eoe特刊

第十一期：Android User Interface Design

Android UI设计

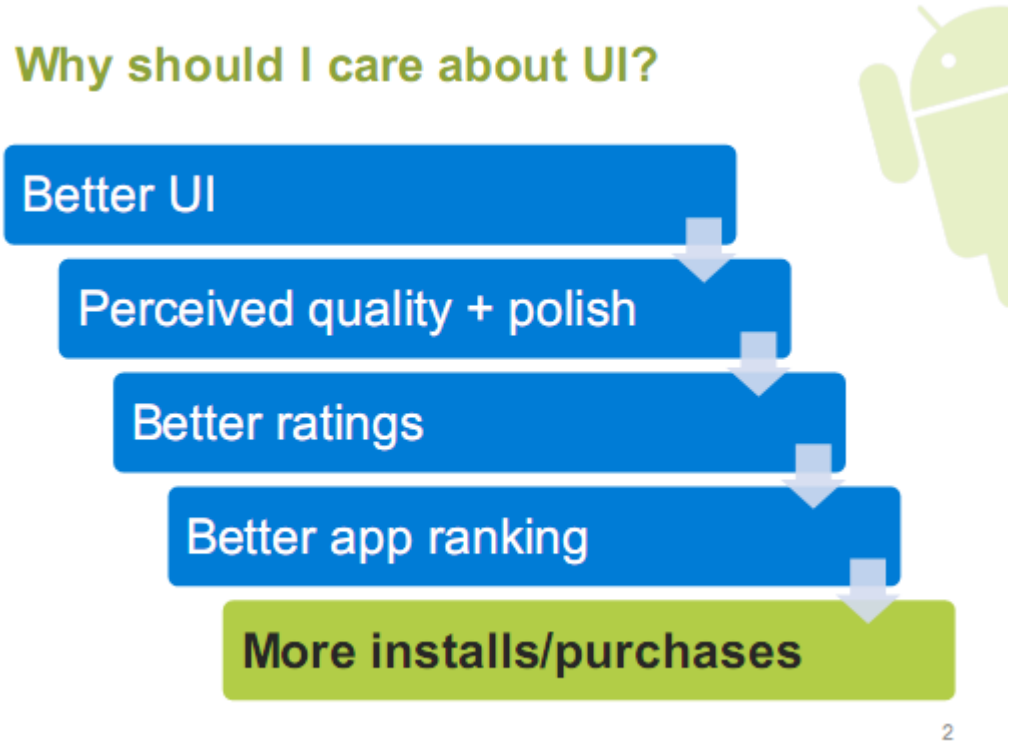
eoeandroid.com
做最棒的Android开发社区

MOBILE eoe 易联致远

优亿市场
Android应用发布与分享平台

目录

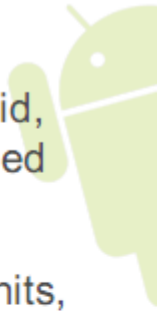
【Google 官方 Android 设计指导】



Agenda – Android UI design tips

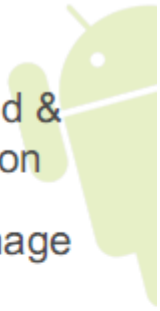
1. Do's and don'ts
2. Design philosophy and considerations
3. UI framework features you should *definitely* be using
4. New UI design patterns
5. Icons and guidelines

3



- **DON'T** simply port your UI from other platforms
  - Users should **feel right at home** with your app **on their device**
  - Balance your brand and platform look
- **DON'T** overuse modal progress & confirmation dialogs
- **DON'T** create rigid, absolute-positioned layouts
- **DON'T** use `px` units, use `dp` (or `sp` for text)
- **DON'T** use small font sizes

5



- DO create versions of **all** resources for high density screens
- DO make large, obvious tap targets (buttons, list items)
- DO follow Android icon guidelines
- DO use proper margins and padding
- DO support D-pad & trackball navigation
- DO properly manage the activity stack
- DO properly handle orientation changes
- DO use theme/style, dimension, color resources to reduce redundancy

6

DO work with visual and interaction designer(s)

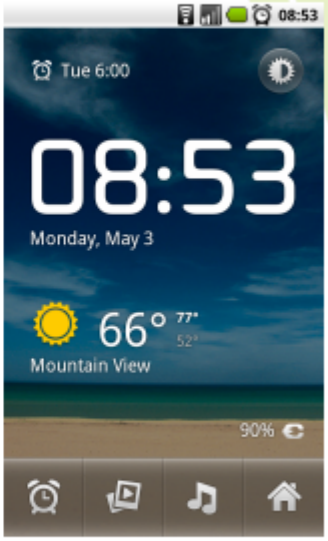
7

Design philosophy and considerations

Android design philosophy

Android设计哲学

- Clear vs. “simple”
- Content vs. chrome
- Consistent yet engaging
  - Elegant variation
- Enhanced by the cloud
  - Maintain user’s context across desktop and mobile



Principles of good interface design*

1. Focus on the user
2. Make the right things visible
3. Show proper feedback
4. Be predictable 可预见
5. Be fault-tolerant 可容错

* Some material borrowed from Donald Norman's *The Design of Everyday Things*

1. Focus on the user

- Know your users
  - Age, skill level, culture, disabilities, etc.
  - What they want to do with your app
  - What kinds of devices they'll be using
  - Where/when/how they'll be using their devices
- Design with a 'user-first' mentality
  - Users are *generally* task-driven
- Test on real users, early and often

2. Make the right things visible

- The most common operations should be immediately visible and available
- Secondary functionality can be reserved for the **MENU** button

12

3. Show proper feedback

- Have at least 4 states (<selector>) for all interactive UI elements:

- Make sure the *effects* of an action are clear and visible
- Show adequate yet unobtrusive progress indicators

进度条，要完全显示，但不要过分突出

13

4. Be predictable

- Do what the user expects
  - Properly interact with the **activity stack**
  - Show information and actions users expects to see (requires testing or observation)
- Use proper affordances
  - If something is clickable, make sure it looks clickable!

活动stack 交互

If complex instructions are required, rethink your design.

5. Be fault tolerant

- Constrain possible operations to only those that make sense
  - Disable UI elements when appropriate
- Limit the number of irreversible actions
- Prefer 'undo' to confirmation dialogs
  - In fact, use as few *modal* dialogs as possible. They're obtrusive.

16

"If an error is possible, someone will make it."

– Donald Norman, author,
The Design of Everyday Things

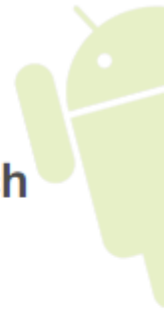
17

Design considerations

- Physical screen size
- Screen density
- Portrait & landscape orientations
- Primary UI interaction method
  - Touch-screen
  - D-pad/trackball
- Soft & physical keyboard

18

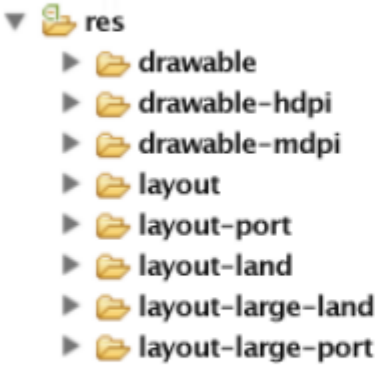
Design considerations



- **Awareness about the ways in which devices can vary is very important**
- Read through the CDD to learn about possible device UI variations
  - http://source.android.com/compatibility
- Screen size & density breakdown
  - http://developer.android.com/resources/dashboard/screens.html

19

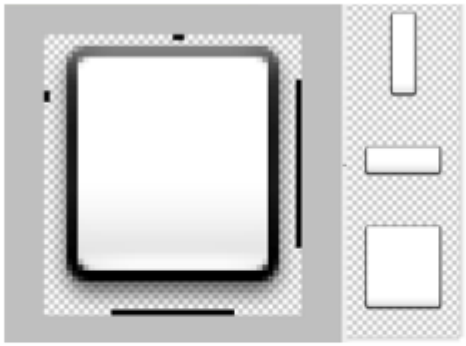
Resource qualifiers



- One .apk contains all resources
- System chooses at runtime which resources to use

22

9-patch drawables – `foo.9.png`



- Similar to CSS3 `border-image`
- Border pixels indicate stretchable regions
- Make both `-mdpi` and `-hdpi` versions!

23

Selector (state list) drawables

▶ drawable

foo.xml:

```
<selector>
    <item android:drawable="@drawable/foo_disabled"
          android:state_enabled="false" ... />
    <item android:drawable="@drawable/foo_pressed"
          android:state_pressed="true" ... />
    <item android:drawable="@drawable/foo_focused"
          android:state_focused="true" ... />
    <item android:drawable="@drawable/foo_default" />
</selector>
```

24

Selector (state list) drawables

▶ drawable-mdpi

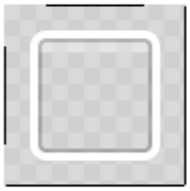
foo_default.png foo_disabled.png foo_focused.png foo_pressed.png

▶ drawable-hdpi

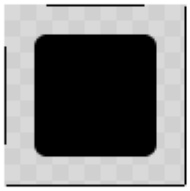
foo_default.png foo_disabled.png foo_focused.png foo_pressed.png

25

Layer drawables – XML + PNGs



foo_border.9.png



foo_mask.9.png

```
Drawable.setColorFilter(
    0xA4C639,
    ...);
```

26

Layer drawables – XML + PNGs

Rendered output
(resizable w/ 9-patch)

27

New UI design patterns

Borrowed from the
Android UI design patterns
talk at Google I/O 2010

http://code.google.com/events/io/2010/sessions/android-ui-design-patterns.html

29

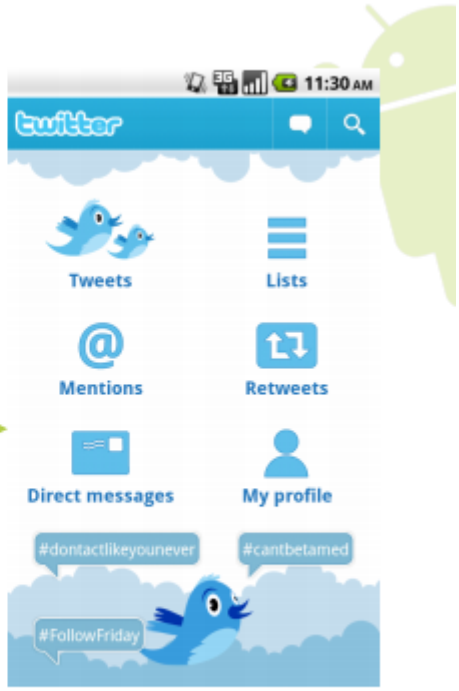
New UI design patterns

- Dashboard
- Action Bar
- Quick Actions

30

New UI design patterns

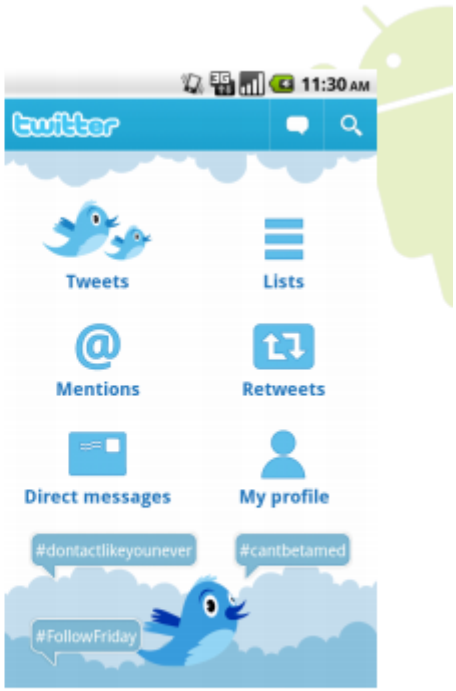
- Dashboard
- Action Bar
- Quick Actions



31

New UI design patterns

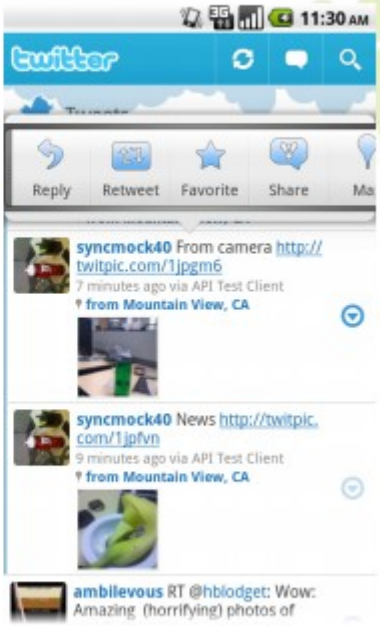
- Dashboard
- Action Bar
- Quick Actions



32

New UI design patterns

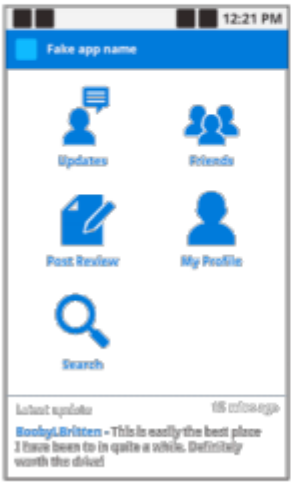
- Dashboard
- Action Bar
- Quick Actions



33

Dashboard

"What can I do with this app?"
"What's new?"



34

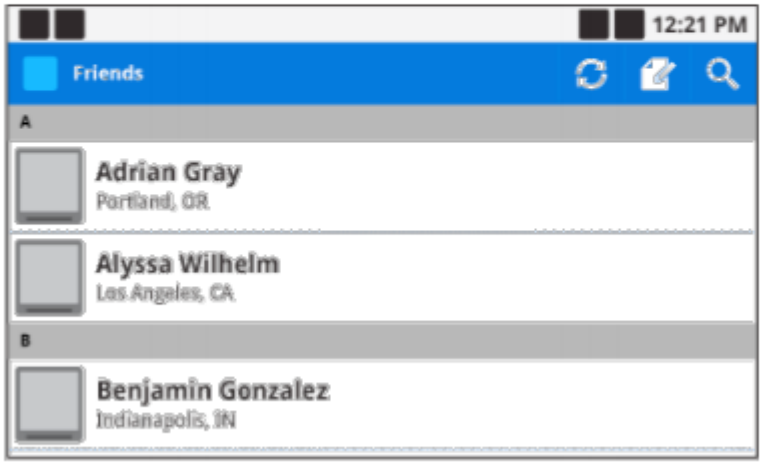
Dashboard – Recommendations

- Focus on 3-6 most important app sections
- Highlight what's new
- Be flavorful – it's your **first impression**

35

Action Bar

"How can I do <common action> quickly?"



36

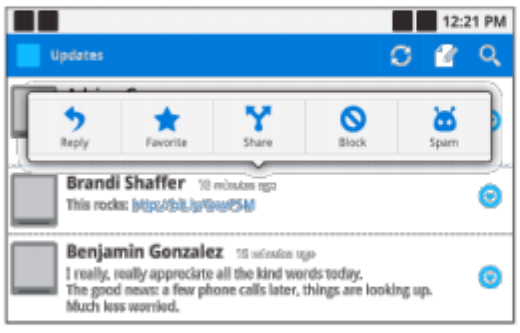
Action Bar – Recommendations

- Bring key, app-wide actions onscreen
- Help to convey a sense of place
- Use consistently within your app
- Provide 'home' mechanism – logo or dedicated button
- **DON'T** use for contextual actions

37

Quick Actions

"What can I do with <this object>?"



