Handelshögskolan
VID GÖTEBORGS UNIVERSITET
Institutionen för informatik
2003-06-04

Multimedia med J2ME i mobiltelefonen

Abstrakt

Multimedia i persondatorer är något som har användts ofta i applikationer och presentationer allt sedan dess genombrott runt mitten av 1990 talet. Många av dagens mobiltelefoner har fått förmågan att visa multimedia och har dessutom fått möjlighet att kunna ladda ner och köra program skrivna i Java. Då dessa möjligheter har kommit så blir det intressant att undersöka hur Java för mobiltelefoner stöder multimedia. Undersökningarna i denna uppsats bygger på litteraturstudier och experiment. Uppsatsen undersöker hur Java 2 Platform, Micro Editions och dess profil, Mobile Information Device Profile version 2.0, MIDP 2.0, stöder utvecklandet av mutimedia i mobiltelefoner. Uppsatsen definerar multimedia till text, bild, ljud, video och animation. I begreppet multimedia så ingår också att kunna leverera ett meddelande därför görs också en undersökning av nätverksfunktionalliteten. Slutsatsen av uppsatsen var att MIDP 2.0 inte stöder multimedia fullt ut utan bara på fem punkter av sex.

Nyckelord: J2ME, Java, mobiltelefon, MIPD, CLDC, Multimedia

Författare: Fredrik Dyrvold
Handledare: Johan Magnusson
Examensarbete I, 10 poäng

1 Inledning

1.1 Bakgrund

Den här uppsatsen behandlar området multimedia i mobiltelefoner från programutvecklarens perspektiv, och då i språket Java. Meningen är att med hjälp av litteraturstudier och experiment försöka klargöra hur stödet för multimedia ser ut i Java för mobiltelefoner.

Under mitten av 1990 talet så slog multimedia presentationer och applikationer igenom stort på persondatorer. Det är svårt att idag tänka sig en datorpresentation utan någon form av multimediainslag. Multimedias genomslag under 1990 talet berodde bland annat på att datorerna blev kraftfullare både på grafik, minnes och processor fronten. Det blev helt enkelt möjligt att kunna visa upp avancerad grafik i form av animationer och video och att spela upp ljud på en persondator. Idag har utvecklingen gått åt samma håll inom utvecklingen av mobiltelfoner, de har på bara några år fått snabbare processorer, mer minne och större färgskärmar. I och med denna utveckling så är det frestande att förutspå ett likartat genomslag av multimedia inom mobiltelefoner som det som vi har sett inom persondatorer.

Java har kommit med en ny plattform, Java 2 Platform, Micro Edition (J2ME), som bland annat vänder sig till mobiltelefoner. Då många av de mobiltelefoner som introduceras idag stöder denna plattform, så öppnas också möjligheten att kunna ladda ner nya program till dem. Många av dessa program kommer att kunna bredda mobiltelefonernas användningsområdet. I från att från början bara ha används till traditionell talkommunikation så öppnas nu möjligheten för mobiltelefoner att hantera multimedia. I och med att dessa möjligheter öppnas så kan vi förvänta oss att det kommer uppstå en mängd olika multimediaapplikationer till mobiltelefoner skrivna i Java.

Om de program som laddas ner skall kunna hantera multimedia på ett bra sätt så är det viktigt att språket det är skrivet i ger ett fullgott stöd för multimedia. Den del av J2ME som används till mobiltelefoner kallas MIDP. Uppsatsens mål är sådeles att genom litteraturstudier och experiment undersöka om MIDP version 2.0, har ett fullgott multimediastöd. Förhoppningen är att läsaren genom uppsatsen skall bilda sig en bra uppfattning om hur MIDP 2.0 stöder multimedia i mobiltelefoner.

1.2 Syfte

Syftet med denna uppsats är att undersöka hur MIDP 2.0 stöder utvecklandet av multimediaapplikationer för mobiltelefoner.

1.3 Problemformulering

Vid utveckling av multimediapresentationer och applikationer är det viktigt att språket som används har ett fullgott stöd för multimedia. Det är därför intressant att undersöka hur MIDP version 2.0 stödjer multimedia. Den här uppsatsen skall försöka besvara nedastående frågeställningar.

- Stöder MIDP 2.0 texthantering?
- Stöder MIDP 2.0 bildhantering?
- Stöder MIDP 2.0 ljudhantering?
- Stöder MIDP 2.0 videohantering?
- Stöder MIDP 2.0 animering?
- Stöder MIDP 2.0 närverkshantering?

1.4 Avgränsning

Java 2 Plattform, Micro Edition (J2ME) är Javas plattform för elektronisk apparatur såsom mobiltelefoner, handdatorer, TV boxar, kommunikationssystem i fordon och en stor del inbäddad apparatur. Många av dessa system har även förmågan att kommunicera multimediellt, men att undersöka J2MEs stöd för multimedia även inom dessa system skulle bli ett alldeles för stort och brett område att skriva om. Jag har därför valt att avgränsa uppsatsen till att bara gälla J2MEs profil MIDP version 2.0 och dess stöd för multimedia på mobiltelefoner.

2 Teoretiskt ramverk

2.1 Multimedia

2.1.1 Historik

Ser man på hur multimedia, i dess digitala form, har utvecklats historiskt så startade det enligt Educational Service District 101 med Vannevar Bush artikel från 1945, ”As We May Think”.[1] I sin artikel beskriver Bush en maskin som är konstruerad för att stödja sammankopplingen mellan stora lager av information. Maskinen gör detta genom att länka samman olika typer av information genom deras gemensamma samband. Bush koncept ses som den begreppsmässiga grunden för hypertext och World Wide Web.

Ted Nelson publicerade 1974 sina idéer om just hypertext och la grunden för Xanadu projektet. Tanken bakom Xanadu projektet var att man skulle utveckla ett system som skulle innehålla hela världens litteratur och övrig information på ett enda ställe och skapa en mekanism för att koppla samman dem. Xanadu Operating Company köptes 1988 av AutoDesk och det gavs ut en prototyp samma år men sedan dess har ingenting hänt och AutoDesk har avsagt sig ansvar.[2]

I mitten av 1980 talet så skapade en grupp utvecklare vid Browns Universitet under ledning av Norman Meyrowitz Intermedia, vilket var ett multimediaverktyg för undervisning och forskning. Intermedia fungerade på liknade sätt som World Wide Web genom hypertextlänkar som sammankopplar applikationer och program.

Allt eftersom utvecklingen av program fortskred under 1970 och 1980 talen så utvecklades programmen från att vara enskilda forskningsprojekt avsedda för högre utbildning, som exempelvis Intermedia, till att bli utvecklingsprogram tillgängliga för allmänheten.

