参考文献

1 彭晨,郭静,裴灵犀.无线传感器网络中的节能策略.控制工

(下转第 239 页)

Android 应用中一种 Activity 窗口管理系统①

夏德冰，陈庆奎

(上海理工大学 光电信息与计算机工程学院，上海 200093)

摘 要：Android 应用程序是由多个组件组成的，Activity 作为 Android 程序的重要组件之一，用于显示可视化的用户界面，接收与用户交互所产生的界面事件。本文提出了 AWMS(Activity Window Management System)机制，用于在多 Activity 窗口的应用中存储已开启的 Activity 窗口，以方便程序调用 Activity 句柄，对其数据及状态进行操作。实现了对 Activity 实例的获取以及对任务中每一个 Activity 的获取。

关键词：Android 操作系统；Activity 生命周期；Activity 窗口管理；多任务

Activity Window Management System in the Android applications

XIA De-Bing, CHEN Qing-Kui

(School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China)

Abstract: Android applications are composed of multiple components, Activity as an important component of Android applications, used to display a visual user interface and receive user interface events generated by interaction. In the paper, AWMS(Activity Window Management System) mechanism is used to store the Activity window in multiple applications, which simplified operation for calling Activity handle to operate its data and state. Thus implement the simplified operation for getting the Activity instance and each of the Activity in the tasks.

Key words: Android operating system; activity life cycle; activity window management; multiple tasks

随着手机功能日趋强大，手机应用软件已成为用户直接操作和应用的主体，美观实用、操作便捷的用户界面会为应用软件带来巨大的市场前景。Google 公司推出的 Android 是开源的手机操作系统，为其应用程序的开发提供了广阔的发展空间[1]。而 Android 应用程序是由多个组件构成的，其中组件 Activity 是 Android 应用程序生命周期的重要组成部分，其运行与管理是整个系统的基础。本文通过对 Activity 生命周期以及栈和任务的研究，设计了一种 Activity 窗口管理系统，介绍了其原理、组成、算法及优势；为 Activity 的动态加载、程序的快速退出以及 Activity 的数据获取及控制等实际开发中的重要问题提供了一种良好的解决方案。

1 Android相关技术简介

1.1 Activity 生命周期

Android 提供的每个生命周期方法都有不同用途，这些生命周期方法分别在活动的前台阶段、可见阶段或整个生命周期阶段被调用。生命周期每个部分中常用的方法和前后台阶段涉及到的方法如图 1 所示[3]。Activity 的生命周期是从创建到销毁的过程，包括活动状态，暂停状态，停止状态，非活动状态[2,3]，其主要方法如下：

onCreate()：创建 Activity 时调用，对所有的“全局”状态做初始设置。并以 Bundle 的形式提供对以前存储的任何状态进行访问。

onRestart()：重新启动 Activity 时调用。该 Activity

① 基金项目:国家自然科学基金项目(60970012);上海信息技术领域重点科技攻关项目(09511501000);上海重点科技项目(09220502800);

上海市重点学科建设项目(S30501)
收稿时间:2011-10-08;收到修改稿时间:2011-11-07

仍在栈中，而不是启动一个新的 Activity。

onStart()：Activity 变为在屏幕上对用户可见时调用。

onResume()：Activity 开始与用户交互时调用（启动和重新启动 Activity，该方法都被调用）。

onPause()：Activity 被暂停或收回 CPU 和其他资源时调用。该方法是用来保存 Activity 状态的地方，以便 Activity 被重新启动时可以具有与其退出时相同的状态。

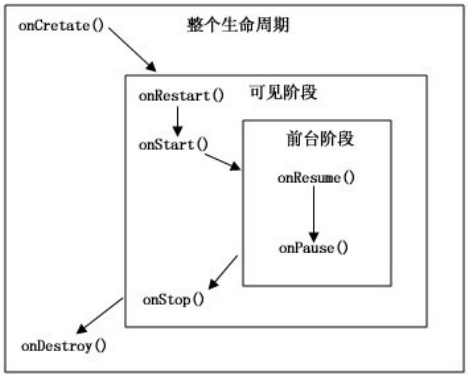

图 1 Activity 生命周期以及前后台阶段状态图

onStop()：Activity 被停止并被转换为不可见阶段及后续的生命周期事件时调用。

onDestroy()：Activity 被完全从系统内存中移除时调用。该方法被调用可能是因为有人直接调用 onFinish()方法或者系统决定停止该 Activity 以释放资源。