38

【Android UI 设计】

Android 系统图标设计原则

创造一个统一外观，感觉完整的用户界面会增加你的产品附加价值。精炼的图形风格也使用户觉得用户界面更加专业。本文档提供了一些信息，帮助你如何在应用界面的不同部分创造图标来匹配 Android 2.x 框架下的普遍风格。遵守这些原则会辅助你为用户创造一个流畅而统一的体验。为了使你创建图标的工作进行的更加快速，你可以下载 Android 图标模板包。更多信息请浏览 Android 图标模板包的使用。

Android 系统被设计在一系列屏幕尺寸和分辨率不同的设备上运行的。当你为自己的应用设计图标时，必须知道，你的应用有可能在任何设备上安装运行。正如支持多屏幕文档中所描述， Android 为你直接提供这样的图标，他们会在任何设备上正确的显示，无论这些设备的屏幕大小和分辨率如何。

一般来说，推荐的方式是为三种普遍的屏幕密度（如表 1）中的每一种都创造一套独立的图标。然后，把他们储存在你的应用中特定的资源目录下。当你的应用运行时，Android 平台将会检查设备屏幕的特性，从而加载特定密度资源目录下相应的图标。想要了解更多如何存储特定密度资源的信息，请参阅创造合格屏幕尺寸和密度的办法目录。

Android 设备的屏幕密度基线是中等。因此，一种被推荐的为多种屏幕密度创造图标方式是：

1. 首先为基准密度设计图标（看表一为实际的像素尺寸设计的图标）。
2. 把图标放在你的应用的默认可绘制资源中，然后在 Android 可视化设备（AVD）或者 HVGA 设备如 T-Mobile G1 中运行应用。
3. 根据需要测试和调整你的基准图标。
4. 当你对在基准密度下创建的图标感到满意的时候，为其他密度创造副本。
   把基准图标按比例增加为 150%，创造一个高密度版本。
   把基准图标按比例缩小为 75%，创造一个低密度版本。
5. 把图标放入你的应用的特定密度资源目录中。例如：
   中密度版本在 res/drawable-mdpi/ 目录下运行（或在默认 res/drawable/ 目录下运行）
   高密度版本在 res/drawable-hdpi/目录下运行。
   高密度版本在 res/drawable-ldpi/目录下运行。
6. 如果需要，测试和调整高密度和低密度的图标。

表 1. 对三种普遍屏幕密度中每一种密度的所需要的成品尺寸图标的摘要

Icon Type　　对于普遍的屏幕密度标准版本尺寸(像素表示),

	低密度屏幕(ldpi)	中密度屏幕(mdpi)	高密度屏幕(hdpi)
启动器	36 x 36 px	48 x 48 px	72 x 72 px
菜单	36 x 36 px	48 x 48 px	72 x 72 px
状态栏	24 x 24 px	32 x 32 px	48 x 48 px
标签	24 x 24 px	32 x 32 px	48 x 48 px
对话	24 x 24 px	32 x 32 px	48 x 48 px
列表视图	24 x 24 px	32 x 32 px	48 x 48 px

启动器图标

启动器图标是一个图形，代表了设备的主页和启动器窗口中的应用。

用户会在点击主页底部的图标中打开启动器。启动器打开，显示所有已经安装应用的图标。他们被以格状排列。用户选择一个应用，通过任何可以得到的硬件导航控制，例如轨迹球点击启动器图标。

用户也可以把一个图标从启动器窗口中拖出来，放在主页上，来更方便的访问应用。再这种情况下，系统会显示你的应用的启动器图标在主页墙纸上的映射。此渲染的映射尺寸与在启动器中渲染的尺寸相同。

系统控制了所有启动器图标的缩放，所以他们被渲染为统一的高和宽。被渲染的启动器图标的实际像素尺寸在随着设备屏幕的像素尺寸和屏幕密度的不同而显示的不同。为了保证你的图标渲染效果最佳，请提供为低密度，中密度和高密度屏幕制作的图标。想得到更多信息，请参阅上面的提供特定密度图标集或下面的为设计师的建议。

风格

你创造启动器图标应该符合以下原则的一般风格。这个准则并不限制你可以做的图标，而是强调你的图标可以在其他设备上共享的普遍的办法。图 1 提供了例子。

图 1.启动器图标风格的插图

干净和现代:

启动器图标应该是现代的，有时有点古怪，但是他们不应该是过时和粗糙的。如果可能的话，你应该避免过度使用象征性的隐喻。

简单和标志性的:

Android 启动器图标应该是自然的抽象表现；你的图标应该高度简化和夸张，以至于他们可以在小尺寸时合适显示。你的图标不应该过于复杂。
尝试用一个简单的部分作为整体的象征性的代表（例如，音乐图标以扬声器作为特征）。
考虑使用自然轮廓和形状，包括几何的和有机的，与现实（不是照片般的真实）映射。
你的图标不应该呈现一个对更大的图像不正确的观点。

触觉和质感:

图标应该表现为不平淡的，有质感的材料。更多信息请看下面的材料和色彩。

面向前方和顶部照明:

Android 2.0 和以后的平台的新规定: Android 启动器图标应该面向前方，透视非常小，而且应该顶部照明。

此外，注意所有图标应该有独立的文字标签，而不是把文字设计嵌入到图标里面，把努力用在使图标有特色和难忘中去。
要看更多 Android 系统下应用的启动器图标的案例，请参阅标准启动器图标。

做什么和别做什么

以下有一些在你为自己的应用设计一个图标的过程中“可以做的的和不要做的”例子

Android 启动器图标是...

现代的，简约的，不平坦的，有触感，和质感的
面向前方和顶部明亮，整体而言，色彩在一定的色谱中

Android 启动器图标不是...

过时的，过分复杂，光滑的，平坦的
旋转的, 剪裁不正的,过于深色的

图 2. 启动器中"做什么和别做什么"的例子

质感和颜色

启动器的图标应该利用触觉，顶部照明，使用纹理材料。即使你的图标只是一个简单的形状，你应该尝试把它们当做真实世界的材料渲染。

该平台的默认应用中启动器图标用了以下图 3 所示的材料。 你的图标可以使用这些材料也可以创建新的材料。

Android 启动器图标经常包括由较小的形状组成一个较大的形状，并结合成一个中立的形状和一个中立的颜色。图标可能会使用中性色彩的组合，但保持较高的对比度。如果可能的话，每个图标不应该使用超过一个原色。

启动器的图标应该使用一个又限制的调色板，包含一系列中立的颜色和原色。该图标不应该过分饱和。

推荐的启动器图标调色板应用，如图 4 所示。你可以使用条侧板中的基本颜色和高亮元素。你可以使用白色到黑色垂直线性渐变叠加一起的调色板的颜色。这产生的印象是，光从顶部照射进来，且保持颜色的低饱和度。

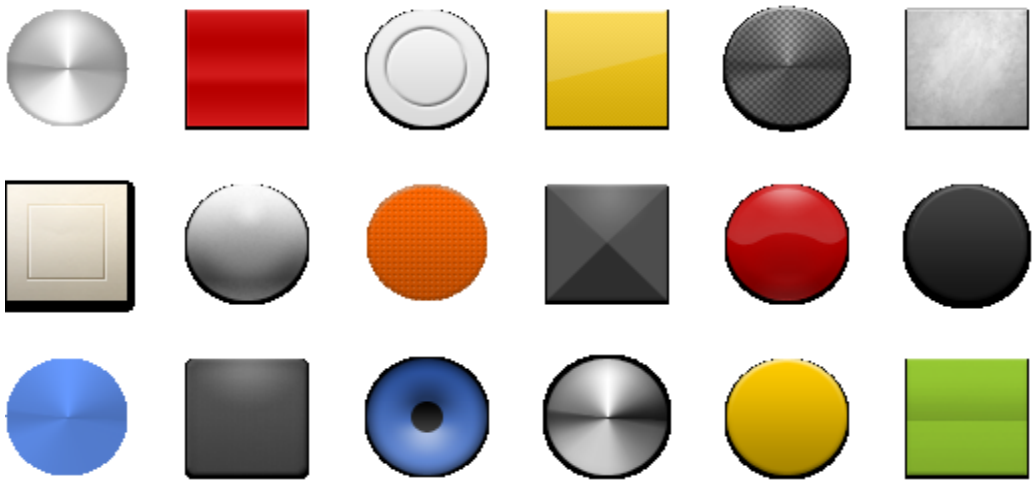


图 3.你可以用它来创建你的图标的材料的例子.

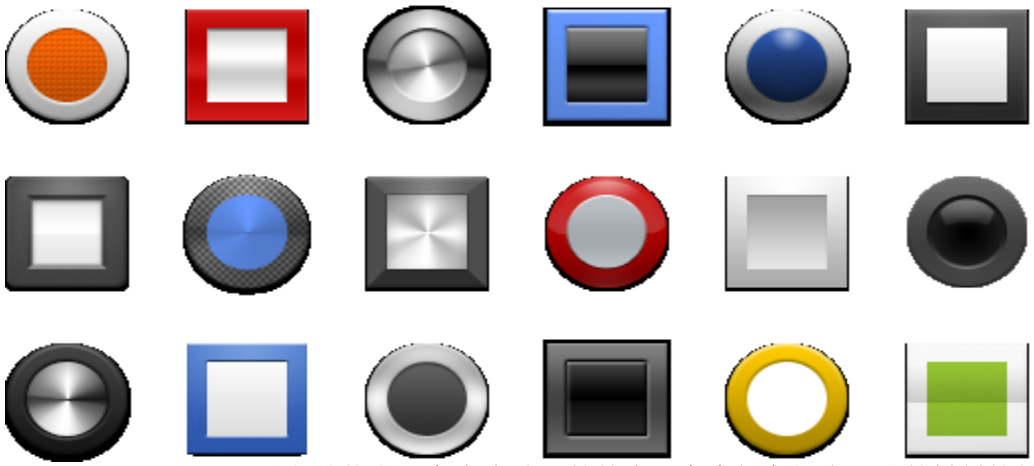


图. 从推荐调色盘中选出的基本和高亮颜色组合形成的材料的例子

当你从简易的调色盘中取出一个高亮颜色组成材料时，你可以创造如图 5 所示的材料组成。为了帮助你开始，图标包（icons pack）包括一个 Photoshop 模板文件（Launcher-icon-template.psd），文件提供了默认的材料，颜色和梯度。

5. 推荐图标调色板.

尺寸和位置

启动器的图标应该使用不同的形状和形式，而且这些必须缩被缩放和定位来创建一致的视觉重量。

图 6 展示了图标放置在各版本中的不同的方式。至于更详细的描述，就是为了制造一个一致的直觉质量，并允许加入阴影，你应该使图标比实际版本中的范围小一些。如果你的图标是方形或近方形，尺寸应该更小。

- 为全版本边界框显示为红色。
- 推荐的实际图标边界框显示为蓝色。该图标框的大小比完整版中的尺寸更小，以便有空间包含阴影和特殊的图标处理。
- 对于方形图标，推荐的边界框是橙色显示的。为正方形图标框比较小是因为要在两种类型的图表中建立同样的视觉重量。

1、 高密度屏幕(hdpi) 的图标尺寸:

1) 全版本: 72 x 72 px
2) 图标: 60 x 60 px
3) 正方形图标: 56 x 56 px

2、中密度屏幕(mdpi) 的图标尺寸:

1) 全版本: 48 x 48 px
2) 图标: 40 x 40 px
3) 正方形图标: 38 x 38 px

3、低密度屏幕(ldpi) 的图标尺寸:

1) 全版本: 36 x 36 px
2) 图标: 30 x 30 px
3) 正方形图标: 28 x 28 px

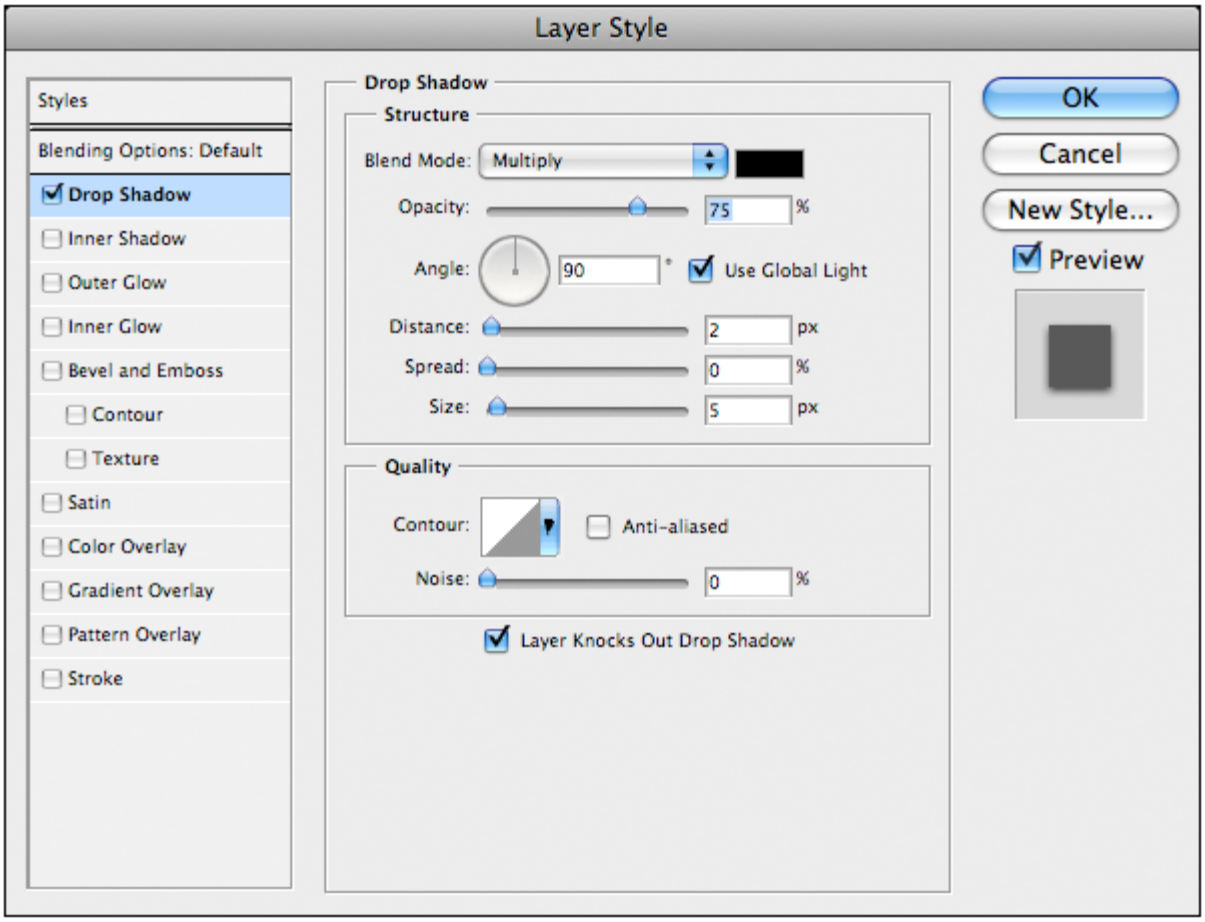
使用启动器图标模板

Android 的图标模板包 2.0 是一个包含默认图标的材料和颜色调色板的模板。该模板为 psd 格式，方便 Photoshop 或相似的图像编辑器编辑。

To get started, fir 要开始使用，首先下载 Android 的图标模板包 2.0.

