GamerDroid

Using Android Phone as a Game Controller

1. Introduction

The basic idea of the project is to use an android device as a game controller to enhance the fun in the gaming experience of a user. Gaming is more fun if it is more intuitive. All gamers feel more comfortable and realistic when they play games on gamepad or gamecontrollers rather than on keyboards. In fact, the gaming experience is the best when one uses a motion sensing device. Using this idea as our target, we tried to make an application for the android devices which will make the user feel like he/she is playing with a real motion sensing game controller.

And the most fantastic thing is that the pre-requisites are only you need to have an android device, that's it and then you will be able to experience the pleasure and excitement and the fun of playing by motion sensing devices.

About :

Language : Java

Tool : ADK (Android development Kit),Connectify(For creating WiFi Hotspot).

IDE : Eclipse

Emulator : AVD (Android virtual device)

2. What is Android??

Android is a Linux-based operating system designed primarily for touchscreen mobile devices such as smartphones and tablet computers. It is currently developed by Google in conjunction with the Open Handset Alliance. Initially developed by Android Inc, whom Google financially backed and later purchased in 2005,Android was unveiled in 2007 along with the founding of the Open Handset Alliance, a consortium of 86 hardware, software, and telecommunication companies devoted to advancing open standards for mobile devices.

It is a software stack for mobile devices that includes an operating system, middleware and key applications. The Android SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language.

2.1) Specifications :

This operating system is based on version 2.6 of Linux, so it has a monolithic system kernel, what means that all system functions and drivers are grouped into one block of code.

2.2) Architecture

Android consists of five layers:

-The Linux kernel 2.6-which includes useful drivers that allow for example WiFi or Bluetooth.

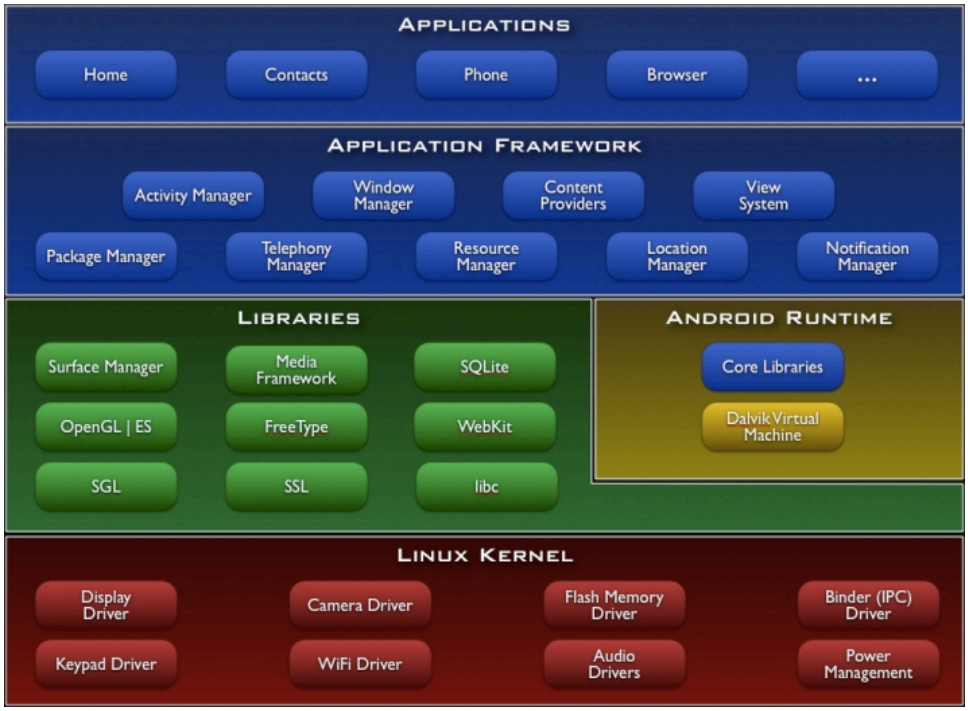
-The library written in C and C + + that provide higher level functionality such as an HTML engine, or a database (SQLite).

-A runtime environment for applications based on a virtual machine, made for inefficient machines such as telephones. The aim is to translate JAVA in machine language understood by Android.

-A JAVA framework that allows applications running on the virtual machine to organize and cooperate.

-The user applications written in Java (Web browser, contact manager etc. ..)

3. Architecture & Marketshare



Android Architecture

Characteristic of the market:

3.1) Competitors

-The principal competitor is iPhone OS. It is mainly for competing with Apple that Android has been created.

-Palm OS devices on PDA.

-Blackberry: which team the same name smartphones

-Windows Mobile: which team smartphones and PDAs.

-Symbian: Current Market Leader but now losing its position

3.2) Key partners

To help launch Android, Google has created an alliance of thirty companies in order to develop standards for mobile devices. There is, among others:

-Operators such as NTT Dokomo, T-Mobile or Bouygues Telecom.

