구글의 안드로이드와 안드로이드마켓

김정훈 (용인송담대학)

차 례

1. 서론

1.1 스마트 폰 시장 동향

WIPI 의무화의 폐지로 해외의 다양한 플랫폼이 국내에 들어 올 수 있게 되었다. 현재 해외에서는 스마트 폰이 대세이다. 스마트 폰의 특징이라면 PC에서 할 수 있는 많은 일을 스마트 폰에서도 할 수 있다는 것이다. 여기에 스마트 폰의 경쟁력이 있다.

전 세계적으로 많이 사용되는 모바일 플랫폼에는 노키아의 심비안, RIM의 블랙베리, 애플의 OS X, 마이크로소프트의 윈도우즈 모바일, 구글의 안드로이드 등이 있다. 다음 표는 각 회사의 모바일 플랫폼의 시장 점유율을 나타낸 것이다.

표 1. 스마트 폰의 점유율과 성장률

회사	2007년 3분기 점유율	2008년 3분기 점유율	성장률
심비안	63.1%	49.8%	-12.0%
RIM	9.7%	15.9%	81.7%
애플	3.4%	12.9%	327.5%
마이크로소프트	12.8%	11.1%	-3.0%

<표 1>을 살펴보면 1위는 2007년이나 2008년 모두 심비안이 차지하였다. 그러나 2008년 점유율은 49.8%로 전년 동기 63.1%에 비해 많이 줄어 들었다. 2008년 심비안은 처음으로 50% 미만의 점유율을 기록한 것이다. 대신 이 자리를 RIM과 애플이 차지하였다[1].

RIM은 시장 점유율을 전년 동기 대비 81.7% 증가시킨 15.9%를 차지하였다. 놀라운 것은 애플의 맥 OS X인데 성장률이 전년 동기 대비 무려 325%나 증가하였다. 시장 점유율 자체는 12.9%로 그리 크지 않지만 시작한 지 얼마 되지 않은 점을 고려하면 1위나 2위를 넘볼 수도 있을 것으로 예상된다.

애플의 맥 OS X는 12.9%의 점유율로 4위인 마이크로소프트의 윈도우즈 모바일을 제치고 3위의 자리에 올랐다. 1.8%의 격차를 두고 3위로 올라섰으며 2위와의 격차도 불과 3% 밖에 나지 않는다. 이렇게 애플이 크게 성장할 수 있었던 이유는 아이폰 판매에 따른 것이다.

1.2 OHA(Open Handset Alliance)

1)애플의 아이폰 못지 않게 최근 주목 받고 있는 스마트 폰이 있다. 바로 구글에서 제작하고 있는 안드로이드 폰이다. 정확히 표현하면 안드로이드 폰은 구글에서 제작하고 있는 것은 아니다. 안드로이드 폰 하드웨어는 세계 유명 단말기 제조회사에서 만들고 그 폰에 탑재되는 모바일 플랫폼만 구글에서 만들고 있다. 구글은 이 모바일 플랫폼을 안드로이드(android)라고 명명하였다.

아직 시장 파급력은 크지 않지만 구글은 노키아의 심비안처럼 OS 플랫폼과 소스 코드를 개방하고 있기 때문에 안드로이드는 전세계 통신사업자들에 의해 빠르게 확산될 것으로 보인다. 더욱이 안드로이드는 개방형인 리눅스를 기반으로 하고 있기 때문에 하드웨어와 서비스 개발의 폭이 넓다는 장점이 있다.

또 구글이 주도하는 안드로이드 이용모임 “OHA(Open Handset Alliance)”에 최근 보다폰, 소프트뱅크모바일, 소니에릭슨, 도시바, 삼성전자, LG전자 등 많은 기업들이 참여하고 있어 안드로이드의 확산 속도는 더 빨라지고 있다[2].

1) 본 고는 저자가 2009년 4월 8일자 주간기술동향 (www.iita.re.kr)에 기고한 글을 요약 정리한 내용입니다.

구글에서는 OHA 정신에 입각하여 안드로이드 출시 후 안드로이드 소스를 공개하였다. 또한, SDK를 공개하여 안드로이드 환경에서 동작할 수 있는 소프트웨어를 개발할 수 있게 하였다. 마지막으로 이것을 유통시킬 수 있는 안드로이드마켓 또한 오픈하여 다양한 소프트웨어를 공급할 수 있게 하였다.

