(float r, float g, float b) {
mRed = r;
mGreen = g;
mBlue = b;
}

private float mRed;
private float mGreen;
private float mBlue;
}
```

上面的程序会在每一帧中清除屏幕显示，当你点击屏幕的时候，它会根据你点击时的事件在你当前坐标点清楚颜色.注意在使用queueEvent()方法在ClearGLSurfaceView. onTouchEvent()中. The queueEvent() 方法是用来在UI 线程操作和渲染线程操作中进行安全的信息交流。如果愿意，你可以用一些其他的Java 跨线程联合技术，例如synchronized方法自己渲染。但是queueing events通常是处理跨线程信息交流的最简单的方式

7.3 GLSurfaceView 其他例子:

你是否厌倦了只会用来清除屏幕的方式?你可以找到更多有趣的有关API Demo的例子在SDK中.所有的OpenGL ES 例子都可以转换成使用GLSurfaceView:
. GLSurfaceView - 一个类三角形。
. Kube - 一个立方体拼图演示
. 半透明 GLSurfaceView - 显示如何显示三维图形的透明背景
. 纹理三角型 - 显示如何绘制出质感的3D三角
. Sprite Text - 展示如何在字体上应用纹理并且应用到3D场景当中.
. 触摸旋转 - 显示如何旋转三维物体响应用户输入。

7.4 选择 一个 Surface

GLSurfaceView 用于你选择渲染surface的类型.不同Android 器件支持不同的surfaces类型,他们是没有共同的子集.不过这样一来给每个器件选择最佳的surface带来了麻烦.在默认情况下GLSurfaceView试图找到一个surface ， 尽可能的接近16位的RGB帧缓冲和16位的深度缓冲. 这取决于您的应用程序的需求,你也可能需要改变这种方式. 例如:半透明的GLSurfaceView这样的例子需要一个alpha通道渲染半透明的数据.
GLSurfaceView提供了一个重载setEGLSurfaceChooser()方法提供给开发者控制选择surface的类型:

setEGLConfigChooser(boolean needDepth)
选择一个接近R5G6B5或者不在 16位的帧缓冲范围的配置.
setEGLConfigChooser(int redSize, int greenSize,int blueSize, int alphaSize,int depthSize, int stencilSize)
选择用数量最少的每像素位数，至少和创建者规定的每通道中bit数一样。Choose the config with the fewest number of bits per pixel that has at least as many bits-per-channel as specified in the constructor.

setEGLConfigChooser(EGLConfigChooser configChooser)
允许完全控制选择配置。通过执行EGLConfigChooser ，会检查设备的功能和选择配置。

7.5 Continuous Rendering vs. Render When Dirty

在大多数3D应用程序，例如游戏或模拟器.以及连续的动画,但是一些3D 应用程序大部分的反应是:他们只是在消极的等待直到用户去操作然后回应给应用程序.对于这些应用程序,默认的GLSurfaceView 方式会不断的浪费时间用于重绘屏幕.如果你正在开发一种反馈式的应用，你可以调用GLSurfaceView.setRenderMode (RENDERMODE_WHEN_DIRTY)方法 关闭持续的动画. 然后调用GLSurfaceView.requestRender()方法当你想重新渲染的时候.

7.6 Help With Debugging

GLSurfaceView有一个方便的内置功能可以调试OpengGL ES应用. 在GLSurfaceView.setDebugFlags()方法可以用来启用日志记录和/或错误检查您的OpenGL ES的调用.在GLSurfaceView的构造函数中先于setRenderer()方法前调用此方法:

```
public ClearGLSurfaceView(Context context) {
super(context);
// Turn on error-checking and logging
setDebugFlags(DEBUG_CHECK_GL_ERROR | DEBUG_LOG_GL_CALLS);
mRenderer = new ClearRenderer();
setRenderer(mRenderer);
}
```

8. 其他

翻译人员

- apcwowo
- IceskYsl(http://iceskysl.1sters.com)
- haiyangjy(http://www.haiyangjy.com)
- 404(http://1404.blogspot.com)
- zhoubo5262
- binbinming

BUG提交

如果你发现文档中翻译不妥的地方，请到如下地址反馈，我们会定期更新、发布更新后的版本
http://www.eoeandroid.com/viewthread.php?tid=355&extra=

参加翻译

如果你有兴趣参加我们的协同翻译，请参考如下链接
http://www.eoeandroid.com/forumdisplay.php?fid=39

关于eoeAndroid

当前，3G商业，传统互联网与移动互联网也呈现出全业务发展的融合趋势，电信与互联网行业已经踏入继单机计算时代、传统互联网时代之后的第三个纪元。
由于看好移动互联网和Android手机平台的商业前景，同时也拥有专业而独特的产品、技术服务能力，我们聚集了一群热爱Android的技术英才，组建了eoeMobile团队。
eoeMobile是一支专注于Android平台应用开发、产品运营和相关商业与技术服务的团队，立志于建立中国最大的Android应用开发专业社区eoeAndroid.com ，想为Android在中国的发展尽自己的微薄之力。