一旦你下载了模板包，解压缩，并在 Adobe Photoshop 或类似的图片编辑器中中打开 Launcher-icon-template.psd ，注意调色板的材料和颜色。您可以使用该模板创建一个启动器的图标作为起点。

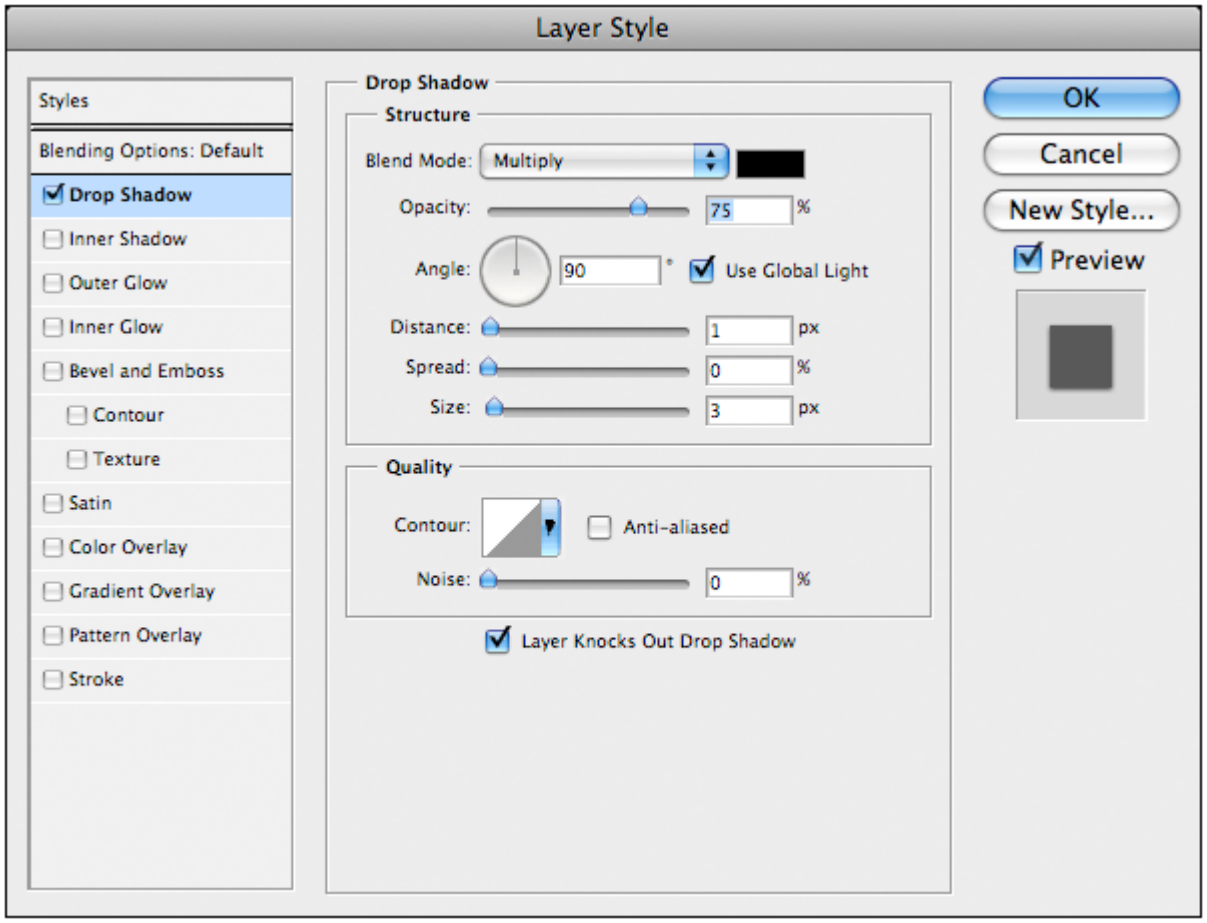
在创建您的图标之后，你可以按照以下规范添加阴影效果，作为你创造的合适的图片大小。



WVGA (高密度) 屏幕的阴影:

1. 效果: 底部阴影
2. 颜色: #000000
3. 混合模式: 多重
4. 不透明度: 75%
5. 角度: 90°
6. 距离: 2px

7. 扩展: 0%
8. 尺寸: 5px



HVGA (中密度) 屏幕阴影:

效果: 底部阴影
颜色: #000000
混合模式: 叠加
不透明度: 75%
角度: 90°
距离: 1px
扩展: 0%
尺寸: 3px

当添加了阴影，图标制作完成后，输出一个格式为 PNG 的透明文件，以确保您的图标在高密度屏显示大小为 72 x72 像素和在中密度屏显示大小为 48 x48 像素。关于为什么你应该为高，中，低密度的屏幕提供不同的启动器版本，参阅支持多种屏幕.

菜单图标

菜单图标是一个图形元素，当用户按下菜单按钮时在向用户显示菜单，在弹出菜单里显示。他们是平面展示的。菜单图标元素不能表现为 3D 或者透视的。

正如提供特定密度图标集中所描述的,你应该为低，中，和高密度的屏幕制作相应的图标集。这可以确保你的图标在一系列安装你的应用的设备中正常显示。见表 1 所建议的为每种密度所创造的图标尺寸。此外，请参阅对设计师建议 中关于如何使用多组图标。

结构

为了保持一致性，所有的菜单图标必须使用相同的原调色板和相同的效果。欲了解更多信息，参阅菜单图标颜色调色板 。
菜单图标应包括圆角，要保证逻辑正确。例如，在图 7 中，合理表现圆角的部分是房顶而不是建筑余下的部分。
所有这个页面上的特定尺寸是建立在一个 48x48 像素的画板，6 像素安全边栏的基础上的。
图标菜单效果（外发光）在灯光，效果，阴影 中被描述，它在必要时可以与 6px 安全边栏重叠。而基础形状必须始终留在安全边栏内。
最后的图形必须导出为一个透明的 PNG 文件。
在 Adobe Photoshop 中制作的菜单图标模板可以在图标模板包中得到。

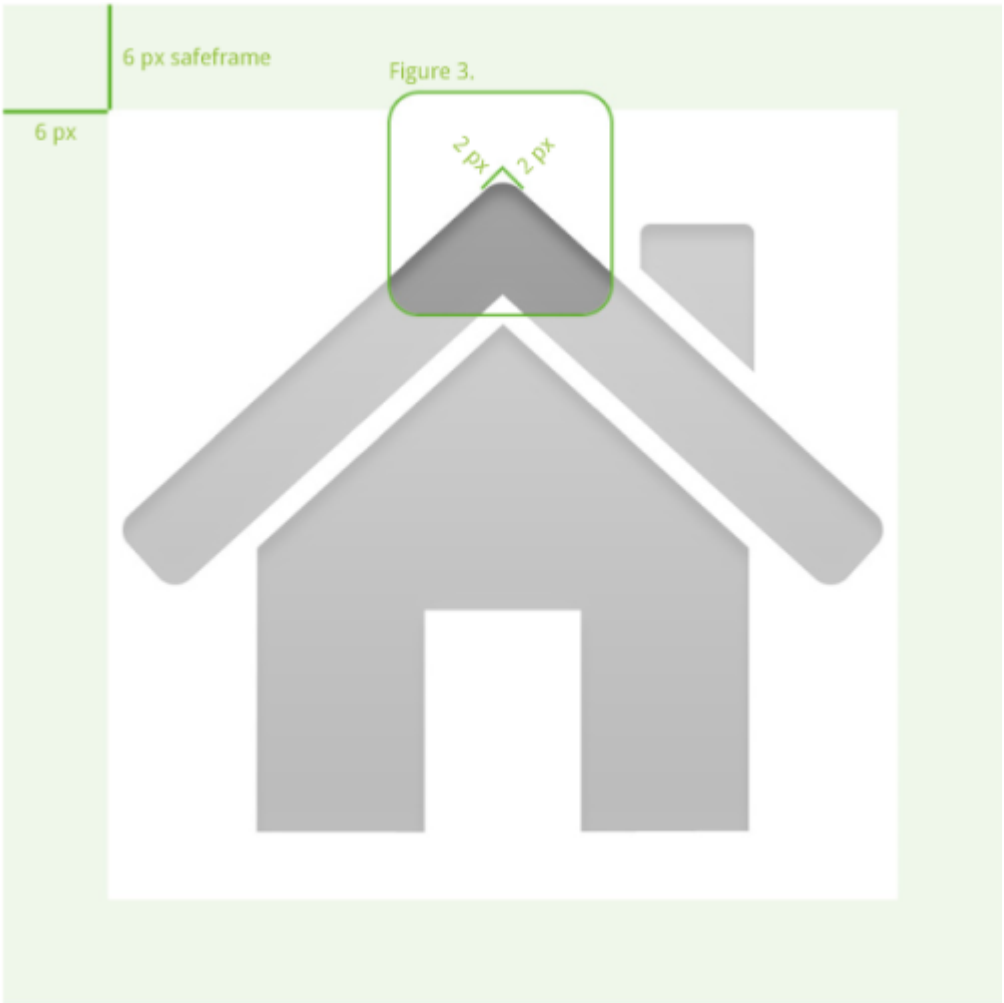


图 7.
菜单图标中的安全栏和圆角 。 图标尺寸是 48x48.

灯光，特效和阴影

菜单图标是平的。轻微凹陷和一些其他特效，如下所示，可以创造纵深感。

图 8.菜单图标的灯光，特效和阴影.

1、前部：从原色调色板中挑出颜色填充渐变
2、内投影：黑色 | 20 % 不透明度
角度 90° | 距离 2px
大小 2px
3、外发光：白色 | 55% 不透明度
扩展 10% |大小 3px
深度 1%
5、内斜角：角度 90° | 高度 10°
发光白色 70%不透明度
阴影黑色 25% 不透明度

颜色调色盘

白色
r 255 | g 255 | b 255
用于外发光和斜面的高光。

渐变填充
1: r 163 | g 163 | b 163
2: r 120 | g 120 | b 120
用作颜色填充。

黑色
r 0 | g 0 | b 0
用于内部阴影和斜面的阴影。

步骤

1. 使用工具如 Adobe Illustrator 创建基本形状。
2. 导入到一个像 Adobe Photoshop 的工具，把它放置在 48x48 像素的透明的背景的图像上。注意安全栏。
3. 增加如图 8 所描述的效果。
4. 导出一个 48x48 像素的透明 PNG 格式的图标。

"做这些和不要做这些"

在为你的应用做菜单图标时，下面是一些“做这些和不要做这些”要考虑的例子。

状态栏图标

状态栏图标用来在状态栏中展示你的应用中的通知，他们与菜单图标非常相似，但是更加小，对比度更高。

正如提供特定密度图标集中所描述的，您应该为低，中和高密度的屏幕制作独立的图标集。这可以确保你的图标在一系列安装了你的应用的设备中显示正常。见表为每个密度屏幕所建议的尺寸。此外，请参阅对设计师的建议中关于如何创建图标集的描述。

结构

圆角必须始终被应用到基础的形状中和状态栏图标细节中，如图 9 所示。
所有这个页面上的特定尺寸是建立在一个 25x25 像素的画板，2 像素安全边栏的基础上的。
状态栏图标可以在必要时与安全栏左右重叠，但绝不能与安全栏的顶部和底部重叠。
最后的图形必须导出为一个透明的 PNG 文件。
在 Adobe Photoshop 中制作的菜单图标模板可以在图标模板包中得到。

图 9.状态栏图标中的安全栏和圆角 。 图标尺寸是 25x25.

灯光，特效和阴影

状态栏图标略有凹凸感的，高对比度的，绘制的图形可以加强对小尺寸的清晰度。

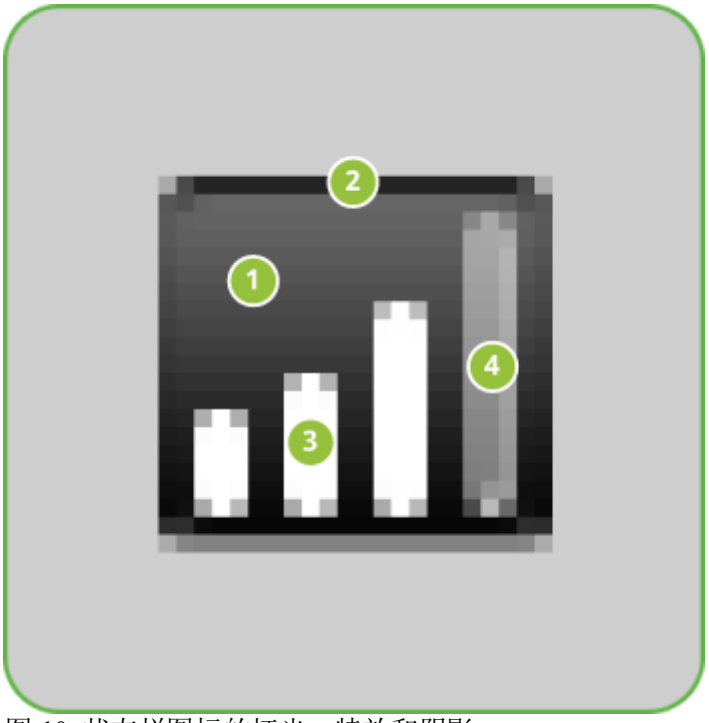


图 10. 状态栏图标的灯光，特效和阴影.

1、前部：用原色板种颜色渐变填充
2、内部斜角：深度 100% | 方向 向下
尺寸 0px |角度 90° |
角度 30°
高光 白色 75% 不透明度
阴影 黑色 75% 不透明度
3、细节：白色
4、缺失细节：色板中灰色渐变
+ 内部斜角: 平滑 | 深度 1% | 方向向下| 大小 0px | 角度 117° |
海拔 42° | 高光 白色 70% | 无阴影
在为你的应用做状态栏图标时，下面是一些“做这些和不要做这些”要考虑的例子。

标签图标

标签图标是用来表示在一个多选项界面里的单独的标签元素的图形。每个标签图标有两种状态：未选中和选中。

正如提供特定密度图标集中所描述的,你应该为低，中，和高密度的屏幕制作相应的独立图标集。这可以确保你的图标在一系列安装你的应用的设备中正常显示。见表 1 所建议的为每种密度所创造的图标尺寸。此外，请参阅对设计师的建议 中关于如何使用多组图标。

结构

未选定的标签图标和菜单图标具有相同的填充渐变和特效，但没有外部发光。
被选择的标签图标，看上去像未选择的标签图标，但有一个暗淡的内部阴影，且和对话图标有着相同的前部渐变。
标签图标有一个 1 像素的安全边栏，且安全边栏只应该和抗混淆圆形的边缘重叠。
此页面上指定的所有尺寸都基于一个大小为 32x32 像素的画板。在 Photoshop 模板内，对边缘栏保持 1 像素的填充。
最后的图像必须导出为一个 32x32 像素的透明 PNG 文件。
在 Adobe Photoshop 制作的标签图标模板可以在标签模板包中得到。


图 11. 未选中的标签图标的安全边栏和填充渐变。图标大小为 32X32。


图 12. 已选中的标签图标的安全边栏和填充渐变。图标大小为 32X32。

未选中的标签图标

灯光，特效和阴影

未选中的标签图标 看上去像以选中的标签图标, 但是有一个微弱的内部阴影, 而且和对话图标有着相同的前部渐变。

图 13. 未选中的标签图标的灯光，特效和阴影.

