Google Android -ohjelmistokehitys

Santeri Vaara & Visa Vaara
Tietotekniikan teemaseminaari, kevät 2011
Jyväskylän yliopisto

1. Esipuhe

Meillä kummallakaan ei juurikaan ollut aiempaa kokemusta Android-kehityksestä. Alusta alkaen oli melko selvää, että teemme seminaariesityksen mobiililaitteiden käyttöjärjestelmistä. Toisena vaihtoehtona meillä oli Applen iPhone OS. Päädyimme kuitenkin Androidiin aiemman Java-kokemuksen vuoksi. Opiskelutaustaa IT-puolelta meillä on suhteellisen vähän. Toinen meistä tekee tietotekniikan kandidaatin työtä ja toinen aineopintotason kokonaisuutta. Kuitenkin kiinnostus Javaa ja mobiiliohjelmointia kohtaan antoi valmiuden paneutua asiaan tarkemmin.

2. Johdanto

Tämä seminaariraportti kertoo Android-ohjelmistoalustan historiasta ja kuinka sille voidaan kehittää ohjelmia. Android on vielä varsin nuori ohjelmistoalusta, joten se kehittyy ja kasvaa varsin nopeasti. Android-marketin leviäminen luo jatkuvasti laajempaa näkyvyyttä Androidille. Androidin kehittämiseen pääsee helposti mukaan, koska se perustuu avoimeen lähdekoodiin. Lisäksi Android-pohjaisten laitteiden nopea yleistyminen tuo laitteet lähes jokaisen ulottuviin. Seminaariraportti johdattaa lukijan Androidin perusteisiin ja mahdollistaa ensiaskeleet Android-kehitysprojektien parissa. Lopun esimerkkikoodit auttavat kehityksen alkuunpääsemisessä. Tämä raportti ei ole tarkoitettu kaikenkattavaksi ohjeeksi vaan lähinnä helpottamaan alkuun pääsyn vaikeutta. Android-kehitysalusta mahdollistaa myös paljon monimutkaisempienkin rakenteiden toteuttamisen.

3. Historia

Android-ohjelmistoalustan kehitti alunperin yhdysvaltalainen Andy Rubin vuonna 2003. Rubin kumppaneineen päätti perustaa yhtiön Android Inc., joka toimi ohjelmistoalustan kehittäjänä aina vuoteen 2005, jolloin Google osti yrityksen. Kauppa aiheutti huhun, jonka mukaan Google oli siirtymässä mukaan mobiilimarkkinoille. [9]

Rubinin johdolla Android kehittyi nopeasti pohjautuen Linuxin kerneliin (Linux-ydin). Sen luvattiin olevan joustava ja helposti päivittyvä käyttöjärjestelmä. Vuoden 2006 lopulla huhut mobiilimarkkinoiden uudesta haastajasta voimistuivat ja käyttöjärjestelmä tuli suuremman joukon tietoisuuteen. Lopulta vuonna 2007 Open Handset Alliance, johon kuului useita suuria yrityksiä, kuten Google, HTC, Texas Instruments, LG ja Intel julkistivat ensimmäisen Android-käyttöjärjestelmään perustuvan puhelimen. Open Handset Alliancen idea oli tuoda avoin mobiiliohjelmistokehitys kaikkien ulottuville. [9]

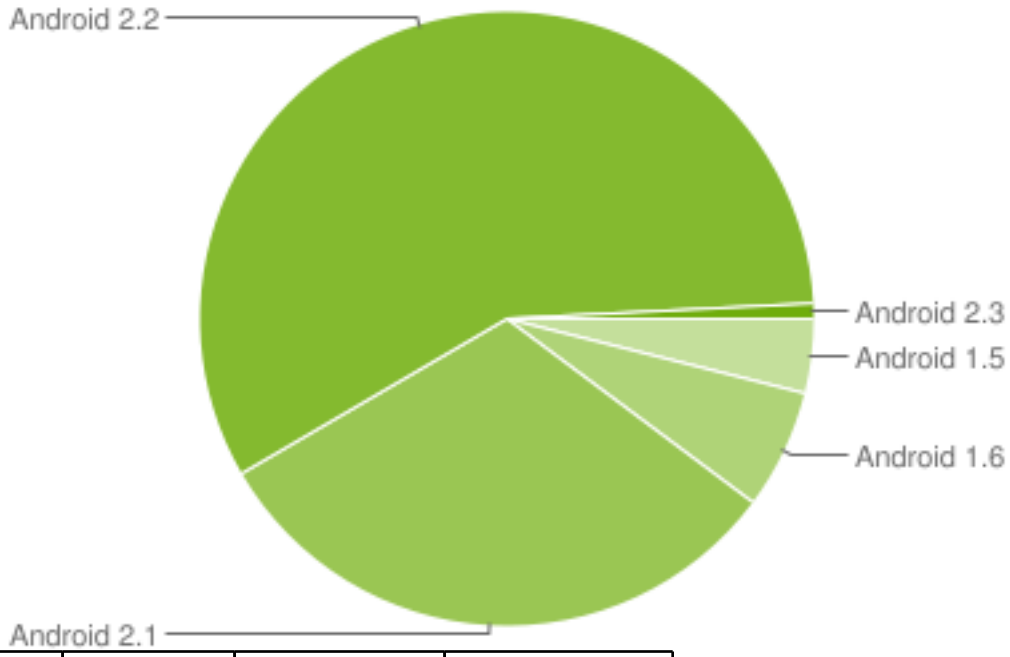
Nykyään Android-käyttöjärjestelmän suosio on voimakkaassa kasvussa ja tuoreimman uutisen mukaan se on syrjäyttänyt Black Berryn RIM-järjestelmän Yhdysvaltojen kärkipaikalta. Euroopassa Androidin suosio on myös kasvussa ja näyttää siltä, että siitä on tulossa lähiaikoina suosituin myös täällä. [11]

