Kingston University London

DEVELOPING OF A MULTIMEDIA GUIDE FOR TECHNOLOGICAL EDUCATIONAL INSTITUTE OF PIRAEUS USING THE ANDROID PLATFORM

IOANNIS KAVVALOS

Master of Science in Networking and Data Communications

THESIS

Kingston University London

Thesis Title
Developing of a Multimedia Guide for Technological Educational Institute of Piraeus using the Android Platform

Dissertation submitted
for the Degree of Master of Science in Networking and Data Communications

By
IOANNIS KAVVALOS

SUPERVISOR
DR. DIONISIS ADAMOPOULOS

KINGSTON UNIVERSITY, FACULTY OF COMPUTING, INFORMATION SYSTEMS & MATHEMATICS
TEI OF PIRAEUS, DEPARTMENTS OF ELECTRONICS AND AUTOMATION

JULY 2011

ACKNOWLEDGMENTS

I would like to thank my family and my friends for all of their encouragement and support.

TABLE OF CONTENTS

Abstract

This dissertation aims to develop a multimedia application that contains information about the Technological Educational Institute of Piraeus. The application was developed using the Android Platform and it was designed mainly in order to be a tool for all the undergraduate students of the educational institute. The application is useful for students so as to learn all the news about their lessons. Also, it contains information for each one of the departments. The code of the application was developed using the Eclipse SDK and the Android SDK with the AVD Manager. There are also some extra features in the main application such as the option to search a book directly from the database of the library.

1. Introduction

Android is a mobile operating system, similar to Symbian, iOS, Windows® Mobile. It was initially developed by Android Inc which was bought by Google in 2005. Google released the main part of the Android code under the Apache License and with this licence vendors can add proprietary extensions without submitting them back to the open source community. Android uses a modified Linux® kernel and allows applications to be developed in Java™ technology using Java libraries. While Android applications are written in the Java language, there's no Java Virtual Machine in the platform, and Java byte code is not executed. For developers, Android SDK provides a rich set of tools, including debugger, libraries, handset emulator, documentation, sample code, and tutorials. Android applications can be easily developed using Eclipse (Android's official development platform) with the help of a plug-in called Android Development Tools (ADT).

1.2 The Android history

The Android operating system was created in response to commercial versions of operating systems for mobile devices. The Android is a modern

operating system for mobile devices that based in the core of the Linux operating system. Originally, it developed by Google and later by the OpenHandset Alliance. It allows developers to create a code by using the programming language Java and controlling the devices through software libraries developed by Google.

In July 2005, Google acquired Android Inc. , a small company based in Palo Alto, California, USA. The co-founders of Android went to work at Google, including Andy Rubin (co-founder of Danger), Rich Miner (co-founder of Wildfire Communications, Inc), Nick Sears (former vice president of T-Mobile), and Chris White (head design and development interface in WebTV). At that time, little was known about the functions of Android Inc. except that it made software for mobile phones. This information created rumors that Google would plan to construct products for the mobile market.

Figure1.1

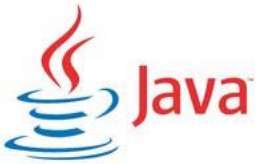

Figure 1.2

In Google, the team led by Rubin, who developed a mobile platform based on Linux kernel, promoted by providing a flexible and upgradeable system. It is reported that Google had already amassed a range of partners' hardware and software providers and noted that it was open to varying degrees of cooperation. More theories that Google would enter the mobile market started in December 2006. Publications from the BBC and The Wall Street Journal informed that Google wanted its searches and applications to mobile phones to develop as fast as possible. Print and electronic media soon reported rumors that Google developed a Google-branded device. More theories followed reporting that Google had laid down the technical specifications and prototypes seemed to mobile phone manufacturers and Network operators.

On September 2007, the InformationWeek covered an evaluation study indicating that Google had filed several patents in the field of mobile telephony. Finally Google launched the Nexus One, a smartphone that uses the Android open source operating system. The device was manufactured by HTC Corporation of Taiwan, and was released on January 5, 2010.

The Android operating system has been accepted several updates since its initial release. These changes update the core operating system, correct errors and add new features.

1.5 (Cupcake) based in the Linux Kernel 2.6.27	On April 30, 2009, released the official 1.5 (Cupcake) update for Android operating system. There are several new features and UI updates which included in the 1.5 update: 1. The ability to record and play video 2. Uploading videos on Youtube and pictures in Picasa directly from the phone 3. A new keyboard with "autocomplete" characteristic 4. Support of BluetoothA2DP (which made available the Bluetooth connectivity with many popular cars and headphones) 5. Ability to automatically connect to a Bluetooth headset within a certain distance 6. News widgets and folders can be placed on the desktop 7. Animations between screens 8. Expanded ability to copy and paste including websites
1.6 (Donut) based in the Linux Kernel 2.6.29	On September 15, 2009, Google released the 1.6 (Donut) SDK which included the following updates: 1. An enhanced experience of Android Market 2. A built-in camera, camcorder and photo albums

	3. The photo album now allows users to select multiple photos for delete. 4. Updated Voice Search, a faster response and deeper integration with native applications, including the ability to make calls to the contacts 5. Update search experience to allow the search to your bookmarks, history, contacts and the Internet from main screen. 6. Information technology support for CDMA / EVDO,802.1x, VPN, Gestures and a Text-to-speech sevice 7. WVGA support 8. Improved search engine and camcorder
2.0/2.1 (Eclair) based in the Linux Kernel 2.6.29	On October 26, 2009, Google released the 2.0 (Eclair) SDK released. Among the changes are the followings: 1. Optimizing the speed of hardware 2. Support for several screen sizes and resolutions 3. Updated UI 4. New browser UI and HTML5 support 5. New contact lists 6. Best white / black for the background 7. Improved Google Maps 3.1.2 8. Microsoft Exchange Support 9. Integrated supports for camera flash 10. Digital zoom 11. The MotionEvent optimized for monitors multi-touch events 12. Improved virtual keyboard 13. Bluetooth 2.1 14. Live wallpapers

2.2.2 (FroYo) Based on Linux Kernel 2.6.32	1. System: Speed, memory, and performance optimizations 2. Additional application speed improvements courtesy of JIT implementation 3. Integration of Chrome's V8 JavaScript engine into the Browser application 4. Improved Microsoft Exchange support (security policies, auto-discovery, GAL look-up, calendar synchronization, remote wipe) 5. Improved application launcher with shortcuts to Phone and Browser applications 6. USB tethering and Wi-Fi hotspot functionality Added an option to disable data access over mobile network 7. Updated Market application with batch and automatic update features 8. Quick switching between multiple keyboard languages and their dictionaries 9. Voice dialing and contact sharing over Bluetooth 10. Support for numeric and alphanumeric passwords 11. Support for file upload fields in the Browser application 12. Support for installing applications to the expandable memory 13. Adobe Flash support 14. Support for extra high DPI screens (320 dpi), such as 4" 720p

1.3 The Android platform

The following chapter refers to the main features of the Android platform. This chapter explains why Android is so famous and represents the basic reasons that make Android an easy platform for developing mobile applications. When

someone develops an android application can publish it in the Android Market and the application will be available for everyone around the world. The programming environment based on an open-source platform and the hardware specifications that required are accessible to everyone when someone is developing an application.

The Android operating system is a new platform in the global market, but the appeal to the general public and developers communities is high and this is shown by the multiple of the applications that are created every day. The Android Market publishes Android applications in one place and it is available for everyone all over the world who uses a simple web explorer (Google Chrome, Mozilla Firefox etc). In addition, Android Market is pre-installed on every Android device and users have access to the applications instantly. This feature is one of the most important reasons that make popular every application in a short period of time. In Android Market, every user can easily search and find applications that are useful or he just wants to have them in his device. In essence, the user can find an application by describing succinctly a few words about the application or by typing exactly the name of the application. Then, the device downloads and setups the application automatically. Another way is to use the menu that separates applications based on their content.

One of the main characteristics of the Android operating system is that the platform is open-source. Each hardware manufacturer and each provider may amend the code of Android to be compatible with their own features. The source code of the Android is available on the Internet and can be accessed by anyone. This feature gives equal opportunities to all developers. This means that the applications that go with the Android platform do not outweigh in anything compared to those which are realized by a developer who has access to the components of the device. Furthermore, the Android platform overmatches because it has the possibility of multi-tasking and that means that multiple applications run simultaneously, while it is not required to close any application if the user chooses to exit because the application continues to run in the background.

The Android applications can run on different devices with different characteristics. Specifically, an application can run on a device with small screen and low resolution but also can be performed in a device with big screen and high resolution. Irrespective of their status which called cross-compatible, the Android platform provides the tools which are necessary for the developers in order to build applications that run only on compatible devices. For example, if an application requires a camera in front of the phone, then the application cannot run on a phone that does not have a camera. This arrangement is known as the “feature detection”. Generally, devices that are compatible with the Android operating system have a camera, GPS, blue-tooth and compass.