-Of-equipment manufacturers like Sony Ericsson or Samsung.

-Manufacturers of semiconductors, including Intel and Nvidia.

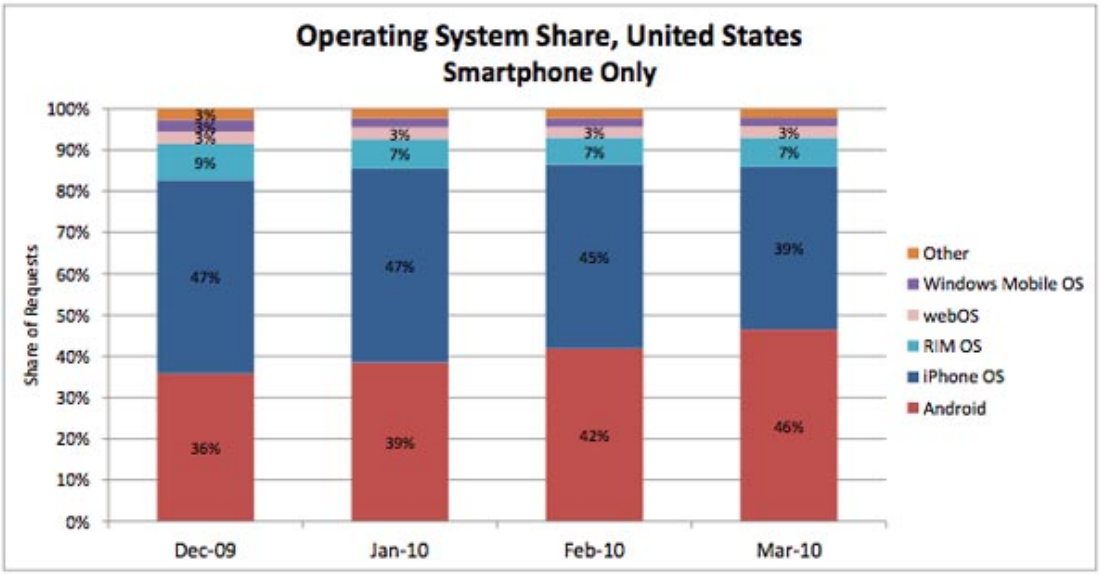
-Corporate businesses.

3.3) Market share

The android market share continues to increase since its inception, and is likely to continue climbing because it is favored by big players like HTC , Sony Ericsson, Samsung, LG, Motorola, Dell, Acer .

Moreover, according to IDC, android will be the 2nd mobile operating system used of the market in 2013.

Here is the state of the market from 2006 to 2009. You have to know that the first mobile phone appeared in android date October 2008.



Source : Screen Digest

Why Android is better ?

• Applications

- Google applications

Android includes most of the time many Google applications like Gmail, YouTube or Maps.

These applications are delivered with the machine most of the time, except in certain cases, such as some phones running android on which the provider has replaced Google applications by its own applications.

-Widgets

With android, it is possible to use widgets which are small tools that can most often get information. These widgets are directly visible on the main window.

-Android Market

This is an online software store to buy applications. Developers who created applications can add them into the store, and these applications can be downloaded by users, they can be both free and paid.

- **Multitasking**

Android allows multitasking in the sense that multiple applications can run simultaneously.

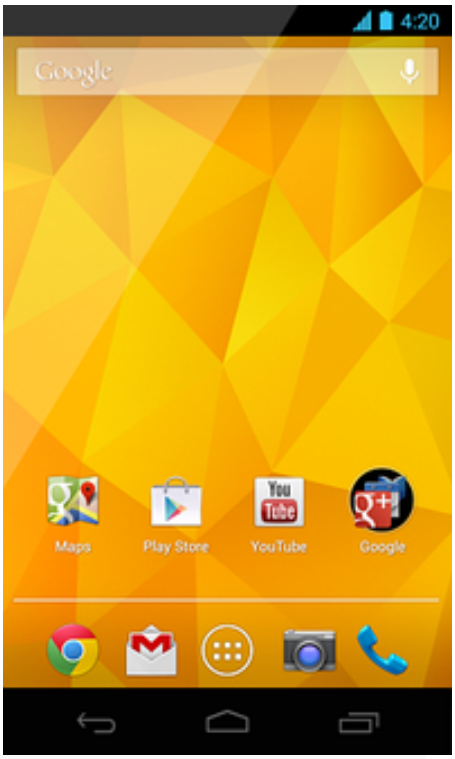
With Task Manager it is possible view all running tasks and to switch from one to another easily.

- **SDK**

A development kit has been put at disposal of everybody. Accordingly, any developer can create their own applications, or change the android platform. This kit contains a set of libraries, powerful tools for debugging and development, a phone emulator, thorough documentation, FAQs and tutorials.

- **Modifiability**:

This allows everyone to use, improve or transform the functions of Android for example transform the interface in function of uses , to transform the platform in a real system embedded Linux.



