Beteckning:_____

HÖGSKOLAN
I GÄVLE

Akademin för teknik och miljö

Applikationsutveckling för Android

Fredrik Andersson
VT 2012

Examensarbete, 15hp, nivå B
Datavetenskap Internetteknologi

Examinator Carina Pettersson
Handledare Ann-Sofie Östberg

Applikationsutveckling för Android

av

Fredrik Andersson

Akademin för teknik och miljö
Högskolan i Gävle

S-801 76 Gävle, Sweden

Email:
Freddeandersson86@hotmail.com

Abstrakt

Detta projekt handlar om ett applikationsutvecklingsprojekt för Androidplattformen vilket innefattar Androids uppbyggnad, Javaprogrammering, design i XML miljö, grunder i Eclipse samt publicering på Google Play (före detta Android market). Projektet resulterar i en Androidapplikation i form av ett enkelt Pongspel med olika funktioner som highscorelista samt att uppsatsen beskriver det man behöver kunna för att utveckla applikationer för Android.

Innehåll

1 Inledning

1.1 Arbetets syfte

Mobila enheter har under de senaste åren blivit allt mer vanliga i vår vardag.
Android är ett av operativsystemen som är utvecklat just för att hantera dessa enher på bästa möjliga sätt, och i takt med detta så har även applikationer för detta operativsystem blivit allt mer populärt att ladda ned, eller utveckla själv, då många ser det som en väldigt stor marknad och möjlig inkomstkälla.
Syftet med detta projekt är att lära sig utveckla och förstå hur Android applikationer fungerar från grunden.

Applikationsutvecklingen kommer att handla om ett litet spel för Android, ett spel som nog de flesta känner till "Pong". Spelet ska styras med pekskärmen och den grafiska motorn kommer vara i enkel 2D, då 3D grafik är mycket mer avancerad att utveckla.

1.2 Frågeställning

1. Bygger Android applikationen helt och hållet på XML i sitt användargränssnitt?

2. Behöver applikationen någon form av informationslagring som highscore värden synligt på nätet via webservices?

3. Vilken typ av målgrupp av användare är Androidapplikationen främst riktad mot?

4. Kan man använda andra programmeringsspråk än Java för att utveckla Android applikationer?

5. Behöver man ha någon rekommenderad Androidenhet för att använda applikationen?

2 Tekniska termer bakom projektet

2.1 Eclipse

Eclipse är en texteditor som många programmerare använder när de utvecklar applikationer. Med hjälp av Eclipse får man bland annat mycket bra hjälp för felsökningar samt att man får en bra formulerad struktur i sin kodning. Eclipse kan användas till många olika programmerinsspråk, där Java, C++ och C# är några av dem. Texteditorn är också helt gratis och finns att ladda ned på www.eclipse.org. Rapporten kommer gå igenom lite mer om Eclipse för Android längre fram.

2.2 XML

XML är förkortning på Extensible Markup Language och används till stora delar för designuppbyggnaden för Androidapplikationer. Man kan med hjälp av XML skapa formulär, knappar, bildrutor, tabeller och mycket annat.
I övrigt används XML för att man på ett enkelt sätt kunna skicka och ta emot datainformation mellan olika typer av datasystem.
XML är likt HTML uppbyggt av "taggar" och är ett väldigt enkelt scriptspråk att lära sig. Läs mer om XML på www.w3schools.com om ni är intresserade av detta.

2.3 Java

Java är ett programmeringsspråk på hög nivå, och används till väldigt många olika typer av program och enheter.
Till skillnad från andra programmeringsspråk som C++ eller C# så är Java "plattformsoberoende", vilket betyder att de program som utvecklas i Java fungerar även på andra enheter och operativsystem än det system man själv utvecklade applikationen på, exempel på operativsystem kan vara Windows, Mac, Linux mm.

Java är uppbyggt av såkallade programpaket och bibliotek som innehåller mängder av olika fördefinierade metoder och hjälpmedel som behövs vid programmering.
Mer information om Java finns att läsa på internet på sidan www.oracle.com.

2.4 Android SDK

Genom att ladda ned och installera Android SDK, får man tillgång till allt som har med Androidutveckling att göra. Android SDK är ett tilläggspaket för Java, och innehåller de speciella biblioteken för Android utveckling. Det är med hjälp av Javas grundfunktioner och Android SDK som gör det möjligt för oss att skriva applikationer för Android. Android SDK kan man ladda ned från http://developer.android.com/sdk/index.html [1].

På den sidan hittar man även dokumentationen för hela Androidpaketet som kan vara till bra hjälp.

2.4.1 Android Virtual Machine

Denna uppskattade modul följer med tillsammans med Android SDK biblioteket och gör det möjligt för utvecklare att testa Androidapplikationer direkt på datorn utan att behöva ansluta till en enhet som kör Android. Man testar då helt enkelt hur applikationen som är under utveckling fungerar på en virtuell Android enhet.

2.5 Photoshop

Photoshop är ett poppulärt bildredigeringsprogram som många fotografer och designers i hela världen använder sig utav. Photoshop är utvecklat av företaget Adobe och en licens för Photoshop kostar pengar men man kan även ladda ned en tidsbegränsad testverison på 30 dagar helt gratis
Ladda ned Photoshop på www.adobe.com.

2.5.1 Filtypen .png

En speciell filtyp för digitala bilder. Det speciella med denna filtyp till skillnad från .jpeg och .gif är att man inte förlorar någon kvalitet på själva bilden man arbetar med under sin redigering. Det är också möjligt att använda sig av transparenta bakgrunden när man arbetar med .png, vilket kan vara till stor fördel många gånger.

2.6 SQLite Databas

SQLite är namnet på databasen som används för Android och även Iphone. Med hjälp av en databas kan man lagra och hämta datainformation som sparats sedan tidigare. Informationen som behandlas på databasen kan vara personuppgifter för olika användare på en websida eller en lista över en webshops alla produkter.
Databaser används i princip alltid i någon form till varje program eller websida i dagens läge. När man arbetar med databaser så bör man behärska språket för att komminucera med en databas. Detta språk kallas SQL, och är ett helt kapitel för sig. En väldigt bra internetguide för att lära sig SQL finns på:
www.thenewboston.com - tutorials – MySQL [2]

2.7 Linux

Linux är ett mycket smidigt operativsystem likt Windows och Mac OS, den stora fördelen med Linux är att det är helt gratis för alla att ladda ned och installera på sina datorer, vilket har gjort Linux till ett väldigt populärt växande operativsystem genom åren. Android bygger sin grund på just Linux stabila kärna, som är själva "grundmotorn" för Android. Denna rapport täcker inte Linux, men det är bra att veta att grunden till Android bygger på Linux när man utvecklar sina applikationer.

2.8 Google Play

Med hjälp av Google Play kan man publicera sina Androidapplikationer för hela världen, så att andra användare kan ladda ned dem på sina egna Androidenheter. Tidigare fanns Android market, men uppdaterades och bytte namn till Google Play våren 2012.

3 Genomförande

3.1 Förberedelser

Projektet startade med att först och främst lära sig till stora drag hur man utvecklar sina egna Androidapplikationer.
Tiden gick därför åt för att studera Androidutveckling genom internetguider samt läsa böcker.