4. Eclipse + Android + Google Api - kehitysympäristö

Valitsimme kehitystyökaluksi Eclipsen, joka on meille tuttu aiemmilta ohjelmoinnin kursseilta. Android Developers-sivuilla oli selkeät ohjeet kuinka Android SDK ladataan ja asennetaan sekä ohjeet Eclipsen lataamiseen ja pluginin asentamiseen. SDK on mahdollista asentaa seuraaville alustoille Windows, Mac OS X (Intel) ja Linux (i368). Android Development Tools (ADT) pluginin asentaminen onnistuu Eclipsessä Help-valikosta Install new Software ja asettamalla osoitteeksi https://dl-ssl.google.com/android/eclipse/. [1]

Käyttöjärjestelmäversioita on useampia, joille ohjelmia voi kehittää. Uusin versio puhelimelle on Android 2.3, jota käyttää 0.8 prosenttia päätelaitteista. Tällä hetkellä yleisimmät käytössä olevat laitteet käyttävät Android 2.1 tai Android 2.2 käyttöjärjestelmää. Meidän testialustana on ZTE Blade-puhelin,

jossa pyörii Android 2.1. Tablet-kehityksessä mennään Android-versiossa 3.0. [1]



Platform	API Level	Distribution	Release date
Android 1.5	3	3.9%	30.4.2009
Android 1.6	4	6.3%	15.9.2009
Android 2.1	7	31.4%	12.1.2010
Android 2.2	8	57.6%	20.5.2010
Android 2.3	9	0.8%	6.12.2010

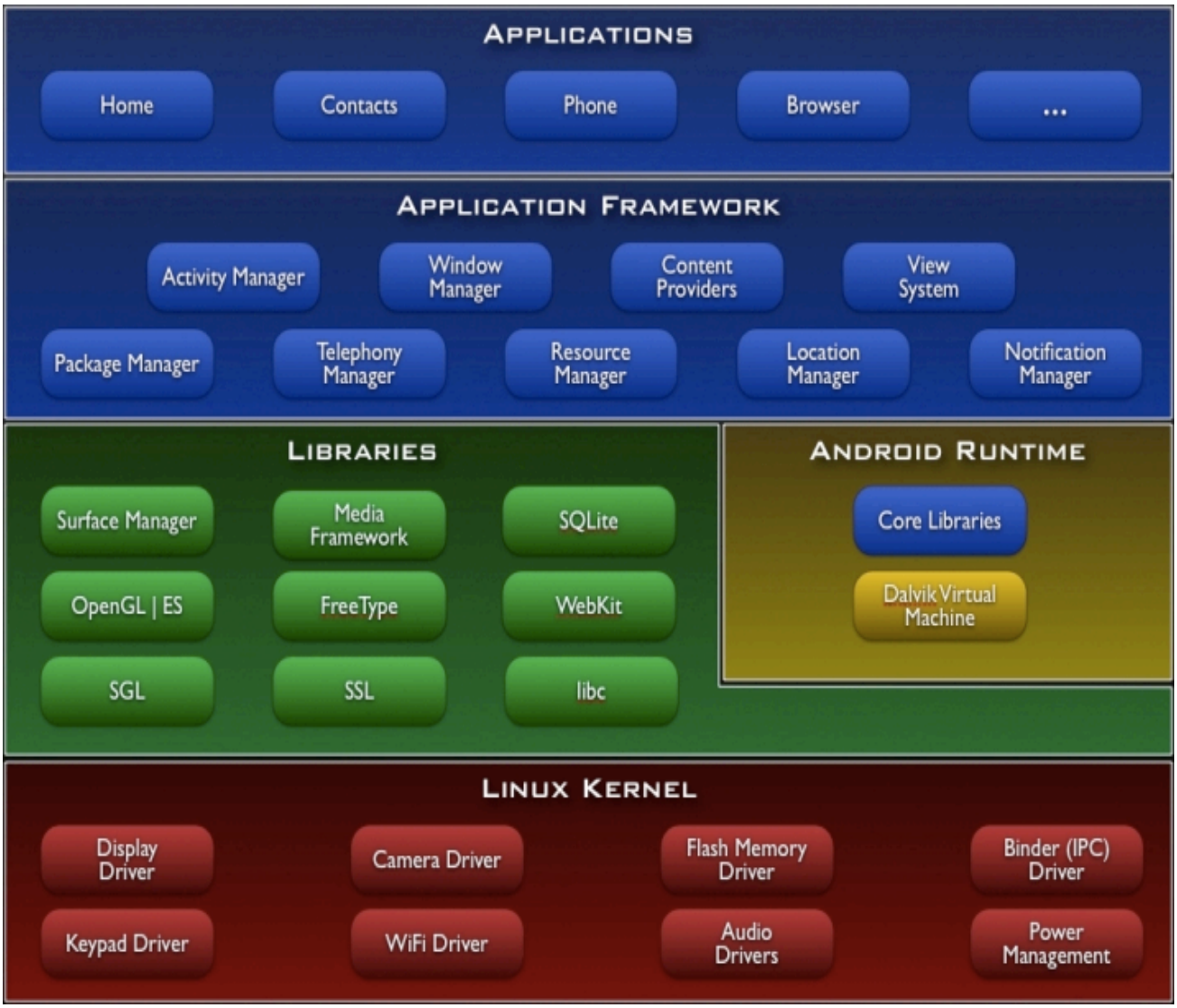
Kuva 1: Data collected during two weeks ending on February 2, 2011 [1, 12]