Android 4.2 "Jelly Bean" on the Nexus 4

Company /developer	Google Open Handset Alliance Android Open Source Project
Programmed in	C, C++, Java[1]
OS family	Unix-like, Linux
Initial release	September 23, 2008[3]
Latest stable release	4.2 "Jelly Bean" (November 13, 2012; 15 days ago)[4][5] [±]
Kernel type	Monolithic (modified Linux kernel)

4. About the Project

A lot of PC users normally play games using their keyboard and mouse. A more desirable and realistic(for some games) way to play games would be to use a game controller. A game controller may be a joystick or a game pad. But normally, good game controllers are pretty expensive. It would be handy if a gadget which someone already owns could be used as a controller.

Our Project “GAMER DROID” is provides a new gaming experience to users,imagine playing NEED FOR SPEED with Android's Accelarator,using Motion Sense Technology to control the direction of the car.

The idea is to make a generic controller for every game,even provide an interface on the Client side installed with button's ,which could be adjusted according to the keyboard/controller controls .

With buttons installed a lot of different games could be played and apart from the motion sensing game experience ,games such as FIFA would live out a new experience.

4.1) Funtioning

Our project will be using wifi connection of android to establish a link between the application used as Client on Android phone and a server running on the computer the game is installed upon.

We are using the wifi ports of the computer and a signal would be send by the Droid Controller to the machine using the ports of computer ,this signal is recognised by the GamerDroid program running on the computer.

The goal is to make the controller generic so that it works for all games and cross-platform for all Operating Systems. We would also like to provide multiple methods of connectivity which would be useful for systems that don't support or have a particular connection type, to connect via another connection type. Ex. Desktops which don't have bluetooth can connect through a usb cable.

5. Phases of the project

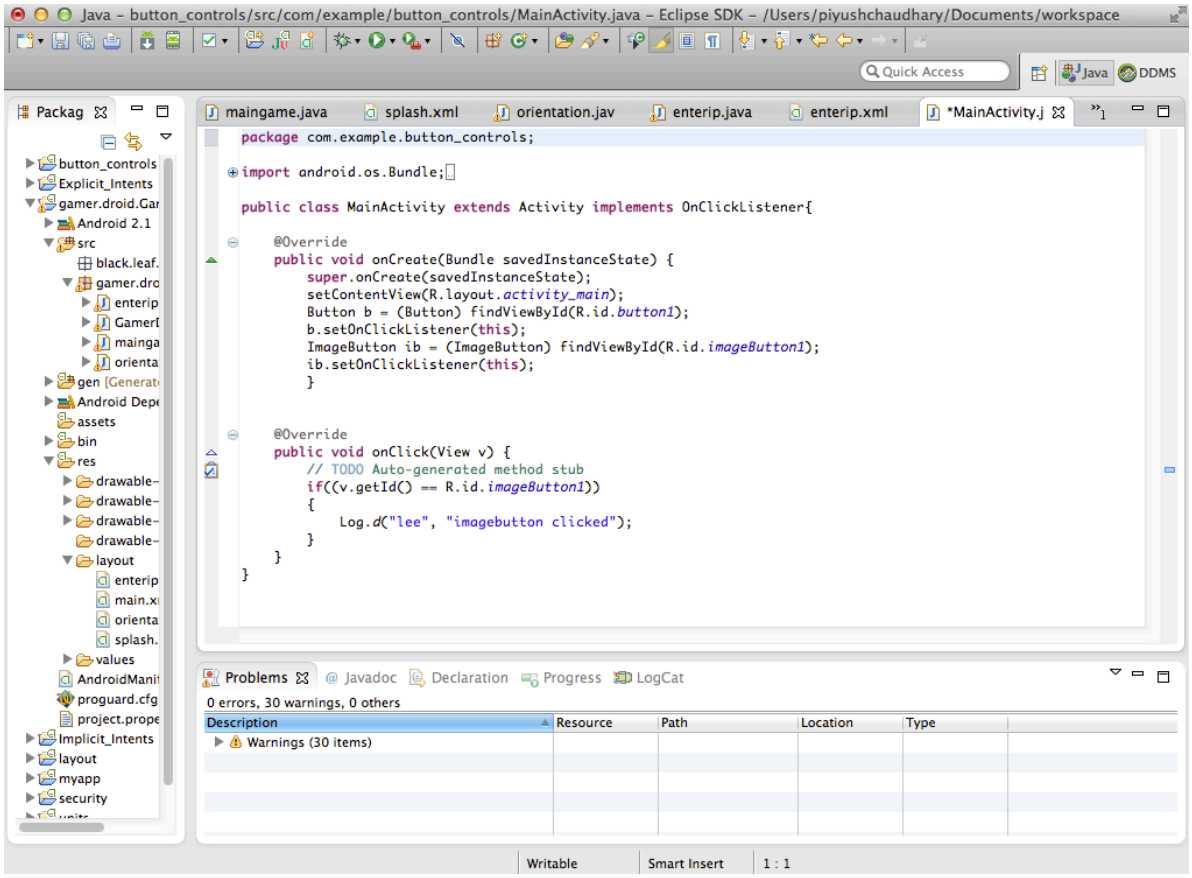
5.1) Phase I

Development of Android Application Environment.