2009년에는 구글 주도의 OHA에 가입한 여러 제조업체들이 안드로이드를 탑재한 단말기를 생산할 것으로 보인다. 삼성전자, LG전자, 모토로라, 소니에릭슨 등이 2009년에 안드로이드를 탑재한 단말기를 출시했거나 할 예정이다.

2. 안드로이드는 무엇인가

2.1 안드로이드 개념

안드로이드는 기존의 WIPI, BREW, GVM 등과 같은 모바일 디바이스를 위한 플랫폼이다. 아주 단순하게 생각하면 PC 위에 돌아가는 Windows와 같은 운영체제라고 생각해도 된다.

좀 더 정확히 설명하면 안드로이드 플랫폼은 운영체제, 미들웨어, 키(key) 애플리케이션들을 포함한 모바일 디바이스를 위한 소프트웨어 집합이다. 개발자들이 Windows에서 애플리케이션을 개발하듯이 안드로이드 SDK를 사용하면 안드로이드 폰에서 동작하는 애플리케이션을 만들 수 있다[3].

애플리케이션들은 Java 프로그래밍 언어로 작성해야 하고 Dalvik 위에서 실행된다. Dalvik은 구글이 만든 가상머신인데, Linux 커널의 최상위 영역에서 동작한다. 자바가상머신 같은 역할을 하는 것이라고 생각하면 이해가 쉬울 것이다.

2.2 안드로이드 특징

안드로이드의 특징은 다음과 같다[4].

1) 애플리케이션 프레임워크의 제공

Windows와 비슷하게 프로그래밍 시 일종의 마법사가 프로그래밍 할 수 있는 프레임워크(framework)를 제공해준다. 프레임워크는 일종의 틀이다. 빵 굽는 틀이 있으면 일정한 모양의 빵을 쉽게 구워낼 수 있다. 서로 다른 애플리케이션 일지라도 그 기본적인 틀은 비슷하기 때문에 대부분의 SDK에서는 그 틀을 만들 수 있는 마법사를 제공하고 있다.

2) Dalvik 가상 머신

자바가상머신(Java Virtual Machine)과 같이 일종의 가상 머신인데 모바일 디바이스를 위해 최적화되었다. 왜 Dalvik이라는 생소한 이름을 사용하는 것일까? Dalvik 이름을 명명한 사람은 Bornstein이라는 사람이다. Bornstein의 조상들이 아이슬랜드의 한 어촌인 Dalvik에서 살았다고 해서 그렇게 명명하였다고 한다.

3) 최적화된 그래픽

안드로이드에서는 기본적으로 2D 그래픽 라이브러리를 제공하고 있다. 또 OpenGL ES 1.0 스펙에 기반한 3D 그래픽 라이브러리도 제공한다.

4) SQLite

데이터를 저장하고 검색하기 위해 사용된다. 안드로이드에서 제공하는 데이터베이스 시스템이라고 생각하면 된다.

5) 미디어 지원

일반적인 오디오, 비디오, 그리고 정지 이미지 포맷들을 지원한다. 현재 구글에서 밝히고 있는 지원가능한 포맷은 MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF 등이다.

6) GSM 테크놀로지, 블루투스, EDGE, 3G, WiFi, 카메라, GPS, 나침판, 가속도계의 지원

하드웨어 의존적이기는 하나 지원 가능하다.

7) 풍부한 개발 환경

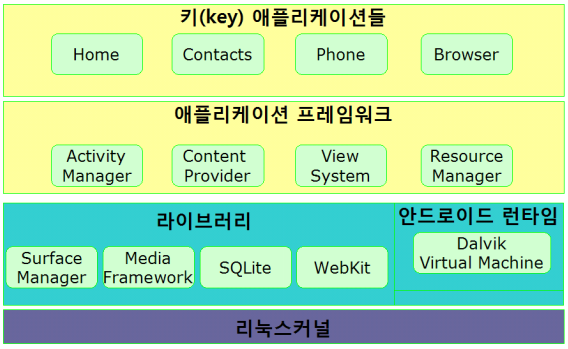
디바이스 에뮬레이터, 디버깅 도구, 메모리 및 성능 프로파일링, Eclipse IDE(Integrated Development Environment)를 위한 플러그인을 제공한다.

8) 웹 브라우저 제공

2.3 안드로이드 아키텍처

(그림 1)은 안드로이드 운영체제의 주요 구성 요소를

보여준다. 이 그림은 안드로이드 아키텍처를 필자가 간략하게 재구성한 것이다[5].