An important fact which favors the development of applications is that the Android SDK package works with eclipse. So the developer can quickly and easily see through every change in the code on the emulator which is provided by the Android SDK, without having to export each time the application and to install it in a mobile device. Also, the emulator is very reliable and has exactly the same behavior with the situation that the application has been installed on an Android mobile phone. Finally, another important advantage is the fact that the software application upgrades still work without the need to redesign some of the key parts of the code for interaction with the application components of the mobile phone hardware.

An important feature of the Android operating system is the fact that they can be combined two or more services in order to compose one application. This feature is called mashup. For example, it can be combined a service that enables the camera with the service that activates the GPS. The result is the creation of an image which contains the coordinates of the place that was captured. Another example of a mashup is an application that combines the geolocation service with a social networking service. These two services produce an application, which can publish every ten minutes the place of the device in a social networking website.

1.4 Basic parts of Android platform

The default programming language of Android operating system is Java and small parts of the framework include the XML language. These are quite common languages and therefore they are accessible to a large number of developers who want to participate in the development of software. The Android operating system is based on the Linux kernel but it is not a Linux. The kernel acts as an abstraction layer between the hardware and the rest of the software stack. The libraries of the Android operating system include a set of huge C / C + + libraries that are used by various components of the system. These are available to developers through the Android application framework.

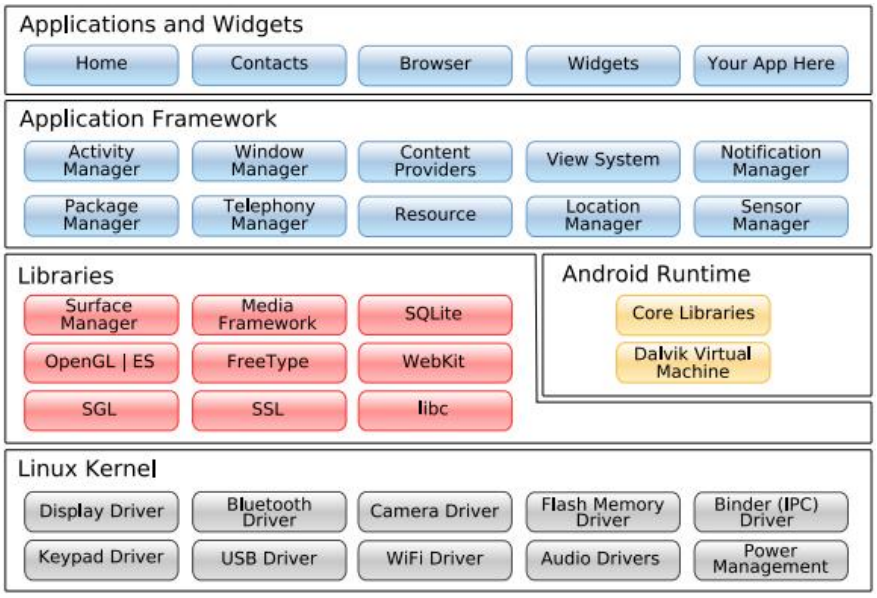

Figure 1.3 : The Android Platform

An Android application has a different philosophy in its structure compared with conventional structures. Usually there is an executable file that contains the program logic using external resources, i.e. local files or images, or texts to be retrieved from the network, etc. The Android application is not an executable file, but as an application it is meant to be a group of files that represents the Activities of the application. These can be accessed by other applications which are written from other developers.

As it was mentioned before, one of the most popular programming languages for application development of the Android operating system is the programming language Java. An Android package includes data files and resources. These two parts are linked with the compiled Java code using the tool aapt and finally they produce a file with extension “apk”. An “apk” file can be installed on Android and can be distributed via Internet (Websites, Android Market). In essence, the “apk” file is a standalone application.

Each application runs in its own separate "box" (sandbox):

a. Each application runs in accordance to the process architecture of Linux. The Android operating system launches the commands when any piece of the code of the application requires execution and ends when it is no longer useful. One of the facts that it is very important is that the Android operating system terminates the application when system resources, that were committed, are required by another application.

b. Each process runs autonomously within a VM (Virtual Machine), so it is isolated from the source of all other applications.

c. Each application has a unique user ID (Linux User ID). Each application uses rights, which are configured so that the files are visible only to the specific user and application. But there are ways that the rights can be shared and can be used by other applications.

1.5 Application’s pieces

One of the main features of Android is that an application can use the data to another application (provided that it is allowed). For example, if our application needs to display a scrolling list of images and another application has implemented such a list, which is also available in other applications, can be called instead to

implement a new one. The application does not incorporate our code with another application, or connected with him. Simply start the execution of this code whenever is necessary. For this reason, the system must be able to start a process of the application wherever it is required by any part of it. So, unlike most systems, Android applications have a unique entry point - implementation (eg main () function). In contrast there are the main ingredients (components) which the system can implement and execute whenever it is necessary. There are four basic types of components:

1.5.1 Activities

An Activity represents a structured graphical interaction with the user. For example, an Activity can display a list of items in order to choose one of them or it presents photographs with their titles. An application for sending text messages can have an activity in order to highlight the contacts, a second activity for writing the message and other activities to display old messages or change the settings. All the activities work together and they coordinated by a single graphical interface. An important feature is that each one of them is independent and is implemented as a subclass of the base class Activity. An application may consist of a single activity or many activities, as in the example of text messaging. There are many activities which depend on the design of the application. Basically, the activities that are described as the first one will be shown to the user when the application starts. The transition from one activity to another is accomplished by a boot to the next activity. A main window displays each activity. Usually, this window covers the full screen but it can also be smaller than the others. An activity may also use additional windows, such as a pop-up dialog box or a window. Usually, pop-up dial boxes request a response from the user through the operation of an activity and the windows inform the user about his actions. A hierarchy of visual objects (views) provides the visual content of a window. These are derived from the base class View. Each View controls a specific rectangular area within the window. The parents' views (if we consider the hierarchy as a tree) contain and organize the formation of their children. The latest views of the classification plan control

rectangular regions and when something happens in this space, they respond to users' actions. In addition, the views exist only during an activity, which takes place at the user interface. For example, a view can display a small image and starts an action when the user selects the image. The Android contains a set of ready-views which we can use - including buttons, text fields, scroll bars, menu items, check boxes and others. A hierarchy of views place in the window one activity with the method Activity.setContentView (). The content view is an object of type View at the top (root) of the hierarchy.

1.5.2 Services

A service has no Graphical User Interface, but it runs in the background for an indefinite period. For example, a service can play background music as the user is doing other things such as transfering data over a network or making calculations in order to provide results in required activities. Each service extends the base class Service. A common example is a media player that plays songs selected from a list. The implementation of the player might have one or more of activities which enable the user to select in order to listen to a song. However, the music service cannot be handled by an activity, since users will expect that the music must continue to play, even when they let the player, in order to start dealing with a different activity. In order to continue playing music, the activity of the media player must place a running service in the background. Furthermore, the system will allow the service to run even it is close to the triggered activity. It is possible to connect or group a service. When the services are connected we can communicate with them via an interface. For example, the music service interface may allow the user to stop, change or play the song from the beginning. The services "run" in the main thread of the process of our application like the activities and all the other components of Android platform. Finally they often use other threads so as to perform time consuming operations. This feature is very important in order not to exclude the operation of the other components or interface with the user.

1.5.3 Broadcast Receivers

A broadcast receiver is a component that receives and responds to communications - messages (broadcast announcements). Many of them come from the same code system, for example: when users change the time, when they get the battery level, when someone takes a picture or when they change the system language. Applications can also create such broadcast messages in order to warn the user, for example: when some data is "downloaded" to the device and it is available for use. An application can have a large amount of broadcast receivers in order to respond to every important communication. All broadcast receivers extend the base class BroadcastReceiver. The broadcast receivers do not have a graphical interface. Nevertheless, one broadcast receiver can start an activity in response to information received or it can use the NotificationManager in order to alert the user. Alerts (notifications) may distract the user in many ways such as flashing the backlight, using the vibration device, playing sounds, etc. Usually, there is an icon in the status bar which provides extra information.

1.5.4 Content providers

A content provider offers the possibility to make available an application specific data to other applications. This data can be stored in the filesystem or in a database SQLite, or anywhere else. Each content provider extends the class ContentProvider in order to implement a basic set of methods. These methods allow other applications to retrieve and store data. Usually, these methods are called directly, but indirect they are called through the object ContentResolver. This object (ContentResolver) can communicate with any content provider and it can cooperate to manage this communication. Android ensures that the process of component is "running", whenever there is a request to be handled by a specific part. Otherwise, the appropriate instance of the component is available during the boot.