Ett väldigt bra sätt att förbereda sig innan man börjar utveckla i Android, är att skaffar sig grundkunskaper i vanlig Javaprogrammering.
Att studera xml kan också vara till stor fördel, men detta språk är mer lättförståerligt än Java och man lär sig xml ganska snabbt bara genom att börja arbeta med det.

Rekomenderade internetguider för nybörjare inom Java och Android:
www.Thenewboston.com - tutorials – Android for beginners [3]

3.2 Nedladdning av programvaran

Efter lite grundstudier är det dags att ladda ner de programvaror som behövdes för att utföra projektet.

* Ladda ned och installera Java for developers.

* Ladda ned och installera Eclipse - *"Eclipse for java developers"*.

* Ladda ned och installera Android SDK paketet.

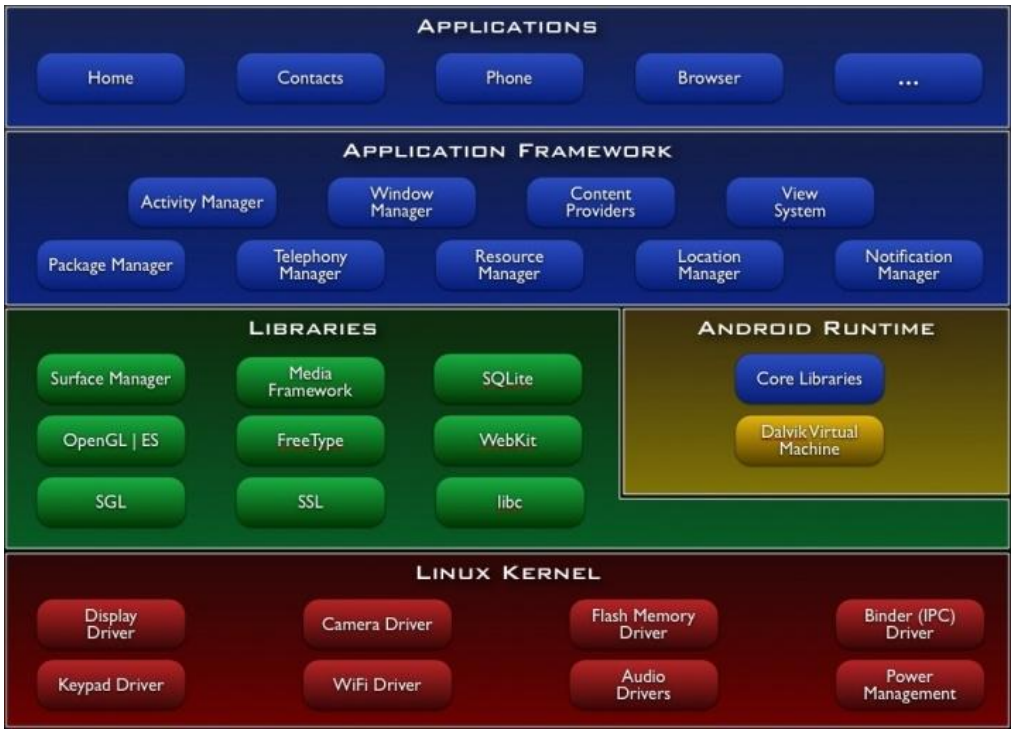
3.3 Android Arkitekturen

Operativsystemet Android bygger sin arkitektur i fyra olika lager där vi hittar Linux kärnan i det understa lagret och agerar grundstommen i hela operativsystemet.
Linux kärnan hanterar alla drivrutiner mot hårdvaran som grafik, ljud, WiFi, kamera samt drivrutiner för knappsats och pekskärm.

Sedan kommer biblioteks lagret, och där hittar man alla olika Javaklasser som är i grunden för Androidplattformen. Här hittar man bland annat databashanteringen, OpenGL klasser för 3D grafik, Audio klasser för uppspelning av musik och ljud, samt ett specielt lager som har hand om själva hanteringen för Androids kärnbibliotek, dvs de klasser som gör det möjligt för Android operativsystemet att köras. Detta lager arbetar tillsammans med det vanliga bibliotetkslagret mot det tredje lagret i arkitekturen.

Det tredje lagret beskriver alla de olika enheter som hanteras av operativsystemet, och tillåter sedan alla applikationer att använda sig utav dem. Det kan vara tillgång till gpshantering, enhetsresurser, telefonfunktioner, och hantering för att rita upp en grafiskt layout.

Det översta lagret i arkitekturen är alla olika applikationer som en användare kan använda som webbläsare, spel, kamera applikationer mm. Detta lager är det enda man ser som användare, och med hjälp av det underliggande lagret så kan alla applikationer använda sig av alla de olika funktioner som Androidsystemet erbjuder.

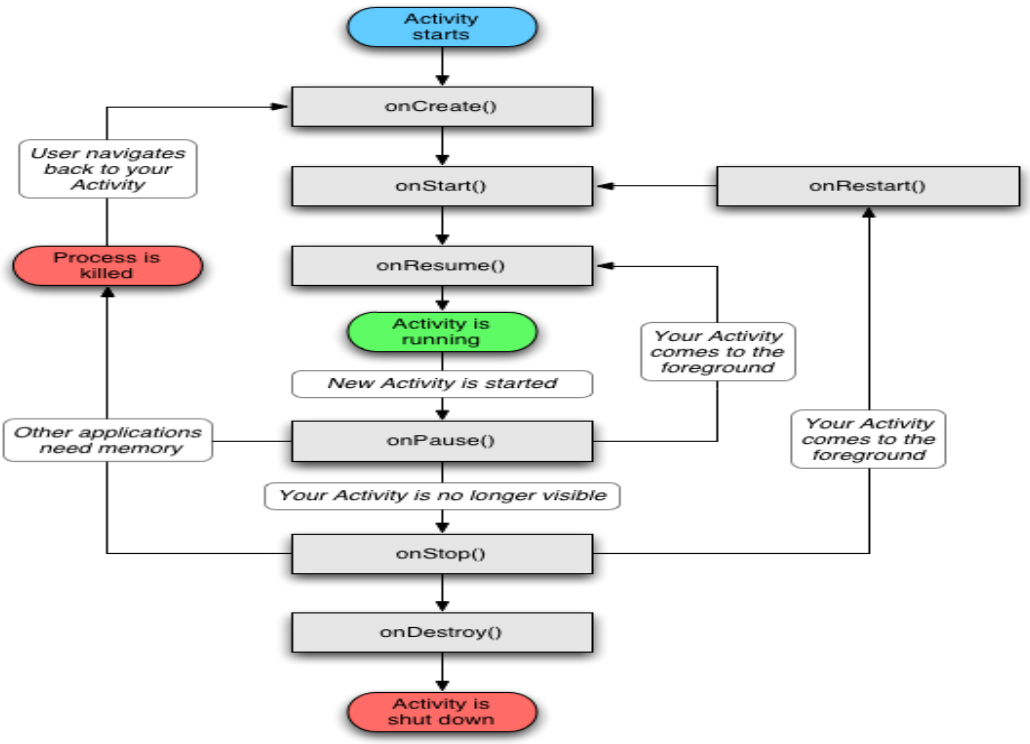


Figur 1. En bild över Androidplattformens arkitektur. [4]

3.3.1 Aktiviteter

När man arbetar med Androidapplikationer så kommer man direkt att springa på något som kallas för aktiviteter.
En aktivitet är med väldigt enkla ord, det man som användare kan se på skärmen. En enkel startbild för en applikation är en aktivitet, ett simpet inmatningsformulär är en aktivitet osv, och tillsammans skapar dessa olika aktiviteter en komplett applikation.
Man kan även bläddra tillbaka likt en webbläsare till föregående aktivitet som inte avslutats.