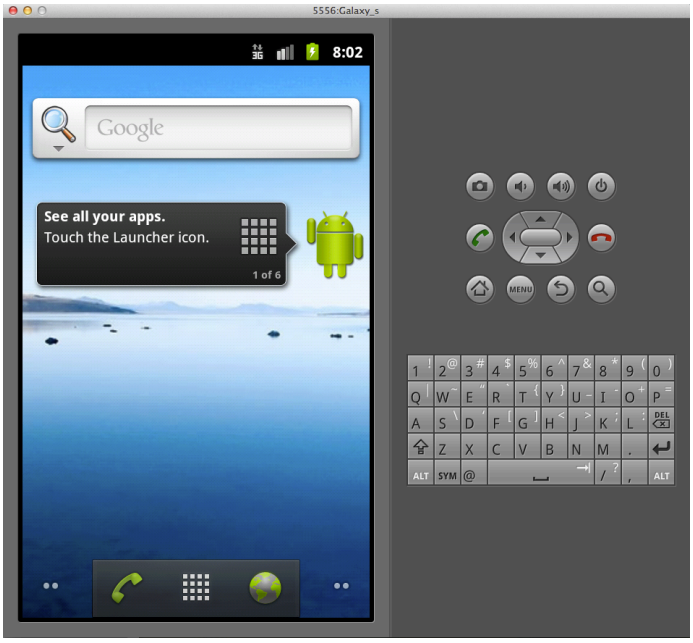
Implementing a application in Android Version 2.1 with the use of Eclipse SDK Version: 3.5.2,with the required android -sdk and android development tookit plugin ,eclipse generates .apk files for android which constitutes of buttons which are directions for the current Game (for now Need For Speed),this .apk is hard coded in the mobile,on installation a controller is setup in the phone and with connection to the computer with help of bluetooth pairing,the controller could then be used to play NFS.

It also comes with an inbuilt emulator i.e android virtual device on which we can also test our application before installing onto a real device.

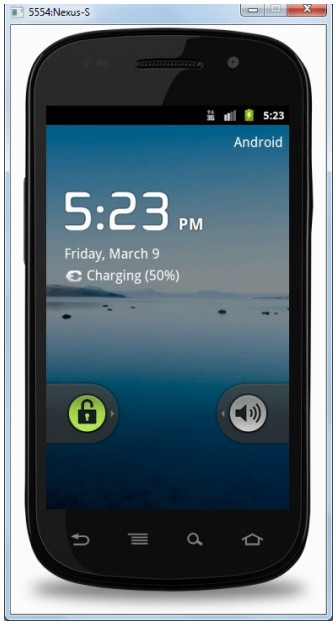
Development of Application in Eclipse



Emulators



Phone Emulator 2.3.x



Nexus - S

Tablet showing Honeycomb 3.x

5.2) Phase II

Establishment of link

We have to create Wifi hotspot in our laptop so as to turn it into a local server.
The next step is to establish a link between PC and “Gamer Droid” on mobile ,this step is one of the crucial steps.

Problems Faced

One of the serious problems with establishment of link was the application which was running fine on emulator but not running when installed in an Android phone,connections not being detected by the application and no connection established between “Gamer Droid” and the PC interface even though bluetooth device being paired.Although 2 PC's were able to interact individually but the same process for Android as Client was a problem faced.

Solution

A big problem was solved by going through the basics of coding with eclipse,the connection problem that is even though the pair up of devices was there ,no packets were exchanged and we double checked the code and finally came across with the error,the manifest of file was changed now which finally led to the connection between “Gamer Droid” and Game.

5.3) Phase III

Game control

When the link is established between Android and PC,the big question is the use of game controls to operate the game with the help of "Gamer Droid",with the help of Gamer Droid interface on the PC and various Java functions and classes ,the signal that the bluetooth link will bring would be used to see which key is pressed rather is the key pressed ,say < | | > are directions for left,up,down and right in the game (say NFS)then if in the android "Gamer Droid" interface in button right is pressed then the game would read > and hence a right direction motion for the car.

The interface of "GamerDroid" on the PC is used here and it is that interface used for recognition of buttons.

5.4) Phase IV

Implementing Buton Controls

Initially we just wanted to check that if the game can be playable wirelessly on an Android device using gamepad.

As the application starts and you connect with the PC, you have the option of playing the game using a generic gamepad.

After all the mapping process we were finally able to play NFS wirelessly through gamepad.