Det dröjde tills runt mitten av 1990 talet innan begreppet multimedia blev känt hos den bredare allmänheten. Detta till trotts att hemdatorer redan hade funnits sedan 1980 talet men datorerna blev inte tillräckligt kraftiga för att köra större multimedia presentationer eller applikationer förens mitten av 1990 talet. [3]

2.1.2 Definition

Ordet multimedias egentliga betydelse kommer från latinets multus, vilket står för många eller flera och medium, vilket betyder en kanal eller ett system av kommunikation, information eller underhållning.[4]

Multimedia betyder enligt Peltonen information i olika format så som text, bilder, ljud, musik, video och animation. [5] Enligt vissa definitioner så betyder multimedia en integrerad presentation av minst tre av de ovan nämnda formaten. Enligt Keep, McLaughlin och Parmar så betyder multimedia att datorinformation kan representeras genom ljud, video och

---

[1] Christopher Keep, Tim McLaughlin, Robin Parmar, *Ted Nelson and Xanadu,* (The Electronic Labyrinth, 2000) http://jefferson.village.virginia.edu/elab/hfl0155.html

[2] Christopher Keep, Tim McLaughlin, Robin Parmar, *Ted Nelson and Xanadu*

[3] Educational Service District 101, *Brief history of multimedia,* (2002) http://www.esd101.net/web/training/multimedia/history.htm

[4] Andreas Holzinger,. *Definition of Multimedia,* (Multi Medial Learning, 2001-07-18), http://www-ang.kfunigraz.ac.at/~holzinge/mml/mml-multimedia-definition.html

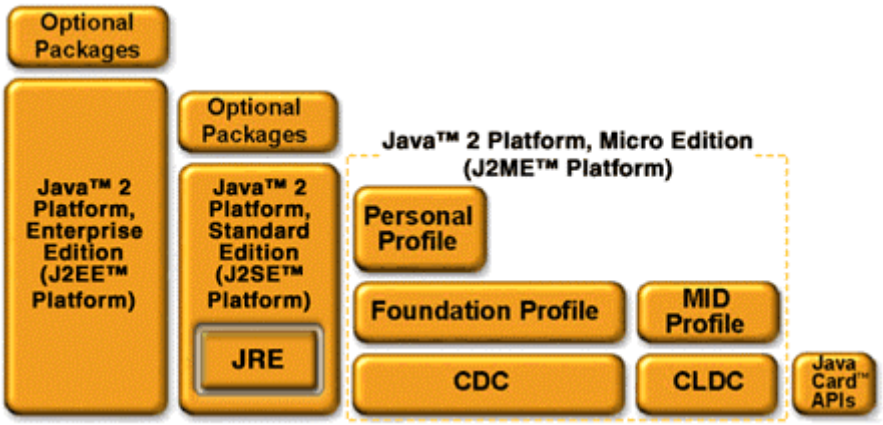
[5] Antti Peltonen, *What is multimedia*. (Finland: University of Oulu) http://edtech.oulu.fi/T3/material/package2/multim01.htm

animation förutom de traditionella medierna, text, grafik och bilder.[6] Holzinger inkluderar även interaktivitet i multimedia förutom ljud, bild, text animation och video i sin definition.[7] Multimedia är inte en pedagogisk idé utan helt enkelt ett sätt att leverera ett meddelande. När man använder sig utav multimedia inom applikationsutveckling så kan man göra sina applikationer mer effektiva genom att exempelvis byta ut text mot ljud eller video. Utvecklare kan med hjälp av multimedia skapa applikationer som ger användare möjlighet att interagera med applikationer på flera sätt, exempelvis med ljud och bilder.

2.2 Java 2 Platform, Micro Edition (J2ME)

2.2.1 Java

J2ME ingår som en av tre plattformar i Java. Java är baserad på idén om att samma mjukvara skall kunna köras på många olika sorters datorer, maskiner och övrig apparatur. Java släptes för kommersiellt bruk 1995 och har sedan dess växt i popularitet.
Anledningen till att Java kan köras på så många olika sorters system beror på att en av Java plattformens komponenter kallad Java virtual machine (JVM) som fungerar som en översättare mellan Java och den aktuella maskinen. [8]
Javas tre plattformar är **Java 2 Platform, Standard Edition (J2SE)** vilket tidigare var det som var Java. Denna plattform är designad för att utveckla applikationer för i huvudsak arbetsstationer. Innehåller alla grundklasser inom Java. Dess räckvidd ligger mellan de två övriga plattformarna. Den nästa plattform är **Java 2 Platform, Enterprise Edition (J2EE)** vilken är den plattform som är den största av de tre. J2EE är designad för att utveckla omfattande affärssystemoch är riktad åt att programmering av serverdelen i distribuerade system. Den sista plattformen, **Java 2 Platform, Micro Edition (J2ME),** är den plattform som är framtagen för att programmera applikationer för inbäddade system och små resursknappa apparaturer. Dess huvudsakliga funktion är att vara ett medium för att distribuera information genom att använda resurser så effektivt som möjligt. [9]



Figur 1: Java 2 Platform, Micro Edition

[6] Christopher Keep, Tim McLaughlin, Robin Parmar, *Ted Nelson and Xanadu*
[7] Andreas Holzinger,. *Definition of Multimedia*
[8] Java.sun, 2003. *What is Java™ Technology,* http://java.sun.com/java2/whatis/
[9] Mikko Kontio, *Mobile Java™ with J2ME*, (Finland: Edita Publishing Inc, 2003)

2.2.2 J2ME arkitekturen

I och med de breda spektra utav inbäddade system och mobila enheter som finns så delades J2ME in i subsektioner, så kallade konfigurationer. De enheter som har liknade särdrag delar på samma konfiguration. Att dela in J2ME i konfigurationer räcker inte på grund utav att enheter som har liknade konstruktioner ändå är skapade för olika ändamål. För att lösa detta så har profiler skapats som läggs ovanpå konfigurationen.
J2Me arkitekturen definierar konfigurationer, profiler och optionella paket som element för att bygga kompletta Java runtime enviorments som möter kraven för en bred räckvidd av apparatur. Varje kombination är optimerad för minne och processor kraft för varje kategori av enheters. [10]

2.2.2.1 Konfigurationer

Konfigurationer är sammansatta av en virituell maskin och ett minimalt set av klass bibliotek. De ger grundfunktionalliteten för en speciell sorts apparatur som delar på liknande egenskaper. För närvarande finns det två J2ME konfigurationer: the Connected Limited Device Configuration, och the Connected Device Configuration.[11] Anledningen att det bara finns 2 konfigurationer beror på att skaparna av J2ME ville undvika en uppsjö av inkompatibla plattformar.[12] En konfiguration definierar den minimala Java runtime environment för en familj av apparatur. [13]

CLDC som är den mindre av de två konfigurationerna är designad för apparatur med periodiskt återkommande nätverks anslutningar, långsamma processorer och begränsat minne. Denna apparatur har vanligtvis antingen 16- eller 32-bitars CPU, och ett minimum av 128 KB till 512 KB minne tillgängligt för Javaplattformens implementation och tillhörande applikationer.[14]

CDC är designad för apparatur som har mer minne, snabbare processorer, och större nätverksbandbredd, såsom TV apparater, mobila telesystem, and avancerade handdatorer. CDC inkluderar en komplett Java virtuell maskin, och en mycket större del utav J2SE plattformen än vad CLDS gör. [15]

2.2.2.2 Profiles

För att kunna ge en komplett runtime environment riktad åt specifika kategorier av viss apparatur så måste konfigurationerna vara kombinerade med ett set av högnivå APIer, så kallade profiler, som ytterligare definerar applikationens livscykelmodel, användargränssnitt, och ger tillgång till apparaturspecifika egenskaper. [16]

Mobile Information Device Profile, MIPD, som är den profil som är designad för mobiltelefoner och enkla typer av handdatorer. Den ger den grundläggande kärnfunktionaliteten som krävs av mobila applikationer, det vill säga användargränssnitt, nätverkskoppling, lokal datalagring och applikations hantering. Kombinerat med CDLC så