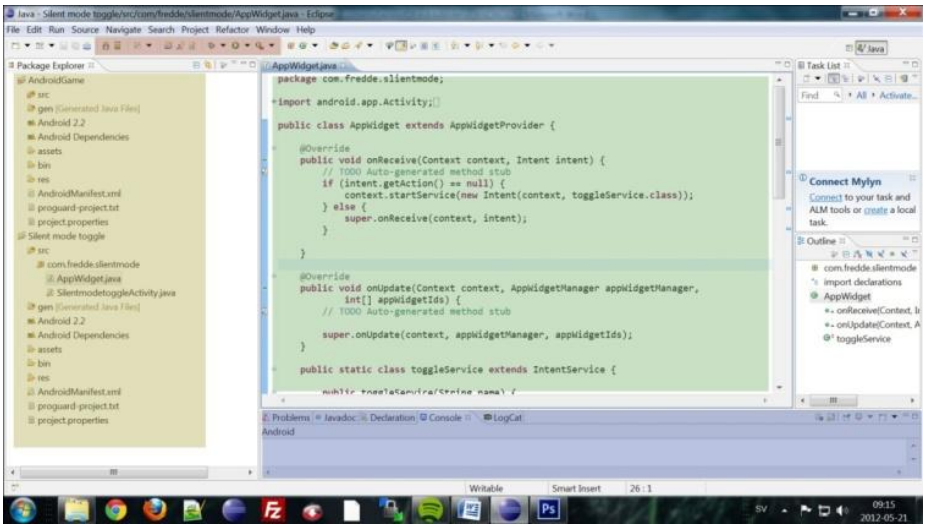
De aktiviteter som ligger under den ”aktiva” dvs synliga aktiviteten, får ett statusläge som kallas pause, detta är en Javametod som automatiskt blir anropad och utför de koder som bestämmer vad som ska hända med denna aktivitet när den hamnar i pause läget.
Man kan även ”återkalla” en aktivitet från pauseläget såvida den aktiviteten inte har blivit helt avslutad.
På detta sätt spar man på enhetes resurser, som batteri och cpu.
En Androidapplikation fungerar i väldigt enkla drag på detta sätt och kallas *”Androids livscykel”.*
Rapporten kommer gå igenom mer om just aktiviteter senare, för att få en bättre inblick och förståelse i detta då det är ett svårt ämne.



Figur 2. Androids livscykel för en aktivitet.[5]

3.4 Strukturen i Eclipse

Strukturen i Eclipse är väldigt simpelt upplagd, men är man inte bekant med vare sig Eclipse eller Java så kan det vara väldigt svårt att förstå vad allt betyder, Följande stycken kommer handla om Eclipse samt filstrukturen för ett Androidprojekt.



Figur 3. Utvecklingslayouten i Eclipse.

Bilden ovan beskriver Eclipse och de färgade rutorna är fälten som är mest relevant.

Röda fältet är några viktiga menyknappar som utvecklaren kommer använda sig mycket utav. Här hittar man den virtuella Android maskinen för att köra sin applikation samt verktyg för felsökningar mm.

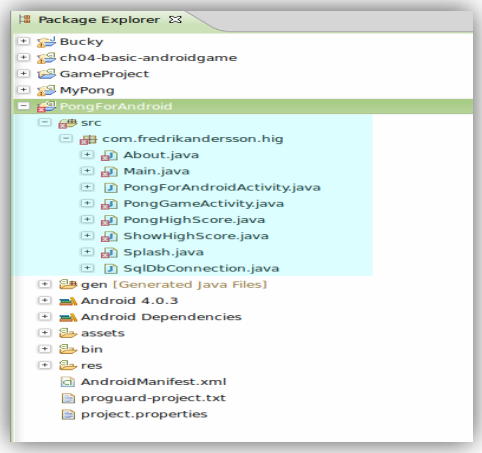
Gula fältet beskriver själva mapp och filstrukturen för projektet (har man fler projekt så visas även dem här) och här hittar man alla filer som man arbetar med under utvecklingen.

I det gröna fältet skriver man helt enkelt sina koder för respektive fil, det kan vara Javakodning eller XML kodning i vårt fall.

Det lila fältet hittar man "konsolen", vilket skriver ut allt som händer i applikationen när vi testkör den i textformat. Det är just fliken som heter *"LogCat"* som man använder vid Androidutveckling. I dessa texter kan man upptäcka eventuella fel och dylikt.

3.4.1 Src

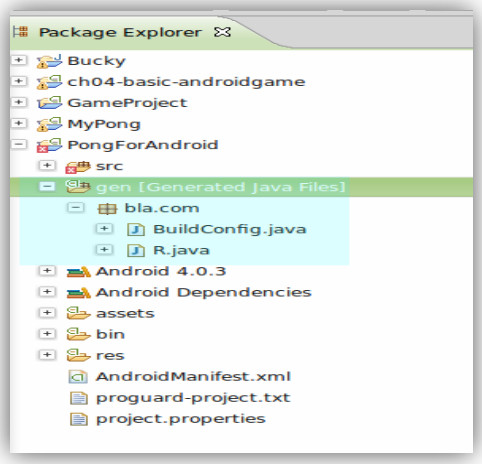
Src är förkorttning för Source, och det är i denna mapp man lagrar sina Javafiler. Alltså själva grundkodningen för vår applikation.



Figur 4.Det markerade området beskriver Src mappen.

3.4.2 Gen

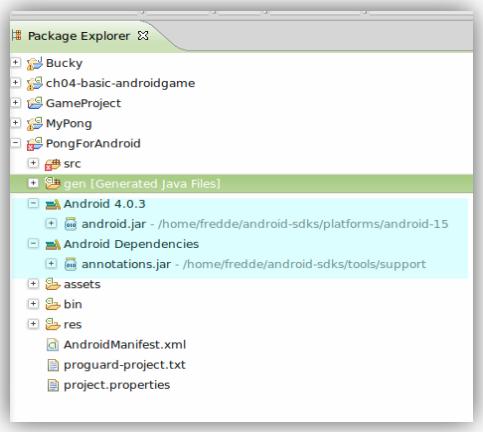
Denna mapp är väldigt speciell, och innehåller endast två Javafiler. Filen *R.java* hanterar länkningen mellan XML dokumenten och övriga resurserfiler som bilder, ljudfiler, typsnitt mm till alla Javafiler i Src mappen. Dessa två javafiler är automatiskt kontrollerade av Android SDK och uppdateras hela tiden när man lägger till nya XML filer eller någon annan av resurserna. Utvecklaren behöver alltså inte alls fundera över detta, utan Eclipse hjälper utvecklaren med det.