5.5) Phase V

Implementing motion sensing using accelerometer

The most fun part of a game is when playing the game comes intuitively to the player. We thought of having motion sensing in GamerDroid because of this idea.

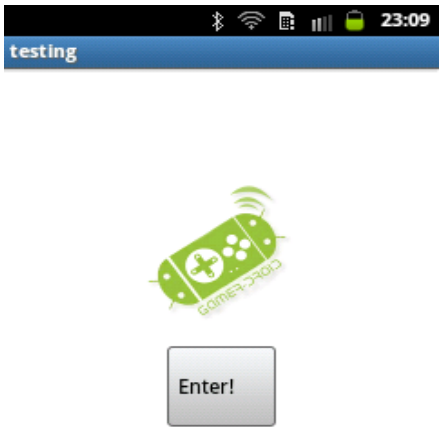
As the application starts and you connect with the PC, you have the option of playing the game using either motion sensing or a generic gamepad.

We implemented motion sensing (or more specifically, orientation sensing) using the accelerometer of Android.

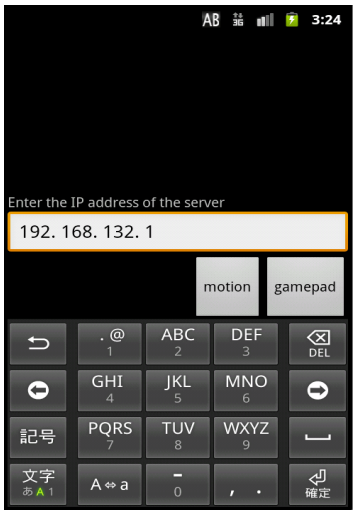
6. Features and highlights

- The techniques used in developing GamerDroid are generic, so the controls can be used for any game.
- The server side has been implemented in java, which by it's very nature is cross platform and hence, GamerDroid can be used on any Operating system.
- The structure of the application is such that it is divided into a number of independent, but communicating modules. So, the application never crashes as a whole.
- The size of the programs is small, so neither the client, nor the server side hogs any space in the device.

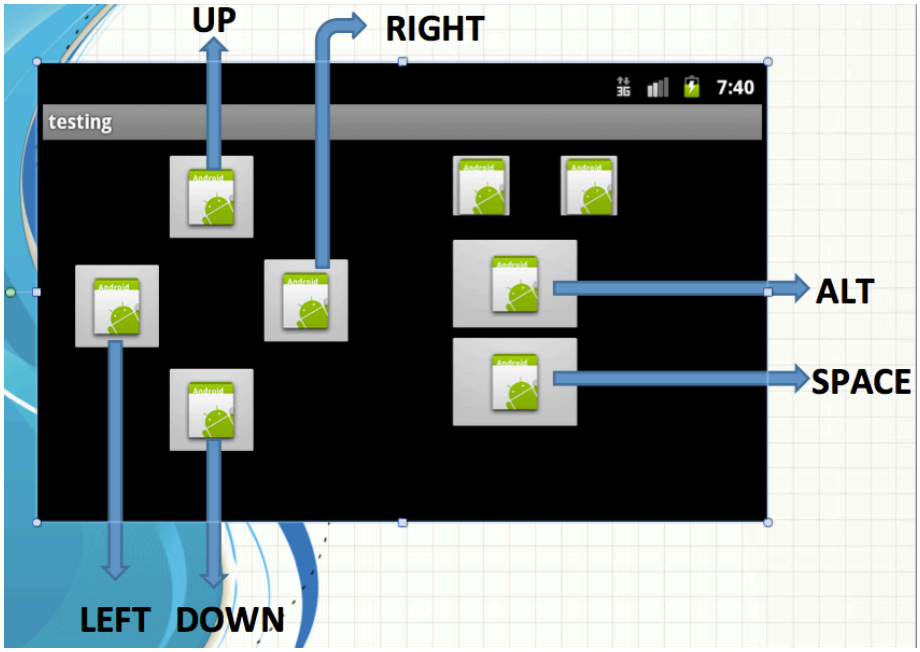
6.1) Screenshots of working application



Front Page



Entering Ip Address



Gamepad

7. Conclusion & Future Scope

- This project is a new concept and a revolution in the world of gaming, a new experience that has a wider access, as it does not require any hardware component.

- Our vision is to implement the same module for multiplayer games thus to only increase the pleasure, we wish to implement multiple Droid connections with PC and hence unique and yet parallel transfer of signals through multiple Android connected through GamerDroid.

Appendices

Details of Team Members

This project was carried out by team of 4 members, Piyush Chaudhary, Sagar Rana, Himanshi Girdhar & Abhinav Kumar Singh. All the members are from the Computer Science Engineering department (7th Semester).A great amount of effort was required by all members in order to learn and understand Android application development from scratch. All team members will work together on achieving major milestones of the project.