---

[10] Java.sun.com, *Datasheet Java™ 2 Platform, Micro Edition,* (2003) http://java.sun.com/j2me/j2me-ds.pdf
[11] Java.sun.com, *Datasheet Java™ 2 Platform, Micro Edition*
[12] Prentice Hall, *Introduction to the Java 2 Micro Edition (J2ME) Platform,* (Developer.com, 2002) http://www.developer.com/ws/j2me/article.php/10945_1475521_2
[13] Eric Giguere, *J2ME Optional Packages*, (Dec 2002) http://wireless.java.sun.com/midp/articles/optional/
[14] Java.sun.com, *Datasheet Java™ 2 Platform, Micro Edition*
[15] Java.sun.com, *Datasheet Java™ 2 Platform, Micro Edition*
[16] Java.sun.com, *Datasheet Java™ 2 Platform, Micro Edition*

ger MIDP en komplett Java runtime environment som ger tillgång till mobiltelefoners och handdatorers funktioner samt minimerar både deras minnes och batteri konsumtion.[17]

MIDP har kommit i en ny version, **MIPD 2.0**, som inkluderar ett stort set av nya finesser. Den tidigare versionen av MIPD (version 1.0) innehöll redan grunden för vad som krävdes för system av affärstyp. Vad som saknades var dock stöd för så kallat ”end-to-end security”, ett brett antagande av MIPD från trådlösa nätverksleverantörer och attraktiva MIDP-baserade enheter – detta resulterade i en brist av användbara affärsapplikationer. Med MIPD 2.0 så stöds nu end-to-end security. MIPD 2.0 introducerar säkerhetskonceptet ”*application signing* and *privileged domains”*. Genom ”application signing” så kan en applikation bli betrodd eller inte baserat på applikationens förmåga att bekräfta sitt ursprung och sin integritet, detta sker genom användandet utav ett certifikat. MIPD 2.0 Stödjer förutom HTTP även HTTPS till skillnad från version 1.0 vilken bara stödjer den tidigare. HTTPS är HTTP som skickas över SSL, Secure Socket Layer. SSL är ett socket protokoll som krypterar data som skickas över nätverk och ger autensitiering vid varje nätverksände. [18]
Version 2 förbättrar också Media stödet genom att inkludera Audio Building Block (ABB), vilken är en del utav det optionella paketet Mobile Media API (MMAPI). ABB ger möjligheten att lägga till ljud som toner, tonsekvenser och WAV filer utan att behöva använda sig utav MMAPI.[19] Ytterligare förbättringar i MIPD 2.0 är Game API, vilken hjälper utvecklare att utveckla snabbare avändargränssnitt med bättre användningsområde på ett resurssnålare sätt. Den grundläggande iden med GameAPI är att skärmen består utav flera lager vilka kan hanteras oberoende utav varandra.[20] Game API inkluderar även klassen Sprite(se stycke Aktuella klasser)

MIDlet står för Mobile Information Device Application.[21] Huvudklassen av varje MIDP applikation måste ärva MIDlet klassen. Denna klass innehåller inte många metoder men tre av dem är viktiga, nämligen: startApp(),destroyApp (boolean) och pauseApp(). Dessa metoder används för att ändra statusen på applikationen. Metoden startApp() anroppas för att aktivera den aktuella applikationen, pauseApp anropas för att pausa densamma och slutligen anropas destroyApp(boolean) för att avsluta MIDleten.[22]

2.2.2.3 Optional Packages

Från början så bestod J2ME bara av konfigurationer och profiler. Men allt eftersom plattformen utvecklades så stod det klart att utvecklarna behövde en tredje typ av J2ME komponent, nämligen optionella packet. Ett optionellt packet är ett set av APIs men det definierar inte en full utvecklings miljö utan används alltid tillsammans med en konfiguration eller en profil.[23]

Mobile Media API, MMAPI, är ett optionellt paket som är riktade till CLDC baserade enheter. Det stödjer ton generering, uppspelning och inspelning av tidbaserade medier.

---

[17] Java.sun.com, *Datasheet Java™ 2 Platform, Micro Edition*
[18] Ortiz, C. Enrique. *The J2ME Mobile Information Device Profile 2.0,* (developer.com, 2003) http://www.developer.com/ws/j2me/article.php/1574791
[19] Java.sun.com, *What's new in MIDP 2.0,* http://java.sun.com/products/midp/whatsnew.html
[20] Mikko Konti,. *MIDP 2.0: The Game API,* (MicroJava.com, 2003-01-09)
[21] Midlet Review, *MIDlet*, (Midlet Review, 2003).
[22] Mikko Kontio, *Mobile Java™with J2ME*, (Finland: Edita Publishing Inc, 2002).
[23] Eric Giguere, *J2ME Optional Packages*

MMAPI delar in multimedia processering i 2 delar, nämligen behandlingen av protokollet för mottagandet utav data och behandlingen utav själva datan. Mottagandet utav data kan ske ifrån fil, inspelnings apparatur eller streaming från en server och behandlingen av data syftar på hur data behandlas i form av exempelvis dekryptering. MMAPI använder sig utav klassen DataSource för mottagandet av data och klassen Player för behandlingen av data.
MMAPI har metoder för att kontrollera att olika media metoder finns på den aktuella enhet som kör en MIDP-applikation. Dessa metoder kan kontrollera om mixning, minst två olika toner kan spelas samtidigt, ljudupptagning, videoupptagning, inspelning stöds, det finns även kontroll utav vilka olika format for video och ljud som stöds samt om metod för stillbildstagning stöds. Även kamera stöd och tongenerering finns.
Alla implementationer av MMAPI stödjer inte alla multimedia typer. Vissa implementationer ger bara stöd för vissa utvalda typer och protokoll. MMAPI kräver att viss support stöds av enheten, medans övriga enligt specifikationen antingen bör stödjas eller kan stödjas. [24]

2.2.3 J2ME runtime environment

J2ME runtime environment består utav en konfiguration tillsammans med en profil.[25] Java runtime environment innehåller de hjälpmedel som behövs för att kunna köra Javaprogram på en dator.[26]

2.2.4 Aktuella klasser

Nedan stycken är med för att förklara några för experimenten viktiga klasser i MIDP 2.0 som används vid multimedia hantering. Samtliga klasser som tas upp har använts i experimenten. Avsikten med detta stycke är inte att fullt ut förklara klasserna och dess innebörd utan bara att ge läsaren en grundläggande förståelse för dessa klasser.

Image en klass som används för att skapa objekt som lagrar grafiskbilddata. Bildobjekt existerar endast i minnet och ritas inte ut till skärm om inte ett direkt kommande anropas.
Klassen **Sprite** används för att skapa objekt som är visuella element och har förmågan att visa upp en av flera lagrade bildrutor. Olika bildrutor kan visas upp för att ge effekten utav animering. Spriteobjekt kan även spegelvändas och roteras för att ytterligare variera dess utseende. Man kan ändra dess synlighet från synlig till osynlig. De bildrutor som används för att visa upp ett Spriteobjekt finns lagrade i ett Imageobjekt. Bildrutorna ligger i en sekvens som kan spelas upp.
Klassen **Graphics** används för att skapa objekt som ger tvådimensionella renderingsmöjligheter.
Canvas är en grundklass som används när man skriver applikationer som behöver hantera lågnivå händelser och erbjuder grafiska metoder för att rita till skärm.
Klassen **GameCanvas** ärver ifrån Canvasklassen men ger utökat stöd för spelspecifika behov såsom möjligheten att buffra grafik i minnet utanför skärm och möjligheten att känna av knappstatus.
Klassen **TextBox** används för att skapa objekt som ger användaren möjlighet att skriva in och ändra text. Det har en maxstorlek av antal tecken som kan skrivas in, denna storlek bestämms av programmeraren.
InputStream är en abstrakt klass som används för att läsa in strömmar av bytes.
String är en klass som används för att representerar teckensträngar. Ett strängobjekt är konstant och dess värde kan alltså inte ändras efter det har skapats. Klassen innehåller bland