1、前部：渐变 | 角度 90°
底部颜色: r 223 | g 223 | b 223
顶部颜色: r 249 | g 249 | b 249
底部颜色 位置: 0%
顶部颜色位置: 75%
2、内部投影：黑色 | 10 % 不透明度 | 角度 90° 距离 2px |大小 2px
3、内斜面：深度 1% | 方向 向下 | 大小 0px | 角度 90° | 高度 10°
高光 白色 70% 不透明度
阴影 黑色 25% 不透明度

步骤

1. 用如 Adobe Illustrator 或类似的工具创建基本图形。
2. 导入到 Adobe Photoshop 或类似的工具中，把它在 32x32 像素，透明背景的图片中合适放置。
3. 如图 13，用合适的滤镜增加特效
4. 把图标导出为一个 32x32 像素的 PNG 透明文件。

被选择的标签图标

已选择的标签图标和菜单图标具有相同的填充渐变和特效，但没有外部发光。

图 14. 被选择的标签图标的灯光，特效和阴影.

1、前部： 全渐变.
2、内部投影：黑色 | 20% 不透明度 |
角度 90° | 距离 2px |
大小 2px
3、内斜角：深度 1% | 角度 | 大小 0px | 角度 90° |
高度 10°
高光 白色 70% 不透明度
阴影 黑色 25% 不透明度

对话图标

对话图标有一个 1 像素的安全边栏。基本图形必须在安全边栏内部合适放置, 但是抗混淆的圆形可以和安全边栏重叠。
此页面上指定的所有尺寸都基于一个 Adobe Photoshop 中建立的，大小 32x32 像素的画板上的。在 Phontoshop 模板中，对边栏保持保持 1 像素的填充。
最后的图片必须导出为一个透明的 PNG 文件。
在 Adobe Photoshop 制作的对话图标模板可以在标签模板包中得到。

图 15. 对话图标安全边栏和填充渐变.图标大小 is 32x32 像素.

灯光，特效和阴影

对话图标 是平的且面向前的图片.为了在黑色背景中突出, 他们用了光线渐变和黑色投影。

图 16. 对话图标的灯光，特效和阴影.

1、前部：渐变叠加 | 角度 90°
底部色彩: r 223 | g 223 | b 223
顶部色彩: r 249 | g 249 | b 249
底部色彩位置: 0%

顶部色彩位置: 75%
2、内部投影：黑色 | 25% 不透明度 |
角度 -90° | 距离 1px | 大小 0px

列表视图图标

列表视图图标看起来像 对话图标， 但是他们用了一个光线来源于物体上方的内部投影特效。他们也只用在列表视图中。在包括 Android 市场应用主屏和在地图应用中的导向屏中使用。

正如提供特定密度图标集中所描述的,你应该为低，中，和高密度的屏幕制作相应的独立图标集。这可以确保你的图标在一系列安装你的应用的设备中正常显示。见表 1 所建议的为每种密度所创造的图标尺寸。此外，请参阅对设计师的建议 中关于如何使用多组图标。

结构

列表视图图标通常有一个 1 像素的安全边栏。抗混淆的圆形可以和安全边栏重叠。
此页面上指定的所有尺寸都基于一个 Adobe Photoshop 中建立的，大小 32x32 像素的画板上的。在 Phontoshop 模板中，对边栏保持保持 1 像素的填充。
最后的图片必须导出为一个透明的 PNG 文件。
在 Adobe Photoshop 制作的列表视图图标模板可以在标签模板包中得到。

图 17. 列表视图图标的安全边栏和填充渐变. 图标大小 is 32x32。

灯光，特效和阴影

列表视图图标是平的且面向前的图片.为了在黑色背景中突出, 他们用了光线渐变和黑色投影。

图 18. 列表视图图标的灯光，特效和阴影.
1、内部投影： 黑色 | 57 % 不透明度 | 角度 120° | 混合模式 正常 | 距离 1px | 大小 1px
2、背景：黑色 | 标准系统颜色
这些图标只在列表视图中使用。

注意: 列表视图图标在 Photoshop 中 32x32 像素的画板中放置，没有安全边栏。

用常用的命名习惯为图标版本命名

尝试命名文件，当他们按照字母排序的时候，有关的版本将会在一个目录内聚集在一起。特别是它有助于为每个图标类型的使用共同的前缀。 例如:

版本类型	前缀	例子
图标	ic_	ic_star.png
发射器图标	ic_launcher	ic_launcher_calendar.png
菜单图标	ic_menu	ic_menu_archive.png
状态栏	ic_stat_sys or ic_stat_notify	ic_stat_notify_msg.png
标签图标	ic_tab	ic_tab_recent.png
对话图标	ic_dialog	ic_dialog_info.png

请注意，你不需要使用任何类型的共享前缀— 这样做只是为了您的方便。

为制造多密度版本图标，建立一个空间储存文件

为不同屏幕密度发展多个版本集意味着制作文件的多个尺寸副本。为了保持文件多个副本的安全和更容易被发现，我们建议您在您的工作空间创建一个目录结构，来组织文件版本。例如:

```
assets/...
    ldpi/...
      _pre_production/...
        working_file.psd
      finished_asset.png
```

```
    mdpi/...
        _pre_production/...
            working_file.psd
        finished_asset.png
    hdpi/...
        _pre_production/...
            working_file.psd
        finished_asset.png
```

这种结构与特定密度结构相同，你可以在你的应用源文件中存储最终的版本。因为你的工作空间的结构与应用的结构类似，您可以迅速确定哪些版本应该复制到每个应用源文件目录下。为不同密度制作的独立版本可以使你根据密度检测不同的文件名，这非常重要，因为为不同密度制作的对应的版本必须有同样的文件名。

为了做比较，下面是一个典型的应用资源的目录结构：

```
res/...
    drawable-ldpi/...
        finished_asset.png
    drawable-mdpi/...
        finished_asset.png
    drawable-hdpi/...
        finished_asset.png
```

首先制作中密度版本

r.由于中密度是 Android 的基线，因此你的设计工作应该从制作中密度版本开始。见上面的表 1，为不同图标类型的实际像素尺寸。如果可能，使用矢量图形或在 Photoshop 建立路径，使其更容易修改版本尺寸。

Activity 和 Task 的设计思路和方法

Activity 和 Task 是 Android Application Framework 架构中最基础的应用，开发者必须清楚它们的用法和一些开发技巧。本文用大量的篇幅并通过引用实例的方式一步步深入全面讲解它们的基础原理(underlying principles)和架构(mechanisms)，例如：Navigation、Multitasking、activity re-use、intents 和 activity stack 等…大部分与其相关的应用模块。重点讲解开发过程中如何更准确的体现用户交互性的便捷和高效，同时也帮助分析 Designers 和 Developers 在开发期间所要面对的问题。
文中涉及到的实例有一部分是属于平台自带的 application（例如：拨号程序等），另外也有 Google 产品线中的一些有代表性的应用（例如：Google Map 等）。建议大家亲自利用 Emulator 或者 Android-powered device 测试实例中的效果，这样可以帮助更加清晰的理解一些模块的含义。（注意：可能会因为硬件对于某些功能无法提供支持，所以有一些实例可能无法在你的测试机中正常浏览）

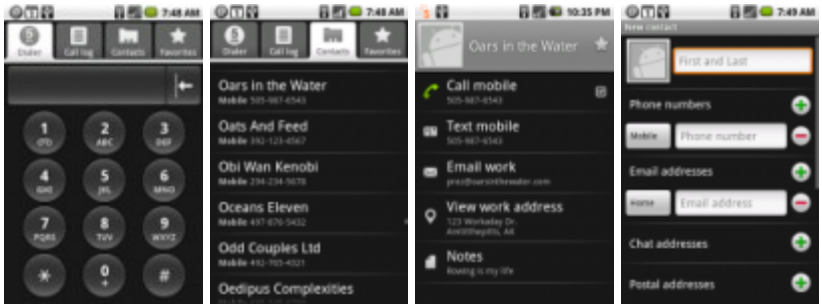
首先需要清楚一些基础模块:

- Applications
- Acitivities
- Activity Stack
- Tasks

以上这四个模块对于理解这篇文章非常重要，下边就来逐一的简单介绍其具体的含义和用法（也可以通过其链接直接查看官方文档）。

Applications
任何一个 Android Application 基本上是由一些 Activities 组成，当用户与应用程序交互时其所包含的部分 Activities 具有紧密的逻辑关系，或者各自独立处理不同的响应。这些 Activities 捆绑在一起成为了一个处理特定需求的 Application, 并且以“.apk”作为后缀名存在于文件系统中。Android 平台默认下的应用程序 例如：Email、Calendar、Browser、Maps、Text Message、Contacts、Camera 和 Dialer 等都是一个个独立的 Apps。

Activities
上边已经提到 Activities 是构成 Applications 的主要组成部分，其实可以更为具体的理解为 Application 仅仅是一个抽象的标签，它将系统内一部分 Activities 关联在一起，协同完成用户的特定需求。安装 Application 的过程也可以简单理解为将其所包裹的 Activities 导入到当前的系统中，如果系统中已经存在了相同的 Activities，那么将会自动将其关联，而不会重复安装相同的 Activities，避免资源的浪费。Application 卸载的过程也会检查当前所关联的 Activities 是否有被其它 Application 标签所关联，如果仅仅是提供当前的 Application 使用，那么将会彻底被移除，相反则不做任何操作。
用户与 Application 的交互行为大部分都是通过 GUI 来完成，在 Android 平台可以有两种方式定义 GUI，其中可以利用 XML 来预置静态的 GUI 元素，或者在 Activity 类的内部动态定义 GUI 元素。这两种不同的方法都是由 Activity 作为驱动和响应用户交互事件的主体。当启动 Application 之后，至少需要一个包含有 GUI 信息的 Activity 实例被创建。
Activity 的主体包括两个主要部分，其中一个是 Content(data)，另外一个是响应用户交互事件的行为。列举一个 Dialer 例子的截图，其中包括四个部分：Dialer 主界面、通讯录、查看联系人信息和添加新联系人。



下面列举了更多比较有代表性的 Applications 和其所包含的 Activities：

- Email – activities to view folders, view list of messages, view a message, compose a message, and set up an account

- Calendar – activities to view day, view week, view month, view agenda, edit an event, edit preferences, and view an alert
- Camera – activities for running the camera, viewing the list of pictures, viewing a picture, cropping a picture, running the camcorder, viewing the list of movies, and viewing a movie
- Game – one activity to play the game, typically another for setup
- Maps – one activity to view a location on a map, a second for lists (such as turn list or friend list), and a third for details (friend location, status, photo)

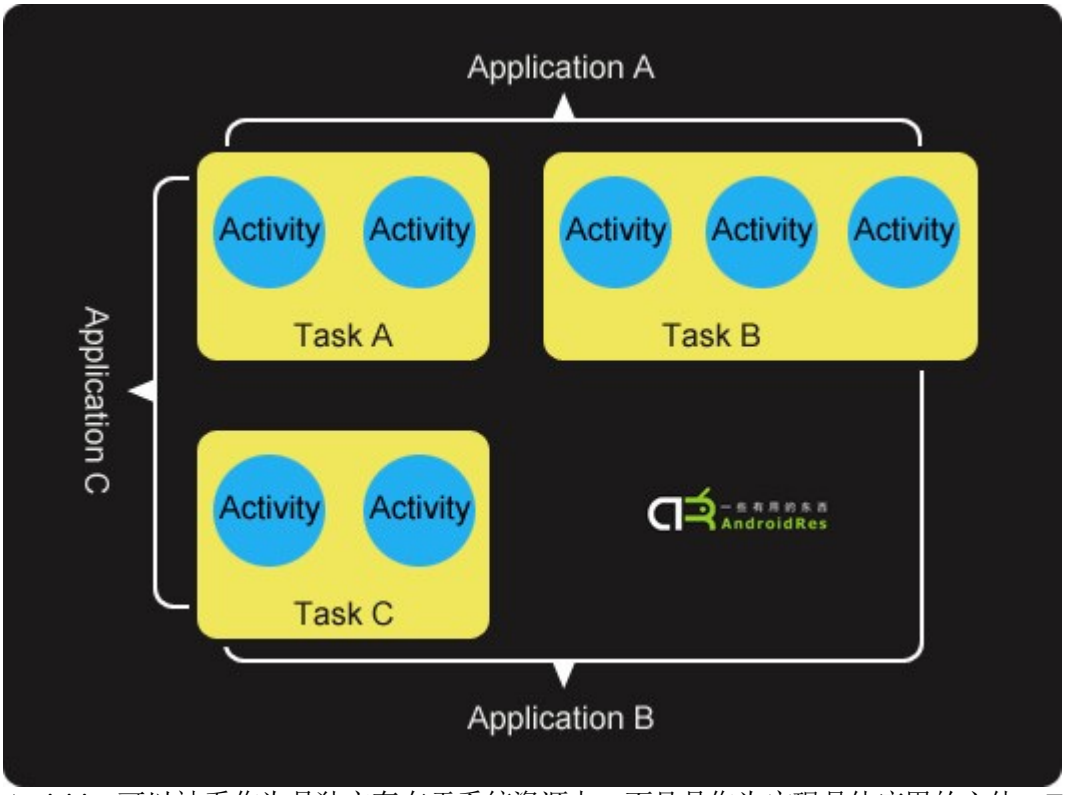
Application 基本上是由四个模块组成：Activity、Service、Content Provider 和 Broadcast Receiver，其中 Activity 是实现应用的主体。

Activity Stack

操作应用程序时，有时需要调用多个 Activities 来完成需求，例如：发送邮件程序，首先是进入邮件主界面，然后启动一个新的 Activity 用于填写新邮件内容，同时可以调出联系人列表用于插入收件人信息等等。在这个操作过程中 Android 平台有一个专门用于管理 Activities 堆栈的机制，其可以方便的线性记录 Activities 实例，当完成某个操作时，可以通过这个导航功能返回之前的 Activity（通过按操作台的“Back”）。
每次启动新的 Activity 都将被添加到 Activity Stack。用户可以方便的返回上一个 Activity 直到 Home Screen，到达 Home Screen 后，将无法再继续查看堆栈记录（俗话说：到头了- Androidres.com）。如果当前 Task 被中止(Interrupting the task)，返回到系统主界面后启动了其它操作，当希望返回到前一个 Task 继续执行时，只需要再次通过主界面的 Application launcher 或者快捷方式启动这个 Task 的 Root Activity 便可返回其中止时的状态继续执行。
相对于 Views、Windows、Menus 和 Dialogs 而言，Activity 是唯一可被记录在 History stack 中的数据，所以当你所设计的应用程序需要用户由 A 界面进入到次一级界面 B，当完成操作后需要再次返回 A,那么必须考虑将 A 看作为 Activity，否则将无法从历史堆栈中返回。

Tasks

在 Android 平台上可以将 Task 简单的理解为由多个 Activities 共同协作完成某一项应用，而不管 Activities 具体属于哪个 Application。通过下边的图示可以更清晰的理解 Applications、Tasks、Activities 三者之间的关系



Activities 可以被看作为是独立存在于系统资源中，而且是作为实现具体应用的主体，Task 将一些 Activity 关联起来实现一个更复杂的应用,单独或者多个 Tasks 可以被定义为一个 Application。
通常实现一个 Task 都会存在一个 Root Activity，但并不是所有情况都如此，通过 Application launcher、Home screen 的快捷方式或者 由 “Recent Tasks”（长时间按住 Home 键） 最近使用过的 Task 记录中启动。当从一个 Activity 中启动另外一个 Activity 时，Back 键将作用于返回前一个 Activity,与此同时新开启的 Activity 将被添加到 Activity Stack 中。