Figur 5. Det markerade området beskriver Gen mappen.

3.4.1 Android X.X och Android Dependencies

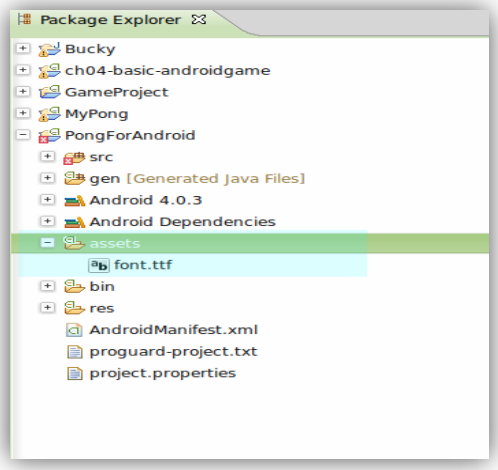
Dessa två mappar innehåller de Javapaket som följer med Android SDK, dessa paket innehåller i sin tur alla de olika fördefinierade metoder för att man som utvecklare ska kunna skriva egna applikationer för Android. Man ser även vilken version av Android man utvecklar sin applikation med, i detta fall Ver 2.2. Utvecklaren behöver ska inte heller här ändra på något utan bara veta vad mappen innehåller.



Figur 6. Androids standard javabibliotek.

3.4.2 Assets

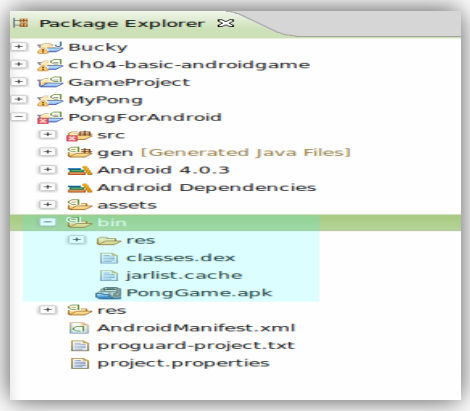
Resursförvaring i form av typsnitt, ljudfiler mm. Denna mapp kommer vi endast använda för förvaring av ett typsnitt.



Figur7. Det markerade området beskriver Assets mappen.

3.4.3 Bin

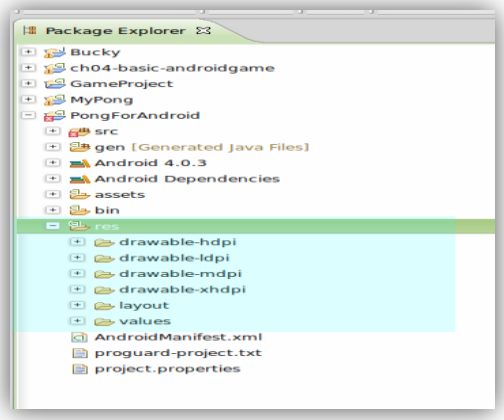
Bin mappen innehåller själva installationsfilen för Androidapplikationen, och slutar med filändeslen *.apk*. Denna installationsfil skapas varje gång man exekverar (testkör) applikationen via den virtuella maskinen, och om man vill testa sin applikation på en vanlig mobilenhet så laddar man över .apk filen till enheten och installerar sedan med sin telefon eller surfplatta. I denna mapp hittar man även lite andra filer som textfiler som informerar lite om applikationen.



Figur 8. Det markerade området beskriver Bin mappen.

3.4.4 Res

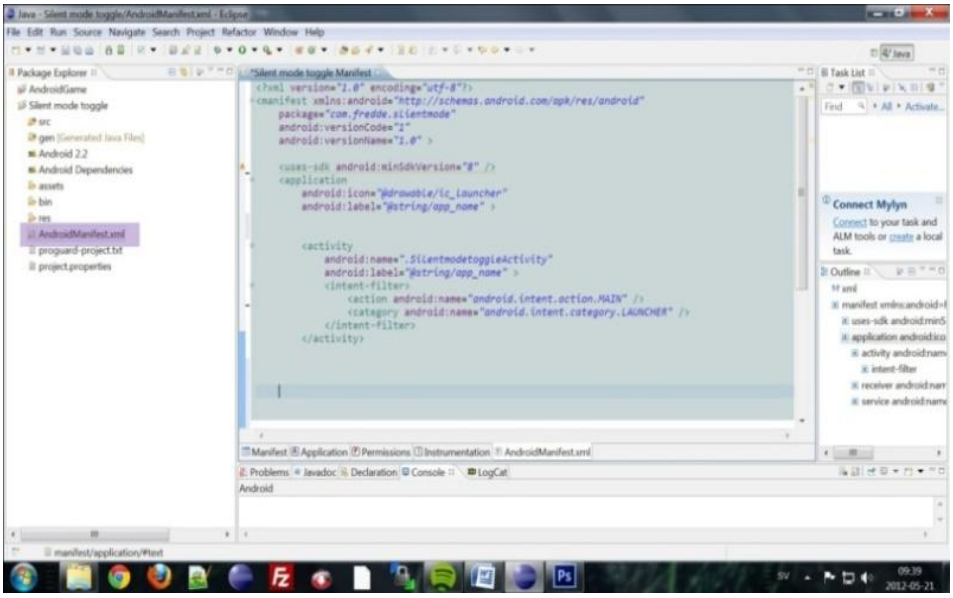
Res är förkorttning för Resource, och här förvarars alla XML dokument för designen/layouten, bildfiler, och även ljudfiler. Res mappen innehåller även ett antal underkataloger som börjar med namnet *”Drawable”* och slutar sedan på *”-hdpi”, ”-ldpi”, ”-mdpi*” och *”-xhdpi”*.
I dessa undermappar sorterar man de bilder man använder för olika upplösningar om man så önskar. I vanliga fall så förvarar man bildfiler i *”Drawable-hdpi”* mappen, och varje resurs får ett unikt idnummer som skapas automatiskt i gen mappen som vi nämnde tidigare för länkning till Javakoden.



Figur 9. Det markerade området beskriver Res mappen.

3.4.5 Androidmanifest

Viktigaste filen i hela applikationen. Androidmanifest.xml styr hela applikationen från början till slut. Denna fil beskriver bland annat version, namn, vilka aktiviteter som finns mm och utan denna fil så går det helt enkelt inte att starta sin Androidapplikation.
Varje gång man skapar en ny aktivitet för sin applikation så måste man deklarera detta i Androidmanifest. Man beskriver även i denna fil om applikationen ska ha några tillgångar för olika speciella inbyggda funktioner som tex. internetanslutningsmöjligheter, gps mm.
De som är bekant med Javaspråken sen tidigare kan tänka sig Androidmanifesten som den körbara mainmetoden i Java.