[24] Java.sun.com, *Mobile Media API (JSR-135),* http://jcp.org/aboutJava/communityprocess/final/jsr135/
[25] Prentice Hall,. *Introduction to the Java 2 Micro Edition (J2ME) Platform*
[26] Jan Skansholm, *Java Direkt*, (Lund: Studentlitteratur, 1999)

annat metoder för att undersöka enskilda tecken i strängen, jämföring av strängar och för konkatinering av strängar.
Maneger är en klass som ger tillgång till systemberoende resurser såsom Playerobjekt för multimediaprocessering. Klassen ger möjlighet att skapa Playerobjekt.
Player är ett gränssnitt. Ett objekt som implementerar Player gränssnittet ges möjligheten att kontrollerar uppspelningen utav tidbaserad media. Objektet får metoder för att handskas med bland annat uppspelnings möjligheter.[27]

2.2.5 MIDP 2.0s stöd för multimedia

Enligt MIDP 2.0s api så stöds text i form av enskilda tecken genom variabeltypen char och hela textsträngar genom klassen String.
När det gäller bildstöd så kräver MIDP 2.0 specifikationen att telefontillverkaren erbjuder stöd för bildformatet PNG och dess tranperanta förmåga, vidare så kan tillverkare även erbjuda annat bildformatstöd.
Ljudstöd har MIDP 2.0 fått genom klassen Manager, som undersöker vilka typer av media filer en mobiltelfon klarar av att spela upp, och klassen Player som gör det möjligt att kontrollera dessa media stöd för flera olika ljudformat bland annat sekvensspelning utav toner och wav filer.[28] MIDP 2.0 specifikationen kräver att telefontillverkaren erbjuder stöd för tongenerering och 8 bitars linjär PCM wav format, vidare så tillåter specifikationen att tillverkaren även erbjuder andra format för förinspelat ljud.[29]
Stöd för videofunktionallitet saknar MIDP 2.0 enligt apin[30].
Animeringsstöd erbjuder MIDP 2.0 bland annat genom sitt gameapi och då först och främst genom klassen Sprite som gör det enklare att skapa animering genom funktioner för traversering av bild och flyttning av bild över skärmen.[31]
Nätverksstöd erbjuder MIDP 2.0 bland annat genom sitt gränssnitt HttpConnection som används för nätverkskoppling mot en internetserver.[32]

2.3 Ordlista

Animation
En animering är en samling bilder vilka ger illusionen utav att förändras över tiden när de sätts samman.[33]

API
Application Programming Interface, är en serie av funktioner som program kan använda för att låta operativsystemet göra grovjobbet.[34] Det är också specifikationen för hur en programmerare skriver en applikation och hur han kommer åt metoderna och attributen i de olika klasserna och objekten.[35]

---

[27] Java.sun.com, *Mobile Information Device Profile 2.0*
[28] Java.sun.com, *Mobile Information Device Profile 2.0*
[29] MIDP 2.0 Specification, Final (JSR 118)
[30] MIDP 2.0 Specification, Final (JSR 118)
[31] Java.sun.com, *Mobile Information Device Profile 2.0*
[32] Java.sun.com, *Mobile Information Device Profile 2.0*
[33] Manning, Mark. *Animations and Chipmunk Basic*
[34] CNET Glossary, *API,* (2003)
[35] Java.sun.com , *CLOSSARY OF JAVA TM TECHNOLOGY-RELATED TERMS*, (2003)

JPC (Java Community Process)
JPC är det sätt på vilken Java plattformen utvecklas. Det är en öppen organisation av Java utvecklare och licensinnehavare från hela världen vars mål är att utveckla och granska Java tekniken. Både Java och JPC var från början skapade av Sun Microsystems men JPC har utvecklats från en informell process som Sun använde sig utav 1995 till en formell process med insyn av många organisationer inom Javas community. [36]

PNG (Portable Network Graphics)
World Wide Web Consortium skriver "PNG is an extensible file format for the lossless, portable, well-compressed storage of raster images." [37]
PNG stödjer bilder med indexerad färg, gråskalor, truecolor och en alphakanal för transperans.[38]

[36] Java Community Process. *The Java Community Process SM (JPC)*, (2003)

[37] W3C, *PNG (Portable Network Graphics),* (1995-02-14)
[38] W3C, *PNG (Portable Network Graphics) Specification,* (1996-10-01)

3 Metod

3.1 Val av metod

Det vetenskapliga perspektiv som valdes för denna uppsats var det hermeneutiska och den metod som valdes för att genomföra uppsatsen var den kvalitativt ansatsen. Metoden kommer här att beskrivas och förklaras, vidare kommer även tillvägagångssättet för det empiriska[39] arbetet beskriva i detalj. Detta för att åstadkomma replikation[40] och evaluering[41].

3.1.1 Positivism och hermeneutik

Positivistisk vetenskapsuppfattning innebär enligt Wiedersheim-Paul och Eriksson[42] att man utgår ifrån att verkligheten är objektiv och lika för alla. Denna uppfattning har dominerat forskningen under en lång tid, antagligen för att potivismen betytt mycket i naturvetenskaperna där emperiska mätningar snabbt ökade våra kunskaper. En positivistisk ansats bygger på formell logik och fakta som är resultat av mätning. Den formella logiken grundas på definitioner, antaganden och satser ifrån vilka teorier bildas som kan testas i olika hypoteser. I grunden utgår man ifrån termer så som ord och symboler, bakom vilka det ligger en verklighetsuppfattning som benämns begrepp. Hur begreppen definieras blir avgörande för deras innebörd. En teori uttrycks i att antal inbördes relaterade satser så kallade teorem vilka bildar ett system. Vissa av satserna i teorin utgör dess antaganden och andra satser är de vars giltighet man vill testa. De satser som skall testas kallar man för hypoteser. Utifrån teorin så bevisas hypoteser som jämförs med observationer vilket kan leda till att teorin omformuleras. Det centrala är att få fram hypoteser som stämmer överens så mycket som möjligt med de observationer som gjordes innan hypotesen ställdes upp.