1.5.5 Activating components: Intents

The Content Providers are activated when a ContentResolver sends them a request. The other three components - activities, services and broadcast receivers are triggered by asynchronous messages called intents. The intent is an object of Intent type which includes the content of the message. On activities and services is required an action in order to run and specify the URI of the data and as a result to act together with other actions. For example, the intent may request a transfer of an activity in order to display an image to the user or to allow him to edit a text. In broadcast receivers, the Intent object identifies the announcement of the action. For example, the intent can inform the user that the camera button has pushed. There are three different methods of activating each type of component:

The first method is: An Activity is switched (or assigned to perform something new) on by passing an Intent object using the method of Context.startActivity () or Activity.startActivityForResult (). The Activity monitor investigates the original intent that is triggered by calling the method getIntent (). The method calls Android onNewIntent () of activity in order to pass the subsequent intents. Typically, an activity causes the next start. If the first procedure “waiting for the results” is returned from the second, the system calls the method startActivityForResult () instead of startActivity (). For example, if you want to start an activity that allows a user to choose a photo, the user will wait until the system return him the selected photo. The result is a subject of Intents which passed to the method onActivityResult () when a user uses an activity.

The second method is: A service starts (or gives new orders) through an Intent object to Context.startService (). The method calls the Android service onStart () and transmits the Intent object. Similarly, an object of type Intent may be the method Context.bindService () to establish a permanent connection between the called and the service component that called it. The service receives the Intent object by calling the method onBind () (if the service is not running already, optional method bindService () can start). For example, an activity can establish a

connection with a (playback) service that plays music to provide the user the means (eg an interface - gui) in order to handle the playback. This activity can invite the bindService () to install the connection and then calls are offered by the service methods.

The third method is: An application can initiate a broadcast by sending an object Intent on methods within the object using the type Context. SendBroadcast (),. SendOrderedBroadcast () and. SendStickyBroadcast () in any version of the SDK. The Android platform sends the intent to all the interested broadcast receivers using the method onReceive ().

1.5.6. Disabling Components

A content provider is still turned on until it responds to a request from a ContentResolver and a broadcast receiver so as to respond to a broadcast message. Therefore, there is no need to explicitly disable these components. On the other hand, the activities provide the user interface. These activities interact with the user for a long time and can stay active, even if they are idle. Similarly, the services may remain active for a long time too. So, Android platform contains methods to disable the activities and services in a methodical way:

1) An activity can be disabled by calling the method finish (). Furthermore, an activity can disable another by calling the method finishActivity ().

2) A service is terminated by calling the method stopSelf () or by calling the method Context.stopService (). The components can also be terminated by the system if it is no longer in use or when the Android platform must obtain memory for more active components.

1.6 The file Manifest

Before the Android launches an ingredient it provides a control test in order to realize if that component already exists. This is the reason why the applications define the components in a manifest file which are included in the Android package APK and they also contain executable code and data. This file is a structured XML file and has the same name for all the existing applications, AndroidManifest.xml. Apart from the statement of components, the AndroidManifest performs other functions such as naming libraries that we will need to apply (except for the default Android library) and identify all these rights which are allowed by the application.

The main task of the manifest is to show the Android operating system the parts (components) of the application. For example:

```
Xml version = "1.0" encoding = "utf-8"?>
<?<Manifest...>
<Application..
<Activity
. Activity>.
</
.>. .
</
</ Application>manifest>
```

The field name of the item <activity> contains the name of the subclass that implements the Activity. The icon and the label field indicate the resource files containing an image and a label and it represents the activity. The remaining components are defined in a similar way - <service> information on services, <receiver> data for broadcast receivers and <provider> data for content providers. The activities, the services and content providers that are not declared on the manifest file will not be visible to the system and therefore do not always run. However, the broadcast receivers can either be declared in the manifest file or can

be created dynamically in code (as objects BroadcastReceivers) and recorded by the system by calling the method Context.registerReceiver ().

2. MULTIMEDIA DEVELOPING

2.1 User Interface (UI)

The user interface, in an Android application, created using objects View and ViewGroup. There are many types of views and viewgroups. Each of them is a descendant of the Class View. View objects are the basic building blocks of user interface of the Android platform. The Class View is a basis for subclasses and it is usually called "widget". Class View offers fully delineated UI objects, such as text fields and buttons. Heading ViewGroup runs as a base for subclasses called "layouts", which provide different kinds of architectural layout, such as linear, tabular and relative. A View object is a data structure. It has attributes which store the parameters of the layout and the content of a given rectangular region of the screen. A View object handles the size, layout, design, focus, the scrolling, and interactions buttons / gestures for the rectangular display area in which it is located. The View object is also a point of interaction for the user and receiver of interaction events when it is developed as a UI object.

2.2 Hierarchy View

The platform Android defines the user interface (UI) as an Activity using a hierarchy of View nodes and ViewGroup. This hierarchical tree can be simple or complex. Developers can create an hierarchical tree either by using the predefined widgets and layouts of Android or with their own custom Views. When the hierarchical tree of Views is connected to the screen with the method that is referred as setContentView we can make the graphics performance (rendering) and pass by reference to the root node of the tree. The Android system receives this report and uses it to invalidate, calculate and design the tree. The node father-hierarchy asks his children-nodes to be designed – respectively. Each node view group is

responsible for applying to their children (nodes view) to be designed. Children nodes may require a size and a place in their father-hierarchy but only the father takes the final decision on how much space it occupies each child. Android platform analyzes the data layout of our series (from the top of the hierarchical tree).It can also create Views and adds them to their parents.