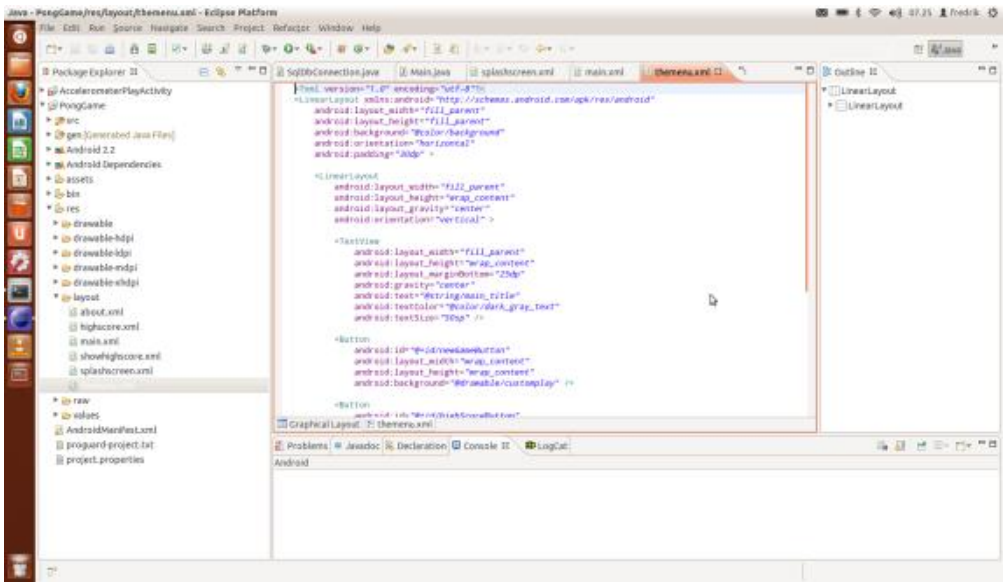
Figur 10. En enkel variant av Androidmanifest.xml.

3.5 Design av användargränssnitt i XML

Som tidigare nämnt, är layouten och andvändargränssnittet i en Androidapplikation uppbyggt med hjälp av XML, och detta handlar om hur desigen för layouten ser ut för applikationen som utvecklades under projektets gång.

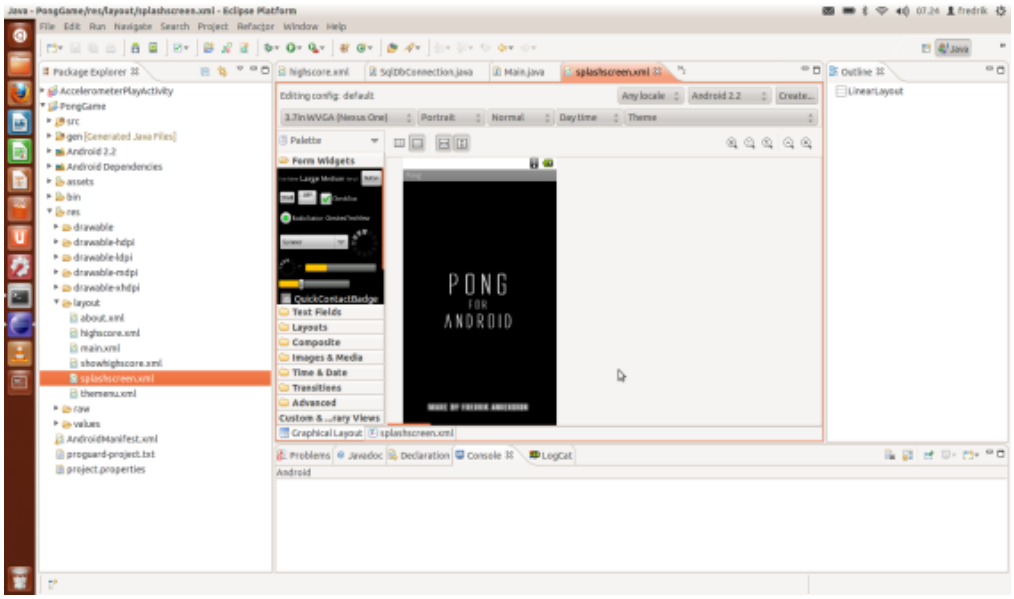
3.5.1 XML för Android

Vid utveckligen av användargränssnittet kan man välja två olika lägen hur man vill utveckla sin design, grafiskt eller textbaserat. Oftast så designar man sin layout med hjälp av den textbaserade delen, då man kan få mycket bättre struktur över hela desigen på detta sätt.
Den grafiska delen är dock väldigt bra om man vill förhandsgranska sin design utan att behöva köra igång sin applikation via den virtuella android maskinen varje gång man ändrat något för att se resultat.
Vid kodning så använder man som sagt "taggar" för att beskriva vad de olika objekten på skärmen är. En enkel XML-design för android kan innehålla en XML-tagg som kallas *"<LinearLayout />"*. Linearlayout är som en ram där man sedan fyller med textfält, knappar, bilder mm, som i sin tur också beskrivs med taggar. Det står även på första raden en *deklaration* om xml, vilket enbart talar om att dokumentet är ett XML dokument.



Figur 11. Bilden visar en XML layout i textdesignläge

För varje tagg lägger man även till s.k attribut, för att modifiera den specifika taggen till det man önskar.
Ett attribut börjar alltid med texten "android:" och sedan lägger man till det man vill modifiera eller beskriva. Vanliga attribut för en knapp eller textfält är "*text*, *bredd*, *höjd*," och ett "*id*".
Attributet "*id*" är väldigt viktigt då det sedan länkar xmlobjekten till Javakodningen.
Man kan även lägga till bakgrundsbild "*background*" för en knapp som man kanske har ritat i Photoshop, för att få en mer passande design för sin applikation.
På bilderna nedan ser man hur projektets design är uppbyggt.



Figur 12. XML-design i grafisktläge.

3.6 Programmering i Java

Java används som basprogrammeringen för Androidapplikationer.
Det är Java som gör alla beräkningar, grafik/ljudhantering och utför händelser när man som användare klickar på knappar eller dylikt på skärmen.

Men som nämnt så saknar en Androidapplikation en körbar metod som Javas *main()* metod, utan ugår ifrån Androidmanifest.xml som bestämmer vilken av de olika aktiviterer som ska starta applikationen, och även vilka av klasserna som är aktiviteter.

Med Android SDK fick vi mängder av olika klasser för att kunna utveckla en Androidapplikation, vi kommer gå igenom de viktigaste metoder för just vårt spel i detta fall.
Det som skilljer sig mellan programmeringen i Java för vanliga datorer mot Androidprogrammering är egentligen ganska liten, då många av Javas standardpaket kan användas även i Android.

3.6.1 Klasser

En klass är en programmeringsterm och man kan säga att det är en objektbeskrivning på hur en detalj eller del i ett dataprogram fungerar. Man kan tänka sig att en vanlig personbil i verkligheten skulle kunna vara en typ av klass.
Klassen har sedan en eller flera uppsättningar av metoder som kan utföras, för vårt personbilsexempel i detta fall är gasa, bromsa och svänga.
Detta kallas objektorienterad programmering och Java är helt uppbyggt efter mängder av klasser, och detsamma gäller då också för Androidutveckling. Projektets klasser innehåller metoder för bland annat spellogik, 2D grafik, användarkontroller, databashantering mm.