▶▶ 그림 1. 안드로이드 아키텍처

1) 애플리케이션

안드로이드에서는 이메일을 확인할 수 있는 클라이언트, SMS 프로그램, 캘린더, 지도, 브라우저, 주소록 등을 키 애플리케이션으로 제공한다. 모든 애플리케이션들은 자바 언어로 작성되었다.

2) 애플리케이션 프레임워크

애플리케이션들이 사용하는 프레임워크를 제공한다. 이 프레임워크를 사용하여 자신만의 애플리케이션을 개발할 수 있다. 위 그림에서 살펴본 것처럼 애플리케이션 프레임워크에서는 애플리케이션을 개발하기 위한 각종 클래스와 메소드들이 제공된다.

3) 라이브러리

시스템 C 라이브러리, 미디어 라이브러리, Surface 관리자, LibWebCore, 2D 그래픽 엔진, 3D 라이브러리, 경량화된 관계형 데이터베이스 엔진 등이 라이브러리 형태로 제공된다.

4) 안드로이드 런타임

안드로이드는 Java의 핵심 라이브러리 기능들을 대부분 포함하고 있다. 모든 안드로이드 애플리케이션은 Dalvik 가상 머신 내에 자신의 인스턴스를 가지고 동작한다.

Dalvik에서는 최소 메모리만을 사용하도록 최적화된 Dalvik Executable(.dex) 포맷의 파일들을 실행한다. Dalvik VM은 자바 언어 컴파일러에 의해 컴파일된 클래스를 'dx'라는 도구에 의해 .dex 포맷으로 변환시켜 실행한다. Dalvik 가상 머신에서는 스레딩과 저수준 메모리 관리와 같은 기능을 리눅스 커널에 의존한다.

5) 리눅스 커널

안드로이드 플랫폼은 보안, 메모리 관리, 프로세스 관리, 네트워크 관리, 드라이버 모델 등의 핵심 서비스를 리눅스에 기초하여 구현되었다. 이 리눅스 커널은 하드웨어와 나머지 소프트웨어 스택 간의 추상화된 계층 역할을 한다.

3. 애플리케이션 구조와 사용자 인터페이스

안드로이드 애플리케이션은 Activity, Intent Receiver, Service, Content Provider의 4가지 구성 요소로 이루어져 있다[6]. 모든 애플리케이션이 위 4가지 모두를 필요로 하는 것은 아니다. 이들의 조합으로 애플리케이션은 구성된다. 예를 들어 애플리케이션은 Activity로만 구성될 수 있고 Activity와 서비스로 구성될 수도 있다. 물론 애플리케이션은 위 4가지 구성요소 모두로 이루어질 수도 있다.

애플리케이션에서 어떤 구성 요소들을 사용할지 결정하였다면 이 구성 요소들의 목록을 AndroidManifest.xml 파일에 기록해야 한다. 이 AndroidManifest.xml 파일은 어느 애플리케이션에서 어떤 구성 요소들을 선언했는지, 그들의 기능과 요구사항은 무엇인지를 기록하는 파일이다.

3.1 하나의 화면, Activity

3.1.1 Activity란 무엇인가

애플리케이션 내의 하나의 스크린 또는 화면을 일컫는 말이다. Activity에서는 UI 컴포넌트를 화면에 표시하고 시스템이나 사용자의 반응을 처리할 수 있다. 윈도우즈에서 윈도우와 비슷한 기능을 하는 것이라고 생각해도 되겠다.

윈도우즈에서 애플리케이션은 여러 개의 윈도우를 갖게 되는데 안드로이드 애플리케이션에서도 여러 개의 Activity를 가질 수 있다.

프로그래머가 구현해야 할 각 Activity는 안드로이드에서 제공하는 Activity 클래스로부터 상속받아 구현해야 하며, 사용자에게 View와 이벤트 처리를 할 수 있는

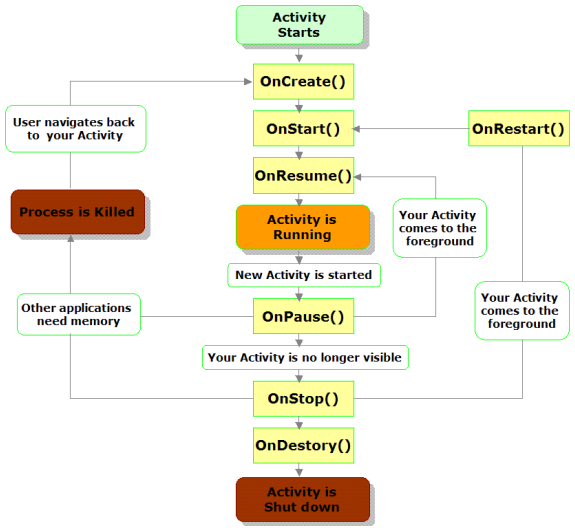
인터페이스를 제공한다.