1.2 Activity 和任务

Android 的核心特点是一个应用可以使用其他应用的组件（需要应用授权），而应用不需要包含用到的另一个应用的代码也不需要连接这些代码，只是在需要时启动其包含的组件。一个 Activity 可以启动另一个 Activity，包括不同应用中的 Activity。并可以通过 BACK 键后退使之前的 Activity 重现在屏幕上。Android 通过将这两个 Activity 放进一个任务中来实现这种用户体验[4]。任务是栈中的一组相关的 Activity，栈中的根节点是任务的初始 Activity(通常为用户首先启动的 Activity)。栈顶的 Activity 是当前正在运行的 Activity，当一个 Activity 启动另一个 Activity 时，新的 Activity 被压入栈中，成为运行的 Activity。当用户按 BACK 键时，当前 Activity 弹出栈，先前的 Activity 恢复为运行的 Activity 显示在屏幕上。Activity 不会在栈中重组，而只是“后入先出”地入栈与出栈。Android 支持多任务切换。一个任务里所有 Activity 组成一个单元，整个任务可以在前台或后台运行，应用程序的切换就是任务的切换。任务不是在 AndroidManifest.xml 文件里的一个类或元素，所以没办法为一个任务里的 Activity 独立设置值。

以上为 Activity 和任务的默认行为，Android 提供其他方法修改这种机制的所有方面。Activity 和任务的关联以及任务中 Activity 的行为都要受到启动 Activity 的 Intent 对象的 Flag 和 AndroidManifest.xml 文件中的 Activity 属性集合共同控制。主要的 Intent 标记为：FLAG_ACTIVITY_NEW_TASK,FLAG_ACTIVITY_CLEAR_TOP,FLAG_ACTIVITY_RESET_TASK_IF_NEEDED,FLAG_ACTIVITY_SINGLE_TOP 。主要的 Activity 属性为：taskAffinity,launchMode,allowTaskReparenting,clearTaskOnLaunch,alwaysRetainTaskState，finishOnTaskLaunch。

同时，Android 提供 Activity 四种加载模式[2]：standard：标准默认模式，通过调用 startActivity()方法产生一个新的实例；singleTop：如果有一个实例位于 Activity 栈顶，则不产生新的实例，而只是调用 Activity 中的 newInstance()方法。否则，产生一个新的实例。singleTask：会在一个新的任务中产生实例，以后每次都会该实例，不会去产生新的实例。singleInstance：与 singleTask 基本相同，只有一个区别：在该模式下 Activity 实例所处的任务中，只能有这一个实例不能有其他实例。这些启动模式都需要在功能清单 Android Manifest.xml 中进行设置 launchMode 属性。

2 AWMS的设计

2.1 AWMS 设计原理

在实际应用中，一个 Android 应用程序往往包括多个 Activity，且多个 Activity 窗口之间可以相互跳转，虽然彼此结合形成一个共同的应用接口，但是每个 Activity 都是独立的，都是 Activity 的子类。要将每个 Activity 窗

口进行管理，则必须将这些实例保存起来。AWMS 中选择 ArrayList<Activity>作为容器，装载打开的 Activity 窗口，其中 ArrayList 是 List 接口的一个可变长数组的实现，可以满足随时添加或移除 Activity 的要求。AWMS 将用户开启 Activity 窗口的轨迹进行“记忆”，将 Activity 从容器底端按照开启的顺序依次存储到容器中。容器的进出遵循“后进先出”的原则，位于容器顶端的 Activity 为正在运行状态的 Activity 显示在屏幕上；处于容器顶端以下的 Activity，但并没有销毁，而是处于暂停/停止状态；当按 BACK 键返回时，就将容器顶的 Activity 移出并销毁，同时将之前开启的 Activity（即此时位于容器顶的 Activity 窗口）转成运行状态，显示在屏幕上，而不需要重新创建之前的 Activity 了；每个被存储的 Activity 都有自己的句柄(再次开启之前的 Activity 窗口，会被重新存储在容器中)；当退出应用程序时，从容器顶到容器底，依次将压入容器的 Activity 移出并销毁。AWMS 设计原理图如图 2 所示。

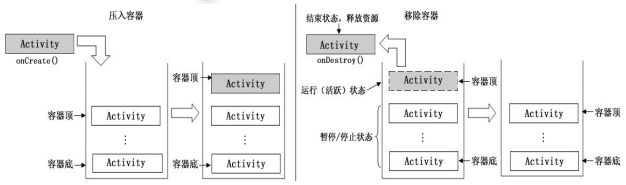

图 2　AWMS 设计原理图

2.2 AWMS 的组成和算法

AWMS 主要由 8 个部分组成，一个容器采用 ArrayList<Activity>和 7 个重要函数模块：

（1）创建容器：private static ArrayList<Activity> actiList_ = null;