För att få tillgång till Javapaketen som är specialicerade för just Android så måste varje Javaklass ärva (*extends*) från klassen *Activity*. Det är just iActivityklassen som alla viktiga metoder för att få en Androidapplikation att fungera då den innehåller bland annat metoderna *onCreate*, *onPause* och *onResume*.
Alla klasser ärver inte direkt av Activityklassen, vill man använda sig av databasverktyg så ärver de nyskapade klasserna från en annan Androidklass som hanterar databaskopplingar.

Klasser sparas som *.java* filer och man kan även ha fler klasser i samma .java fil
Är man mer intresserad av ämnet kan man läsa detta i *Java direkt med Swing* [6]

Följande Javaklasser har utvecklats i detta projekt, och alla utom en av dem ärver från superklassen *Activity*. Databasklassen ärver från en speciell databasklass som finns i Android.

PongGameActivity.java
Spelmotor med grafik, spellogik samt kontrollfunktionen för pekskärmen. Denna java fil innehåller två klasser, där ena klassen är själva aktiviteten för Android och den andra klassen hanterar spellogiken och grafiken.

PongHighscore.java
Tar hand om poängen när spelet är slut och begär användaren att skriva in sitt namn, för att sedan kunna lagras i databasen.

ShowHighscore.java
Visar poängtabellen om användaren klickar på den knappen i huvudmenyn.

Splash.java
Anropas direkt när man startar applikationen, denna javafil visar endast startbilden i 4,5 sekunder och spelar upp ett startljud.

SqlDbConnection.java
Denna klass hanterar databaskopplingen.

Main.java
Hanterar huvudmenyn och vad användaren väljer för allternativ där. Man kan säga att Main.java kopplar samman alla de andra javafilerna med varandra.

About.java
Visar endast en liten informationsruta om användaren klickar på "How to play" i huvudmenyn.

3.6.1 OnCreate metoden

OnCreate Anropas direkt vid start av en viss aktivitet, och det är i denna metod som man ska skriva alla de moment man vill att sin aktivitet ska utföra. Man deklarerar även sina referenser till xml dokumenten man vill använda sig av. Man hittar en metod som heter *"setContentView(R.layout.xml)"*, som refererar och bestämmer den specifika layouten som skapas i XML så att den visas när aktiviteten körs.
Just texten "*R.layou.xml"* betyder att Java ska hämta länkningen till layoten för aktiviteten från filen *R.java* i Gen mappen.
När man sedan har layouten på plats, så måte även knappar och textfält eller andra objekt som finns med på skärmen också länkas till java på samma sätt som layouten i sig.

Detta gör man genom att man först och främst deklarerar i javakoden variabler av respektive objekt (knapp, text mm). Om man skulle deklarera en knapp som sedan skall länkas till XML i java så skriver man följande sätt:

```
private Button knapp;

knapp = (Button) findViewById(R.id.knappid);
```

Nu har vi en länkning till vårt knappobjekt mellan javakod och xml.

Den viktigaste onCreate metod som är utvecklad i detta projekt finner man i klassen PongGameActivity.java, då den utför flera olika saker som att starta upp spelplanen med grafik, starta spelmotorn med logik, samt refererar till en s.k ”lyssnaremetod” för Spelkontroller.

3.6.2 OnPause metoden

Tidigare i rapporten vid beskrivningen för aktiviteter så nämndes att en aktivitet gick i ”pauseläge”. Det är just denna speciella metod som hanterar detta och i här skriver man sina koder man vill att aktiviteten ska utföra när denna metod anropas.
I aktiviteten som man hittar i *Splash.java* finner vi ett exempel på *onPause();*
När startskärmen visats klart och huvudmenyn kommer fram så anropas *Splash.java* onPause som utför en metod för att stoppa musiken samt en metod som heter *finish(),* och den metoder stoppar hela aktiviteten helt och hållet vilket betyder att man inte kan gå tillbaka till denne aktivitet utan att behöva starta om hela applikationen.
Metoden är överlagrad från superklassen ”*Activity”*, och vill man inte att sin applikation ska utföra något speciellt vid pauseläge så behöver man inte ens deklarera denna metod i sin aktivitet.

3.6.3 OnResume metoden

Denna metod anropas alltid när en ny aktivitet startar, och även när man går tillbaka till en underliggande aktivitet än den man för tillfället ser. Denna metod är även den överlagrad från superklassen *”Activity”* och behöver inte deklareras.
Ett bra exempel där *onResume()* används är när man ska gå tillbaka till huvudmenyn från highscorelistan. Då återkallas samma instans av huvdmenyn utan att den startas om helt från början.

3.6.4 OnclickListener

OnClickListener är ett *interface* (tillägg) för en javaklass och skall alltid finnas med om man vill att något skall hända när man klickar på en knapp.
När man har implementerat detta *interface* så måste man lägga till en metod i den aktuella javaklassen som heter *onClick().* Det är i denna metod man kodar det man vill skall hända vid knapptryckning. Man måste även bestämma vilken knapp som skall hanteras av denna metod genom att skriva "*knapp.setOnClickListener(this);"* i *onCreate()* metoden. Självklart kan man använda sig av hur många knappar man vill i samma aktivitet, som i projektet där fyra olika knappar används bara i huvudmenyn.
För källkod för huvudmeny se bifogad bilaga 1.

3.6.5 OnTouchListener

OnTouchListener fungerar precis likadant som *onClickListener* , enda skillnaden är att med hjälp av detta *interface* kan man använda pekskärmen för att dra och släpp funktioner för Androidapplikationer.
Denna metod används i detta fall för att styra applikationens spelkontroll.
Android Application Development for dummies[7]

3.6.6 Spelmotor

Applikationens spelmotor finner man i filen *PongGameActivity.java* och i den innerliggande klassen *Spelyta*. Här finner man en väldigt speciell *while loop*, som går om enda tills användaren avslutar spelet genom att förlora eller avbryta på back knappen på sin enhet.
Loopen utför många beräkningar och anrop till grafikritare, poängräknare samt logiken för spelet. Här skapas även en s.k ”tråd” i Java *Thread*, som medgör att Java kan utföra fler uträkningar samtidigt. Detta behövs då grafik, musik måste uppdateras hela tiden i *while loopen.*

Logiken för spelet hanterar hur bollen ska studsa mot spelramen och paddorna, missar man bollen med en av paddarna så återställs svårighetsgraden och spelet börjar om från början igen till användaren själv bestämmer för att sluta spela. Värt att nämna är att vid 200 poäng eller högre så startas en ny aktivitet om man skulle missa bollen, så man kan skriva in sitt namn om man vill vara med på higscorelistan.