Ominaisuudet:

•**Application framework** mahdollisuus käyttää järjestelmän komponentteja ja ydinsovelluksia
•**Dalvik virtual machine** kääntäjä Java-luokille, optimoitu mobiililaitteille
•**Java support** tuki javalle
•**Integrated browser** perustuu avoimen lähdekoodin WebKit- ja v8 JavaScript-moottoreihin
•**Optimized graphics** perustuu räätälöityyn 2D-grafiikkakirjastoon. OpenGL tarjoaa 3D-grafiikkakirjaston
•**SQLite** relaatiotietokanta
•**Media support** yleisimmät ääni-, video- ja kuvaformaatit (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
•**GSM Telephony**
•**Bluetooth, EDGE, 3G, and WiFi**

•**Camera, GPS, compass, and accelerometer**
•**Multitasking** moniajo
•**Market for application** Android Market
•**Rich development environment** plugin Eclipse IDE:lle, laite-emulaattori, virheidenetsintätyökalu, muistin- ja suorituskyvynhallintatyökaluja [1]

Seuraava arkkitehtuurikaavio kertoo Androidin tärkeimmistä komponenteista:



Kuva 2: Android-arkkitehtuurikaavio [1]

Applications Järjestelmä tarjoa ydinsovelluksia(core applications), joita ovat esimerkiksi SMS, Contact ja Calendar. Ohjelmatason ohjelmat kirjoitetaan Java-kielellä.
Application framework Kehittäjät voivat vapaasti käyttää hyväkseen laitteistoa, paikkatietoa, ajaa taustapalveluita, asettaa hälytyksiä ja huomautuksia. Kehittäjillä on täydet oikeudet käyttää niin framework-ohjelmistoja kuin ydinsovelluksillakin. Tämä taso huolehtii muunmuassa aktiviteettien

elinkaaresta.
Libraries Android sisältää myös C/C++ kirjaston, joita voidaan hyödyntää ylemmän application frameworkin kautta. Android sisältää myös seuraavia kirjastoja;

- **System C library**
- **Media Libraries** - perustuu PacketVideon OpenCORE:n; kirjasto tarjoaa toistamisen ja nauhottamisen suosittuihin ääni- ja videoformatteihin, kuten MPEG4, H.264, MP3, AAC, AMR, JPG ja PNG
- **Surface Manager** - hallinnoi pääsyä näkymiin ja yhdistää 2D ja 3D grafiikkakerrokset
- **LibWebCore** - nettiselainmoottori, joka tarjoaa Android-selaimen sekä web wiev:n
- **SGL** -2D grafiikka moottori
- **3D libraries** -perustuu OpenGL ES 1.0 APIs; kirjastot käyttävät joko 3D-laitteistokiihdytintä tai optimoitua 3D-sovelluskiihdytintä
- **FreeType** - bittikartta ja merkkivektorien renderöintiin
- **SQLite** - tietokanta kaikkien ohjelmien käyttöön

Android runtime tarjoaa toiminnallisuuden Java-kirjastojen kautta. Jokainen Android-ohjelma on oma prosessinsa, jolla on oma tilansa Dalvikin Virtuaalikoneessa. Dalvik on toteutettu siten, että voidaan suorittaa useita tiloja tehokkaasti ja hyödyntää tehokaasti muistin käyttöä. Virtuaalikone on rekisteripohjainen ja ajaa käännettyjä Java-luokkia.
Linux Kernel Android pohjautuu Linux versioon 2.6, jossa on ydinpalvelut: turvallisuus, muistin hallinta, prosessien hallinta, verkkopino ja ajurit. Kernel toimii myös abstraktiokerroksena laite- ja sovelluskerroksen välissä. [1]