3.1.2 Activity 라이프사이클

안드로이드의 Activity 역시 MIDlet과 유사하게 라이프사이클 관리를 위한 8개의 메소드를 제공한다. 각각의 메소드들은 다음과 같다[7].

onCreate	onStart	onRestart	onResume
onFreeze	onPause	onStop	onDestroy

각 메소드들의 자세한 의미는 참고문헌의 주간기술동향을 참조하기 바란다[8]



▶▶ 그림 2. Activity 라이프사이클

Activity가 시작되고 실행 중에 있을 때 다른 Activity가 시작되면 그때까지 실행되던 Activity는 pause 상태로 전환되어야 한다. pause 상태로 존재하다가 다른 애플리케이션에서 메모리를 많이 요구하게 되면 pause 상태에 있던 Activity는 죽을(killed) 수도 있다.

만약 pause 상태에 있던 Activity가 사용자의 선택에 의해 다시 포그라운드로 돌아가게 되면 resume 상태가 된다. 또 pause 상태에 있던 Activity가 더 이상 보이지 않으면 stop 상태로 전환된다. stop 상태에 있던 Activity가 다시 포그라운드로 되면 restart에 의해 start 상태로 돌아간다. 물론 stop 상태에 있던 Activity도 다른 애플리케이션이 메모리를 요구할 때 죽을 수도 있다. 또는 destory 상태로 전환될 수도 있다.

위와 같이 한 Activity가 시작되고 멈추어지고 다시 시작되고 그리고 결국은 죽는 과정이 마치 라이프사이클과 같다고 해서 Activity 라이프사이클이라고 한다.

3.1.3 Intent

안드로이드는 화면과 화면 사이를 이동할 때, Intent라는 특별한 클래스를 사용한다. 이 Intent에는 애플리케이션이 무엇을 해야 하는지에 대한 정보가 담겨져 있다. A Activity에서 B Activity로 화면 전환이 이루어질 때 Intent 클래스가 사용된다. Intent 클래스는 다음과 같이 사용된다.

```
Intent i = new Intent(this, NoteEdit.class);
startActivityForResult(i, ACTIVITY_CREATE);
```

호출하는 측은 this 이고 호출 당하는 측은 NoteEdit 클래스이다. this 클래스에서 NoteEdit 클래스의 화면으로 전환이 이루어진다. 화면 전환은 startActivity()이나 startActivityForResult() 메소드를 통해 이루어진다.

3.2 Intent Receiver, Service, Content Provider

3.2.1 Intent Receiver

Intent Receiver는 핸드폰으로 전화가 걸려오거나, 데이터 네트워크 접속이 활성화되는 것과 같이 외부에서 이벤트가 발생했을 때의 상황을 처리하기 위해 사용된다. Intent Receiver는 Activity와 같이 UI를 그려주는 것은 아니고, NotificationManager를 이용하여 사용자에게 어떤 일이 발생했는지를 알려준다. Intent Receiver도 AndroidManifest.xml 파일에 등록해야 하는데, Context.registerReceiver()를 이용하여 코드 상에서 등록해 줄 수도 있다.

3.2.2 Service

Service는 UI와 상관없이 오랫동안 존재하면서 실행되는 코드이다. 예를 들면 재생 목록에서 노래를 재생하는 미디어 플레이어 같은 것을 Service라고 할 수 있다. 미디어 플레이어 애플리케이션은 사용자가 곡을 선택하고 재생을 시작하게 하는 하나 이상의 Activity를 가지고 있지만, 음악을 재생하는 것은 Activity에 의해 이루어지는 것은 아니다.

사용자는 새로운 화면으로 이동하고 나서도 음악이 계

속 재생되기를 기대한다. 이럴 때 미디어 플레이어의 Activity에서 Context.startService() 문장을 실행하면, Service로 실행된다. 안드로이드 시스템은 음악 재생 서비스를 멈출 때까지 계속 재생해 준다. Context.bindService() 메소드는 서비스에 연결하거나 아직 시작하지 않은 Service를 시작할 때 사용한다. 일단 Service에 연결되면 Service에 접근 가능한 인터페이스를 통해 멈춤, 다시 재생 등의 기능을 사용할 수 있다.