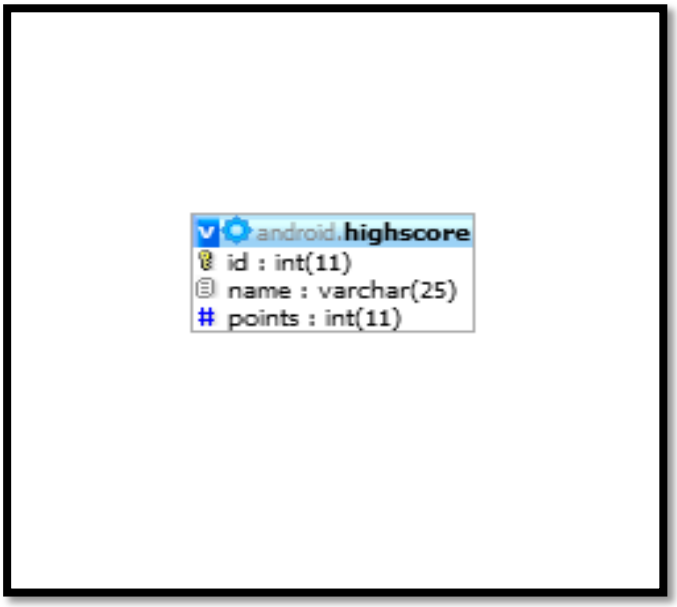
3.6.7 Spelgrafik i 2D

Grafiken för applikationen hanteras även den i *PongGameActivity.java* och i klassen *Spelyta*. Hela bilden man ser ritas upp på skrämen med hjälp av en Androidklass som heter *Canvas*. I detta canvasobjekt som skapats kan man rita alla de spelmoduler som paddar, boll och även texten som visar poängställningen.
Paddarna och bollen är objekt från klassen *Rect()* och beskriver en enkel rektangel. Sedan med hjälp av inbyggda metoder för just denna Rectklass så kan man utveckla boll och paddar att se ut precis som man själv önskar.
Man sätter även ut positionen i koordinater vart på skärmen dessa moduler skall placeras. Med hjälp av den speciella *while-loopen* som tar hand om körningen av spelet så uppdateras hela spelytan konstant och grafiken ritas om hela tiden, vilket gör att användaren upplever att saker å ting rör på sig i realtid och man får en spelbar applikation.

3.6.8 SQLite Databas

Databasen som användes i projektet heter *SQLite* och är en väldigt enkel version av en databas för just mobila enheter. Databasen för detta projekt fungerar på sådant sätt att den sparar tre värden i sin tabell.
Man har ett ID attribut som är en primär nyckel, och gör varje inlägg i databasen unikt. Sedan har man med ett attribut för spelarnamn och det sista attributet för poängen.
Kopplingen mot databasen är kodad i filen *SqlDbConnetcion.java* och hanterar insättning, hämtning och uppkoppling mot databasen.
Själva databasen i sig skapas först när man installerat applikationen och sparat in värden första gången. En metod kontrollerar om det redan finns en databas och om inte så är fallet skapas databasen med rätt attribut genom s.k SQL koder.

Många som arbetat med databaser förr har säkert arbetat med någon form av hjälpverktyg som *PHPmyAdmin* eller dylikt, där man grafiskt skapar och hanterar sina databaser.
Något sådant verktyg finns ännu inte för Android, utan man får skriva sina *SQL koder* för hand.
Databaser är som många andra av de verktygen ett ämne i sig, och man bör lära sig detta då det är väldigt kraftfullt verktyg att arbeta med i sina applikationer.
Rapporten kommer inte täcka mer om databaser.



Figur 13. Beskrivning av databasen med sina tre tabeller.

3.7 Google Play

Uppladdningen och publiceringen av applikationen är väldigt enkel.

Steg för steg guide hur man publicerar sin Android applikation.

1. Registrera ett eget Google Play konto (kostar 25 dollar, ca 175kr).

2. Signera applikationen genom Eclipse.

3. Sedan ska man skriva information om sin applikation på Google Play, följt av att man laddar upp den signerade versionen av sin applikation.

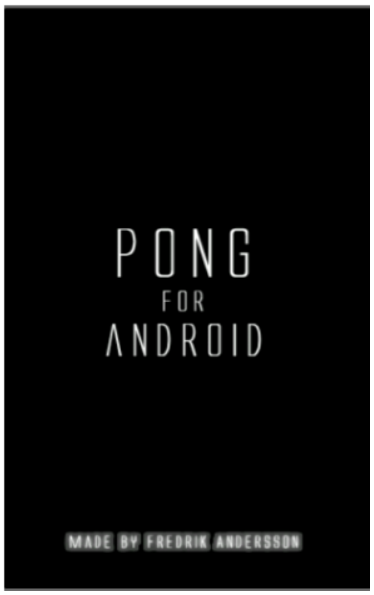
4. Klart! Nu finns er alldelens egna Android applikation tillgängligt för alla Androidanvändare i hela världen.

Länkning till Google Play http://developer.android.com/index.html [8].

4 Resultat

4.1 XML layout

Applikationens slutliga layout för användargränssnittet för startskärm och huvudmenyn.
Resultatet blev bra, där framförallt själva huvudmenyn var den viktigaste layouten i detta projekt då den ska vara lättförståerlig för alla. Fyra enkla knappar som beskriver tydligt vad dem utför, samt en text som beskriver namnet på applikationen.



Figur 14 . Applikationens startskärm.

Figur 15 . Applikationens huvudmeny.

4.2 Javaprogrammering

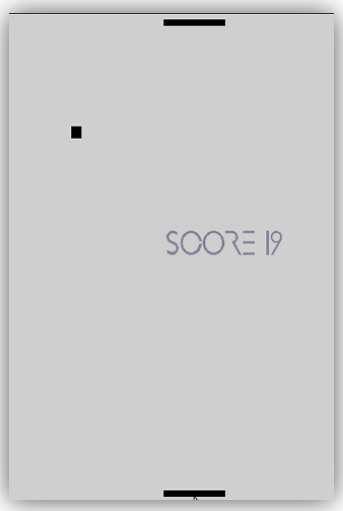
Programmeringen i Java gick också väldigt bra och resultatet för 2D grafiken blev enkel och smidig, spelytan anpassar sig efter skärmens storlek, vilket betyder att man kan spela på alla enheter från mobiltelefoner till surfplattor.

Får man högre än 200 poäng och efter det missar bollen så startas en ny aktivitet som tar med sig värdet av poängen och man blir sedan frågad om att lägga in detta med namn i en databas, som man sedan från startmenyn kan titta på.

Alla knappar fungerar och utför det som de är programmerade till samt att musik och ljud fungerar enligt mina önskemål.

I övrigt så fungerar all koppling mellan de olika aktiviteter som är utvecklade för applikationen utmärkt utan några körningsfel.

.



Figur 16 . Applikationens spelyta med boll och paddor

4.3 Google Play

Publiceringen på Google Play gick relativt bra, själva uppladdningen av applikationen var enkel och beskrivningen på Google Plays hemsida var lätt att följa.
Det som var dåligt med detta var att det blev svårt att hitta Applikationen på Google Plays lista efter publiceringen. Detta berodde på att det redan finns vädligt många applikationer som heter något med ”pong” eller ”Android” och att de mest popoläraste applikationerna listas först. Det man hade kunnat gjort annorlunda i detta fall är att döpa applikationen till något mer unikt namn.