这里有两个被表示为 Task 的例子：

- 发送带有附件的邮件
- 查看 YouTube 视频，并且通过 Email 的方式共享给其他联系人。

- Interrupting the Task
这是 Task 一个非常重要的特性，用户可以实时中止当前为完成的 Task，新开启一个不同的 Task，当新 Task 完成操作后，依然可以返回当上一次中止的 Task 继续完成余下操作。这个特性大大方便了同时运行多个 Tasks，并且可以方便的在他们之间切换。这里有两种方式可以从当前 Task 跳转为其它 Task（应用这两种方式切换 Task，都允许返回到 Task 最初中止前的状态）。

- 系统抛出一个 Notification，当前 Task 会被终止，跳转为 Notification 的 Task。
- 用户强制中止

当然，除了这两种方式以外，还有另外一个特殊情况，算作为第三种方式来启动一个新的 Task：Activity 本身被定义为一个 Task。例如： Maps 和 Browser 就是属于第三种情况的 Application，通过邮件中的一个地址来启动 Maps Activity 作为一个新的 Task，或者通过邮件中的链接启动 Browser 来启动一个新的 Task。当处在这种情况下，Back 按键被触发后，将返回到上一个 Task(邮件），因为这些新的 Tasks 并不是通过 Home Screen 中的 Application launcher 或者快捷方式来启动。

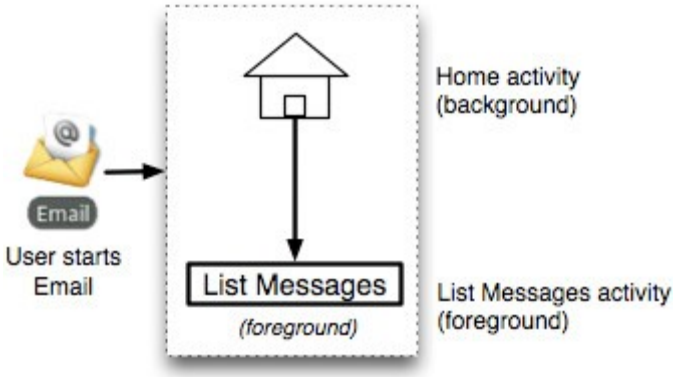
了解 Activities 和 Tasks 的基本原理

请大家一定首先理解之前所提及的内容，如果对某些概念依然含混不清，请及时查阅更多资料（官方文档是最好的学习资料），否则无法快速理解接下来将要讲述的例子，甚至丧失阅读兴趣。

接下来，将通过一些有代表性的实例了解关于 Applications、Activities、Activities stack、Tasks 和 Intent 等一些模块的最基本原理。从各个角度分析系统对于用户在不同模式下操作的反应原理。

从 Home 启动一个 Activity

绝大部分的 Application 都由此启动（也有一些 Application 是通过其它 Application 启动）。具体的方式有两种，其一是从系统的 Application Launcher 启动，另一种是直接由 Home Screen 的快捷方式。启动 Application 后，Root Activity 会显示在当前窗口，并可直接供用户操作界面元素。官方给出了一个有关这个过程的图示，其实我感觉这个描述的还不够直观，凑合着用吧。大体的过程是由 Home 下启动 Email Application，在这个应用程序中可以直接提供给用户操作的是 List Messages Activity，Home Activity 切换为后台运行。

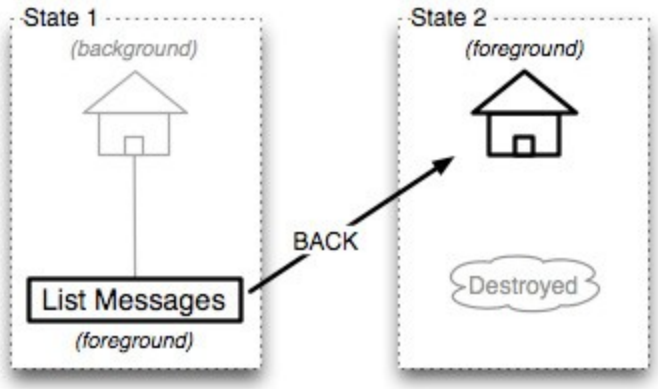


应用 Back 或 Home 键离开当前 Activity 的区别

应用 Back 或者 Home 都可以离开当前 Activity（基于 Application 的 Root Activity），Home activity 重新切换到 foreground，然而二者最根本的区别在于用户是否还需要保留当前 Activity 的 state。

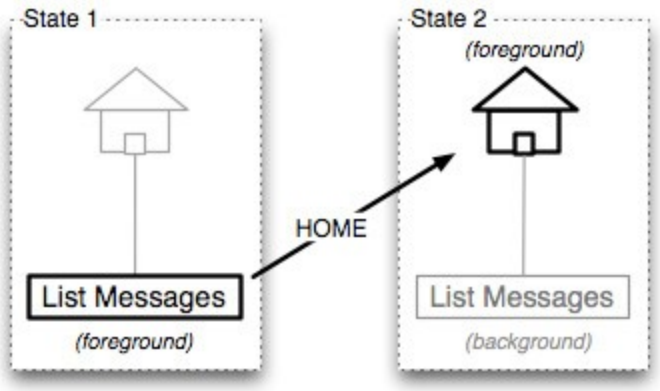
- Back:

将会终止（Destroy）当前正在运行的 Activity，返回到之前的 Activity（如果是 Root Activity，那么将会直接返回到 Home Activity）。官方给出了一个相关过程的图示，当用户正在操作 List Messages Activity 时，下拉邮件列表（改变了 Scrolling 状态），通过 Back 键返回到 Home Activity 之后，当再次通过 Email Icon 启动 List Messages Activity 时，将会看到列表处在初始位置。通过这个演示可以了解到通过 Back 键离开当前 Activity 时，无法暂时保留住其 State 数据，当再次启动时相当于重新创建了一个实例。



-Home:

利用 Home 取代 Back 返回的方式，当前 Activity 将被切换到 Background，而不是被 Destroied。这样的好吃是可以暂时保留这个 Activity 的 State 信息，当再次通过 Application launcher 或者快捷方式启动时，可以返回到最后离开的状态。对比在 Back 中引用的例子，当再次由 Home 返回到 Activity 时，将会看到最后一次操作所记录的 Scroll 状态，而不是默认的初始位置。



Exception(例外情况）

前边列举了两种典型的情况，同时还存在一些例外的情况，某些 Activity 从 Background 被“召唤”到 foreground 之后依然是相当于重新创建了新实例，其有区别于前边所论述的结果。即便是暂时保存在 Background 模式下（没有被 Destroied），其 State 数据也将丢失。例如：Contacts 和 Gallery 等。当用户启动了 Contact 应用程序，并点选某个条目查看详细信息，如果通过 Home 键返回后，再次重复启动 Contact 应用程序时，看到的并不是之前所打开的特定条目的详细信息，而是初始的默认界面。这个例子说明不是所有情况下通过 Home 键返回后都可以保存当前 Activity 的 State 信息。

另外一种是与 Back 键有关的特殊情况。前边提及到大部分的 Activity 通过 Back 键返回到 Home Activity 时，其自身将被彻底销毁，默认情况下 Activity 响应 Back 按键的方法被定义了 Destroy 行为。但对于某些特别情况，开发者可以根据需求将相应 Back 按键事件的行为重新“override”，撤消默认的 Destroy 行为。音乐播放器是与其相关的一个典型应用，当用户在播放器的 Root Activity 中触发 Back 按键后，转为 Background 模式下继续播放当前的音乐，同时 Home Activity 转为 Foreground。

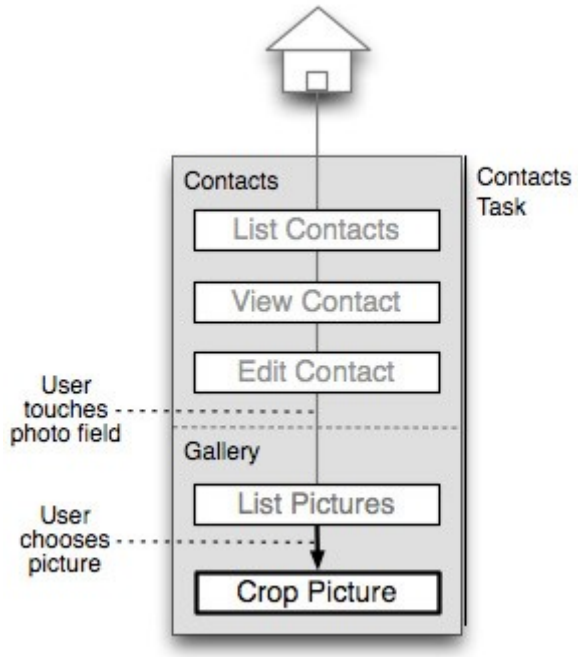
Activity 的复用

在多个不同的 Applications 中，当遇到有相同目的应用时，会涉及到 Activity 的复用性问题，这在开发过程中是一个非常普遍的情况。复用性一直被众多开发机构强调为节约成本，优化资源的最有效的机制。对于移动应用平台更加看重资源的最优化利用，复用性的应用在 Android 平台上无处不在，通过两个比较基础的例子来具体的说明。

- Contacts 利用 Gallery 获得图像资源

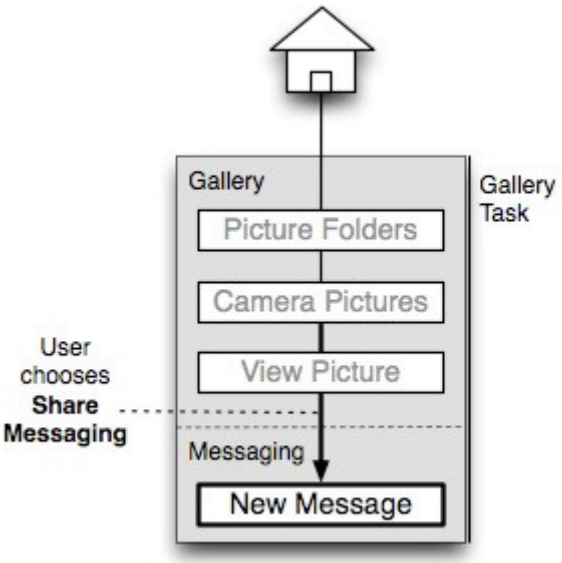
众所周知 Contacts 是手机中最常用的应用程序，主要用于存储当前用户的联系人信息，其中需要包含联系人的头像信息。在 Android 平台中的图像信息是由 Gallery 管理，所以 Contacts 必然需要复用 Gallery Activity 来获取相应的图像信息。

针对于 Android 或者其它平台开发应用程序都需要有良好的复用性意识，这个需要贯穿于项目的整个开发过程。包括如何利用当前系统的现有资源，或者考虑到将来可能会被其它应用程序用于完成特定的需求。当用户正在调用的 Intent filter 不唯一时，系统将弹出一个供用户选择的对话框，这的确是一个完美的解决方法。



- 利用 Messaging 扩展 Gallery 共享功能

用户通过 Gallery 查看当前系统中的图像资源，每次单独打开一幅图像资源都可以通过 Menu -> Share 将当前的资源以附件形式插入新创建的 Messaging 中，并且以正常发送信息的方式将其共享给收件人。如果取消当前的共享行为，只需要通过 Back 按键返回到 Gallery Activity。相比较前一个例子的区别在于，Message Activity 完成发送或者被取消操作，其不会返回任何信息。



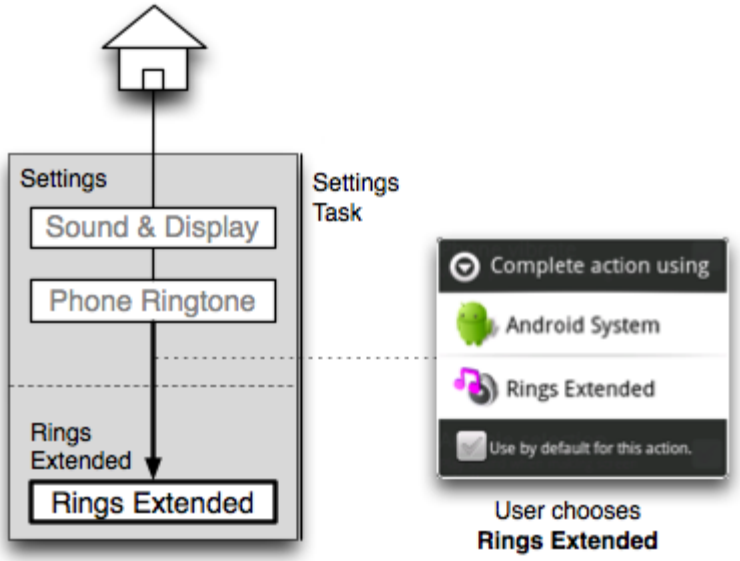
以上两个例子分别讲解了利用一系列的 Activities 来完成某一项需求，并且它们都调用了外部的 Application 资源。

Replacing an Activity

目前要介绍的内容是关于在不同的 Applications 中，有相同 Intent filter 属性的 Activities 可相互间替换，这对于习惯 Windows 等操作系统的用户比较不容易理解。其实如果您足够细心，就可以发现之前的例子中有关于这里所提及情况。

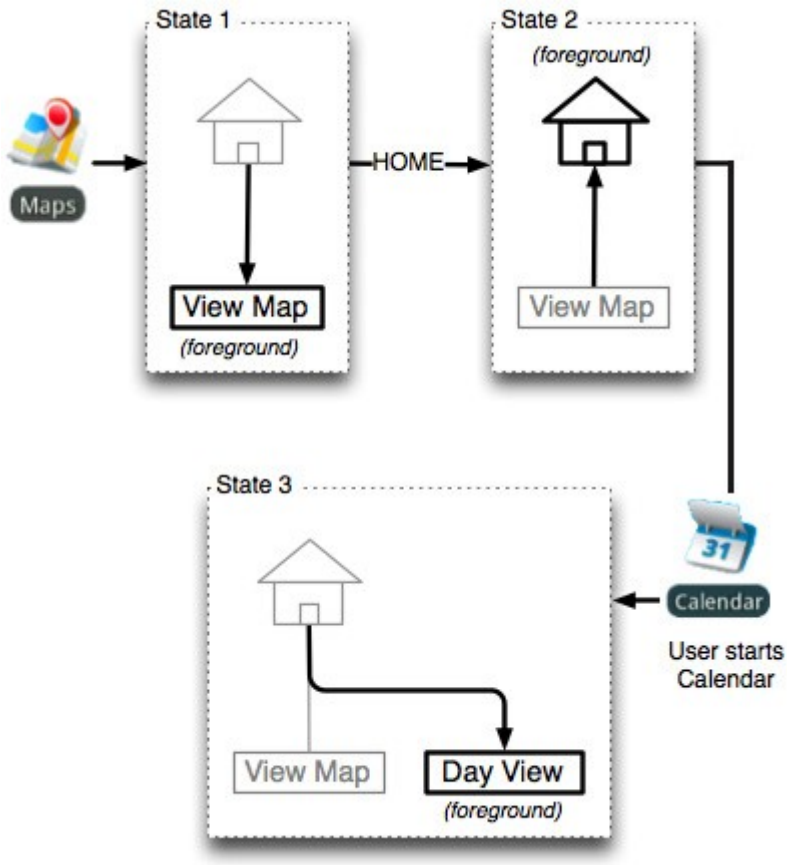
通常遇到这种情况发生时，一般都是因为外部具有相同功能的 Activity A 在处理问题的能力方面要优于当前 Application 中默认的操作行为 Activity B，系统会抛出一个可供选择的对话框，用户根据主观判断来选择最优