2.3 Layout

The most common way to define the layout is through a XML file layout. The XML provides a humanly comprehensible structure for the layout, such as the HTML. Each element in the XML object is either View or ViewGroup. The View objects are the leaves of the tree and the objects ViewGroup are the branches. The name of an XML element is equivalent to the Java class that represents. <TextView> creates an element TextView in the UI and <LinearLayout> creates an element called LinearLayout view group. When you load a layout resource, the Android system initializes these objects (created in runtime) which are based on elements of our layout. For example, a simple vertical layout code with a text view and a button would be similar to the code that is given below:

```
<?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 this example, the LinearLayout contains a TextView and a Button. We can put inside the code and another LinearLayout (or other type of view group) in order to prolong the hierarchy view and therefore create thereby a more complex layout. We can also design objects View and ViewGroups based on Java code, by using the methods addView (View) to dynamically assign new objects View and ViewGroup. There are many ways that we can create the layout of our view. Using more and different type of view groups, fields can construct views and view groups in infinite ways. Some predefined view groups that provide the Android (called layouts) include the LinearLayout, the RelativeLayout, the TableLayout, the GridLayout and others. Each one of them offers a unique set of parameters that are used to determine the location of fields and views in the structure of the layout.

2.4 Widgets

A widget is a View object that acts as an interface for interaction with the user. The Android provides a fully delineated set of widgets, like buttons, checkboxes and text-entry fields, so we can quickly build our UI. Some widgets provided by Android are more complex, like the date picker, the clock and the zoom controls. If we want to do something more sophisticated, we can create our own widgets by extending and combining existing widgets. The package android.widget contains a list of widgets provided by Android.

2.5 UI Events

Once we add some Views / widgets in the UI, we want to know the user interaction with them, so we can execute some actions. For information on the UI events, we need to implement one of the following:

1) First of all, we must define an event listener and we enter it in the View. The View class contains a collection of interfaces called On <something> Listener,

each one with a callback method called On <something> (). For example, View.OnClickListener (to handle "clicks" in a View), View.OnTouchListener (for handling touch screen events in a View), and View.OnKeyListener (for handling the touch buttons in a View). So if we want the View to be informed when it is "clicked" (ie when a button is selected), we must realize the OnClickListener and set the onClick () callback method (which will execute the energy for a click), and register it in through the View setOnClickListener ().

2) We can override an existing callback method for the View. For example, we can confront events by touching the screen, (onTouchEvent ()), when the trackball moves (onTrackballEvent ()), or when a button is pressed in the device (onKeyDown ()). This allows us to define the default behavior for each event in our View and to identify when an event should be transferred to another child View. These are callbacks inside a Class View, so the only chance to define them is when we are creating a custom component.

2.6 Menu

The application menu is another important part of the User Interface application. Menus offer a reliable interface that brings out the functions and application settings. The most common application menu appears when you press the MENU button on the device. However, we can add Context Menus, which appear when the user selects and holds an item. The menu is also constructed through an hierarchy of View. On the other hand, we define the callback method onCreateOptionsMenu () or callback method onCreateContextMenu () for our Activity and declare objects that we want to include in our menu. In the first case, the Android will automatically generate the required hierarchy for the View menu and will plan every object which is included in it. Menus handle their own events, so developers do not need to register event listeners for the items menu. When an object is selected, the method onOptionsItemSelected () or the method onContextItemSelected () will be invited from the environment.

2.7 Statement Layout

The Layout is the architecture of the user interface to an Activity. The Activity defines the structure of the layout and contains all the items displayed to the user. We can define the layout with the following ways:

1. We can declare User Interface elements in XML. The Android provides a simple XML vocabulary that corresponds to the classes and subclasses of View, such as those on the widgets and layouts. Then we can create a data layout in runtime. The Android framework gives us the flexibility to use one of these two methods or both of them in order to declare and manage the UI of our application. For example, we can declare the default layouts of our application in XML, including details of the screen that will contain and their properties. After adding code to our application we can modify the position of the objects on the screen, such as we have already referred in XML, during runtime.

2. The ADT Plugin for Eclipse offers a preview of the layout that we have stated in the XML file.

3. We can test and tool Hierarchy Viewer, for debugging layouts. The Hierarchy Viewer displays the values of the properties of the layout, wireframes indicators padding / margin, and full rendered views while we debug on the emulator or the device.

4. The tool layoutopt allows us to analyze and layout hierarchies easily in order to identify deficiencies and other problems. The benefit of declaration of the UI in XML is that it allows better separation in the representation of our implementation of the code. The descriptions of the UI are out of code enforcement, which means that we can modify / customize without having to change and recompile the code enforcement. For example, we can create XML layouts for different screen orientations, different size monitors and different languages.

Moreover, stating the layout in XML makes it easier to display the structure of the User Interface and it is easier to correct the errors. In this format, the text focuses on teaching us how to declare the layout to XML. Creating a View object in runtime, we can look at reviews of classes and ViewGroup View. Generally, the XML vocabulary is used in the statement of UI elements and follows the structure and nomenclature of the classes and methods. In this case, the element names correspond to class names and the names of the properties in these methods.

2.8 Graphics

The graphics in the Android operating system are included in a custom library of two-dimensional (2D) graphics and the OpenGL ES 1.0 3D graphics for high performance. The most common API for 2D graphics is in the package drawable. The API on OpenGL is available from the package Khronos OpenGL ES, along with some OpenGL utilities for Android. When a project starts, it is important to define precisely the requirements for the graphics. Different graphic processes are best achieved by different techniques. For example, graphics and animations for a relatively static application must be carried out in a different way than in a game or interactive applications with 3D rendering.

2.9 Analysis Options

The 2D graphics can be created with one of the following two ways:

The first way is: We design the graphics or animations in an object View. In this way, the manipulation of the graphic design (and animation) is made by the design process of normal View. We must simply specify the graphics into the View.

The second way is: We design the graphics directly onto a canvas. In this way, we call the method draw(), the appropriate class (passing of the Canvas) or a method of draw ...() (eg drawPicture ()). Following this procedure, we can control any animation.

The first way is better if we want to design simple graphics that do not need to be dynamic and part of a game. On the other hand, we prefer the second way when our application should must do re-draw at regular intervals. Essentially, all games have to be designed to Canvas.

2.10 Design Canvas

Through Canvas, our plan actually performed on a Bitmap, which is later placed in the window. If we plan inside the method onDraw (), we have the Canvas and we should simply tune calls for design. We also request a Canvas from SurfaceHolder.lockCanvas (), when working with an object SurfaceView. However, if we want to create a new Canvas, we must set the Bitmap in order to implement the plan. The Bitmap is always necessary for a Canvas. We can define a new Canvas as the following one:

```
Bitmap  Bitmap.createBitmap  100,=(b);
Bitmap.Config.ARGB_8888);
b =(100, Canvas cnew Canvas
```

So the Canvas was designed over the specified Bitmap. After drawing on it with the Canvas, we can transfer the Bitmap Canvas to another one of the methods Canvas.drawBitmap (Bitmap ,...). It is more logical to design graphics with one of the Canvas provided by View.onDraw () or SurfaceHolder.lockCanvas () as it will be seen below. The Canvas class has a separate set of design methods, such as Bitmap (...), drawRect (...), drawText (...), and many others. Other classes may also use methods draw (). For example, we will have some items Drawable that want to place on Canvas. The Drawable has its own method draw () and accepts the Canvas as an argument.

2.11 View

If the application does not require high processing power and high framerate (for example, a chess game, snakes and ladders, or some other applications by changing the environment slowly), then the implementation can be done by creating a custom View component and the design of the Canvas View . onDraw (). The most convenient in this way is that the Android framework gives us the pre-configured Canvas which will define the methods calls for the design. For starters, we extend the Class View (or descendant thereof) and set the method onDraw (). This method is called the Android framework that the View is designed by itself. Here we perform all calls for the design through the Canvas, so as to get the method onDraw (). The Android framework calls the onDraw () method as necessary. Every time our application is ready to design, we call the View to be invalidated by calling the invalidate (). So we declare that we want the View to be redesigned and Android calls the method onDraw(). Within onDraw () of the View, we use the Canvas that we have been given for the entire design, using different methods Canvas.draw ...(), or other methods draw () by classes that receive the Canvas as a parameter. When onDraw () is completed, the Android framework uses the Canvas in order to draw a Bitmap that will operate the system. We must refer that if we want to invalidate a request from a thread outside the main Activity thread, we have to call the postInvalidate ().

2.12 Surface View

The SurfaceView is a special subclass of View, which offers an area exclusively for design in the hierarchy of View. The main aim is to offer this surface design in the secondary thread to the application, so the application cannot wait to prepare the View system. A secondary thread, that is associated with the SurfaceView, can be designed in the Canvas smoothly. First of all, we can create a class that inherits the SurfaceView. The class must implement to make the SurfaceHolder.Callback. This subclass is an interface that give us the underlying Surface, when it is created, modified, or destroyed. These facts are important

because we have to know the exact time of beginning the design, whether to make adjustments to meet new surface properties, and when it is safe to stop designing, and possibly kill other processes. Within the class SurfaceView it would be ideal to identify a secondary class Thread, which will perform all procedures in Design Canvas. Rather than manipulate the object directly Surface should be handle through a SurfaceHolder. Thus, when initializing the SurfaceView, ask the requesting SurfaceHolder getHolder (). Then we need to know the SurfaceHolder that we want to receive the callbacks in SurfaceHolder (from SurfaceHolder.Callback) requesting addCallback () (with this parameter). After that we overload each method SurfaceHolder.Callback from heading SurfaceView to design the Surface Canvas from the second thread and we must pass along the thread SurfaceHandler and regain the Canvas with lockCanvas ().

2.13 Graphics 2D

The Android operating system offers a custom 2D graphics library for the design and animated shapes and images. Inside the packages android.graphics.drawable and android.view.animation we will find common classes used in the design and movement in two dimensions.

2.14 Drawables

A Drawable is a general deduction for "something that can be designed." The class Drawable inherites and provides a wide range of concrete drawable graphics, including BitmapDrawable, ShapeDrawable, PictureDrawable, LayerDrawable, and many others. Of course, we can inherit them to define our own custom Drawable objects that have unique behavior. There are three ways to define and initialize a Drawable: the first one is by using a stored image of the resources, the second one is by using an XML file which defines the properties of the Drawable and the third one by using the manufacturers (constructors) from the normal class.

2.15 Creation of icons resource

A simple way to add graphics to our application is a simple reference to an image file from the resources. The supported files are PNG (recommended), JPG (acceptable) and GIF (not recommended by the documentation). This technique is suitable for icons, logos, or other similar graphics embedded in games. If we want to use an image from the resource we simply add the image in the directory res / drawable / of our project. Thus, we can refer to it from either our code or the XML layout. In each case the reference image comes through a resource ID, which is the name of the file without the extension (eg, in my_image.png refer to my_image). It should be noted that images placed in res / drawable / can be optimized automatically in a compressed format without losing the tool aapt. For example, a true-color PNG that requires less than 256 colors can be converted to an 8-bit PNG with colored palette. This will produce an image with the same quality that takes less memory. We must remember that the binary images of this list may change during the build. If we intend to read a picture as a bit stream in order to turn it into a bitmap, we must put the image file in res / raw /, where there will be compression optimization.

2.16 Sample Code

The following code segments show us how to create an ImageView by using an image in the drawable resources and how to add it to the layout.