5 Diskussion

5.1 Planering och uppföljning

Planeringen för projektet har gått relativt bra eftersom jag inte har hållit mig till något speciellt schema utan började endast lite lätt med att studera Android till grunden och sedan började jag fundera på vilken typ av applikation som skulle utvecklas. Resultatet blev snabbt ett spel då jag tycker det verkade mest intressant och att ett pongspel inte är så speciellt avancerat men samtidigt väldigt lärorikt då man får använda sig av många olika metoder för att få det hela att fungera som man vill. Uppföljningen har också fungerat bra med handledare, där jag tycker att jag fått den feedback jag behövde för att slutföra detta projekt.

5.2 Diskussion om resultatet

Grundtanken med hela projektet var att lära sig hur en Androidapplikation fungerar samt att sedan utveckla en egen applikation, och jag tycker absolut att resultatet blev väldigt bra.
Resultatet blev en fungerande applikation i form av ett pongspel i 2D grafik som fungerar på Androidplattformen.
Man kan alltid förbättra delar av programmeringen samt designen på layouten för huvudmenyer och dylikt men jag är väldigt nöjd med vad jag åstadkommit.
Det finns självklart också mycket mer att lära sig om tex androids livscykel och de andra termerna men jag tycker jag har förstått hur det mesta fungerar.
En detalj som jag gärna hade viljat haft med i applikationen var ännu en kontrollfunktion för själva spelet, där man bara genom att luta på Androidenheten kunna styra paddorna.

Utvecklingsmässigt har jag utökat mina kunskaper om Java, XML, Eclipse och framför allt Android under projektets gång. Jag har även ökat mina kunskaper i operativsystemet Ubuntu (Linux) som jag till största del utvecklade Android applikationen på.

5.3 Problem under arbetets gång

De största problemen var egentligen mest dataproblem, då jag blev tvungen att flytta fram och tillbaka hela projektet från dator till dator när de av konstiga anledningar slutade att fungera, dock löste sig detta och projektet kunde fortsätta utvecklas på en och samma dator i slutet. Ett annat problem var vid sjävla pekskärmsmetoden då man till en början inte kunde röra paddorna hela tiden med samma dragning, utan man fick "klicka" på skärmen där man ville att paddorna skulle placera sig.
Detta löstes dock på ett väldigt simplet sätt då jag enbart ändrade på returnerings värdet i metoden som hanterar pekskärmshändelser från ”*false”* till ”*true”*. Tipset fann jag i en av youtubeklippen om Android på Thenewboston.com.

Det sista nämnvärda problemet var höjden och bredden på spelytan, då jag i början programmerade fasta värden på höjd och bredd så att det skulle se bra ut på just min mobiltelefon. Sedan kom jag att tänka på att alla enheter har ju olika skärmstorlek, så det var ett problem som behövde lösas. Dock fann jag en ganska enkel lösning på detta då man tydligen kunde hämta bredd och höjd direkt från sitt Javaobjekt i Canvas, med hjälp av en *getX()* och *getY()* metod för detta objekt som nu var hela spelytan. På detta sätt kunde bildstorleken anpassas till vilken enhet som helst.

6 Slutsats

6.1 Vidareutveckling i framtiden

Efter projektets slut så kommer jag nog att lägga just detta pongspel åt sidan, men kommer definitivt att studera Android djupare då jag tycker det var väldigt intressant att jobba med denna fantastiska plattform. Det jag i första hand kommer fördjupa mig mer inom är just spelutveckling, och känner jag att jag blir riktigt duktig på detta så kanske det inte skulle vara någon dum idé att söka ett riktigt arbete inom området.
Jag kommer alltid att spara mitt Pongspel som minne.

6.2 Svar på frågeställningen

1. Bygger Android applikationen helt och hållet på XML i sitt användargränssnitt?

Svar: Nej. XML används endast för menyer samt startskärm, 2D-grafiken sköts helt i Java.

2. Behöver applikationen någon form av informationslagring som highscore värden synligt på nätet via webservices?

Svar: Nej. Det enda som är någon form av lagring i applikationen är en lokal databas.

3. Vilken typ av målgrupp av användare är Androidapplikationen främst riktad mot?

Svar: Alla olika målgrupper kan ta del av denna applikation då den fyller ett simplet roande syfte.

4. Kan man använda andra programmeringsspråk än Java för att utveckla Android applikationer?

Svar: Ja. Java är det främsta och mest utvecklade språket för Android, men man kan även utveckla applikaioner i C och C++.

5. Behöver man ha någon rekommenderad Androidenhet för att använda applikationen?

Svar: Ja, man behöver en enhet som är utrustad med pekskärm.

Referenser

[1] *Developers for Android*, http://www.android.com, besökt senast : 2012-05-28

[2] *MySQL*, http://www.android.com, besökt senast : 2012-05-26

[3] *Android Tutorials*, http://www.thenewboston.com, besökt senast : 2012-05-26

[4] *Android akritekturen*, http://elinux.org/Android_Architecture, besökt senast : 2012-06-03

[5] *Android livscykel*, http://www.androidjavadoc.com/1.0_r1_src/android/app/Activity.html, besökt senast : 2012-06-03

[6] Jan Skansholm, *Java direkt med swing!*, Studentlitteratur, Lund, 2010

[7] D.Felker and J Dobbs, *Application Development for dummies*, Wiley Publishing Inc, Indianapolis, 2010

[8] *Google Play*, http://play.google.com, besökt senast 2012-05-27

[9] E.Burnette, *Hello Android Third Edition*, The Pragmetic Bookshelf, Dallas, 2010.

Bilaga 1: Javakodning för huvudmenyn

```
public class Main extends Activity implements OnClickListener  {

	//Variabler för knapparna.
	private Button highScoreButton,aboutButton,newGameButton,exitButton;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
			super.onCreate(savedInstanceState);

			//Kod för fullskärmsläge.
			requestWindowFeature(Window.FEATURE_NO_TITLE);
	  getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
			WindowManager.LayoutParams.FLAG_FULLSCREEN);

			//Refererar till bakgrunden
			setContentView(R.layout.themenu);

			//Initierar knapparna
			buttons();
	}

	// Metod som länkar och sätter en lyssnare på varje knapp.
	private void buttons() {
			continueButton = (Button) findViewById(R.id.continueButton);
			aboutButton = (Button) findViewById(R.id.aboutButton);
			newGameButton= (Button) findViewById(R.id.newGameButton);
			exitButton = (Button) findViewById(R.id.exitButton);
			highScoreButton= (Button) findViewById(R.id.highScoreButton);
			continueButton.setOnClickListener(this);
			aboutButton.setOnClickListener(this);
			newGameButton.setOnClickListener(this);
			exitButton.setOnClickListener(this);
			highScoreButton.setOnClickListener(this);
	}
```

```
	// OnClick som lyssnar efter knapptryck
	@Override
	public void onClick(View v) {
			switch (v.getId()) {

			case R.id.aboutButton:
				Intent intent = new Intent(this, About.class);
				startActivity(intent);
				break;
			case R.id.newGameButton:
				Intent intent Game = new Intent(this,
					PongGameActivity.class);
				startActivity(intentGame);
				break;
			case R.id.exitButton:
				finish();
				break;
			case R.id.highScoreButton:
				Intent intentHighScore = new Intent(this,
					ShowHighScore.class);
				startActivity(intentHighScore);
				break;

			default:
				break;
			}
	}
}
```