I hermeneutisk vetenskapsuppfattning så försöker man till skillnad mot den positivistiska, som beskriver och förklarar, istället få en helhetsförståelse, en insikt. Forskaren träder här in i den statistiska analysens ställe. Forskarens roll inom hermeneutiken är till skillnad mot positivismen mer aktiv då han/hon är medagerade alltså själv är en aktör och inte är objektiv. En hermeneutiker anser att forskning och utredning är en aldrig avstannande process utan hela tiden fortgår där det väsentliga är att öka sin förståelse.[43]

3.1.2 Kvalitativ och kvantitativ metod

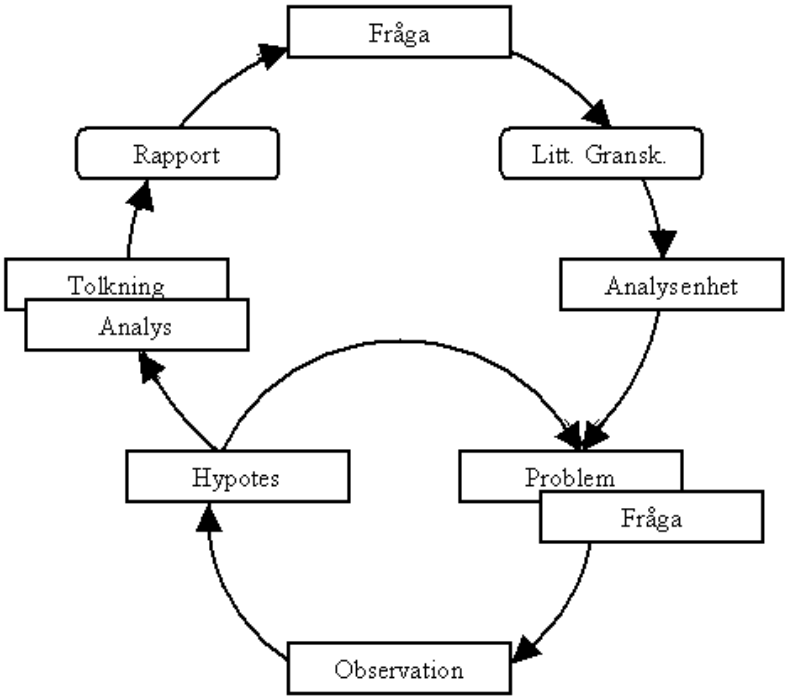
Grovt förenklat kan man säga att det finns två grupper av metoder att tillgå som angreppssätt vid undersökning och forskning. Skillnaden dem i mellan ligger i vad som undersöks. Kvantitativa metoder använder sig utav kvantifiering med hjälp av matematik och statistik. Dessa metoder resulterar i ofta numeriska observationer. Exempel på kvantitativa metoder är experiment, test, frågeformulär etc.
Kvalitativa metoder utgår från en subjektiv verklighet som snarare skall tolkas än mätas. Resultatet av det som undersöks med hjälp av kvalitativa metoder kan inte utryckas i siffror utan utrycks istället med ord.[44]

---

[39] Emperisk betyder beroende av eller grundad på erfarenhet, såsom iakttagelse. (Ekman, 1985)
[40] Replikation betyder att metoden skall vara möjliga att upprepa under identiska förhållanden. (Backman, 1998)
[41] Evalueringa innebär att man värderar det emperiska förfarandet, man lägger synpunkter på vald metodik, dess samband med problemställningen och dess bärkraft för de slutsatser och tolkningar som görs. (Backman, 1998)
[42] Finn Wiedersheim-Paul & Lars Torsten Eriksson, *Att Utreda och Rapportera*, (Sandby: DeTryck, 1989)
[43] Wiedersheim-Paul, Eriksson, *Att Utreda och Rapportera*
[44] Backman, Jarl. *Rapporter och Uppsatser*

3.2 Den kvalitativa forskningsprocessen

Den kvalitativa forskningsprocessen innehåller enligt Backman ett stort mått av flexibilitet och dynamik. Den ger ett stort utrymme för variationer och är inte särskilt standardiserad eller sekventiell, detta kan visa sig i att flera moment kan pågå samtidigt i interaktion med varandra. Nya moment kan påbörjas utan att föregående har avslutats.[45] Enligt Backman kan den kvalitativa forskningsprocessen beskrivas i nedan följande steg.



Figur 2: Den kvalitativa forskningsprocessen

1. **Fråga**: Arbetet inleds med en fråga, vanligtvis ställd ”hur” eller ”varför”, som ofta utgår från en praktisk eller tillämpad situation.
2. **Litteraturgranskning**: Skall i huvudsak klargöra vilken tidigare kunskap som finns inom området, visa betydelsen av problemet, precisera problemet etc.
3. **Val av analysenhet**: Avgränsning av fall, val av konkret fall
4. **Problemformulering**: Sker ofta samtidigt som litteraturinsamlingen. Problemformuleringarna är ofta breda och vaga i början men finslipas under studiens gång. Leder slutligen till definitiv frågeformulering.
5. **Observation**: Observationen av verkligheten är här beroende utav observatören som ett tolkande subjekt. Vanliga metoder för att göra observationen är intervjuer, olika typer av deltagande och/eller naturalistiska/etnografiska studier och dokument. Försökspersoner utväljs selektivt och inte statistiskt i den kvalitativa ansatsen.
6. **Analys**: Analysen sker ofta kontinuerligt med insamling utav data. Strävan att ge en helhetsbild. Det insamlade materialet måste struktureras och tolkas.

[45] Backman, Jarl. *Rapporter och Uppsatser*

7. **Tolkning**: Tolkningsmomentet sker samtidigt som analys och observationsmomentet. Tolkningsmomentet skall ge innebörd och mening åt observationerna och kräver kunskap, insikt och sensitivitet.
8. **Rapportering**: För den kvalitativa rapporten finns det inga rekommendationer eller ramverk man kan följa. Man bör ha rapporten i åtanke under hela forskningsprocessen.

3.3 Praktiskt Tillvägagångssätt

Först etablerades ett generellt problemområde, J2ME, och en fråga formulerades. Efter detta påbörjades insamlingen av sekundärdatan genom litteraturstudierna. Parallellt med insamlingen av sekundärdatan så började insamlingen av primärdata genom först planeringen och sedan utförandet utav experiment. Samtidigt med genomförandet av experimenten så analyserades sekundärdatan för att ge svar på vissa rent programmeringstekniska frågor. Detta var viktigt för att överhuvudtaget kunna genomföra experimenten. Efter genomförandet utav experimenten så analyserades dessa. När analysen var klar så följde en diskussion varpå en slutsats drogs.

3.3.1 Litteraturstudier

Denna del bestod i att samla in och studera sekundärdatan. Stor del av litteraturstudierna har bestått i att hitta ett lämpligt ramverk. Detta för att få den utgångspunkt ifrån vilken den emperiska studien har beskrivits. Litteraturen som användes inhämtades till största delen ifrån Internet och till viss del ifrån böcker. Den litteratur som söktes behandlade ämnena multimedia och J2ME. Allteftersom litteraturstudierna pågick så förändrades problemformuleringen från att ha varit väldigt bred, J2ME, till att bli smalare och mer precis, MIDP 2.0 och dess stöd för utvecklingen av multimediaapplikationer till mobiltelefoner. Denna del tog störst anspråk i tid, speciellt då mycket av det material som samlats in fick sorteras bort allt efter som problemformuleringen smalnade av och blev mer precisare.

3.3.2 Planering av experiment

De experiment som planerades i denna del byggde på det material som hade tagits fram om multimedia och J2ME under litteraturstudierna. Det primära var att experimenten verkligen skulle ge svar på de frågor som hade ställts i problemformuleringen. Under planeringen så togs beslutet att inte utföra experiment om videostöd då litteraturstudierna hade visat att video funktionalitet inte ligger inom MIDP 2.0 utan i det optionella paketet MMAPI. De experiment som under planeringen fastställdes att göra var:

- Teststöd
- Bildstöd
- Ljudstöd
- Animeringsstöd
- Nätverkstöd

3.3.3 Emperisk studie

Experimenten genomfördes i en simulerad miljö i en vanlig persondator av typen AMD Athlon 1200. Operativsystemet var Windows 2000. Datakoden skrevs i ES-Computings texteditor Editplus och kompilerades i Suns utvecklingsverktyg för J2ME, Java 2 Platform Micro Edition, Wireless Toolkit. Resultatet av experimenten visades upp den emulator som ingår i Wireless Toolkit. Förberedelserna under planeringen av experimenten tog ungefär två veckor att genomföra och själva experimenten tog från några timmar till två dagar att genomföra beroende på komplexitet.