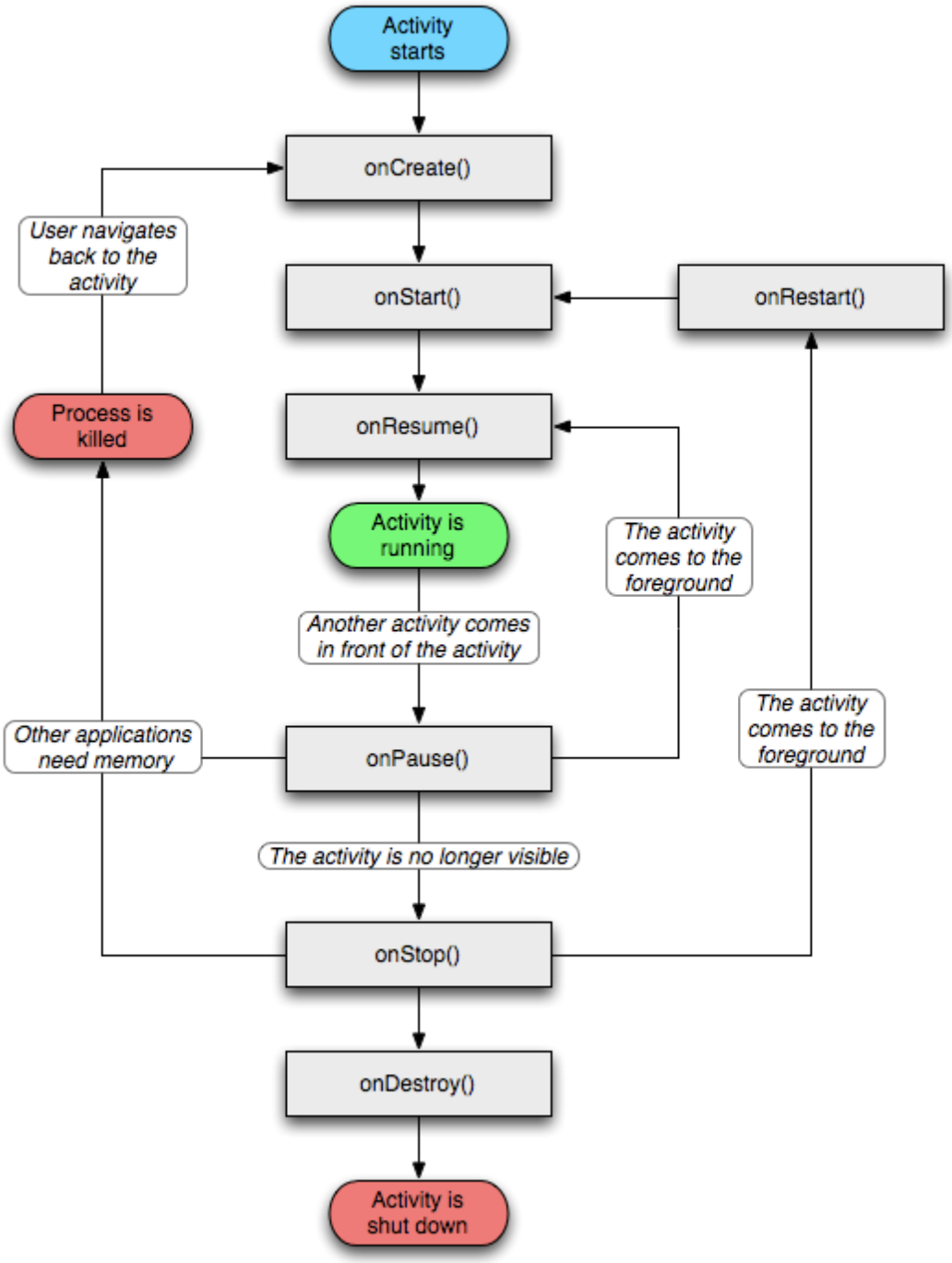
AndroidManifest.xml

Android sovellukset kuvataan AndroidManifest.xml tiedostossa. Tässä tiedostossa tulee kertoa kaikki sovelluksen käyttämät aktiviteetit, palvelut, yleisvastaanottajat (broadcast receivers) ja sisällöntuottajat (content providers). Tässä myös määritellään ohjelman tarvitsemat oikeudet, kuten pääsy Internettiin. AndroidManifest.xml voidaan ajatella olevan kuvaus Android-sovelluksesta. [2]

Aktiviteetit

Aktiviteetti on näkymä käyttäjälle. Aktiviteettien kautta käyttäjä on vuorovaikutuksessa laitteiston kanssa. Activity-luokka huolehtii ikkunoimisesta, johon kehittäjä voi luoda oman käyttöympäristön. Normaalisti aktiviteetti käyttää koko ruutua, mutta sillä on myös muita mahdollisuuksia — esimerkiksi luoda kelluvia näkymiä.

Kaikki aktiviteetit tulee määritellä AndroidManifest.xml tiedostossa. Activity-luokka on tärkeä osa ohjelman elinkaaren kannalta. Kun aktiviteetti käynnistyy ensimmäisen kerran, kutsutaan onCreate-metodia. Aktiviteetin sisällä voidaan avata uusia aktiviteetteja. Alla oleva kuva näyttää aktiviteettien elinkaaren aina onCreate-metodista onDestroy-metodiin asti. [2]



Kuva 3: Aktiviteetin elämänkaari [1]

Context

Luokka android.content.Context tarjoaa yhteyden Android järjestelmään. Se on rajapinta sovellusympäristössä. Context tarjoaa myös getSystemService-metodin, joka mahdollistaa eri laitteiston palvelujen hallinnan, kuten vibraattorin ja virranhallinnan. Aktiviteetit ja palvelut periytyvät Context-

luokasta. Context-luokkaa voidaan kutsua suoraan kutsumalla this-oliota. [2]

Intent

Intentit ovat asynkronisia viestejä, joilla voidaan välittää tietoa aktiviteettien ja palveluiden välillä. Intent-olio voi sisältää tietoa vastaanottavista komponenteista. Esimerkiksi kutsuttaessa Intent-olion kautta selainta, voidaan URL välittää suoraan selainkomponenttiin Intent-olion parametrinä. Intent-olio voi myös kuvata halutun toiminnon. Android-järjestelmä tarjoaa ratkaisut toiminnon toteuttamiseen. Esimerkiksi lähetettäessä sähköpostia, Intent-olio sisältää tiedon sähköpostin lähettämisestä ja järjestelmä tarjoaa kaikki sähköpostin lähettämisen mahdollistavat ohjelmat, joista käyttäjä valitsee itse haluamansa.