（2）addActivity 模块：该函数在 Activity 的 onCreate()方法中调用，实现被创建的 Activity 都压入到容器中。public static void addActivity(Activity activity) {

```
    Log.e("Tag", "addActivity");
    if (activity != null) {
        if (actiList_ == null)
            actiList_ = new ArrayList<Activity>(0);
            actiList_.add(activity);
        }
    }
```

（3）removeActivity 模块：从容器中移除当前的 Activity，实现销毁的 Activity 从容器中移除。

```
public static void removeActivity(int actId) {
    try {
        if (actiList_ == null || actiList_.size() == 0)
                return;
            if (actId >= actiList_.size())
                return;
            Activity act = actiList_.remove(actId);
            act.finish();
        } catch (Exception e) {
        }
    }
```

（4）getHomeActivity 模块：实现获取最早开启的处于容器底端的 Activity。

```
public static Activity getHomeActivity() {
        if (actiList_ == null || actiList_.size() == 0)
            return null;
        return actiList_.get(0);
    }
```

（5）getTopActivity 模块：实现获取容器顶端的 Activity。

```
public static Activity getTopActivity() {
        if (actiList_ == null || actiList_.size() == 0)
            return null;
        return actiList_.get(actiList_.size() - 1);
    }
```

（6）getLastActivity 模块：实现获取上一个 Activity。

```
public static Activity getLastActivity() {
        Log.e("Tag", "getLastActivity");
        if (actiList_ == null || actiList_.size() <= 1)
            return null;
        return actiList_.get(actiList_.size() - 2);
    }
```

（7）clearActivity 模块：获得容器 ArrayList 中压入 Activity 的个数，然后对容器进行遍历，本着“后入先出”的原则，将压入容器的所有 Activity 按照次序一一移出，并调用 Activity 的 finish()方法进行销毁，实现无返回的快速退出整个应用程序。

```
public static void clearActivity() {
        try {
```

```
            int sz = actiList_.size();
            Log.e("Tag","size="+String.valueOf(sz));
            for (int i = sz; i > 0; i--) {
                Activity act = actiList_.get(i - 1);
                if (act == null) {
                    actiList_.remove(i - 1);
                    continue;
                }
            actiList_.remove(i - 1);
            act.finish();
            act = null;
            }
        } catch (Exception e) {
            Log.e("clearActivity fail+e.toString(),"");
        }
    }
```

（8）getActivitys 模块：返回 ArrayList<Activity> 容器，通过调用该函数实现获取压入容器的所有 Activity 实例。

```
public static ArrayList<Activity> getActivitys() {
    return actiList_;
}
```

3 实验结果与优点分析

图 3 为实验的运行过程的界面显示图，图 4 为程序运行通过 Logcat 打印出的 4 个 Activity 的生命周期情况以及运行中的数据信息。其中：

步骤 1：Activity01 中显示了一段从网络获取的数据信息。点击“按钮 2”跳转到 Activity02，Activity02 运行，刷新 Activity01 中的数据，并且获取显示在 Activity02 的屏幕上（经历：01 暂停->02 显示在界面->01 停止）。

步骤 2：Activity02 正在运行的状态下，切换任务，开启拨打电话的应用（经历：02 暂停->02 停止）。

步骤 3：在电话应用界面上，按 BACK 返回，屏幕显示 Activity02，恢复离开前的状态（经历：02 从可见阶段的 onRestart()->onStart() 到前台阶段的 onResume()，02 运行。图 4 蓝色框部分为步骤 2、3）。

步骤 4：Activity02 点击“按钮 3”跳转到 Activity03 窗口（经历：02 暂停->03 运行->02 停止）。

步骤 5：Activity03 点击“按钮 4”跳转到 Activity04 窗口（经历：03 暂停->04 运行->03 停止）。

步骤 6：Activity04 按 BACK 键返回 Activity03 窗口（经历：04 暂停->03 从可见阶段的 onRestart()->onStart()到前台阶段的 onResume()，03 显示在屏幕上->04 停止->04 销毁）。

步骤 7：Activity03 点击“按钮 4”跳转到 Activity04 窗口（经历：03 暂停->04 重新创建运行->03 停止）。

步骤 8：Activity04 点击“退出应用”按钮，退出整个应用程序，显示手机 HOME 界面（经历：04 暂停->03 销毁->02 销毁->01 销毁->04 停止->04 销毁）。