3.2.3 Content Provider

애플리케이션은 자신의 데이터를 SQLite 데이터베이스에 저장하거나 데이터베이스를 사용하지 않고 단순히 파일에 저장할 수도 있다. Content Provider는 어떤 애플리케이션 데이터를 다른 애플리케이션이 공유할 필요가 있을 때 아주 유용하다. 이 클래스는 다른 애플리케이션에서 데이터를 저장하거나, 가져오는 작업을 가능하게 한다.

3.3 화면 구성요소들의 계층구조

안드로이드 애플리케이션의 가장 기본적인 구성 단위는 Activity라는 클래스이다. 이 Activity는 한 화면을 나타내고는 있지만 그 자체로는 아무 것도 보여줄 수 없다. Activity에 View와 ViewGroup 클래스를 사용해야 비로소 화면에 무엇인가를 표시할 수 있다[9,10].

3.3.1 사각 레이아웃, View

View 클래스는 화면 상에서 직사각형 형태의 레이아웃과 각종 정보를 저장하는 자료 구조이다. View 클래스는 화면의 크기 조절, 레이아웃 구성, 그리기, 포커스 변화, 스크롤링과 키 처리 등을 위해 사용된다.

View 클래스는 위젯(widget)의 베이스(base) 클래스로도 사용된다. 위젯은 Text, EditText, Button, RadioButton, Checkbox, ScrollView 등과 같이 화면 내에 컴포넌트 처럼 동작하는 것을 말한다. 위젯을 이용하면 UI를 빠르게 만들 수 있다.

3.3.2 View 들의 집합, ViewGroup

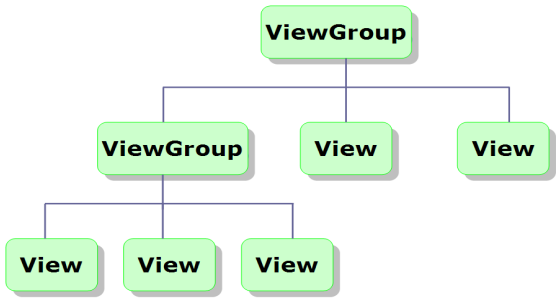
ViewGroup은 말 그대로 View가 여러 개 있는 것이라고 생각하면 된다. 휴대폰 화면에는 하나의 View만 있는 것이 아니라 여러 개의 뷰가 다양한 형태로 존재한다. 이러한 다양한 형태의 View들의 집합을 ViewGroup이라고 한다.

ViewGroup은 View 안에 또 다른 View를 가질 수 있는 구조적 형태를 지원한다. 하나의 ViewGroup은 View나 또 다른 ViewGroup을 포함할 수 있다. 따라서 ViewGroup은 자기가 자기를 포함하는 형태가 가능하다.

ViewGroup 클래스는 레이아웃을 위한 기본 클래스로도 사용된다. 레이아웃은 뒤에서 자세히 살펴보겠지만 View 집합체를 포함하는 구조체를 제공한다.

3.3.3 트리 구조의 사용자 인터페이스

일반적으로 안드로이드에서는 Activity의 UI를 View와 ViewGroup의 트리구조를 이용하여 구성한다. 트리구조의 UI는 안드로이드에서 제공하는 위젯과 레이아웃 또는 직접 제작한 View를 이용하여 구성된다. 당연한 말이겠지만 어떻게 구성하느냐에 따라 트리구조는 간단해질 수도 있고 복잡해질 수도 있다.



▶▶ 그림 3. 트리구조의 사용자 인터페이스

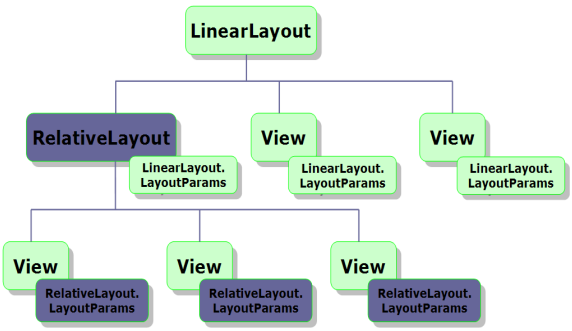
(그림 3)과 같은 트리구조를 화면 상에 그리기 위해서는 트리구조의 루트노드를 파라미터로 하여 setContentView()라는 메소드를 호출하면 된다. Activity가 활성화 되고 포커스를 받으면, 시스템은 루트노드를 화면 상에 그리며, 순서대로 각 ViewGroup은 자기의 하위 노드를 그리게 된다.