4 Resultat

För att undersöka hur MIDP 2.0 stödjer multimedia så har nedan experiment utförts. De beskrivs här i detalj hur dessa experimenten har genomförts.

I och med att alla klasser av typen MIDlet innehåller metoderna startApp(), pauseApp(), destroyApp(boolean) och commandAction() så kommer inte implementeringen utav dessa tas upp förutom i första stycket om Textstöd. Samtliga av experimeten har försökts att utföras på ett så enkelt sätt som möjligt programmeringstekniskt, det har bara testats om den aktuella funktionaliteten stöds och det har inte gjorts några felkontroller. Detta för att underlätta för läsaren.

4.1 Textstöd

Undersökningen av MIDP 2.0 s textstöd gick till enligt följande.
Först skapades en klass, Textstod, som ärvde från klassen MIDlet. I klassen Textstöd så initierades stöd för skärm genom att skapa ett objekt av klassen Display. Sedan skapades ett formulär av klassen form, två textfält av klassen TextField för uppvisning av texten och två textsträngar av klassen String samt en variabel av typen char. Charvariabeln initierades till tecknet ”a”. Den första textsträngen initierades till ”Hello World” och den andra textsträngen skapades som ett nytt objekt med charvariabeln som indata. Detta på grund utav att textfälten inte klarar av att ta emot en charvariabel som indata utan kräver en textsträng. Efter detta initierades de två textfälten med de två textsträngarna. När detta var gjort så lades textfälten till formuläret.
Förutom detta så implementerade jag även metoderna startApp(), pauseApp(), destroyApp(boolean) och commandAction().

4.2 Bildstöd

Undersökningen av MIDP 2.0 s bildstöd gick till enligt följande.
Först skapades två klasser, Bildtest som ärvde från klassen MIDlet och Bild som ärvde från klassen Canvas.
I klassen Bildtest så initierades stöd för skärm genom att skapa ett objekt av klassen Display I Bildtest metod startApp så skapades ett objekt utav klassen Bild. Bildobjektets metod paint(Graphics) anropades genom att anropa metoden repaint, efter detta så lades bildobjektet till displayobjektet genom metoden setCurrent(bildobjekt).
I klassen Bild så skapades ett objekt av klassen Image vilket initierades med en bildfil utav formatet PNG. När detta var gjort så anropades metoderna getWidth och getHeight för att ta reda på bildskärmens bred och höjd. Sedan skapades en paint metod i vilken det ritades ut en vit fyrkant med hjälp utav metoderna setColor(int) och fillRect(int, int, int, int), i fillRect användes värdena som hade tagits reda på genom getWidth och getHeight metoderna. Slutligen ritades bilden ut genom metoden drawImage(Image, int, int, int, int), även här användes värdena från getWidth och getHeight.

4.3 Ljudstöd

Två olika ljudstöd undersöktes, dels att kunna spela enkla toner och dels att kunna spela ljudfiler utav typen wav. Undersökningen av MIDP 2.0 s ljudstöd gick till enligt följande.
Först skapades två klasser, LjudStod som ärvde från MIDlet och Ljud som implementerade den abstracta klassen Runneble, detta gjordes för att kunna tråda programmet och därigenom kunna åstadkomma en pause mellan uppspelningen av tonen och ljudfilen.

I klassen LjudStod så initierades stöd för skärm genom att skapa ett objekt av klassen Display. I klassen LjudStod så skapades ett objekt utav klassen Ljud och i metoden metod startApp anropades ljudobjektets metod start(). I ljudobjektets startmetod så skapades först en tråd för att senare kunna pausa objektet. Sedan skapades ett objekt av klassen manager och dess funktion playTone(int, int, int) anropades. Efter detta så pausades applikationen med hjälp av tråden i två sekunder. När detta var gjort skapades ett objekt utav klassen Player(String) med sökvägen till en wavfil som inparameter. Slutligen så startades playerobjektet med dess metod start().

4.4 Animeringsstöd

Animeringsstödet undersöktes genom att låta ett Spriteobjekt rita ut två olika bilder efter varandra och på så sätt skapa en illusion av rörelse och sedan flyttades dessa bilder över skärmens yta.
Först skapades tre klasser, AnimeringsTest som ärvde från klassen MIDlet, Bakgrund som ärvde från klassen Canvas och implementerade gränssnittet Runnable och klassen Animering som ärvde från klassen Sprite.
I klassen AnimeringsTest så initierades stöd för skärm genom att skapa ett objekt av klassen Display. Efter detta skapades ett objekt av klassen Bakgrund.
I klassen Bakgrund så anropades metoderna getWidth och getHeight för att ta reda på bildskärmens bred och höjd, sedan skapades ett objekt av klassen LayerManeger för att kunna hantera Animeringsobjektet. Efter detta skapades ett objekt av klassen Animering genom att anropa metoden skapaAnimering(). Animeringsobjektet skickades som indata till LayerManegerobjektet. Metoden skapaAnimering() läste in en bild och skapade ett objekt av klassen Animering. Metoden start() skapade en ny tråd av bakgrundsklassen och startade sedan den. Metoden run skapade ett Graphicsobjekt för att kunna rita ut allt på skärmen och anropade sedan i en evighetsslinga Animeringsobjektets metoder dans(), render() och move(int, int) för att skapa illusionen av rörelse0. Metoden render(Graphics) anropade Graphicsobjektets funktioner setColor() och fillRect(int, int, int, int) och skickade sedan in Graphicsobjekt i metoden paint(Graphics, int, int).
I klassen Animering skapades metoden dans() som växlade på bilderna.

4.5 Nätverksstöd

Undersökningen av nätverksmöjligheten skedde genom att programmet kopplade upp sig mot en server på Internet med httpprotokollet och därifrån hämtade en textfil som skulle skrevs ut på skärmen. Undersökningen av MIDP v.2s nätverksstöd gick till enligt följande.
Först skapades två klasser, HttpConnect som ärvde från MIDlet och Kontakt.
I klassen HttpConnect så initierades stöd för skärm genom att skapa ett objekt av klassen Display. Sedan skapades också ett TextBox objekt för att kunna visa upp innehållet i den textfil som hämtades från servern, efter detta skapades ett objekt utav klassen Kontakt. I Klassen kontakt skapades metoden getUrlText som tog en url i form av en textsträng som parameter och returnerade ett objekt av klassen TextBox. I metoden getUrlText så skapades fyra olika objekt av klasserna StreamConnection, InputStream, String och TextBox. StreamConnectionobjektet användes för att öppna kontakten mot servern, InputStreamobjektet användes för att läsa från StreamConnectionobjektet och Stringobjektet för att lagra det som läste från servern. När programmet hade läst klart från servern så skickades Stringobjektet in i det TextBoxobjekt som funktionen returnerade. I klassen HttpConnect så anropades Kontaktobjektets metod getUrlText och lagrade returvärdet i TextBoxobjektet. Till sist så skickades TextBoxobjektet till Displayobjektet och kunde på så sätt skriva texten till skärm.