SDKs are named after a dessert item. Available versions at the time of writing are:

1.5 Cupcake,

1.6 Donut,

2.1 Eclair,

2.2 Froyo,

2.3 Gingerbread

3.x Honeycomb,

4.0 Ice Cream Sandwich

4.2 Jelly Bean

Android Device Analyses

For testing our application we used Samsung Galaxt SII smart phone powered by android 4.1 Ice cream Sandwich.Its specification are as follows

GENERAL	2G Network	GSM 850 / 900 / 1800 / 1900
	3G Network	HSDPA 850 / 900 / 1900 / 2100

		HSDPA 1700 / 2100 / 1900 - for Telus
	SIM	Mini-SIM
	Announced	2011, February
	Status	Available. Released 2011, April
DISPLAY	Type	Super AMOLED Plus capacitive touchscreen, 16M colors
	Size	480 x 800 pixels, 4.3 inches (~217 ppi pixel density)
	Multitouch	Yes
	Protection	Corning Gorilla Glass
		- TouchWiz UI v4.0
MEMORY	Card slot	microSD, up to 32GB
	Internal	16GB/32GB storage, 1 GB RAM
WLAN	Wi-Fi 802.11 a/b/g/n, DLNA, Wi-Fi Direct, Wi-Fi hotspot	
Bluetooth	Yes, v3.0+HS	
NFC	Optional	
USB	Yes, microUSB v2.0 (MHL), USB On-the-go	
FEATURES	OS	Android OS, v2.3.4 (Gingerbread), v4.0.4, upgradable to v4.1
	Chipset	Exynos
	CPU	Dual-core 1.2 GHz Cortex-A9
	GPU	Mali-400MP

Testing Android Applications

Testing is one of the most important phases of the application development cycle. The Android SDK provides a powerful test framework for defining and running a variety of tests to validate different aspects of Android applications. The Android test framework is built on top of the popular JUnit test framework. The Android test framework extends JUnit by incorporating Android-specific instrumentation functionality to allow tests to control the environment surrounding the Android applications. This makes it easy to test all possible use cases.

Important Terms :

Activity :

An activity is a user interface that allows the user to interact with the screen, to perform actions.

For example, a text messaging application could have an activity that displays a list of contacts to send messages. Once the contact is selected , activity could send information to a second activity that could serve to send the message to the contact.

When an application is launched, what it displays is the result of an activity. At the code level, for create an activity , you must create a class that extends the Activity class. An activity has a required inCreate () method. It is the main method. To interact with the program, through the activity, there must be something displayed, that is why the activity, contains what is called views.

View :

A View is the basic building block for user interface components. A View occupies a rectangular area on the screen. View is the base class for widgets, which are used to create interactive UI components (buttons, text fields, etc.).There is differents kinds of views, for example a ListView is able to display only an interactive list of what you want to display, while a WebView allows you to to display a web page. As said before, A view occupies a rectangular area on the screen. To organise these rectangles on the screen, there is a text file written in XML for every different screen.

Xml:

Xml means Extensible Markup Language. Android provides a straightforward XML vocabulary that corresponds to the View classes and subclasses. The goal of using Android's XML vocabulary, is to quickly design UI layouts and the screen elements they contain, in the same way that creating web pages in HTML with a series of nested elements.

Here is an example :

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout_width="fill_parent"

android:layout_height="fill_parent"

android:orientation="vertical" >

    <TextView android:id="@+id/text"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:text="Hello, I am a TextView" />

    <Button android:id="@+id/button"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:text="Hello, I am a Button" />