的方式处理当前任务。通过一个比较容易理解的实例来说明整个过程，建议“动手能力强”的同学可以通过模拟器亲自尝试。
例如：用户在当前系统下加载了最新的 Phone Ringtone Activity，取名为 Rings Extended。如果用户通过 Setting -> Sounds&Display -> Phone Ringtone 来设置当前的铃音属性时，将会弹出一个包含有系统默认的 Phone Ringtone Activity 和最新加载的 Rings Extended 两种可供选择的操作应用，同时在对话框中还提供了一种可以直接启动系统默认的操作方式选项。如果用户选择了 Rings Extended，那么其将会被载入当前的线程中替代原有的默认操作行为，可以根据下面的图示来增强理解。

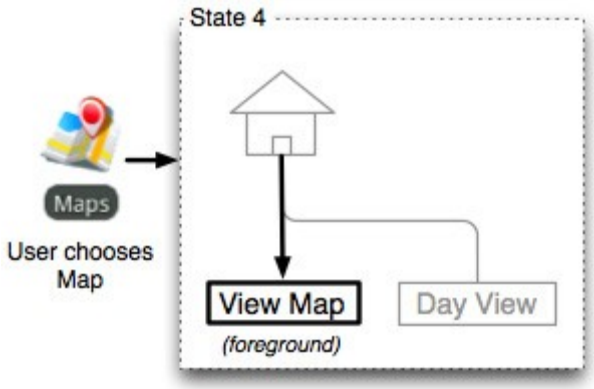


多任务同时运行（Multitasking）
在之前的板块有专门提到关于 Home 和 Back 两种切换到 Home Screen 的方法和它们之间的差异性，这个章节将会重点涉及到系统可以同时处理多个实时运行的任务。如果用户正处于某个 Application A 开启状态时，通过 Home 按键切换回 Home Activity 的同时保留了此前 Application A 运行的状态信息，可以开启新程序的同时，也可以再次将 Application A 切换回 Foreground。
接下来通过一个有关 Map 应用的实例更加具体的了解其所涵盖的过程。
首先的起始阶段分为三个步骤，

第一步，由 Application Launcher 启动 Map 应用程序，并且搜索一个具体的地理位置。假设当前的网络环境非常不理想，需要花费一定的时间 Download 地图数据。
第二步，当系统需要花费较长时间加载当前地图信息数据时，保持当前 Activity 的状态，返回 Home Activity 启动其它的 Applicaton，地图 Activity 切换到 Background，而并不会中断加载进度（依然保持网络连接）。
注意：以上是 Activity 在默认条件下的反应行为，其切换为 Background 状态后直接触发 onStop()事件，开发者可以重新定义其方法。例如：强制 Activity 在转为 Background 状态下，终止网络连接。
第三步，当前 Map activity 已经切换到 Background 状态下运行，Home Activity 切换到 Foreground。这时用户启动 Calender activity，其将自动转为 Foreground 状态，同时获得操作焦点。
将以上三个步骤用图示的方式表述：



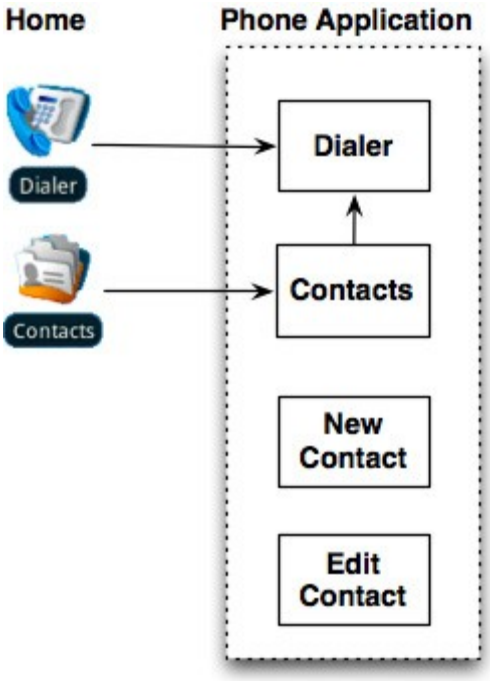
最后，退出当前 Calender activity 返回到 Home，再次通过 Maps 图标将其处在 Background 状态的实例切换到 Foreground。



通过上边的例子看出用户通过 Application Launcher 同时运行多个 Tasks，代表系统具备多任务处理机制 – Running multiple tasks。

启动 Application 的两种不同方式

每个 App 都需要提供至少一个 Entry point（翻译成“入口点”有点别扭，干脆保留原样）供用户或者系统调用其所关联的 Activities，Application launcher 中的小图标就是每个单独 App 的 Entry Point。另外 App 也可以相互间通过 Activity 作为 Entry Point 来启动，可以将 App 所包含的每个 Activity 看作为潜在的 Entry point。系统中的 Phone Application 同样具有两个 Entry Points：Contacts 和 Dialer。下边的图示中可以了解到用户通过 Application launcher 启动 Contacts Activity，选择其中某一个联系人之后，调用 Dialer Activity 拨打其所提供的电话号码。

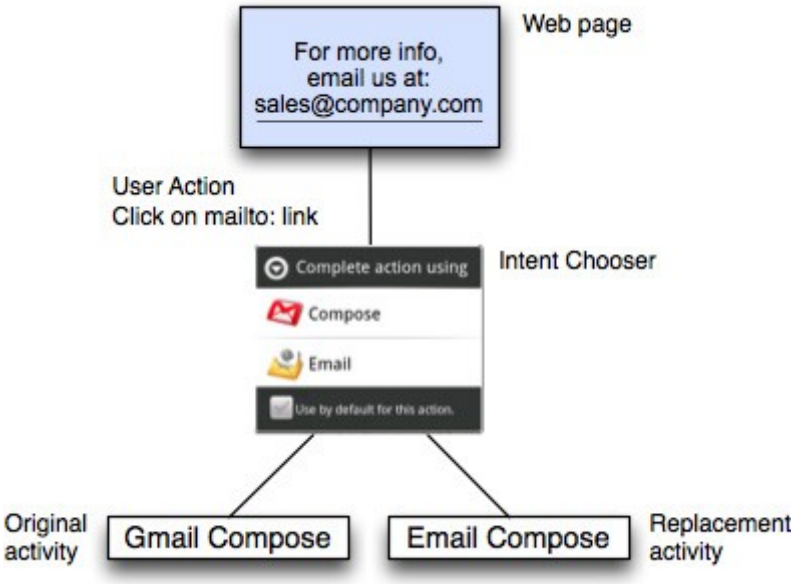


Intents

在现实世界中大家每时每刻都会与周围的环境发生互动，这个互动的过程首先要确定一种意识，例如：感觉到口渴，需要水分补充。这种意识会引导自己以习惯的方式解决口渴问题，采用的方式可以多种多样，吃冰淇淋、喝水、嚼树叶等。类似于口渴的意识形态被抽象为 Intent，并将其看作是一种对象，这就是 Android 响应“意识”的方式。

在 Android 平台上，用户的操作行为是由各种不同的事件组成，系统会将每个事件都抽象为 Intent 对象，寻找解决这项需求的具体方法。抽象的 Intent 对象有两种形式，第一种是“明确”的 Intent（Explicit Intent），在初始化的时候已经为这个 Intent 关联了特定的 Activity。第二种是“不明确”的 Intent（Implicit Intent），代表这个 Intent 没有明确关联 Activity，当它被抛出后，系统在众多 Activities 中根据 Intent filter 来寻找与其匹配的处理方法。如果存在多个结果，用户可以根据需要选择合适的处理方法。

引用一个具体的例子，单击一个 mailto:info@androidres.com 链接后，这个被抛出的 Intent 属于 Implicit Intent，系统抓取了解决这个 Intent 的结果，将所有的结果供用户选择（Gmail 或者 Email）：



下边给出更多系统默认的 Intent 关联列表：

- View the list of contacts – resolves to a contact list viewer activity
- View a particular contact – resolves to a contact viewer activity
- Edit a particular contact – resolves to a contact editor activity
- Send to a particular email – resolves to an email activity
- Dial a phone number – resolves to a phone dialer activity
- View the list of images – resolves to an image list viewer activity
- View a particular image – resolves to an image viewer activity
- Crop a particular image – resolves to an image cropper activity

Intent 对象包含两个元素：

1）Action ：例如 查看、编辑、拨打电话、查看图像资源等等。

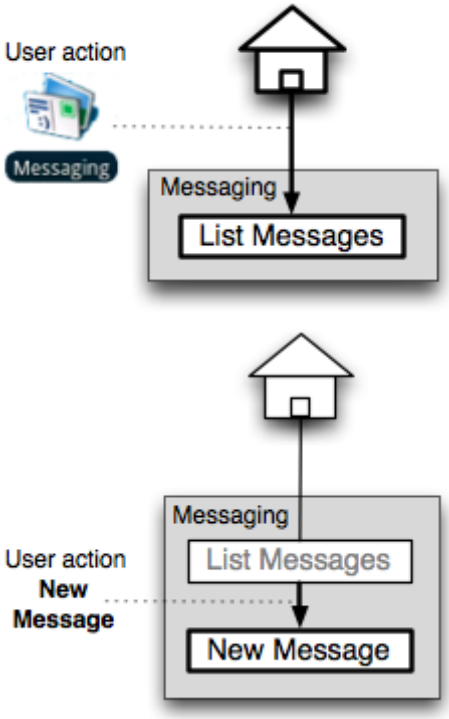
2）Data：提供给某种行为的具体数据。加工果汁饮料，需要提供水果（黑心店除外）。

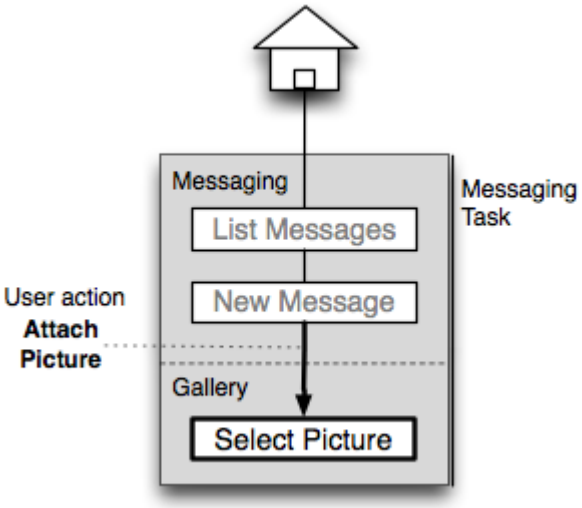
参照官网的解释：Intent Class 和 Intent Filters。

Tasks 相互间切换

依然是应用实例来说明这个切换的过程。在这个例子中，用户编辑一个短消息，并且插入图像附件，但是在发送之前启动 Calendar，随后切换回短消息编辑界面，最后发送信息。

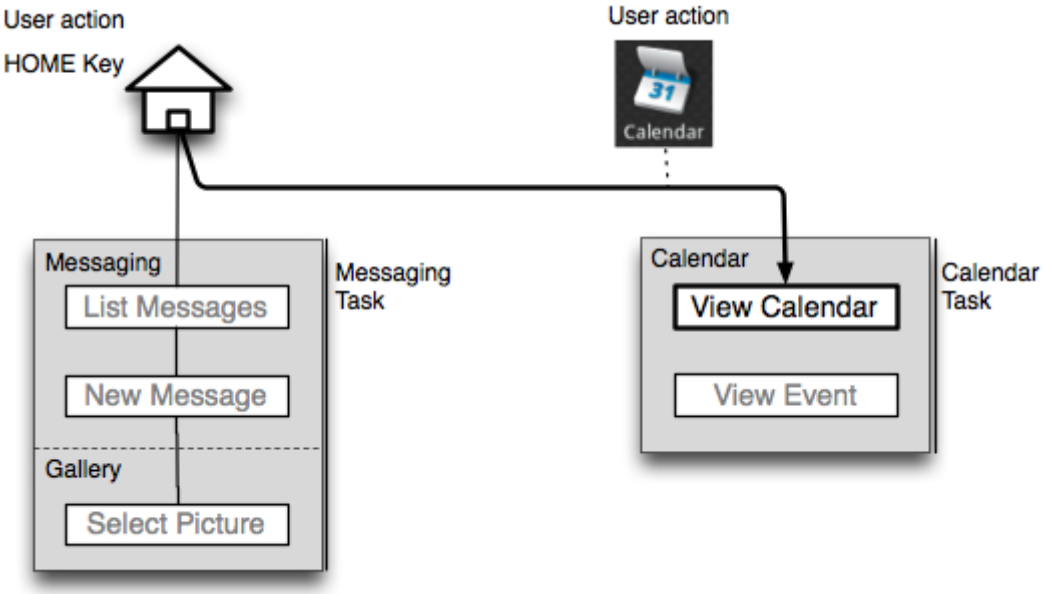
1）启动第一个 Task：Messaging App，Home > Messaging > New Message > Menu > Attach > Picture。插入图片的步骤需要调用 Gallery Activity，它是一个独立的外部程序。



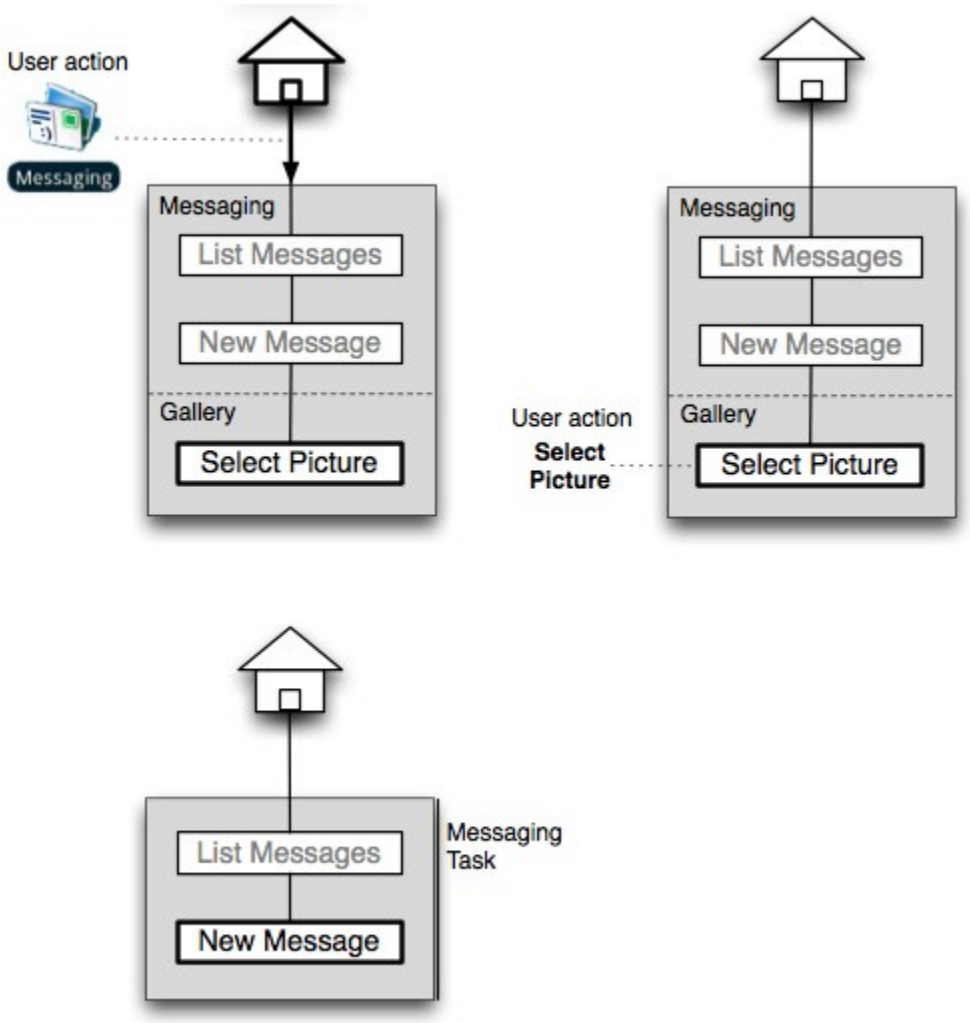


接下来启动另外一个 Task，由于没有直接从当前的 Activity 运行 Calendar，所以需要切换到 Home。

2）启动另外一个 Application（Calendar）：Home > Calendar



3）查看 Calendar 完成后，将 Messaging 由 Background 切换到 Foreground 模式，其中还包括了添加附件，并最终发送消息。



至此，对于 Android 平台中两个比较核心元素： Activities 和 Tasks 的介绍基本告一段落，以后也许会有更多关于这方面的讨论，希望得到您的关注。另外，有些朋友或许已经看过官方的原文，而本站也再次有幸得到了您的通读，如果在某些概念或者论述内容上存在遗漏或者误解，那么真诚的希望能够获得指正和帮助。