5 Analys

5.1 Analys av emperisk data

Analysen påbörjades egentligen redan vid insamlingen av den sekundära datan och fortlöpte samtidigt med planeringen och genomförandet utav experimenten.
Analysdelen bestod i huvudsak av att mot bakgrund av det aktuella ramverket studera resultatet av varje experiment i den emulator som ingick i Wireless Toolkit.

MIDP 2.0s textstöd fungerade i den simulerade miljön utan några problem. Det var inga problem med att skapa ett Stringobjekt med en textsträng som indata och sedan visa upp den på skärmen. Inte heller var det några problem med att skapa en charvariabel och sedan använda den som indata till ett Stringobjekt. Jag anser att stödet för texthantering är fullgott i MIDP 2.0.

Bildstödet fungerade också utan problem. Det som behövdes göra var att skapa ett bild objekt med en sökväg till en bildfil av formatet PNG och sedan skapa en paintmetod som ritade ut bilden till skärm. Det man kan tycka är ett litet minus är att man med säkerhet bara kan används sig utav ett format, nämligen PNG-formatet, när man utvecklar applikationer. MIDP 2.0 specifikationen lägger dock inga hinder för telefonutvecklarna att erbjuda fler format men vill en utvecklare vara säker på att hans applikation är portabel till flera mobiltelefoner så måste han använda sig utav PNG. Jag anser att dock stödet för bildhantering trots detta är fullgott i MIDP 2.0.

De två ljudstöd som prövades var enkel tongenerering och ljudfiler av WAV-format. Det var inga problem att spela upp dessa ljud. Vad som dock skall påpekas är att det finns en mängd andra ljudformat som inte testades, men på frågan om MIDP 2.0 stöder ljud så anser jag helt klart att så är fallet.

Att skapa en animering kan göras på flera sätt, det sätt som valdes var att skapa ett Spriteobjekt och använda dess metoder för att byta bilder och flytta objektet över skärmen. I och med Game APIn som har introducerats i MIDP 2.0 så bland annat möjligheterna att animera bilder underlättats för programutvecklaren som bara behöver göra ett metodanrop för att ändra bild. Att skapa en animering utan klassen Sprite betyder att programmeraren får skriva omständligare kod för att byta ut bilder. Så på frågan om MIDP 2.0 stöder animering så anser jag att det inte bara stödjer utan det även underlättar skapandet av animeringar.

Det nätverksstöd som jag testade var en HTTPkoppling mot en Internetserver där jag hämtade hem en textfil. Detta fungerade bra. Vad som dock bör påpekas var att testet inte omfattade HTTPS, detta till trots så anser jag att MIDP 2.0 har fullgott nätverksstöd.

Videostöd saknas i MIDP 2.0. och det är inget som MIDP 2.0 specifikationen heller kräver att det skall finnas. Stöd för video finns i det optionella paketet Mobile Media API men eftersom detta är ett optionellt paket så anser jag att det står helt klart att det saknas stöd för video i MIDP 2.0

6 Diskussion

6.1 Självkritik

Då uppsatsen experiment har utförts i en simulerad miljö i Sun Microsystems egna utvecklingsverktyg för J2ME, Java 2 Platform Micro Edition, Wireless Toolkit så är frågan hur adekvata experimenten egentligen är rättfärdigad. Vad som verkligen hade varit interessant hade varit att utföra experimenten på ett urval av mobiltelefoner som har stöd för MIDP 2.0. Att genomföra ett sådant skarpt projekt har det dock inte funnits någon möjlighet att göra, ur ekonomisk synpunkt

6.3 Potential

Det är svårt att se någon gräns för vad MIDP profilen tillsammans med optionella kan erbjuda, de kan ju i princip byggas ut hur mycket som helst. Det viktiga är att det verkligen fungerar i den produkt som det är tänkt för. En annan aspekt som är viktig gällande de optionella paketen, är att Javas idé om att mjukvara skall kunna köras på många olika sorters apparaturer, i detta fallet mobiltelefoner, inte går förlorad. Det finns annars en klar risk för detta. Släpps det många olika optionella paket som bara fungerar på en eller ett fåtal telefoner och utvecklare använder sig av dessa paket när de skriver program så går stor del av portabiliteten förlorad.

6.2 Framtid

Då multimediastödet i MIDP 2.0 inte var fullgott men ändå bra, då det erbjuder stöd i fem av de sex testfall som genomfördes, så bäddar det för utvecklande utav multimediaapplikationer. Multimedia möjliggör skapandet utav applikationer som interagerar med användare på flera nivåer så är det intressant att se vad för slags applikationer som kommer att utveckas i framtiden.

Att MIDP i dagsläget inte har stöd för video är inte så mycket Suns fel, det är snarare så att video inte är något som på bred front stöds av mobiltelefonerna. Om detta ändras i framtiden och videostöd blir vanligare bland mobiltelefoner så är det inte otroligt att detta skulle komma att läggas till i en framtida version av MIDP.

6.4 Fortsatt forskning

Några förslag till fortsatt forskning kan vara att titta på hur multimedia stödet i MIDP 2.0 används och då också till vilken typ av applikationer. En annan interessant fråga kan vara om det kan komma nya typer utav applikationer tack vare multimediastödet i MIDP 2.0. En tredje interessant frågeställning skulle kunna vara ur säkerhetssynpunkt, finns det risk för att virus, spyware eller andra illasinnade program kan komma att spridas med multimedia applikationer skirvna i J2ME?

Avslutningsvis i diskussionen vill jag som en personlig reflektion av MIDP 2.0 säga att det är enligt mitt tycke en väl genomtänkt profilversion, det går lätt och fort att programera i den. Som utvecklare så får man väldigt mycket gratis i form av de olika klasserna och dess metoder. Jag tror att detta kommer rendera en uppsjö av olika program i framtiden.

7 Slutsats

Uppsatsens syfte att undersöka om MIDP version 2.0, har ett fullgott multimediastöd anser jag vara genomfört och undersökningen visade att så inte var fallet. De frågeställnigar som ställs i problemformuleringen har blivit besvarade och svaren var att MIDP stödjer text, bild, ljud, animering och nätverk men saknar stöd för video. Litteraturstudierna i uppsatsen visade dock att det inom J2ME finns stöd för video men att detta inte ligger inom MIDP 2.0 utan i det optionella paketet Mobile Media API

Referenser

Böcker

Ekman, Rolf. (1985). *Filosofins Grunder.* Stockholm Nordstedts Tryckeri AB¨

Kontio, Mikko. (2002). *Mobile Java™ with J2ME*, Finland: Edita Publishing Inc. (32)

Skansholm, Jan. (1999). *Java Direkt*. Lund: Studentlitteratur

Nätreferenser

Antti Peltonen, University of Oulu, Finland. What is multimedia.
http://edtech.oulu.fi/T3/material/package2/multim01.htm
Senast besökt 2002-05-06

api\midp\index.html, 2002. *MID Profile*
http://java.sun.com/j2me/docs/
Senast besökt 2002-05-06

CNET Glossary, 2003. *API*
http://www.cnet.com/Resources/Info/Glossary/Terms/api.html
Senast besökt 2002-05-06

Educational Service District 101, 2002. *Brief history of multimedia*
http://www.esd101.net/web/training/multimedia/history.htm
Senast besökt 2002-05-19

Giguere , Eric Dec 2002. *J2ME Optional Packages*
http://wireless.java.sun.com/midp/articles/optional/
Senast besökt 2002-05-06

Hall, Prentice ,2002. *Introduction to the Java 2 Micro Edition (J2ME) Platform*
http://www.developer.com/ws/j2me/article.php/10945_1475521_2
Senast besökt 2002-05-06