</LinearLayout>
```

In the previous example, here's an XML layout that uses a vertical LinearLayout to hold a TextView and a Button.It will be possible to modify attributes the elements in the activity class that use this XML file.

Intent :

An activity can of course start another one, even if it but to do this, it will need a special object called Intent. An intent is basic2 description of an operation to be performed. It can launch an Activity, send a broadcastIntent to any interested BroadcastReceiver components, and communicate with a background Service. An Intent performs binding between the code in different applications. it can be thought of as the link between activities. It is possible to add some informations to an Intent, thanks to an object called bundle, that you add to the intent thanks to the method :

Intent.putExtras(Bundle objetbunble);

Android Manifest :

AndroidManifest.xml file is necessary for all android applications and must have this name in its root directory. In the manifest you can find essential informations about the application for the Android system, informations that the system must have before it can run any of the application's code. Here is what you can find in the Android manifest:

-The name of the Java package for the application. The package name serves as a unique identifier for the application.

-The description of the components of the application : the activities, services, broadcast receivers, and content providers that the application is composed of and under what conditions they can be launched .

-The processes that will host application components.

-The permissions the application must have in order to access protected parts of the API and interact with other applications.

-The permissions that others are required to have in order to interact with the application's components.

-The list of the Instrumentation classes that provide profiling and other information as the application is running. These declarations are present in the manifest only while the application is being developed and tested; they're removed before the application is published.

-The minimum level of the Android API that the application requires.

-The list of the libraries that the application must be linked against

References

- Android. (n.d.). Retrieved on September 30, 2012, from Android web site:
- http://www.Android.com/.
- Android Market. (n.d.). Retrieved on September 30, 2012, from Android
- http://www.android.com/market/.
- Android Market and iPhone App Store Compared. (2009, October 24). Retrieved October 30, 2012, from Android central web site: http://www.Androidcentral.com/Android-market-iPhone-app-store-compared.
- Android Overview. (n.d.). Retrieved on September 16, 2009 from openhandsetalliance web site: http://www.openhandsetalliance.com/Android_overview.html.
- http://developer.android.com/
- Head First Android Development : O'Reilly Textbook.

For socket Programming:

- http://www.anddev.org/socket_programming-t325-s30.html
- http://developer.android.com/reference/java/net/Socket.html
- http://www.javaworld.com/jw-12-1996/jw-12-sockets.html

Glossary

The list below defines some of the basic terminology of the Android platform.

.apk file

Android application package file. Each Android application is compiled and packaged in a single file that includes all of the application's code (.dex files), resources, assets, and manifest file. The application package file can have any name but must use the .apk extension. For example: myExampleAppname.apk. For convenience, an application package file is often referred to as an ".apk".

Related: Application.

Action

A description of something that an Intent sender wants done. An action is a string value assigned to an Intent. Action strings can be defined by Android or by a third-party developer. For example, android.intent.action.VIEW for a Web URL, or com.example.rumbler.SHAKE_PHONE for a custom application to vibrate the phone.

Related: Intent.

Activity

A single screen in an application, with supporting Java code, derived from the Activity class. Most commonly, an activity is visibly represented by a full screen window that can receive and handle UI events and perform complex tasks, because of the Window it uses to render its window. Though an Activity is typically full screen, it can also be floating or transparent.

adb

Android Debug Bridge, a command-line debugging application included with the SDK. It provides tools to browse the device, copy tools on the device, and forward ports for debugging. If you are developing in Eclipse using the ADT Plugin, adb is integrated into your development environment. See Android Debug Bridge for more information.

Application

From a component perspective, an Android application consists of one or more activities, services, listeners, and intent receivers. From a source file perspective, an Android application consists of code, resources, assets, and a single manifest. During compilation, these files are packaged in a single file called an application package file (.apk).

Related: .apk, Activity

Canvas

A drawing surface that handles compositing of the actual bits against a Bitmap or Surface object. It has methods for standard computer drawing of bitmaps, lines, circles, rectangles, text, and so on, and is bound to a Bitmap or Surface. Canvas is the simplest, easiest way to draw 2D objects on the screen. However, it does not support hardware acceleration, as OpenGL ES does. The base class is Canvas.

Related: Drawable, OpenGL ES.

Content Provider

A data-abstraction layer that you can use to safely expose your application's data to other applications. A content provider is built on the ContentProvider class, which handles content query strings of a specific format to return data in a specific format. See Content Providers topic for more information.

Related: URI Usage in Android

Dalvik

The Android platform's virtual machine. The Dalvik VM is an interpreter-only virtual machine that executes files in the Dalvik Executable (.dex) format, a format that is optimized for efficient storage and memory-mappable execution. The virtual machine is register-based, and it can run classes compiled by a Java language compiler that have been transformed into its native format using the included "dx" tool. The VM runs on top of Posix-compliant operating systems, which it relies on for underlying functionality (such as threading and low level memory management). The Dalvik core class library is intended to provide a familiar development base for those used to programming with Java Standard Edition, but it is geared specifically to the needs of a small mobile device.

DDMS

Dalvik Debug Monitor Service, a GUI debugging application included with the SDK. It provides screen capture, log dump, and process examination capabilities. If you are developing in Eclipse using the ADT Plugin, DDMS is integrated into your development environment. See Using DDMS to learn more about the program.

Dialog

A floating window that that acts as a lightweight form. A dialog can have button controls only and is intended to perform a simple action (such as button choice) and perhaps return a value. A dialog is not intended to persist in the history stack, contain complex layout, or perform complex actions. Android provides a default simple dialog for you with optional buttons, though you can define your own dialog layout. The base class for dialogs is Dialog.

Related: Activity.

Drawable