각 ViewGroup은 가용한 공간을 계산하고, 그 하위의 View를 배치한다. 각 하위의 View에서는 Draw() 메소드를 호출하여 스스로 그릴 수 있도록 한다. 각 하위의 View는 자신이 그려질 크기와 위치를 부모에게 문의하는데, 부모는 최종적으로 어느 정도의 크기로, 어느 위치에 그릴 것인가를 결정해 준다.

3.3.4 LayoutParams

LayoutParams 클래스는 뷰 객체들이 어떻게 그려져야 하는지에 대한 정보를 부모 객체에게 알려주는 역할을 한다. 각 객체들의 폭과 높이를 명시할 때 다음 둘 중 하나로 표시된다.

- WRAP_CONTENT: 콘텐츠를 표시하는데 필요한 최소한의 크기로 그린다.
- FILL_PARENT: 부모 객체와의 패딩(여백)을 제외한 나머지 모든 공간을 차지한다.



▶▶ 그림 4. 각 View와 ViewGroup은 부모 속성에 맞게 LayoutParams를 정의

모든 LayoutParmas의 서브 클래스는 값을 설정하는 각자의 방법을 가지고 있다. 각 하위 요소들은 위 그림과 같이 부모의 속성에 맞는 LayoutParams를 정의해야 한다. 모든 ViewGroup은 가로와 세로, 마진, 테두리 속성을 가지고 있으며 프로그래머가 원하는 크기로 지정할 수 있다.

루트 노드가 LinearLayout이므로 3개의 자식 노드는 선형으로 배치되어야 한다. 자식 노드 중 첫 번째 노드는 다시 RelativeLayout이므로 3개의 자식 노드들은 상대적으로 배치되어야 한다.

4. 앱스토어와 안드로이드마켓

현재 국내 휴대폰 사용자들이 휴대폰에서 동작하는 게임이나 애플리케이션을 다운로드 받기 위해서는 자신이 가입한 이동통신사의 서버에 접속해야 한다. 따라서 모바일 콘텐츠를 제공하는 CP(Contents Provider)는 게임이나 애플리케이션을 이동통신사의 서버에 업로드 해야 한다.

그러나 아무나 이동통신사의 서버에 업로드 할 수 있는 것은 아니다. 이동통신사의 검수를 통과한 콘텐츠 만이 서버에 업로드 될 수 있다. CP 입장에서는 개발을 다하고도 이동통신사의 검수 과정을 통과하지 못하면 소비자들에게 자신의 콘텐츠를 판매할 수 없다.

최근 서비스를 시작한 애플의 앱스토어(App Store)나 구글의 안드로이드마켓(Android Market)에서의 콘텐츠 유통 방식은 기존 방식과는 많이 다르다. 이동통신사의 검수 과정이 없어지고 앱스토어의 경우는 좀 다르지만 누구나 콘텐츠를 업로드할 수 있다. 검수 과정이 없어지거나 또는 간략화되면 질 낮은 애플리케이션이 유통될 수 있다는 단점도 있으나 많은 콘텐츠가 사용자들에게 다가갈 수 있다는 장점도 있다.

4.1 애플의 앱스토어

2008년 7월에 애플은 아이폰을 판매함과 동시에 앱스토어를 개장하였다. 이후 오픈 한달 만인 8월에 6,000만 번의 다운로드와 3천만 불의 판매 수익을 올렸다. 다시 한달 만인 9월에는 1억 번 다운로드를, 10월에는 2억 번의 다운로드라는 엄청난 결과를 달성하였다. 또 12월에는 등록된 애플리케이션이 1만 개를 넘어섰고 다운로드도 3억 번을 돌파했다고 한다. 이러한 추세는 2009년에도 이어져 2009년 1월에 5억회, 3월 중에 8억회를 돌파했다고 한다[11].

앱스토어에는 무료 애플리케이션도 있고 유료 애플리케이션도 있다. 유료인 경우는 1~5 달러를 받는 애플리케이션이 많다고 하니 3억 번의 다운로드를 생각하면 판매 금액은 상상을 초월한다. 다운로드 회수만을 보면 애플리케이션을 개발할 수 있는 개발자들에게는 희소식이 아닐 수 없다. 심지어 개발자들의 천국이라는 말이 나올 정도로 앱스토어에 대한 기대는 크다.