Android 最佳实践之流畅设计

即使你的应用程序是快速且响应灵敏的，但一些设计仍然会给用户造成问题——与其它应用程序或对话框未事先计划的交互，意外的数据丢失，意料之外的阻塞等等。避免这些问题，有助于理解应用程序运行的上下文和系统的交互过程，而这些又正影响着你的应用程序。简而言之，你应该竭尽全力去开发一个与系统和其它应用程序流畅交互的应用程序。

一个常见的流畅问题是，一个应用程序的后台处理——例如，一个 Service 或者 BroadcastReceiver——弹出一个对话框来响应一些事件。这可能看起来没啥大碍，尤其是你在模拟器上单独地构建和测试你的应用程序的时候。然而，当你的应用程序运行在真机上时，有可能你的应用程序在没有获得用户焦点时后台处理显示了一个对话框。因此，可能会出现在活跃的应用程序后方显示了你的应用程序的对话框，或者从当前应用程序夺取焦点显示了一个对话框，而不管当前用户正在做什么（例如，正在打电话）。那种行为，对应用程序或用户来说，就不应该出现。

为了避免这些问题，你的应用程序应该使用合适的系统资源来通知用户——Notification 类。使用 Notification，你的应用程序可以在状态栏显示一个 icon 来通知用户已经发生的事情，而不是夺取焦点和打断用户。

另一个流畅问题的例子是未能正确实现 Activity 的 onPause()和其它生命周期方法而造成意外丢失了状态或用户数据。又或者，如果你的应用程序想暴露数据给其它应用程序使用，你应该通过 ContentProvider 来暴露，而不是（举例）通过一个可读的原始文件或数据库来实现。

这些例子的共同点是它们都应该与系统和其它应用程序协作好。Android 系统设计时，就把应用程序看作是一堆松散耦合的组件，而不是一堆黑盒代码。作为开发者来说，允许我们把整个系统看作是更大的组件集合。这有益于我们可以与其它应用程序进行清晰无缝的集成，因此，作为回报，我们应该更好的设计我们的代码。

这篇文章将讨论常见的流畅问题以及如何避免它们。它将囊括这些主题：

1) 别丢弃数据

2) 不要暴露原始数据

3) 不要打断用户

4) 有太多事情要做？在线程里做

5) 不要让一个 Activity 超负荷

6) 扩展系统主题

7) 设计你的 UI 可以应付多屏幕分辨率

8) 假设网络很慢

9) 不要假定触摸屏或键盘

10) 节省设备电池

1）别丢弃数据

一定要记住 Android 是一个移动平台。可以显而易见地说，其它 Activity（例如，“Incoming Phone Call”应用程序）可能会在任何时候弹出来遮盖你的 Activity，记住这个事实很重要。因为这个过程将触发 onSaveInstanceState()和 onPause()方法，并可能导致你的应用程序被杀死。

如果用户在你的应用程序中正在编辑数据时，其它 Activity 出现了，这时，你的应用程序被杀死时可能丢失那些数据。当然了，除非你事先保存了正在进行的工作。“Android 方式”是这样做的：能接收和编辑用户输入的 Android 应用程序应该重写 onSaveInstanceState()方法，并以恰当的方式保存它们的状态。当用户重新访问应用程序时，她能得到她的数据。

进行这种处理方式最经典的例子是 mail 应用程序。如果用户正在输入 email，这时其它 Activity 启动了，mail 应用程序应该把正在编辑的 email 以草稿的方式保存起来。

2）不要暴露原始数据

如果你不想穿着内衣在大街上溜达的话，你的数据也不应该这样。尽管可能存在暴露应用程序的某种形式给其它应用程序，但这通常不是最好的主意。暴露原始数据，要求其它应用程序能够理解你的数据的格式；如果你变更了格式，那么，你将破坏那些没有进行同步更新的应用程序。

“Android 方式”是创建一个 ContentProvider，以一种清晰的、深思熟虑的和可维护的 API 方式暴露你的数据给其它应用程序。使用 ContentProvider，就好像是插入 Java 接口来分离和组装两片高耦合的代码。这意味着你可以修改数据的内部格式，而不用修改由 ContentProvider 暴露的接口，这样，也不会影响其它应用程序。

3）不要打断用户

如果用户正在运行一个应用程序（例如，Phone 程序），断定对用户操作的目的才是安全的。这也就是为什么必须避免创建 Activity，而是直接在当前的 Activity 中响应用户的输入。

那就是说，不要在 BroadcastReceiver 或在后台运行的 Service 中调用 callActivity()。这么做会中断当前运行的应用程序，并导致用户恼怒。也许更糟糕的是，你的 Activity 可能成为“按键强盗”，窃取了用户要提供给前一个 Activity 的输入。视乎你的应用程序所做的事情，这可能是个坏消息。

不选择在后台直接创建 Activity UI，取而代之的是，应该使用 NotificationManager 来设置 Notification。它们会出现在状态栏，并且用户可以在他空闲的时候点击它们，来查看你的应用程序向他显示了什么。

（注意，如果你的 Activity 已经在前台了，以上将不适用：这时，对于用户的输入，用户期望的是看到下一个 Activity 来响应。）

4）有太多事情要做？在线程里做

如果你的应用程序需要执行一些昂贵或耗时的计算的话，你应该尽可能地将它挪到线程里。这将阻止向用户显示可怕的“Application Not Responding”对话框，如果不这样做，最终的结果会导致你的应用程序完全终止。
一般情况下，Activity 中的所有代码，包括它的 View，都运行在相同的线程里。在这个线程里，还需要处理 UI 事件。例如，当用户按下一个按键，一个 key-down 事件就会添加到 Activity 的主线程队列里。事件处理系统需要很快让这个事件出列并得到处理；如果没有，系统数秒后会认为应用程序已经挂起并为用户提供杀死应用程序的机会。

如果有耗时的代码，内联在 Activity 上运行也就是运行在事件处理线程里，这在很大程度上阻塞了事件处理。这会延迟输入处理，并导致 ANR 对话框。为了避免这个，把你的计算移到线程里。在响应灵敏性设计的文章里已经讨论了如何做。

5）不要让一个 Activity 超负荷

任何值得使用的应用程序都可能有几个不同的屏幕。当设计 UI 屏幕时，请一定要使用多个 Activity 对象实例。

依赖于你的开发背景，你可能理解 Activity 类似于 Java Applet，它是你应用程序的入口点。然而，那并不精确：Applet 子类是一个 Java Applet 的单一入口点，而一个 Activity 应该看作是你的应用程序多个潜在入口点之一。你的“main”Activity 和其它之间的唯一不同点是“main”Activity 正巧是在 AndroidManifest.xml 文件中唯一对“android.intent.action.MAIN”动作感兴趣的 Activity。

因此，当设计你的应用程序的时候，把你的应用程序看作是 Activity 对象的集合。从长远来看，这会使得你的代码更加方便维护。

6）扩展系统主题

当谈到 UI 观感时，巧妙地交融非常重要。用户在使用与自己期望相反的 UI 的应用程序时，会产生不愉快的感觉。当设计你的 UI 时，你应该尽量避免太多自己的主题。相反的，使用同一个主题。你可以重写或扩展你需要的主题部分，但至少在与其它应用程序相同的 UI 基础上开始。详细请参照“应用风格和主题”部分。

7）设计你的 UI 可以应对多屏幕分辨率

不同的 Android 设备可能支持不同的屏幕分辨率。甚至一些可以自己变更分辨率，例如，切换到风景模式。确保你的布局和图片能足够灵活地在不同的设备屏幕上正常显示。

幸运的是，这很容易做到。简而言之，你需要做的是为主要分辨率提供不同版本的作品，然后为不同的尺寸设计你的布局。（例如，避免使用硬编码位置而使用相对布局。）如果那样做的话，系统会处理剩下的部分，而且你的应用程序在任何设备上都看起来很棒。

8）假设网络很慢

Android 设备会有多种网络连接选项。所有的都提供数据访问，但之间肯定有更快的。其中，速度最慢的是 GPRS，GSM 网络的非 3G 数据服务。即使具备 3G 能力的设备在非 3G 的网络上也会花费很多的时间，所以，网络很慢仍然是一个长期存在的事实。

这就是为什么你应该按照最小化的网络访问和带宽来编写你的代码。你不能假设网络是快速的，所以，你应该总是计划它是慢的。如果你的用户碰巧在一个快速的网络上，那很好——他们的用户体验会提升。你要避免相反的情形：在不同的地点和不同时间，应用程序有时可用，有时慢得令人抓狂，这样的程序可能不会受欢迎。

还有一个潜在的地方是，如果你正在使用模拟器，那么你很容易受它迷糊，因为模拟器使用电脑的网络连接。这比手机网络快很多，所以，你需要修改模拟器设定来模拟较低的网络速度。你可以在 Eclipse 中做到这点，在启动选项的模拟器设置页里设置或者在启动模拟器时通过命令行选项设置。

9）不要假定触摸屏或键盘

Android 可以支持多种外观形状。也就是说，一些 Android 设备拥有全“QWERTY”键盘，而其它可能会有 40 键、12 键或其它键盘设置。同样的，一些设备可能有触摸屏，但一些也会没有。

当创建你的应用程序的时候，记住这一点。不要假定特定的键盘布局——除非你真的想限定你的应用程序只运行在某些设备上。

10）节省设备电池

如果移动设备经常插在墙上，那么，它也就不是很“移动”。移动设备是电池供电的，如果我们能让每次充电的电池使用得更持久一些，那么每个人都会更加开心——尤其是用户。其中两大耗电硬件是处理器和无线；这也就是我们为什么要写尽可能少做工作、尽可能少去使用网络的应用程序的重要原因。

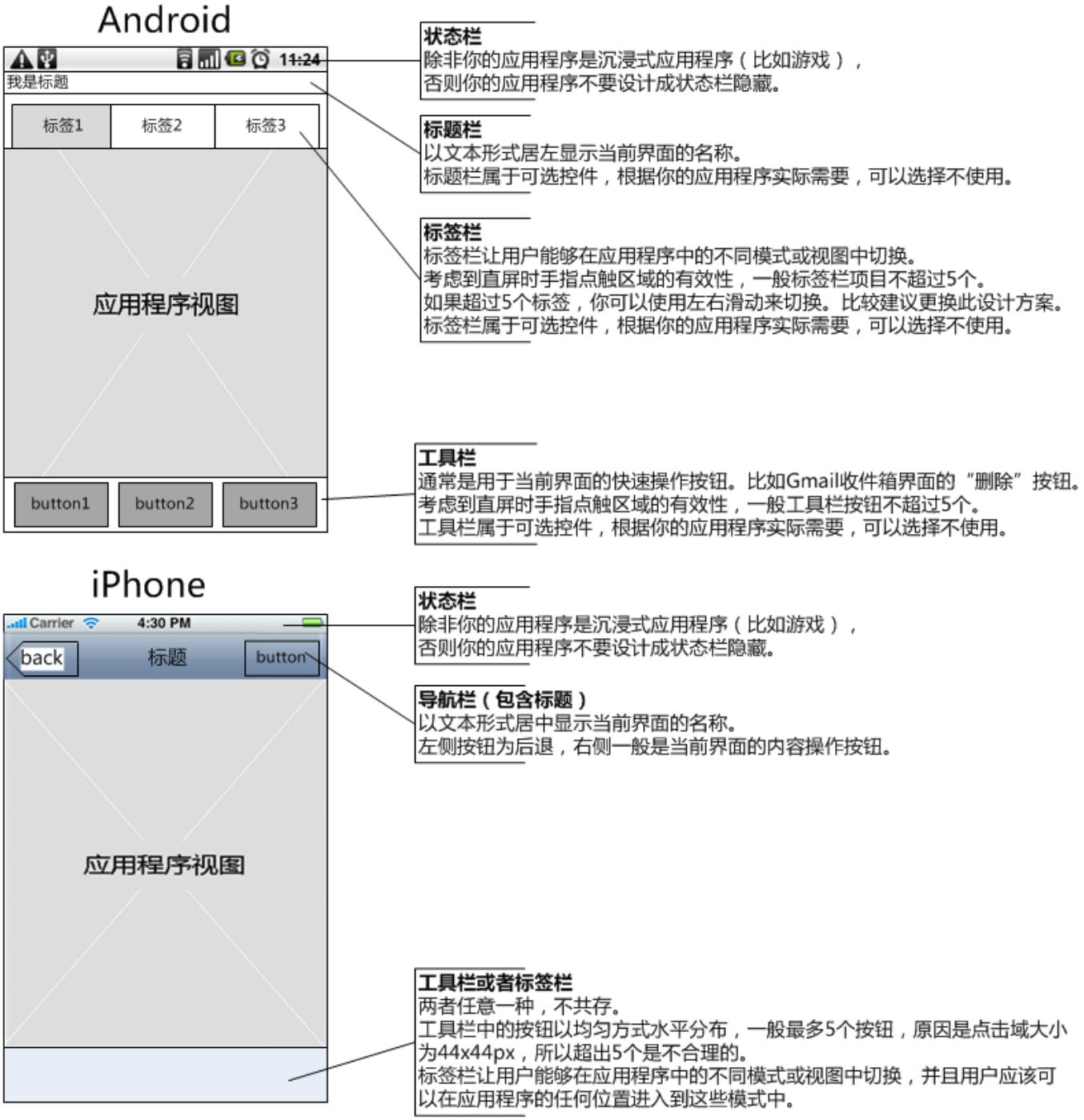
如何让你的应用程序最小化的占用处理器，归根结底还是要写高效代码。为了减少无线的电量消耗，确保对错误条件进行正确的处理，并只获取你要的东西。例如，如果某一个网络操作失败了，不要不断地进行重试。如果失败了一次，有可能是用户不受欢迎，因此，如果你再以正确的方式操作，有可能还会失败；所有你做的都是在浪费电池。