```
(savedInstanceState);LinearLayoutmLinearLayoutLinearLayout
protected void onCreate  (this);
mLinearLayout; {super.onCreate
LinearLayout / Create ain which to add the / /ImageViewImageView
(Bundle savedInstanceState)= new= new
/ImageViewInstantiate anand define its properties
iImageView (R.drawable.my_image);(true);ImageView
(this);i.setImageResource
```

```
i.setAdjustViewBounds  / / set boundsthe
theto matchDrawable's
```

In other cases, we can manipulate the image as an Drawable object. To do this, we can create a Drawable resource with the following way:

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

It should be noted that every single resource in our work can have only one state, many different snapshots of the object that they create. For example, if we have two snapshots of a Drawable object on the same image, and change a property (like an alpha) in one of the Drawables, then the same value will be changed to another one. For this reason, when we deal with multiple instances of an image, instead of changing form in Drawable, we can make a tween animation.

2.17 Creation of resource XML

If someone wants to change the properties of the Drawable when using the application, he must define the object in XML and he can change its properties once initialized. Then, he retrieves an instance of the object by calling the Resources.getDrawable () and giving as a parameter the ID of the file XML. Each Drawable subclass which supports the method inflate () it can be defined in XML and it can initialize the application. Each Drawable that supports XML inflation makes use of special properties of XML that help to define the properties of the object. The following XML defines a TransitionDrawable:

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

```
</ transition>
```

With this XML stored in res / drawable / expand_collapse.xml, the following code will initialize the TransitionDrawable and will set as content

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

This transition can be promoted under one second with:

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

2.18 Shape Drawable

When we want to design some dynamic two-dimensional graphics, we can use an object ShapeDrawable. A ShapeDrawable is an extension of the Drawable, so we can use it whenever we need a Drawable, possibly for a background of a View using the method setBackgroundDrawable (). Of course, we also can design the shape as a custom View. Because ShapeDrawable has its own method draw (), we can create a subclass of View the ShapeDrawable planned during the execution of the method View.onDraw (). A major expansion of the Class View's planned as a ShapeDrawable View is as follows:

```
class  extends View public CustomDrawableView{super
```

```
ShapeDrawable (context);10;10;300; mDrawableShapeDrawable
publicCustomDrawableView{private context)(new(
mDrawable;
x  int=  int=int==
int y
=width
height50;
(Contextnew OvalShape));().(0xff74AC23);height);}canvas)(canvas);}}
mDrawable.getPaint  setColor (x,(Canvas
mDrawable.setBounds  y, x + width, y +

protected void onDraw
{mDrawable.draw
```

In the constructor, the ShapeDrawable defined as a OvalShape. Assigned a color and defined the boundaries of shape. If you did not define the limits, the shape will not be designed, if not define the color will be set to black. A custom View, can be designed in any way they want. With the following code, we can design the shape of a programmatic Activity:

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

If we can design a custom drawable from XML instead of an Activity, then heading CustomDrawable need to overload the constructor View (Context, AttributeSet), which is called when initializing a View by inflation from XML.

Heading ShapeDrawable (like many other types Drawable the package android.graphics.drawable) allows us to define various properties of drawable through public methods. Some of the properties may wish to amend contain alpha transparency, color filter, dither, opacity and color. We can define basic drawable shapes using XML.

2.19 Frame Animation

In a traditional animation there are a set of different images, which appear in succession, such as playing a movie. The base class for frame animations is AnimationDrawable. We can define the frames of animation through the code using the API AnimationDrawable, however, achieved more simply with an XML file that has the list of frames composing the animation. As the tween animation mentioned, the XML file for this type of animation is in the directory res / drawable / the project. In this case, the instructions regarding the number and duration of each frame of animation. The XML file consists of a <animation-list> element as the root node and a series of child <item> nodes, each of which defines a frame: a drawable resource to the frame and the length of the frame. An example XML file for the frame-by-frame animation:

```
drawable/rocket_thrust1" android:= android: drawable = "@ drawable
android:= "@duration"200"/>/
<item android:drawable="@drawable/rocket_thrust2"
android:duration="200" />
drawable <itemrocket_thrust3 =""/>
"android:duration  200
</ animation-list>
```

This animation runs in just three frames. Defining the status android: oneshot from the list to true will recycle once and then stop and keep the last frame. If set to false then the animation will repeat. To XML stored as rocket_thrust.xml

list res / drawable / the project, and may be added as a background image on View and then executed. Here is an example Activity, where the animation is added to the ImageView and then becomes animated when you touch the screen:

```
AnimationDrawable (savedInstanceState);
(R.layout.main);
public void onCreate
rocketAnimation;
{
super.onCreate
(Bundle savedInstanceState)setContentView
ImageView rocketImage =



= ();}(MotionEvent  ()MotionEvent.ACTION_DOWN)();(event)
rocketImage.getBackground true;}

(AnimationDrawable)public boolean onTouchEvent  {rocketAnimation.start
{if(event.getAction==

return

event)returnsuper.onTouchEvent?}
```

It is important to note that the method start () called in AnimationDrawable can not be called during the execution of the method onCreate () of Activity, because AnimationDrawable is still not fully adhering to the window. If you want to run the animation immediately, without interaction, then we must call the method onWindowFocusChanged () of Activity, which will be called when the Android restores the window to focus.

2.20 3D with OpenGL

The Android includes support for high-performance 3D graphics through OpenGL API - namely, through the OpenGL ES API. The OpenGL ES is a version of OpenGL for use in embedded devices. The versions of OpenGL ES have similarities but are not closely linked to the corresponding versions of classic OpenGL. The Android currently supports the OpenGL ES 1.0, which corresponds to OpenGL 1.3. So, if the application you have in mind is implemented with OpenGL 1.3 on a desktop computer should be implemented and Android. This API provided by the Android is similar to the J2ME JSR239 OpenGL ES API. However, it may not be identical, so care is needed deviations.

2.21 Audio and Video

The Android platform provides built-in encoding / decoding for the most common file types media, so you can easily import audio, video and images to your applications. Access to the media capabilities of the platform is quite simple - the use is done with the same intents and activities used for the remainder of Android. The Android lets you play audio and video of several different types. Reproduced sound or video is saved in the application recources (raw resources), regardless of files in file system, or data stream (data stream) which goes through the network. To include these features in your application, use the class MediaPlayer. The platform also allows you to capture audio and video, where supported by hardware device. For recording audio or video, use the class MediaRecorder. Note that the emulator has hardware for audio or video, but actual devices are likely to provide these capabilities, accessible through the class MediaRecorder.

2.21.1 Play audio and video

The Media played from anywhere: from raw material to resources, from files stored on the filesystem, or an available network (URL). The sound reproduction becomes only the standard device for audio output. At present, this is the speaker of the device or headset Bluetooth. Cannot playback the audio of the conversation.

2.21.2 Play from Raw Resource

Perhaps the most common is the playback from within an application. This function is relatively easy to implement:

First step: Put the sound file (or other media resource) to the list res / raw of the project, where the plug-in in Eclipse (or aapt) will find it and transform it into a resource which can refer to Class R

Second step: Create an instance of MediaPlayer, and make reference to the above resource using MediaPlayer.create, then call the start () from the snapshot:

```
MediaPlayer mp = MediaPlayer.create ();
R.raw.sound_file_1);
(context, mp.start
```

To stop playback, call the stop (). If you want to later play back the media, then you need to reset () and prepare () the object MediaPlayer before calling back the start (). (The create () calls the prepare () the first time.) To pause playback, call the pause (). Continue playback from where it stopped with the start ().

2.21.3 Play from File or Stream

An application can play back files from the filesystem or a web URL:

1. Create a new instance of the MediaPlayer using new

2. Call setDataSource () with a String containing the path (to the file system or URL) of the file you want to play
3. Initially then prepare () and then start () in the snapshot:

```
MediaPlayer mp = new MediaPlayer (PATH_TO_FILE);();();
mp.setDataSource
();mp.prepare
mp.start
```

The stop () and pause () you can use the functions as above.

The mp may be null and this is good practice to check if it is null immediately after the new. Also, exceptions IllegalArgumentException and IOException must either catch or become ignored by the setDataSource (), it might also file the petition or absent. If you pass a URL to a file online, this file should support progressive download (progressive download).

2.21.4 Playing content JET

The Android platform contains a mechanism, called JET, which allows you to add interactive reproduction of the contents of the JET files in your application. You can create content for interactive play JET using the application JetCreator found in the SDK. For breeding and management of JET content from your application, use the class JetPlayer. For descriptions of the JET and for instructions on how to use the tool JetCreator, see JetCreator User Manual. The tool is fully equipped operating in OS X and Windows and on Linux supports the creation of new content, but not import and edit existing content.

An example for the regulation of reproduction by a JET file. Jet that is stored on your SD:

```
JetPlayer myJet  JetPlayer.getJetPlayer  sdcard/level1.jet");= (
myJet.loadJetFile / /
=();byte segmentId0;
("/queue segment 5, repeat once, use General MIDI, transpose
by -1 octave
myJet.queueJetSegment 1, -1, -1, 0, 0, 0, segmentId + +);();
/ / queue segment (2,
5, -1,0, segmentId + 2myJet.queueJetSegment
+);myJet.play
```

The SDK contains an application example - JetBoy - showing how to use JetPlayer to create an interactive soundtrack for your game. Also emphasizes how to use JET events to synchronize the music with events in the game. The application is located in <sdk> / platforms/android- 1.5/samples/JetBoy.

3. Android application development

3.1 The Android SDK

The Android SDK is available for downloading in the following website: http://developer.android.com/sdk/index.html.

3.2 Installation of the Android SDK

The following description regards the process of installing the Android SDK and development environment. If you have already installed the Android SDK you can use this tool in order to update the package.

Step 1: Computer preparation – The most important requirement is the existence of the JDK (Java Development Kit). If you intend to develop in Eclipse with ADT Plug-in (Android Development Tools) - which is recommended if you are new to Android - should ensure that you have installed an appropriate version of Eclipse (3.4 or later).

Step 2: Download the Android SDK - The Android SDK, just after installation, are not fully loaded but contains only basic tools, some of which you can download the remaining ingredients (such as platform Android, code examples, notes, etc.). If you have downloaded the Windows Installer (.exe file) to run, that will check if you have the JDK (if not, install it) and will be installed in the default location.

Step 3: Installing ADT Plug-in for Eclipse - The Android offers a plug-in for Eclipse called Android Development Tools (ADT) which is designed to provide an integrated environment in which developing Android applications. Extends the capabilities of Eclipse, allowing developers to quickly create new Android projects, designing the user interface of their application with a graphical way to do debug using the Android SDK Tools, and even export signed (or not) APKs that they can split over. Overall, the development of applications in Eclipse using the ADT proposed, since it is the quickest way to get started with Android. So if we want to use the ADT to develop applications, install it. If you prefer to work with another development environment (IDE), no need to install the Eclipse and the ADT. Instead, we can use directly, the SDK Tools for application development.

Step 4: Adding Platforms and other Components - The last step in order to set up a full SDK is to setup the tool in order to use Android SDK and AVD Manager (which includes the basic tools) and to download all the necessary components. The SDK uses a modular structure which separates and allocated all of its components - ie the versions of the platform Android, the add-ons (additional) tools platform, code examples and documentation - into autonomous parts for

installation. The SDK, after the initial installation, contains only the most recent version of the SDK Tools. To develop an application we need at least one Android platform, ie a specific version and platform tools (platform-tools). If you used the Windows installer, at the end of the installation wizard will open the "Android SDK and AVD Manager" with a default set of platforms and other components, to installation press Install to accept the installation of the above ingredients. We can open the tool Android SDK and AVD Manager with one of the following ways:

● From Eclipse, select Windows> Android SDK and AVD Manager.
● On Windows, press double-click the file SDK Manager.exe the root folder of the SDK.
● On Mac or Linux, open a terminal and points to the tools / the Android SDK and execute:. / android

Then use the tool Android SDK and AVD Manager, as shown below, to navigate to the SDK repository and choose or upgrade components. In this way will be installed in our system components chosen.

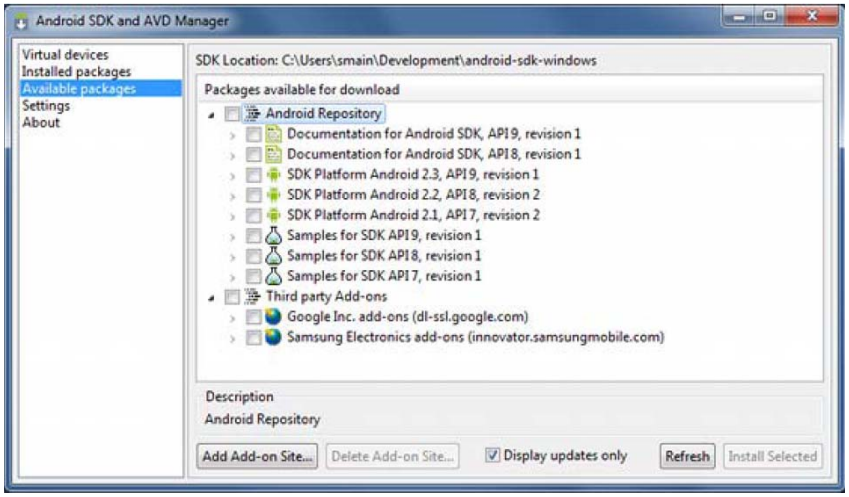

Figure 3.1

3.3 Available Components

There are two sources of components for the SDK: the one is the main source of Android (Android Repository) and the other, an exogenous (third) source of additional components (Third party Add-ons). The Android Repository offers the following types of components:

First type - SDK Tools (preinstalled in the SDK package to remove it) - It includes tools for debug and control applications. They are inside the folder <sdk> / tools / installation of SDK.

Second type - SDK Platform-tools - includes tools that allow users to develop applications in order to debug them (in real device or emulator - emulator android device), to control the device, etc. These tools have been developed along with the Android platform to support the latest features. These tools are updated when a new version of the platform is available and there are inside the folder <sdk> / platform-tools / installation of the SDK.

Third type - Android platforms - Every Android OS version is assigned a version of development environment (SDK), which contains everything needed for developing applications in it (libraries, image operating the simulator, code examples and tools).

Fourth type - USB Driver for Windows (only for Windows) - Includes drivers (driver files) which we can install to run and debug applications do us in a real device with Android software. If you intend to work in real device or not using the OS Windows for software development, we do not need the USB drivers.

Fifth type - Samples - Contains code examples of small or large applications on all platforms Android. It is a good resource for beginners to platform developers.

Sixth type - Documentation - lowers our system copies the latest versions of documentation and explanation of the framework API in Android.

The Third party Add-ons source provides components for creating and developing applications using special, beyond the basic, libraries (such as the library of Google Maps) and even differing versions - versions of the operating system. You can include other external sources by clicking the Add Add-on Site.

Adding location of the SDK's tools / and platform-tools in the environment variable PATH, we can through the command line to execute a program - a tool without having to be at the directory in which the program or to identify the full name (full path name). Depending on which operating system you use, you can include the above two sites in the PATH by the following ways:

First way - In Microsoft Windows environment, press Right click on My Computer and then click the option “Properties”. In the Advanced tab, select Environment Variables and in the window that appears press double click on Path (the system variables). Add to this end, the exact location of tools / and platform-tools.

Second way - In Linux environments, locate the PATH environment variable to any of the files ~ /. Bash_profile, ~ /. Bashrc or ~ /. Profile and add to it the full name of the directory tools / and platform-tools /.

Third way - In Mac OS X, locate the file. Bash_profile and follow the same procedure with Linux, but if the file does not exist, create it.

4. TEI PIRAEUS – Application development

4.1 Use case

The figure ( ) shows the use case diagram of the TEI Piraeus application. One of the main use cases is the procedure which is followed by the user in order to find a book in the library. Initially, the user selects the menu option “Library” from the UI of the application. Then, the user fills the fields in order to find a book. The program displays the results in the screen.

The basic steps are the following:

* User choose the menu “Library”
* The application shows the fields (author, book name etc)
* User fills the fields
* The application searching the database
* The application shows the results in the mobile screen

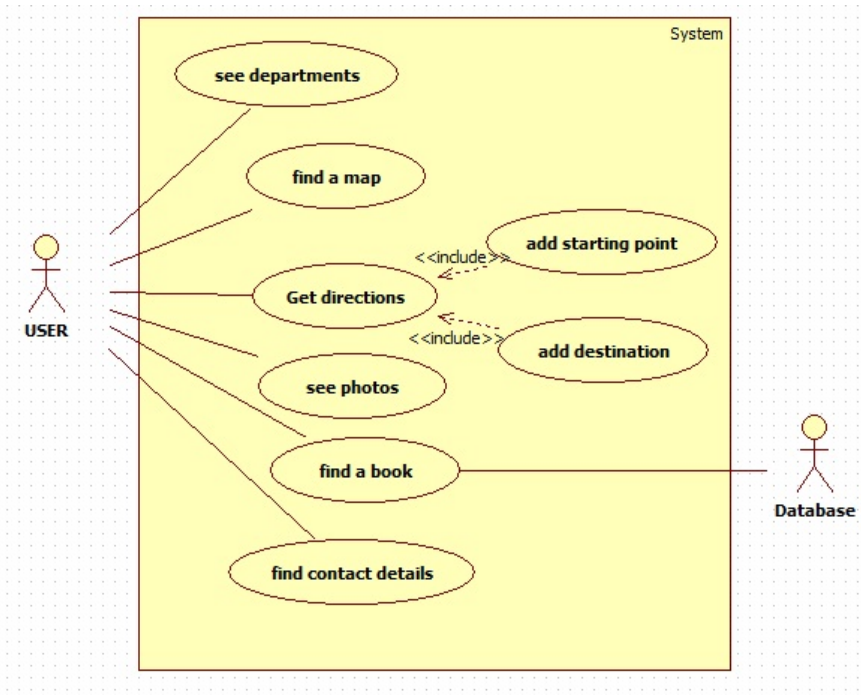

Figure 4.1

4.2 TEI Piraeus app

The main aim of this dissertation was to develop a multimedia application. The result was implemented using the techniques that were described in the previous chapters. The diagram ( ) shows the applications’ screens and the way that they are connected to each other.

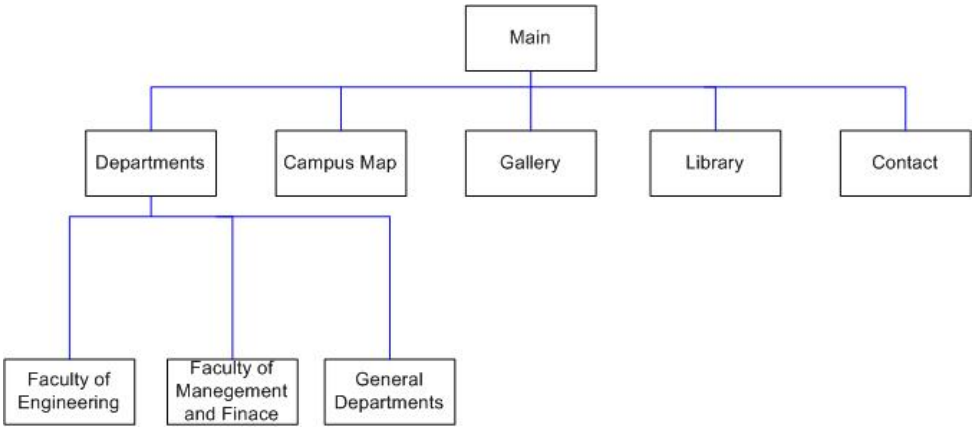

Figure 4.2

The application was implemented slightly different than it was originally planned. This happens because it was necessary to make a theoretical reference to the way that the android operating system displays the multimedia files to the devices. Meanwhile, the application should be built in the form guide. Generally I acquired enough theoretical knowledge. However, I did not use all of them because I had to build several projects that are not related with the android operating system.

The user of the mobile phone can see the main screen by opening the application. The choices that appear on the screen are the following:

1) Departments
2) Campus map
3) Gallery
4) Library
5) Contact

Figure 4.3

The code of this screen is the following:

```
public class teipir extends Activity {
 /** Called when the activity is first created. */
 @Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
Button next3 = (Button) findViewById(R.id.button1);
next3.setOnClickListener(new View.OnClickListener() {
public void onClick(View view) {
Intent myIntent = new Intent(view.getContext(), showdepartments.class);
startActivityForResult(myIntent, 0);
}
});
Button next5 = (Button) findViewById(R.id.button2);
next5.setOnClickListener(new View.OnClickListener() {
public void onClick(View view) {
Intent myIntent = new Intent(view.getContext(), showmap.class);
startActivityForResult(myIntent, 0);
}
});
Button next = (Button) findViewById(R.id.button3);
```

```
next.setOnClickListener(new View.OnClickListener() {
public void onClick(View view) {
Intent myIntent = new Intent(view.getContext(), showgallery.class);
startActivityForResult(myIntent, 0);
}
});
Button next2 = (Button) findViewById(R.id.button4);
next2.setOnClickListener(new View.OnClickListener() {
public void onClick(View view) {
Intent myIntent = new Intent(view.getContext(), showlibrary.class);
startActivityForResult(myIntent, 0);
// TODO Auto-generated method stub
}
});
Button next4 = (Button) findViewById(R.id.button6);
next4.setOnClickListener(new View.OnClickListener() {
public void onClick(View view) {
Intent myIntent = new Intent(view.getContext(), showcontact.class);
startActivityForResult(myIntent, 0);
```

In the first case, the user can see the menu which shows the departments of the Technological Educational Institute of Piraeus. Then, each one of them presents the available websites. If the user clicks on a link he activates the application of the mobile phone so as to open the respective website. This can be achieved by using a parameter in the XML code (android:autoLink=”web”) so that the android operating system can understand that this field is a link to a website. For example, a part of the XML code of the screen “Faculty of Engineering” is the following:

```
<TextView android:layout_width="match_parent"
android:gravity="center_horizontal" android:layout_height="wrap_content"
android:id="@+id/TextView09" android:text="Department of Automation
Engineering">
```

```
</TextView>
<TextView android:layout_width="match_parent" android:autoLink="web"
android:gravity="center_horizontal" android:layout_height="wrap_content"
android:id="@+id/TextView02" android:text="http://auto.teipir.gr">
</TextView>
```

Figure 4.4

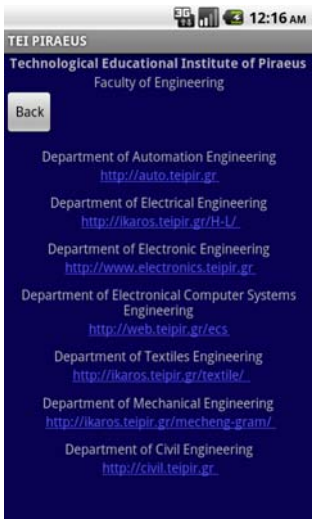

Figure 4.5

In the second case, the user presses the button in order to see the map of the Technological Educational Institute of Piraeus. In this case, the application loads a screen which includes a frame. This frame is set to load via maps.goole.com a specific location on the world map. The code of the program sets the coordinates of TEI Piraeus and also the tools that are available to the user (zoom, add a POI, etc.). The XML code that provides the webframe is the following:

```
<WebView  xmlns:android="http://schemas.android.com/apk/res/android"
   android:id="@+id/webview"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"/>
```

Also, the part of the code that executes the webview of the google map is the following:

```
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.map);
mWebView = (WebView) findViewById(R.id.webview);
mWebView.getSettings().setJavaScriptEnabled(true);
mWebView.loadUrl("http://maps.google.com/?ll=37.978651,23.673574&spn=0.00
674,0.009645&t=h&z=17");
Button next = (Button) findViewById(R.id.button5);
next.setOnClickListener(new View.OnClickListener()
```

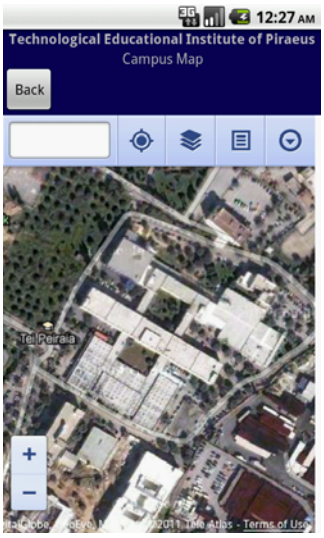

Figure 4.6

In the third case, the user clicks the button in order to see the gallery of the Technological Educational Institute of Piraeus. In this place, the applications show several photographs of the TEI of Piraeus. The photographs are stored within the application (inside the /res/drawable directory). The application displays the images on the screen using the following code:

```
public class showgallery extends Activity {
```

```
/** Called when the activity is first created. */
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.gallery);
Gallery g = (Gallery) findViewById(R.id.gallery1);
g.setAdapter(new ImageAdapter(this));
g.setOnItemClickListener(new OnItemClickListener() {
public void onItemClick(AdapterView<?> parent, View v,int position, long id) {
Toast.makeText(showgallery.this, "" + position, Toast.LENGTH_SHORT).show();
}
});
}
public class ImageAdapter extends BaseAdapter {
int mGalleryItemBackground;
private Context mContext;
private Integer[] mImageIds = {
R.drawable.photo1,
R.drawable.photo2,
R.drawable.photo3,
R.drawable.photo4,
R.drawable.photo5,
R.drawable.photo6,
R.drawable.photo7,
R.drawable.photo8,
R.drawable.photo9,
};
public ImageAdapter(Context c) {
mContext = c;
TypedArray a = c.obtainStyledAttributes(R.styleable.Gallery);
mGalleryItemBackground =
a.getResourceId(R.styleable.Gallery_android_galleryItemBackground, 0);
a.recycle();
```