图 3 运行界面

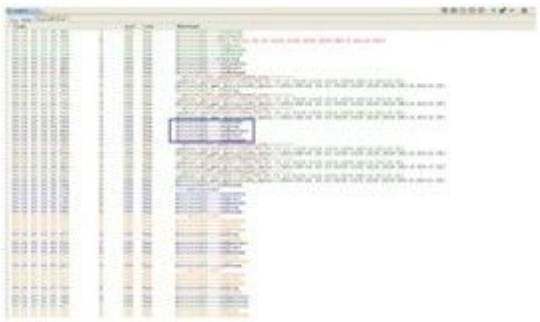

图 4 Activity 的生命状态及数据结果

优势分析：

AWMS 在开发中的可以随开发者需求的不同而有不同的应用方式。比如：添加自己需要的模块，在需要验证用户的时候可以添加登录界面等；在 Activity 比较少的应用中，AWMS 可以只作为一个类来提供其他 Activity 进行调用等等。AWMS 的使用范围不止如此，在游戏应用等多 Activity 的程序中用户界面可能有几十个或者更多，或者浏览器这样的未知窗口数量，需要动态加载 Activity 的应用，AndroidManifest.xml 文件不需要或者不可能对如此多的 Activity 添加定义 <activity android:name=".Acti01" android:label="Acti01" ></activity>。此时，窗口的整体框架基本相同，可以通过一个 Activity 来实现 AWMS 并提供方法作为基类，方便各个 Activity 窗口进行继承来调用其方法，从而实现管理整个应用的所有 Activity 窗口。实际开发中，AWMS 的灵活性和可扩展性可以使开发者根据具体应用的不同，对其进行扩展，产生更多的功能。

总之，AWMS 的优势为：（1）有序存储已开启的 Activity 窗口，便于当前 Activity 调用其他 Activity 句柄，实现了当前 Activity 窗口对已开启的其他窗口的灵活操作；（2）退出程序，可以一步销毁所有 Activity 窗口直接退出，无需将已开启的 Activity 窗口一一返

(下转第 248 页)

运用 Hough 变换提高直线检测效率①

祁宝英

(南昌航空大学 信息工程学院, 南昌 330063)

摘 要：针对 hough 变换检测直线段中的运算速度慢、占用内存多的问题，提出一种改进的检测直线段的方法。首先，对图像进行减半采样处理，采用从大概到精确的检测策略，减小运算量；其次采用 soble 算子进行边缘检测；最后，在经典的 hough 变换的方法上增加了边缘梯度幅度进行限制，减小了坐标转换的次数，并将断开的同一直线上的线段连接起来。

关键词：图像处理；直线检测；hough 变换

Improve Efficiency of Line Detection Using Hough Transform

QI Bao-Ying

(School of Information Engineering,NanChang HangKong University, NanChang 330063, China)

Abstract：Because of the slow detection speed and occupying big system memory for lines in digitial images, a new improved detection method was proposed. Firstly, sampling in half was used to extract process the image, adopted the strategy that from probably to accurate tese, reduce the computational complexity.Then, the soble operator was proceeded on edge detection. Finally, increased the edge gradient on the classical hough transform to reduce the number of coordinate transformation, and joined up the disconnected line.

Key words：image processing; line detection; Hough transform

图像特征提取是使用计算机提取图像信息，决定每个图像的点是否属于一个图像特征。而直线特征是图像中物体的基本特征之一，是机器视觉必须解决的重要问题，有着广泛的应用。在数字图像处理中,直线特征是图像分割的基础，如：道路的识别[1]，建筑物的识别：直线检测和分析在航空和卫星图像[2-4]等方面也有着广泛的运用。总之，直线检测的研究对图像处理和模式识别有着重大的意义。

直线检测算法主要有哈夫曼变换和最小二乘线性拟合算法。但是被广泛应用于图像处理领域的还是哈夫曼变换(Hough Transform)。Hough 变换是由 Paul Hough 于 1962 年提出的，它是将图像空间的点映射到参数空间。霍夫变换[7-9]以其优异的鲁棒性和很强的抗干扰能力被广泛地应用于计算机视觉和模式识别等领域，如直线以及一些参数曲线的检测等。但是由于精度不高，运算速度慢阻止了它在实时性要求很高的领域的广泛应用。