用户是相当聪明的：如果你的程序高耗电，他们是一定会发现的。到那个时点，你唯一可以确定的是，你的程序将很快被卸载掉。

【手机 UI 设计最佳实践】

Android 与 iPhone 应用程序界面布局对比

下图是我根据 Android 和 iPhone 这两个平台的“生产内容型的应用程序”整理出来的界面布局示例。所对比的 4 个点，均是两大平台的应用程序常规界面元素。



状态栏：

Android 和 iPhone 的状态栏均在屏幕顶端的位置，原因无外乎是人的视觉流程是从上到下的。

Android 的状态栏，具有 notification 的功用，当应用程序有新的通知，在状态栏左侧显示通知图标，向下滑动即可打开查看通知详情。

iPhone 的状态栏，包含了活动状态的显示，比如某进程正在运行，会有个转动的动画在这里，但是你不能对这个动画有任何操作。另外，当你在浏览时，轻击 iPhone 状态栏，也能起到快速至顶的作用。

个人认为，Android 通知系统做得很不错，貌似后来的 Windows Phone7 也效仿了此设计。

标题栏：

Android 是纯粹的界面标题栏，这里一般无操作响应。
而 iPhone 的标题栏，承担更多的作用是导航，你能在这里快速后退（Android 使用实体按键来后退），或是针对此界面进行一些操作。

标签栏：

Android 的标签栏位于界面标题之下，一般最多 5 项，在原生的平台界面，视觉效果也较丑。
而 iPhone 的标签栏是明确在屏幕下方的，也是最多 5 项。

相比于 Android，iPhone 对状态栏的处理显得很智能。所有显示图标和文字的标签都是相同的宽度并且显示黑色背景。当标签被选中后，它的背景淡化并且标签中的图片变亮。
如果应用程序的标签栏包含 5 个以上的标签，iPhone OS 会显示其中的 4 个并在第 5 个自动显示为“更多标签”。

我曾经思考过，为什么同是标签，Android 在上，iPhone 在下的问题。

到目前，我也只能猜测：
Android 的设计师认为，标签应该先被看到，并且不能让 menu 键的菜单项挡到。
iPhone 的设计师认为，标签是用于切换当前应用程序不同视图的，应该更容易被按到，所以选择在屏幕下方。

也因此，我已经完全不纠结所谓的“单手持机”的情境设计，因为以上两平台均无法很好地支持。

工具栏：

Android 的工具栏一般居于屏幕下方，一般是 3 个按钮。考虑到可点击域的有效性，一般最多也不建议超过 5 个。

而 iPhone 的工具栏，是与标签栏在同一位置的，即标签栏与工具栏不共存，在屏幕下方，要么是工具栏，要么就是标签栏。

老实说，我现在并没有在设计新的手机平台，我们公司也不会去折腾新的“某某 phone”。因此鉴于我目前站立的只是这个高度，我没有理由去为我的应用程序界面对上述四点进行“创新”。

不过，我总是能发现一些“A mix B” 的应用程序界面设计，让我觉得更加无聊。

UC 浏览器 Android 版，屏幕右下角的菜单按钮显得有些多余。

手机 QQ for Android，典型的 A mix B 设计。
除了底部离奇的菜单栏+标签混合使用之外，屏幕左下角的菜单按钮，似乎是步 UC 的后尘。
另外，列表项目，比如某好友，需要使用双击才能打开（或叫先聚焦再单击）。Android mix Symbian S60 v5。这应该是个不加分析的直接移植版。

天天动听 Android 版，和手机 QQ for Android 一样，底部离奇的菜单栏+标签混合使用之外，屏幕左下角的菜单按钮，似乎是步 UC 的后尘。

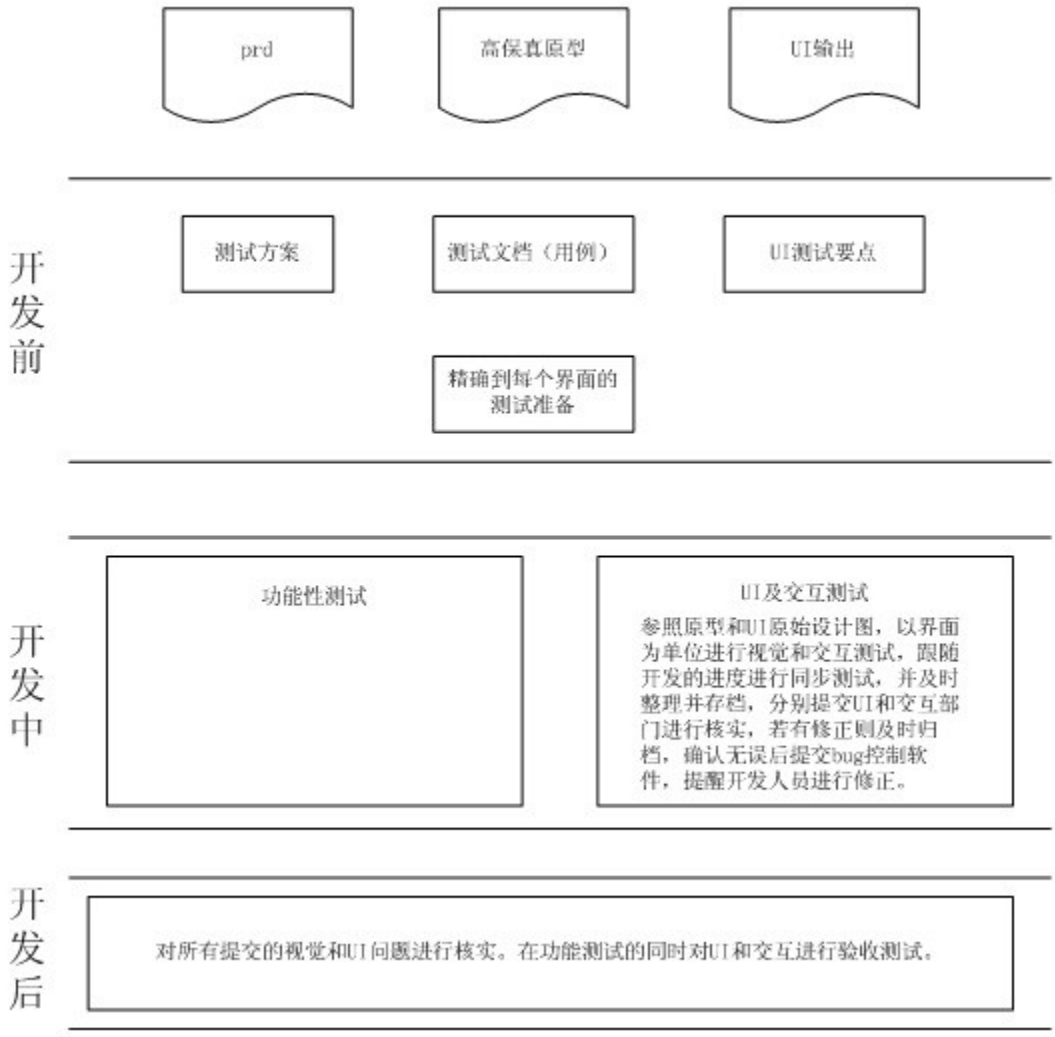
腾讯微博 Android 客户端以及开心网 Android 客户端，不用说了，Android mix iPhone。

无论是哪一个平台的应用程序设计，我个人始终坚持的观点是：遵循 OS 的设计规范，熟知平台特性，最好与平台自带的应用程序保持一致的使用体验。不需要惊喜。

你的创新，应当用在正确的地方。

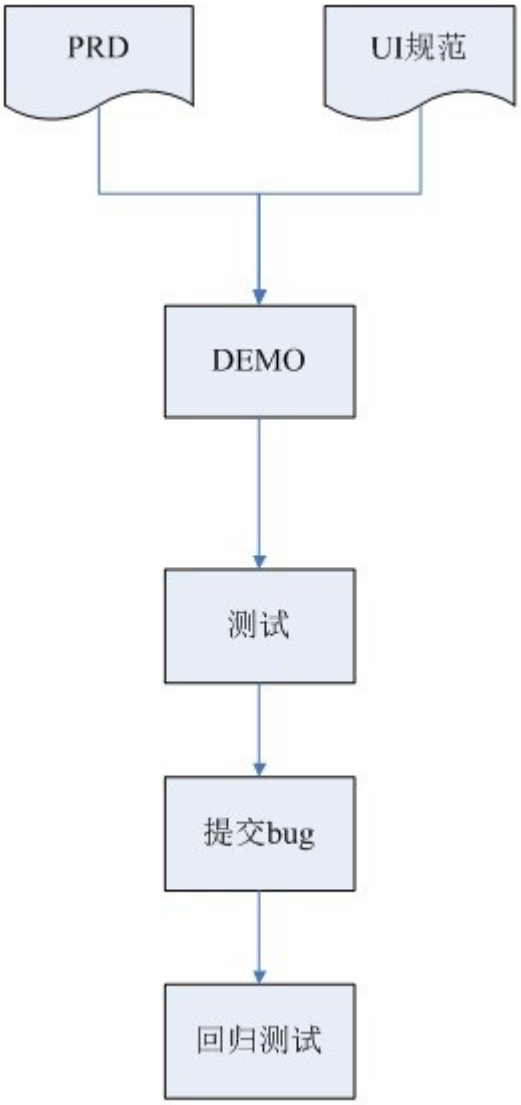
手机客户端 UI 测试分析

随着众多网站不断的在客户端的布局，不论是门户、社区、购物网站等都将手机客户端做为发展的一个方向，手机客户端仅局于一个小小的屏幕内，对于手机客户端的 UI 测试有着与网站不同的测试方式。从客户端起动界面开始，到运行过程，直至退出，UI 测试都有着自己的规范和要求。
手机客户端测试人员在产品过程中的工作：



手机客户端 UI 测试常见的测试点：
1．各种分辨率下，显示正常。现市场上主流的塞班 V3 系统手机为 240*320、320*240，V5 系统手机为 360*480。WM 系统主要为 240*320、480*640、480*800。Android 系统主要为 320*480，480*800。Iphone 系统为 320*480。在产品确定设计前在哪些系统中些屏幕下运行。测试将对不同的屏幕下对 UI 在不同的机型个测试效果。
2．前景色与背景色搭配合理协调，反差不宜太大，最好少用深色，如：大红，大绿等，常用色考虑使用手机系统的界面色调。对于 UI 在设计上的用色，测试可以提出很多宝贵的意见，只有图片跑在手机系统才可以更好的分辩出 UI 设计的图片是否会产生误差，这里的误差是指图片颜色是否与手机系统搭配，是否与视觉设计的想法有出处。

3．与正在进行的操作无关的按钮应该加于屏蔽（在 windows mobile 用灰色显示），或许与 WM 系统的界面有关，对于不同的系统，在 UI 测试上要有所不同，在满足手机特性的情况下，如何做到对于手机界面 UI 测试显得更加重要。
4．控件的焦点与非焦点状态的边框要有明显的区别。对于控件上的焦点掌握，在不同颜色下的边框有着严格的要求。即在选中与未选中下，UI 对于控件不同，这对于 UI 测试的要求更高。
5．长操作（下载，上传，更新，登录等）时，要有明确的动态指示 logo 或文字（例如：loading…等），表明操作正在进行中。手机访问速度没有 PC 快，对于手机小屏幕很容易失去耐心，简短的提示就是为了让用户继续停在当前页面，同时友好的 UI 界面提示也显得很重要。
6．对于非法的输入或操作应有足够的提示说明，提示、警告或错误说明应该清楚、明了、恰当的跳出提示警告画面，但冲击力不能太强。
7．文字描述的准确性：a. 文字描述与对应功能是否一致 ； b. 错别字。
8．文字用语的一致统一：父窗口的选项与子窗口标题统一一致。
9．产品帮助文档：a. 与产品功能和截图配套一致，当重新打包新系统时，及时更新产品帮组文档； b.文档格式； c.帮助中应该提供技术支持方式，一旦用户难于解决可以方便寻求新的帮助方式。
10．产品的版权和商标的 logo 和文字申明（一般在启动界面或者软件产品的“关于”选项里面）；涉及公司的形象和品牌，一定要规范标准化。
11．给用户提供自定义界面风格，由用户自己选择颜色和字体。满足不同用户习惯，同时满足用户对于一些颜色偏差（如色弱用户）。
测试流程：
1．根据提供的 UI 规范文档以及 PRD 了解整个产品的业务和 UI 规范。
2．开发提供已设计好的 DEMO。
3．如果是单一的 DEMO，不涉及业务流程，跳到第 5 否则跳到第 4。
4．结合 PRD 中的业务流程以及 UI 规范进行测试，并提交发现的 BUG，执行完后进行第 7。
5．使用 UI 规范文档对已提交的 DEMO 页面进行测试，并提交 BUG。
6．对后续给出的 DEMO 页面根据 PRD 的业务进行集成测试，并提交 BUG。
7．对 BUG 进行跟踪回归测试。



客户端是由各类的手机控件下组装成的，优秀的 UI 设计同时也抓住对控件的 UI 优化达到产品的优化。针对于一些手机界面常用的控件，在正常状态下、选中状态下等多种状态的常见效果展示：

【其他】

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/thread-34359-1-1.html

关于 eoeandroid

eoeandroid 团队是一个有远大的梦想，有充沛的激情，有高效执行力的团队。北京易联致远无线技术有限公司于 2009 年 8 月成立。eoeandroid 的核心成员有在摩托罗拉、卓望科技、T3g 和手机 design house 的工作经历和相关丰富的行业经验。eoeandroid 致力于让移动互联网的软件开发变得容易，发布变得更加方便，传播变得更加迅捷,让用户以最快的速度获取最适合自己的移动互联网应用。

新版优亿市场上线

优亿市场（eoeMarket）经过 eoe 团队无数个白天黑夜的奋斗，数个版本的调整和升级，我们迎来了一个更加优优秀的新版本（V0.2.2 内测版）；其本土化、纯中文支持；高质量、分类清晰的应用，无论你是狂热的 android 玩家，还是初涉 android 江湖的小白，你都可以在上面轻而易举的找到自己想要的软件和游戏。而这一切都是免费的。
新版优亿市场（eoeMarket） V0.2.2 内测版特性如下：

- 全新设计的炫酷界面
- 智能检测已安装应用的新版本
- 更加精准的智能推荐
- 更加丰富的应用专题
- web 端和客户端收藏列表一键同步
- 前所未有的稳定性

使用方法：

第一步:准备工作

1. 手机在主屏幕状态 按下“MENU”键，选择“设置”菜单
2. 在设置界面中选择“应用程序”
3. 在应用程序设置界面勾选“未知来源”选项，允许安装非电子市场提供的应用程序

第二步:下载并安装 eoe 应用商店 2

a.通过手机浏览器安装

1.输入网址 eoemarket.com/a，按下“转到”按钮

2.下载完成后，点击“eoeMarket2.apk”,启动安装程序

b.通过 PC 安装

1.浏览器输入 eoemarket.com/a 或者点击本页的'立即下载'

2.下载完成后，将“eoeMarket2.apk”拷贝到手机 SD 卡中

3.安装

北京易联致远无线技术有限公司
电话：（010）82176766
E-mail:company@eoemobile.com
网址：www.eoemobile.com
中国最大的android开发者社区：www.eoeandroid.com
中国本土的android软件下载平台：www.eoemarket.com