Android tarjoaa määritellyn intentin ja kuvaavan intentin. Määritelty intentti nimeää komponentin, esimerkiksi Java-luokan, jota kutsutaan. Kuvaava intentti pyytää järjestelmältä palvelua, esimerkiksi avaamaan selaimen. Android-järjestelmä määrittelee sopivan sovelluksen. Jos vaihtoehtoisia on tarjolla, järjestelmä tarjoaa käyttäjälle listan vaihtoehdoista. Tämä määrittely perustuu IntentFilttereihin. IntentFiltterit kuvataan AndroidManifest.xml-tiedostossa. Komponentti käyttää IntentFiltteriä kuvaamaan, minkä tyyppisistä toiminnoista se vastaa. Jos IntentFiltter-määritystä ei ole tehty, Intent-olion tulee olla määritelty. [2]

Emulaattori

Android SDK tarjoaa laite-emulaattoreita sovelluksien koeajoon ja debuggaukseen. Emulaattorin voi valita laitetyypin ja näytön koon mukaan. Emulaattoria käytetään hiirellä ja toiminta vastaa täysin fyysistä Android-laitetta.

Kuva 4: Android-emulaattori

Android-market ja matka Pattayalle voi alkaa

Android-sovelluksia voi myydä Android-marketissa samaan tapaan kuin Applen AppStoressa tai Nokian Ovi-kaupassa. Aluksi kehittäjän tulee luoda Google Market-tili, jonka jälkeen sovellusten ilmainen jakaminen on mahdollista. Jos sovellukselleen haluaa määrittää hinnan, tulee rekisteröidä Merchant-tili marketin kehittäjäsivuston kautta. Aloitusmaksu on 25 dollaria. Tämän tarkoituksena on pitää huolta siitä, että sovellusten taso pysyy hyvänä. [10]

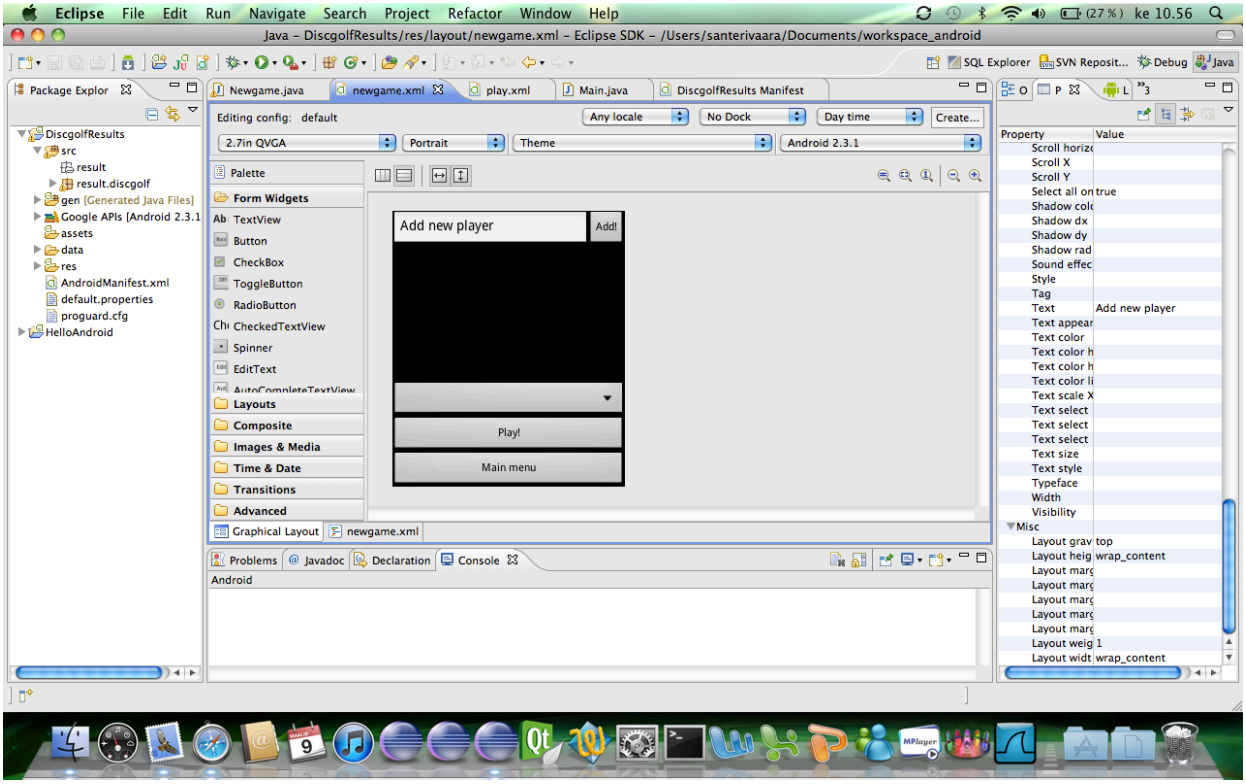
Sovelluksen voi laittaa markettiin ladattavaksi mistä maasta tahansa, mutta sovelluksen myyminen ei onnistu vielä läheskään kaikkialta. Tällä hetkellä sovelluksia voi asettaa myyntiin Android-markettiin seuraavista maista: Argentiina*, Australia, Itävalta, Belgia, Brasilia*, Kanada, Tanska, Suomi, Iso-Britannia, Itävalta, Hongkong, Irlanti, Israel*, Italia, Japani, Meksiko*, Alankomaat, Uusi-Seelanti, Norja, Portugali, Venäjä*, Singapore, Espanja, Etelä-Korea*, Ruotsi, Sveitsi, Taiwan*, Iso-Britannia, Yhdysvallat. [10]