```
Button next = (Button) findViewById(R.id.button5);
next.setOnClickListener(new View.OnClickListener() {
public void onClick(View view) {
Intent myIntent = new Intent(view.getContext(), teipir.class);
startActivityForResult(myIntent, 0);
}
});
```

The results of the previous code are shown in the next figure:

Figure 4.7

In the fourth case, the user acquires access to the library database of the Technological Educational Institute of Piraeus. Here, the screen appears some fields which are part of the code of the library's website. The fields are loading with the same way as in the original website (http://lib.teipir.gr).

Figure 4.8

In the fifth case, the application shows the contact information of the Technological Education Institute of Piraeus. This is similar to the first case which uses the XML field properties in order to provide the appropriate information to Android operating system. In this case, when the user clicks on the telephone number, the application executes the "mobile call" service.

Figure 4.9

All the screens of the application have a "back" button. Android applications are not necessary to have such a button because the devices which are running the

Android operating system they have such a button. This specific application was included in order to be familiar with the users who have not previously experienced in using an Android mobile phone.

Conclusion

We can draw many conclusions from the implementation of this dissertation. First of all, during this procedure I realized that it is not difficult to develop an Android application compared to other platforms. Google developed all the necessary tools in order to help developers to make complex applications in a short time. So, there are no limits except from the developers' imagination. The Android operating system has a great development. Currently, there are countless sources of information on the Internet and many companies deal only with the Android development. In the area of multimedia applications, the Android operating system responds perfectly. This is of course the fact that the Android operating system runs on devices that have very fast processors, large memory and various other features. In any case, any mobile phone that uses the Android is a powerful tool. This tool can be used either for business or pleasure. The Android operating system was built by a huge company in the field of technology and its future will be absolutely bright.

References

Friesen, Geoff. Learn Java for Android Development. [Berkeley, Calif.]: Apress, 2010.

Burnette, Ed. Hello, Android: Introducing Google's Mobile Development Platform; [Android 2]. Raleigh, NC

Android Developers. What is Android, 2010. http://developer.android.com/guide/basics/what-is-android.html

Wikipedia.org. Android Operating System, 2010.
http://en.wikipedia.org/wiki/Android_(operating_system)#cite_note-EnSDK-62

Google Code. Google Maps API Web Services, 2010.
http://code.google.com/apis/maps/documentation/webservices/index.html

Sven Woltmann. “Android Beginners Workshop”. AndroidPIT. February,2010

Wiley Publishing. Professional Android 2 Development. Indianapolis, 2010

IBM Corporation . Introduction to Android development using Eclipse and Android Widgets. 2010.

Google Buys Android for Its Mobile Arsenal., Businessweek.com. 2005-08-17.

Lauren Darcey. Sams Teach Yourself Android™ Application Development in 24 Hours.
Indianapolis, Indiana. 2010
Mark L. Murphy, Android Programming Tutorials, CommonsWare, USA, 2010

Shawn Van Every, Pro Android Media. APRESS, USA, 2009

Appendix

Teipir.java

```
package teipir.thesis.com;

import android.app.Activity;
import android.os.Bundle;
import android.content.Intent;
import android.view.View;
import android.widget.Button;

public class teipir extends Activity {
   /** Called when the activity is first created. */
   @Override

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

      Button next3 = (Button) findViewById(R.id.button1);
      next3.setOnClickListener(new View.OnClickListener() {

   public void onClick(View view) {
   Intent myIntent = new Intent(view.getContext(), showdepartments.class);
   startActivityForResult(myIntent, 0);

            }
});

   Button next5 = (Button) findViewById(R.id.button2);
      next5.setOnClickListener(new View.OnClickListener() {

public void onClick(View view) {
content myIntent = new Intent(view.getContext(), showmap.class);
            startActivityForResult(myIntent, 0);

            }
});
      Button next = (Button) findViewById(R.id.button3);
      next.setOnClickListener(new View.OnClickListener() {


                              public void onClick(View view) {
                                    Intent myIntent = new Intent(view.getContext(), showgallery.class);
                                    startActivityForResult(myIntent, 0);

                              }
      });
                        Button next2 = (Button) findViewById(R.id.button4);
                            next2.setOnClickListener(new View.OnClickListener()
            public void onClick(View view) {
                                                      Intent myIntent = new Intent(view.getContext(),
showlibrary.class);
                                                      startActivityForResult(myIntent, 0);

                                    // TODO Auto-generated method stub

                              }
                        });

                            Button next4 = (Button) findViewById(R.id.button6);
                            next4.setOnClickListener(new View.OnClickListener() {


                                                public void onClick(View view) {
```

```
                                                    Intent myIntent = new Intent(view.getContext(),
showcontact.class);
                                                    startActivityForResult(myIntent, 0);

                                        // TODO Auto-generated method stub

                                }
                        });


    }
}
```

Showmap.java

```
package teipir.thesis.com;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.webkit.WebView;

public class showmap extends Activity {

        private WebView mWebView;

        /** Called when the activity is first created. */
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.map);

                mWebView = (WebView) findViewById(R.id.webview);
                mWebView.getSettings().setJavaScriptEnabled(true);

mWebView.loadUrl("http://maps.google.com/?ll=37.978651,23.673574&spn=0.00674,0.009645&t=h&z=17");

                Button next = (Button) findViewById(R.id.button5);
                next.setOnClickListener(new View.OnClickListener() {
        public void onClick(View view) {
                Intent myIntent = new Intent(view.getContext(), teipir.class);
                startActivityForResult(myIntent, 0);

                }
});
        }}
```

Showmanagement.java

```
package teipir.thesis.com;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class showmanagement extends Activity {
   /** Called when the activity is first created. */
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.management);

      Button next = (Button) findViewById(R.id.button5);
      next.setOnClickListener(new View.OnClickListener() {

                                    public void onClick(View view) {
                                                Intent myIntent = new Intent(view.getContext(), showdepartments.class);
                                                startActivityForResult(myIntent, 0);

                                                // TODO Auto-generated method stub
}

});
   }
}
```

Showlibrary.java

```
package teipir.thesis.com;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.webkit.WebView;

public class showlibrary extends Activity {

            private WebView mWebView;
```

```
        /** Called when the activity is first created. */
         public void onCreate(Bundle savedInstanceState) {
                        super.onCreate(savedInstanceState);
                        setContentView(R.layout.library);

                        mWebView = (WebView) findViewById(R.id.webview);
                        mWebView.getSettings().setJavaScriptEnabled(true);
                        mWebView.loadUrl("http://lib1.teipir.gr/cgi-bin-EN/egwcgi/4076/query.egw;/-
1+195.251.69.61:210/ADVANCE");

Button next = (Button) findViewById(R.id.button5);
next.setOnClickListener(new View.OnClickListener() {
        public void onClick(View view) {
                Intent myIntent = new Intent(view.getContext(), teipir.class);
                startActivityForResult(myIntent, 0);
        }
        });
         }
}
```

Showgallery.java

```
package teipir.thesis.com;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.content.res.TypedArray;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.BaseAdapter;
import android.widget.Button;
import android.widget.Gallery;
import android.widget.ImageView;
import android.widget.Toast;
```

```
public class showgallery extends Activity {


        /** Called when the activity is first created. */
    public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.gallery);

    Gallery g = (Gallery) findViewById(R.id.gallery1);
    g.setAdapter(new ImageAdapter(this));
   g.setOnItemClickListener(new OnItemClickListener() {
   public void onItemClick(AdapterView<?> parent, View v,int position, long id) {
   Toast.makeText(showgallery.this, "" + position, Toast.LENGTH_SHORT).show();
    }
   });
    }

public class ImageAdapter extends BaseAdapter {

        int mGalleryItemBackground;
        private Context mContext;

        private Integer[] mImageIds = {
        R.drawable.photo1,
        R.drawable.photo2,
        R.drawable.photo3,
        R.drawable.photo4,
        R.drawable.photo5,
        R.drawable.photo6,
        R.drawable.photo7,
        R.drawable.photo8,
        R.drawable.photo9,
        };

        public ImageAdapter(Context c) {
        mContext = c;
        TypedArray a = c.obtainStyledAttributes(R.styleable.Gallery);
    mGalleryItemBackground = a.getResourceId(R.styleable.Gallery_android_galleryItemBackground, 0);
        a.recycle();

        Button next = (Button) findViewById(R.id.button5);
    next.setOnClickListener(new View.OnClickListener() {

                                public void onClick(View view) {
                                Intent myIntent = new Intent(view.getContext(), teipir.class);
                                startActivityForResult(myIntent, 0);

                                }
```

```
        });
}
                public int getCount() {
        return mImageIds.length;
        }
        public Object getItem(int position) {
        return position;
        }
        public long getItemId(int position) {
        return position;
        }
        public View getView(int position, View convertView, ViewGroup parent) {
        ImageView i = new ImageView(mContext);

        i.setImageResource(mImageIds[position]);
        i.setLayoutParams(new Gallery.LayoutParams(300, 250));
        i.setScaleType(ImageView.ScaleType.FIT_XY);
        i.setBackgroundResource(mGalleryItemBackground);
        return i;
        }
};

    }
```