애플이 앱스토어를 히트시키자 블랙베리의 RIM도 블랙베리 앱센터(App Center)를 만들었고, PDA로 유명한 팜(Palm)도 소프트웨어스토어(Software Store)를 만들었다. 국내 SKT와 휴대폰 제조 업체인 삼성전자도 모바일 콘텐츠 직거래 장터를 구축하겠다고 한다.

4.2 구글 안드로이드 마켓

애플보다 늦긴 했지만 구글도 애플 앱스토어와 비슷한 구글 안드로이드마켓을 오픈해서 운영해오고 있다. 초기 안드로이드 마켓에서는 무료 애플리케이션만을 서비스하였는데 2009년 1사분기부터는 유료로 애플리케이션을

판매할 수 있다[12].

시장 조사 업체 스트래티지 어낼리틱스(SA)는 최근 발간한 보고서에서 안드로이드가 올해 휴대폰 제조 업체, 개발자, 이동통신 서비스 업체들의 지원을 등에 업고 무려 900%까지 성장할 것으로 전망했다. 그래서 2012년경에는 아이폰을 추월할 것으로 전망하였다.

이는 안드로이드의 개방성에 기인한다. 애플의 OS X와 그 위에서 동작하는 아이폰은 애플에서만 제조할 수 있어 다른 업체들이 참여할 기회가 크지 않다. 모든 수익이 애플에게만 돌아가는 구조이어서 다른 업체들이 안드로이드 진영에 참여할 가능성이 크다.

아무튼 모바일 콘텐츠 개발자들에게는 유료로 애플리케이션을 판매할 수 있는 또 다른 공간이 생기는 것이고, 이에 또 다른 기회가 펼쳐지고 있다. 안드로이드마켓이 활성화 되면 앱스토어에서 서비스되고 있는 많은 애플리케이션들이 안드로이드마켓에서도 판매될 것으로 예상된다.

4.3 두 시장의 비교

모바일 콘텐츠 시장은 앱스토어와 안드로이드마켓이 큰 흐름을 만들어 나가고 있는 것이 사실이다. 이에 두 시장을 비교해 보고자 한다.

먼저 개발에 필요한 SDK를 살펴보자. 아이폰 SDK는 애플에 개발 업체로 등록해야 SDK를 다운로드 받을 수 있다. 반면 안드로이드 SDK는 누구나 다운로드 받을 수 있다.

두 시장 모두 개발자는 판매 금액의 약 70%를 수익으로 가져갈 수 있다. 나머지 30%는 애플의 경우 수수료 명목으로 애플이 징수하고 있으며, 안드로이드마켓에서는 이동통신사가 30%를 수취하게 된다. 구글은 별도로 이익을 취하지 않고 있다. 이익을 추구하는 기업 입장에서는 용납되지 않는 일이다. 대신 구글은 수익을 창출할 수 있는 방법을 다른 곳에서 찾고 있다. 구글은 안드로이드마켓을 활성화시켜 모바일 광고 시장을 통해 수익을 얻겠다는 전략을 가지고 있다.

두 시장의 가장 큰 차이점은 애플의 앱스토어는 애플의 통제하에 운영되고 있고, 안드로이드마켓은 사용자에게 열려 있어 자율적으로 운영된다는 것이다. 따라서 애플의 게시 조건을 만족시키지 못하면 그 애플리케이션은 앱스토어에서 판매될 수 없다.

두 회사가 사용한 용어에서도 미세하게 차이를 느낄 수 있다. 애플은 상점이라는 뜻인 Store란 용어를 사용하고 있고, 구글은 장터의 의미를 갖는 Market이란 용어를 사용하고 있다. 마켓이라는 용어를 선택한 구글이 좀 더 열린 시장이라고 생각된다.

안드로이드마켓이 좀 더 개방적이라고 생각되는 사례는 또 있다. 애플에서는 인터넷 전화(VoIP)를 이동통신사와의 관계 때문에 허락하지 않고 있다. 반면, 구글에서는 인터넷 전화를 이용할 수 있는 애플리케이션이 등록되어 있는데 이 애플리케이션을 이용하면 비싼 전화 대신에 값싼 인터넷 전화를 이용할 수 있다.

다음은 지금까지의 애플의 앱스토어와 구글의 안드로이드마켓을 비교 정리한 것이다.