*Näissä maissa sijaitsevien kauppiaiden tulee luoda ja linkittää AdSense-tili voidakseen vastaanottaa maksuja.

5. Frisbeegolf Results -sovellus

Päätimme tehdä harrastuksen innoittamana ohjelman frisbeegolf-tulosten kirjaamiseen. Android-marketista löytyy jo muutama ohjelma tätä tarkoitusta varten, mutta niiden ominaisuudet ovat riittämättömät. Ideana oli lisätä ohjelmaan myös Google Maps ja siihen ratojen sijainnit. Lisäksi päädyimme liittämään jokaiseen rataan myös nettisivun, joka aukeaa info-nappia painamalla. Tällöin ohjelma sisältää kaikki komponentit, mitä pelaaja saattaa tarvita pelaamisen ohella.

Itse kehitys lähti liikkeelle ensimmäisen näkymän suunnittelusta. Eclipseen saatavan ADT plugin mukana tulevalla XML-editorilla näkymien suunnittelu onnistuu suhteellisen helposti, varsinkin jos on aikaisempaa kokemusta vastaavan editorin käytöstä. Meillä sitä ei ollut, joten alku oli varsin hankala. XML-komponenttien lisääminen onnistuu näytön vasemmalla olevasta palette-valikosta. Kun komponentti on lisätty näkymään, sen määrityksiä voi muuttaa oikealle avautuvasta properties-listauksesta. Jokaiselle komponentille pitää määrittää id, jonka avulla sitä voidaan kontrolloida aktiviteetti-luokassa.



Kuva 5: Eclipsen XML-layout -editori

Esimerkki XML-tiedostosta

Yllä olevan XML-editorissa näkyvän näkymän generoima XML-koodi:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
        android:layout_height="fill_parent" android:layout_width="fill_parent"
        android:orientation="vertical">
        <LinearLayout android:layout_height="wrap_content" android:id="@+id/linearLayout1"
                android:focusable="true" android:focusableInTouchMode="true"
                android:layout_gravity="top" android:layout_width="match_parent">
                <EditText android:layout_gravity="top" android:id="@+id/addplayerEditText"
                        android:text="Add new player"
                        android:layout_height="wrap_content" android:selectAllOnFocus="true"
                        android:layout_width="wrap_content"
                                android:layout_weight="1"></EditText>
        <Button android:layout_gravity="top" android:layout_height="wrap_content"
                        android:id="@+id/addPlayerButton" android:text="Add!"
                        android:layout_width="wrap_content"></Button>
        </LinearLayout>
        <ListView android:id="@+id/playersListView"
                android:layout_width="fill_parent" android:layout_height="wrap_content"
                android:layout_weight="0.6">
        </ListView>
        <LinearLayout android:id="@+id/LinearLayout01"
                android:layout_width="fill_parent" android:orientation="vertical"
                android:layout_height="wrap_content">
                <Spinner android:layout_gravity="bottom" android:layout_width="fill_parent"
                        android:layout_height="wrap_content"
                        android:id="@+id/setcourseSpinner"></Spinner>
                <Button android:layout_gravity="bottom" android:layout_height="wrap_content"
                        android:id="@+id/playButton" android:layout_width="match_parent"
                        android:text="Play!"></Button>
                <Button android:layout_gravity="bottom" android:layout_height="wrap_content"
                        android:id="@+id/mainmenuButton" android:layout_width="fill_parent"
                        android:text="Main menu"></Button>
        </LinearLayout>
</LinearLayout>
```

R-luokka

Kun näkymä on saatu suunniteltua, tulee komponentit liittää toiminnalliseen Java-koodiin. Tämä onnistuu Eclipsen generoiman R-luokan kautta. R-luokka sisältää kaikkien XML-koodissa käytettävien komponenttien, layouttien ja vakiomerkkijonojen id:t.

Aktiviteetti-esimerkki

Aktiviteetin eli Java-luokan rakentaminen alkaa pakollisten metodien suunnittelusta ja luomisesta. Aktiviteetti periytetään Activity-luokasta, joka sisältää jokaiselle aktiviteetille olennaiset yhteiset ominaisuudet ja toiminnot. Halutut nappulat, tekstikentät ja listat määritetään luokan attribuuteiksi.

Jokaisen aktiviteetin pitää sisältää onCreate-metodi, joka määrittää mitä tapahtuu, kun aktiviteetti ensimmäisen kerran käynnistetään. onCreate-metodin sisällä ensimmäiseksi luodaan haluttu layout eli XML-editorissa muotoiltu näkymä. Tämän jälkeen alustetaan attribuutit XML-komponenteiksi R-luokan avulla ja määritetään mahdolliset kuuntelijat ja vastaavat toiminnallisuudet. onCreate-metodin lisäksi aktiviteettien elinkaareen kuuluu muitakin aktiviteetin tilaan liittyviä metodeja, kuten onPause, onDestroy ja onRestart. Näitä metodeja ei kehittäjän tarvitse välttämättä itse määrittää. Kutsut menevät yläluokalle, joka sisältää tarvittavan informaation niiden toiminnasta. Tarvittaessa kehittäjä voi kuitenkin määrittää erikoistoimintoja eri tiloihin luomalla itse kyseiset metodit. Discgolf Results -sovelluksessa ei toistaiseksi ole tarvinnut määrittää kustomoituja toimintoja muille kuin onCreate-metodille.

Kun XML-koodissa oleviin komponentteihin on saatu yhteys, voidaan niille määrittää toiminnallisuus onClick-metodin avulla. Alla olevassa esimerkissä nappeja newgame ja about painamalla avautuu uusi näkymä Intent-luokan avulla. quit-nappia painamalla aktiivisena oleva näkymä sulkeutuu ja siten myös koko ohjelma sulkeutuu.