A compiled visual resource that can be used as a background, title, or other part of the screen. A drawable is typically loaded into another UI element, for example as a background image. A drawable is not able to receive events, but does assign various other properties such as "state" and scheduling, to enable subclasses such as animation objects or image libraries. Many drawable objects are loaded from drawable resource files — xml or bitmap files that describe the image. Drawable resources are compiled into subclasses of android.graphics.drawable. For more information about drawables and other resources, see Resources.

Related: Resources, Canvas

Intent

An message object that you can use to launch or communicate with other applications/activities asynchronously. An Intent object is an instance of Intent. It includes several criteria fields that you can supply, to determine what application/activity receives the Intent and what the receiver does when handling the Intent. Available criteria include include the desired action, a category, a data string, the MIME type of the data, a handling class, and others. An application sends an Intent to the Android system, rather than sending it directly to another application/activity. The application can send the Intent to a single target application or it can send it as a broadcast, which can in turn be handled by multiple applications sequentially. The Android system is responsible for resolving the best-available receiver for each Intent, based on the criteria supplied in the Intent and the Intent Filters defined by other applications. For more information, see Intents and Intent Filters.

Related: Intent Filter, Broadcast Receiver.

Intent Filter

A filter object that an application declares in its manifest file, to tell the system what types of Intents each of its components is willing to accept and with what criteria.

Through an intent filter, an application can express interest in specific data types, Intent actions, URI formats, and so on. When resolving an Intent, the system evaluates all of the available intent filters in all applications and passes the Intent to the application/activity that best matches the Intent and criteria. For more information, see Intents and Intent Filters.

Related: Intent, Broadcast Receiver.

Broadcast Receiver

An application class that listens for Intents that are broadcast, rather than being sent to a single target application/activity. The system delivers a broadcast Intent to all interested broadcast receivers, which handle the Intent sequentially.

Related: Intent, Intent Filter.

Layout Resource

An XML file that describes the layout of an Activity screen.

Related: Resources

Manifest File

An XML file that each application must define, to describe the application's package name, version, components (activities, intent filters, services), imported libraries, and describes the various activities, and so on. See The AndroidManifest.xml File for complete information.

Nine-patch / 9-patch / Ninepatch image

A resizeable bitmap resource that can be used for backgrounds or other images on the device. See Nine-Patch Stretchable Image for more information.

Related: Resources.

OpenGL ES

Android provides OpenGL ES libraries that you can use for fast, complex 3D images. It is harder to use than a Canvas object, but better for 3D objects. The android.opengl and javax.microedition.khronos.opengles packages expose OpenGL ES functionality.

Related: Canvas, Surface

Resources

Nonprogrammatic application components that are external to the compiled application code, but which can be loaded from application code using a well-known reference format. Android supports a variety of resource types, but a typical application's resources would consist of UI strings, UI layout components, graphics or other media files, and so on. An application uses resources to efficiently support localization and varied device profiles and states. For example, an application would include a separate set of resources for each supported local or device type, and it could include layout resources that are specific to the current screen orientation (landscape or portrait). For more information about resources, see Resources and Assets. The resources of an application are always stored in the res/* subfolders of the project.

Service

An object of class Service that runs in the background (without any UI presence) to perform various persistent actions, such as playing music or monitoring network activity.

Related: Activity

Surface

An object of type Surface representing a block of memory that gets composited to the screen. A Surface holds a Canvas object for drawing, and provides various helper methods to draw layers and resize the surface. You should not use this class directly; use SurfaceView instead.

Related: Canvas

SurfaceView

A View object that wraps a Surface for drawing, and exposes methods to specify its size and format dynamically. A SurfaceView provides a way to draw independently of the UI thread for resource-intensive operations (such as games or camera previews), but it uses extra memory as a result. SurfaceView supports both Canvas and OpenGL ES graphics. The base class is SurfaceView.

Related: Surface

Theme

A set of properties (text size, background color, and so on) bundled together to define various default display settings. Android provides a few standard themes, listed in R.style (starting with "Theme_").

URIs in Android

Android uses URI strings as the basis for requesting data in a content provider (such as to retrieve a list of contacts) and for requesting actions in an Intent (such as opening a Web page in a browser). The URI scheme and format is specialized according to the type of use, and an application can handle specific URI schemes and strings in any way it wants. Some URI schemes are reserved by system components. For example, requests for data from a content provider must use the content://. In an Intent, a URI using an http:// scheme will be handled by the browser.

View

An object that draws to a rectangular area on the screen and handles click, keystroke, and other interaction events. A View is a base class for most layout components of an Activity or Dialog screen (text boxes, windows, and so on). It receives calls from its parent object (see viewgroup, below)to draw itself, and informs its parent object about

where and how big it would like to be (which may or may not be respected by the parent). For more information, see View.

Related: Viewgroup, Widget

Viewgroup

A container object that groups a set of child Views. The viewgroup is responsible for deciding where child views are positioned and how large they can be, as well as for calling each to draw itself when appropriate. Some viewgroups are invisible and are for layout only, while others have an intrinsic UI (for instance, a scrolling list box). Viewgroups are all in the widget package, but extend ViewGroup.

Related: View

Widget

One of a set of fully implemented View subclasses that render form elements and other UI components, such as a text box or popup menu. Because a widget is fully implemented, it handles measuring and drawing itself and responding to screen events. Widgets are all in the android.widget package.

Window

In an Android application, an object derived from the abstract class Window that specifies the elements of a generic window, such as the look and feel (title bar text, location and content of menus, and so on). Dialog and Activity use an implementation of this class to render a window. You do not need to implement this class or use windows in your application.