표 2. 앱스토어와 안드로이드 마켓의 비교

앱 스토어	비교대상	안드로이드 마켓
2008년 7월 11일	서비스 시기	2008년 8월 28일
애플	운영 주체	자율적으로
애플의 허락하에	애플리케이션 등록	자율적으로
등록해야만 가능	SDK 다운로드	누구나 가능
CP와 애플이 7:3	수익분배	CP와 이통사 또는 솔루션업체가 7:3
아이폰 2007년 6월 29일	폰 출시 시기	구글폰 2008년 10월 22일

5. 결론

WIPI는 이동통신사(SKT, KTF, LGT)와 관계없이 동일한 플랫폼을 사용함으로써 초창기 서로 다른 플랫폼 때문에 발생한 문제들을 해결해 주었다. 대표적인 것이 모바일 게임 업체들의 중복 개발을 막은 것이다.

또한 단일화된 플랫폼의 사용으로 모바일 산업 보호 효과 역시 컸다. 플랫폼의 국산화와 단일화로 퀄컴 등 해외로 빠져나가는 로열티를 줄였고, 외국산 단말기의 진입을 제한하여 삼성전자, 엘지전자 등 국내 단말기 업체들이 내수 시장을 지배할 수 있게 하였다. 이러한 측면에서 국내 단말기 시장 보호 역시 성공한 것으로 평가할 수 있다.

그러나 독자적인 국내 표준만 고집하던 WIPI는 글로벌 추세를 제때에 반영하지 못하면서 위기를 맞게 되었다. 문제로 지적되었던 것으로는 호환 문제, 국내 단말기 제조사만의 독점, 쉽지 않은 WIPI 플랫폼의 세계화 등이 있었다.

그 동안 삼성전자, LG전자 등은 WIPI 의무화 정책으

로 국내에서 특별한 보호를 받고 있었다. 그러나 WIPI 의무화 폐지로 다양한 외국산 단말기가 국내에 진출하게 되면, 외국 업체들과 치열한 경쟁을 벌여야만 한다.

WIPI 의무화 폐지는 단말기의 무한 경쟁 뿐 아니라 그 위에 올라가는 모바일 플랫폼의 치열한 경쟁을 초래할 것이다. 그동안 우리는 우리만의 표준에 집착하다 보니 해외 동향에 큰 관심을 가지지 못하였다. 그러나 해외의 다양한 모바일 플랫폼들이 국내에 들어 올 수 있어, 이제는 한 치 앞을 내다 볼 수 없는 공간 속으로 빠져들고 있다.

이렇게 급변하는 시대에 받아들여야 할 것이 있다면 받아들여야 하고 변신해야 할 것이 있다면 빠르게 변신해야 할 것이다. 질 좋은 콘텐츠가 많이 유통되어야 부흥할 수 있는 모바일 시장에서는 특히 안드로이드마켓과 같은 콘텐츠 유통방식은 관심있게 지켜봐야 할 것이다.

참고문헌

[1] AppleInsider, http://www.appleinsider.com/
[2] Open Handset Alliance, http://www.openhandsetalliance.com/
[3] What is Android?, http://code.google.com/intl/ko/android/what-is-android.html
[4] 안드로이드 SDK, http://code.google.com/intl/ko/android/intro/installing.html
[5] Anatomy of an Android Application, http://code.google.com/intl/ko/android/intro/anatomy.html
[6] Tutorial: A Notepad Application, http://code.google.com/intl/ko/android/intro/tutorial.html http://code.google.com/intl/ko/android/intro/appmodel.html
[7] Life Cycle of an Android Application, http://code.google.com/intl/ko/android/intro/lifecycle.html
[8] 주간기술동향, 정보통신연구진흥원, 2009년 4월
[9] Implementing a User Interface, http://code.google.com/intl/ko/android/devel/implementing-ui.html
[10] Android Building Blocks, http://code.google.com/intl/ko/android/devel/building-blocks.html
[11] 애플 앱스토어, http://www.apple.com/iphone/appstore/
[12] 안드로이드마켓, http://www.android.com/market/

저자소개

● 김 정 훈(JeongHoon Kim)

- 1991년 2월 : 서울시립대학교 컴퓨터통계학과 (이학사)
- 1994년 2월 : 연세대학교 컴퓨터과학과 (이학석사)
- 1994.01 ~ 1996.08 하이닉스 소프트웨어연구소 대리
- 1996.08 ~ 1999.06 현대정보기술 인터넷사업부 선임연구원
- 1999.07 ~ 2001.10 엔씨소프트 리니지토너먼트팀 팀장
- 2002.03 ~ 2003.08 소프트젠 모바일사업부 이사
- 2003.08 ~ 현재 용인송담대학 컴퓨터게임정보과 조교수

<관심분야> : 온라인 게임, 모바일 게임