```
package result.discgolf;

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

/**
 * Main activity class. Program starts from this class!
 * @author team Danger
 * @version 0.1 7.2.2011
 *
 */
public class Main extends Activity implements OnClickListener {

	private Button newgame, loadgame, addcourse, settings, quit, about;
	private static final String TAG = "GolfScoreResults";
	private Intent act;

	/** Called when the activity is first created. */
	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main);
```

```
        Log.d(TAG, "onCreate started");

        //Initializing attributes
        newgame = (Button)findViewById(R.id.newgame);
        loadgame = (Button)findViewById(R.id.loadgame);
        addcourse = (Button)findViewById(R.id.addcourse);
        settings = (Button)findViewById(R.id.settings);
        quit = (Button)findViewById(R.id.quit);
        about = (Button)findViewById(R.id.about);

        //Adding listeners to attributes
        newgame.setOnClickListener(this);
        loadgame.setOnClickListener(this);
        addcourse.setOnClickListener(this);
        settings.setOnClickListener(this);
        quit.setOnClickListener(this);
        about.setOnClickListener(this);

        Log.d(TAG, "onCreate finished");
}

/**
 * On click method. Reacts when user clicks a button.
 */
public void onClick(View src) {
        switch (src.getId()) {
        case R.id.newgame:
                act = new Intent(this, Newgame.class);
                startActivity(act);
                break;
        case R.id.loadgame:
                break;
        case R.id.addcourse:
                break;
        case R.id.settings:
                break;
        case R.id.quit:
                finish();
                break;
        case R.id.about:
                act = new Intent(this, About.class);
                startActivity(act);
                break;
        }
```

```
	}
}
```

Google Maps -esimerkki

Tämä on esimerkki siitä, miten Google Mapsin saa käyttöön omaan sovellukseen. AndroidManifest.xml tiedostoon tulee lisätä <uses-library android:name="com.google.android.maps" /> -rivi, jolloin saadaan käyttöön maps-kirjasto.

```
package result.discgolf;

import com.google.android.maps.MapActivity;
import com.google.android.maps.MapView;
import android.os.Bundle;

/**
 * GMaps activity controls the mapview.
 * @author team Dangerous
 * @date 21.2.2011
 */
public class GMaps extends MapActivity {

	private MapView mapView;

	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.gmaps);

		//Initializing attributes
		mapView = (MapView) findViewById(R.id.mapview);
		mapView.setBuiltInZoomControls(true);
}
	@Override
	protected boolean isRouteDisplayed() {
		return true;
	}
}
```

Alla olevassa XML-tiedostossa tulee määritellä android apiKey. Google-tilin omaavat käyttäjät saavat haettua Android Maps API Keyn syöttämällä palveluun oman sertifioidun sormenjälkensä (MD5). Ohjeet ja hakulomake löytyy osoitteeesta: http://code.google.com/intl/fi/android/maps-api-signup.html

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
	android:orientation="vertical" android:layout_width="fill_parent"
	android:layout_height="fill_parent">
	<com.google.android.maps.MapView android:id="@+id/mapview"
		android:layout_width="fill_parent" android:layout_height="fill_parent"
		android:apiKey="Oma avain tulee tähän!" />
</RelativeLayout>
```

WebView -esimerkki

WebView-luokka tarjoaa kehittäjälle selaintoiminnot. Alla olevassa esimerkissä luodaan WebView-näkymä ja asetetaan siihen osoitteeksi http://www.fgr.fi.

```
package result.discgolf;

import android.app.Activity;
import android.os.Bundle;
import android.webkit.WebView;

public class WebBrowser extends Activity {
	private WebView www;

	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.webbrowser);

		www = (WebView) findViewById(R.id.www);
		www.getSettings().setJavaScriptEnabled(true);
		www.loadUrl("http://www.fgr.fi");
	}
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
	android:layout_width="match_parent" android:layout_height="match_parent">
	<WebView android:layout_width="match_parent" android:layout_height="match_parent"
	android:id="@+id/www"></WebView>
</LinearLayout>
```