本文是从减少图像不必要的分辨率以及 hough 变换的改进方面着手，检测出图像中的直线特征并标注出来。

① 基金项目:航空科学基金(2009GZS0090)
收稿时间:2011-10-26;收到修改稿时间:2011-11-18

1 经典的哈夫曼变换基本原理

Hough 变换的基本原理在于利用点与线的对偶性，将笛卡尔坐标空间中的直线变换到极坐标空间中。这样就把原始图像中给定曲线的检测问题转化为寻找参数空间中的峰值问题。

Hough 变换的性质如下：

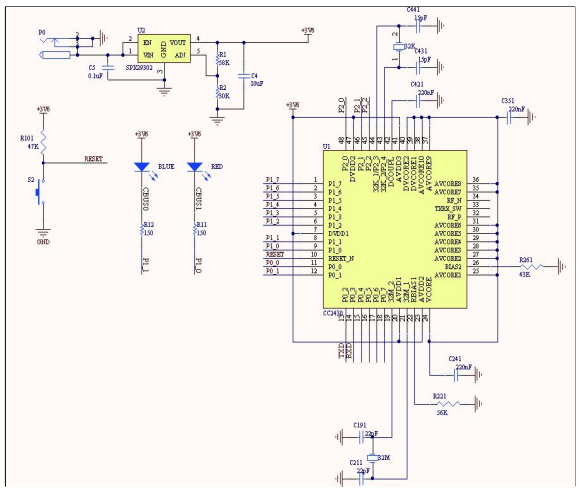

图 8　Zigbee 模块原理图

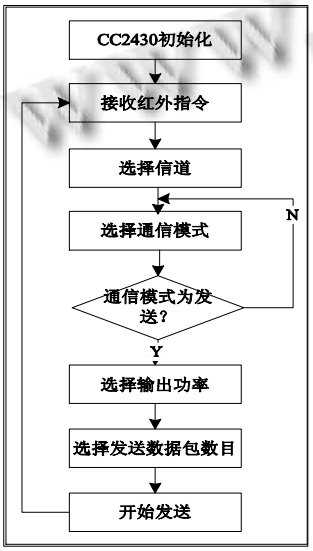

图 9　Zigbee 模块程序流程图

Zigbee 模块首先使用串口接收红外指令，然后通过射频通信，将红外指令发送到各个分控节点。

3　结语

随着 Intemet 技术的发展以及 32 位微处理器时代的来临，信息家电、信息家庭的概念已开始深入人心。提出以 ARM9 的 32 位微处理器 S3C2440A 作为家庭网关的中央处理器，软件上以嵌入式 Linux 为平台，实现了短信接收、遥控器自学习、红外指令发送等功能。该家庭网关成本较低，易于升级，便于推广应用。

参考文献

1 黄布毅,张晓华.基于 AMR-Linux 的 SQLite 嵌入式数据库技术.单片机与嵌入式系统应用,2005,(4):21–24.
2 王京谦,万花新.开源嵌入式数据库 Berkelye DB 和 SQLite 的比较.单片机与嵌入式系统应用,2005,(2):5–7.
3 杨利平,龚卫国,李伟红,王华华,周留洋.基于网络技术的远程智能家居系统.仪器仪表学报,2004,25(4):308–311.
4 张靖强,杜彦亭,杨进.基于家用计算机的远程家电控制的设计与实现.计算机应用,2010(6):394–395.
5 冯东,陈俊杰.基于 Linux 的智能家居网关管理软件设计与实现.舰船电子工程,2004(6):116–120.
6 夏玉杰,张栓记.基于 ARM 的嵌入式家庭网关研究与设计.ARM 开发与应用,2009(7):104–106.
7 郭文慧.智能家电控制器的虚拟实现.安徽理工大学,2009,06.

(上接第 227 页)

回显示，再退出；（3）支持多任务情况下切换任务，可以按 BACK 键直接回到之前应用窗口，恢复当时的状态，无需重新开启应用程序。

4　结语

本文先介绍了 Android 的系统架构及其组件 Activity 的生命周期，然后阐述了设计 Activity 窗口管理系统的原理和关键模块并简要介绍了使用方法和应用范围；最后展示实验结果，总结了优势；为 Android 平台下应用开发中的 Activity 管理，提供了一种简单，灵活，易操作的管理方案。

参考文献

1 Butler M. Android: Changing the Mobile Landscape. Pervasive Computing, IEEE, 2011.4–7.
2 Android Developer.http://developer.android.com.
3 W.Frank Ableson, Charlie Collins, Robi Sen. Google Android 揭秘.北京:人民邮电出版社,2010:58–60.
4 隆益民.Android 应用开发.广州:中山大学出版社, 2010:38–41.