AndroidManifest.xml -esimerkki

AndroidManifest.xml-tiedostossa määritellään aktiviteetit, palvelut ja erilliset kirjastot. Alla olevassa erimerkissä on tuotu Google Maps -kirjasto sekä sallitaan sovelluksen käyttävän yhteyttä Internettiin.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
	package="result.discgolf" android:versionCode="1" android:versionName="0.1">
	<application android:icon="@drawable/icon" android:label="@string/app_name"
		android:debuggable="true">
		<activity android:name="Main" android:label="@string/app_name">
			<intent-filter>
				<action android:name="android.intent.action.MAIN" />
				<category android:name="android.intent.category.LAUNCHER" />
			</intent-filter>
		</activity>
		<activity android:name="About">
		</activity>
		<activity android:name="Newgame">
			<intent-filter>
				<action android:name="android.intent.action.Newgame" />
				<category android:name="android.intent.category.LAUNCHER" />
			</intent-filter>
		</activity>
		<activity android:name="Play">
			<intent-filter>
				<action android:name="android.intent.action.Play" />
				<category android:name="android.intent.category.LAUNCHER" />
			</intent-filter>
		</activity>
		<activity android:name="WebBrowser">
			<intent-filter>
				<action android:name="android.intent.action.WebBrowser" />
				<category android:name="android.intent.category.LAUNCHER" />
			</intent-filter>
		</activity>
		<activity android:name="GMaps">
			<intent-filter>
				<action android:name="android.intent.action.GMaps" />
				<category android:name="android.intent.category.LAUNCHER" />
			</intent-filter>
		</activity>
```

```
            <uses-library android:name="com.google.android.maps" />
        </application>
        <uses-permission android:name="android.permission.INTERNET" />
</manifest>
```

6. Yhteenveto

Android ohjelmointiin pääsee helposti käsiksi Android developer sivuston esimerkkien kautta. Sieltä löytyvät ohjeet niin Android SDK:n lataamiseen kuin Android Development Toolkitin (ADT) asennukseen Eclipseen. Kehitysympäristö antaa hyvät valmiudet luoda kilpailukykyisiä ja toimivia ohjelmia. Lisäksi tuki SQL-tietokannalle on erittäin hyvä lisä. Periaatteessa tietorakenteen voi myös rakentaa itse, mutta sen toimivuuden kannalta on suositeltavampaa käyttää valmista SQLite-relaatiotietokantaa.

Perustaidot Java-ohjelmoinnista ja tietämys Eclipsen toiminnasta ovat tarpeen kehityksen alkuunpääsemisessä. Lisäksi on hyvä tietää jotain XML-kielestä layoutteja tehdessä. Suurimmat ongelmat alussa meillä oli juuri XML-koodin ja XML-editorin toiminnan ymmärtämisessä. Toiminnallinen Java-koodi on pääosin puhdasta Javaa. Android-kirjasto tuo valmiita rakenteita mukavasti kehittäjän saataville ja näin ollen vähentää tarvetta rakentaa komponentteja itse. Jonkin verran kohtasimme ongelmia kehitysympäristön kanssa. Muutaman kerran Eclipse väitti, että projektissa ja jossain tietyssä luokassa on virhe, vaikka loppujen lopuksi näin ei ollutkaan. Eclipsen ja koko käyttöjärjestelmän uudelleenkäynnistys on siis jopa suositeltavaa aika ajoin tällaisten haamuvirheiden estämiseksi. On erittäin epämukavaa etsiä virhettä koodista, mitä ei oikeasti ole olemassa.

Kaiken kaikkiaan Eclipse hyöstettynä Android-työkaluilla on pätevä ympäristö Android-sovellusten kehittämiseen. XML-editori helpottaa näkymien suunnittelua ja mahdollistaa niiden tekemisen vaikka ei aikaisempaa XML-kokemusta juurikaan olisi. Yksinkertaisten ohjelmien tekeminen on suhteellisen helppoa. Monimutkaisempien ohjelmien tekemisessä pitää Android-kirjaston komponentteihin tutustua hieman tarkemmin.

5. Lähteet

1. Android-kehittäjien sivusto
   http://developer.android.com/index.html
2. Lars Vogel, Android Development Tutorial - Gingerbread
   http://www.vogella.de/articles/Android/article.html
3. Android ListView Multiple Choice Example http://www.androidpeople.com/android-listview-multiple-choice-example/
4. Mathias Reisch, Episode #11 – Intents: Multi Activity Applications
   http://www.xtensivearts.com/2010/04/16/episode-11-intents-multi-activity-applications/
5. James Reed, Android App Course, Lab 5, Google Maps esimerkki
   https://sites.google.com/site/androidappcourse/labs/lab-5
6. Shane Conder & Lauren Darcey, Android User Interface Design: Linear Layouts
   http://mobile.tutsplus.com/tutorials/android/android-sdk_linear-layouts_2/

7. Fabrizio Chami, Android Google Maps Tutorial
http://www.javacodegeeks.com/2011/02/android-google-maps-tutorial.html
8. Lars Vogel, Location API and Google Maps in Android - Tutorial
http://www.vogella.de/articles/AndroidLocationAPI/article.html#device_setting%20the%20geoposition
9. Simon Hill, History of Android
http://www.brighthub.com/mobile/google-android/articles/18260.aspx
10. Android Market kehittäjille
https://market.android.com/support/
11. comScore: Android now 33% of US smartphones, iPhone still up
http://www.electronista.com/articles/11/04/01/comscore.says.android.at.33pc.iphone.up.february/
12. Wikipedia
http://fi.wikipedia.org/wiki/Android
13.