Holzinger, Andreas, 2001-07-18. *Definition of Multimedia*
http://www-ang.kfunigraz.ac.at/~holzinge/mml/mml-multimedia-definition.html
Senast besökt 2002-05-06

Java.sun.com, 2003. *What is Java™ Technology*
http://java.sun.com/java2/whatis/
Senast besökt 2002-05-06

Java.sun.com, 2003. *GLOSSARY OF JAVA™ TECHNOLOGY-RELATED TERMS*
http://java.sun.com/docs/glossary.html
Senast besökt 2002-05-06

Java.sun.com, 2003. *Datasheet Java™ 2 Platform, Micro Edition*,
http://java.sun.com/j2me/j2me-ds.pdf
Senast besökt 2002-05-06

Konti, Mikko. 2003-01-09. *MIDP 2.0: The Game API*
http://www.microjava.com/articles/techtalk/game_api
Senast besökt 08-05-2003

Ortiz, C. Enrique, 2003. *The J2ME Mobile Information Device Profile 2.0*
http://www.developer.com/ws/j2me/article.php/1574791
Senast besökt 2002-05-06

Java.sun.com, *What's new in MIDP 2.0*
http://java.sun.com/products/midp/whatsnew.html
Senast besökt 2002-05-06

Java Community Process, 2003. *The Java Community Process SM (JPC)*
http://www.jcp.org/en/home/index
Senast besökt 2002-05-06

Keep, Christopher & McLaughlin, Tim & Parmar, Robin, 2000. *Ted Nelson and Xanadu*
http://jefferson.village.virginia.edu/elab/hfl0155.html
Senast besökt 2002-05-19

Manning, Mark. *Animations and Chipmunk Basic*
http://www.nicholson.com/rhn/basic/animate-note.html
Senast besökt 2002-05-08

Midlet Review. (2003). *MIDlet*, Midlet Review
http://www.midlet-review.com/navi/lexicon.htm#midlet
Senast besökt 2002-05-19

MIDP 2.0 Specification, Final (JSR 118)
http://www.jcp.org/aboutJava/communityprocess/final/jsr118/index.html
Senast besökt 2002-05-06

Java.sun.com, *Mobile Information Device Profile 2.0*
http://jcp.org/aboutJava/communityprocess/final/jsr118/index.html
Senast besökt 2002-05-21

Java.sun.com, Mobile Media API (JSR-135)
http://jcp.org/aboutJava/communityprocess/final/jsr135/
Senast besökt 2002-05-06

Mobile Information Device Profile, v2.0 (JSR-118)
JCP Public Draft Specification
http://java.sun.com/j2me/docs/

W3C, 1995-02-14. *PNG (Portable Network Graphics)*
http://www.w3.org/Graphics/PNG/
Senast besökt 2002-05-06

W3C, 1996-10-01. *PNG (Portable Network Graphics) Specification*
http://www.w3.org/TR/REC-png.html
Senast besökt 2002-05-06

Bilaga: Kod från experimenten

Textstöd

Klassen Textstod

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class Textstod extends MIDlet implements CommandListener
{
 private Display display;
 private TextField tfHello;
 private TextField tfChar;
 private Command cmExit;
 private Form fmMain;
 private String texten, str;
 public Textstod(){
  display = Display.getDisplay(this);
  fmMain = new Form("Hello World");
  char data[] = {'a'};
  texten = "Hello World";
  str = new String(data);
  cmExit = new Command("Exit", Command.SCREEN,1);
  tfHello = new TextField("String",texten ,15,TextField.ANY);
  tfChar = new TextField("char",str ,15,TextField.ANY);
  fmMain.addCommand(cmExit);
  fmMain.append(tfHello);
  fmMain.append(tfChar);
  fmMain.setCommandListener(this);
 }

 public void startApp(){
  display.setCurrent(fmMain);
 }
 public void pauseApp(){
 }
public void destroyApp(boolean unconditional){
 }
 public void commandAction(Command c, Displayable s){
  if (c == cmExit)
  {
     destroyApp(false);
     notifyDestroyed();
  }
 }
}
```

Bildstöd

Klassen Bildtest

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class Bildtest extends MIDlet {
   private Display d;
   private Bild a;
   public Bildtest() {
     d = Display.getDisplay(this);
     a = new Bild();
   }
   protected void destroyApp(boolean p0){
   }
   protected void pauseApp() {
   }
   protected void startApp() throws MIDletStateChangeException {
     a.repaint();
     d.setCurrent(a);
   }
}
```

Klassen Bild

```
import javax.microedition.lcdui.*;
public class Bild extends Canvas{

                Image bild;
                int x,y;
   public Bild() {
     try{
                bild = Image.createImage("/bild.png");
     } catch (Exception e){
       System.out.println("Error: " +e.getMessage());
     }
     x        = getWidth();
     y        = getHeight();
   }
   protected void paint(Graphics g) {
                g.setColor(0xffffff);
   g.fillRect(0, 0, x, y);
                g.drawImage(bild, x/2, y/2, g.TOP|g.LEFT);
   }
}
```

Ljudstöd

Klassen Ljudstod

```
import java.io.IOException;
import java.io.*;
import javax.microedition.media.*;
import javax.microedition.midlet.*;
```

```
import javax.microedition.lcdui.*;

public class Ljudstod extends MIDlet {
   private Ljud a;
   public Ljudstod() {
     a = new Ljud();
   }
   protected void destroyApp(boolean p0){
   }
   protected void pauseApp() {
   }
   protected void startApp() throws MIDletStateChangeException {
                a.start();
   }
}
```

Klassen Ljud

```
import java.io.IOException;
import java.io.*;
import java.lang.*;
import javax.microedition.media.*;
import javax.microedition.lcdui.*;

public class Ljud  implements Runnable {
                public Ljud(){
                }
                public void start(){
                                Thread t = new Thread(this);
                                t.start();


                try {
                                Manager.playTone(60, 500, 100);
                }
                catch (MediaException me) {}
                  try { Thread.sleep(2000); }
     catch (InterruptedException ie) {}
                  try {
                                 InputStream is = getClass().getResourceAsStream("/Boo.wav");
                                 Player p = Manager.createPlayer( is, "audio/x-wav" );
                                 p.start();
                                 }
                                 catch (MediaException me) {}
                                catch ( IOException ioe ){System.out.println("Error " + ioe);}
                }

                public void run() {
                                while (true) {
                                }
                }
```

```
}
```

Animeringsstöd

Klassen AnimeringsTest

```
import java.io.IOException;
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class AnimeringsTest extends MIDlet {
   private Display d;
   private Bakgrund a;
   public AnimeringsTest() {
     d = Display.getDisplay(this);
                  try {
     a = new Bakgrund();
                    }
     catch (IOException ioe) {
       System.out.println(ioe);
     }
   }
   protected void destroyApp(boolean p0){
   }
   protected void pauseApp() {
   }
   protected void startApp() throws MIDletStateChangeException {
     a.start();
     d.setCurrent(a);
   }
}
```

Klassen Animering

```
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;

public class Animering
   extends Sprite {
  private int danssteg = 0;
  public Animering(Image image, int w, int h) {
    super(image, w, h);
  }
  public void dans() {
                    danssteg++;
                    if(danssteg>1)
                                  danssteg=0;
    setFrame(danssteg);
  }
}
```

Klassen Bakgrund

```
import java.io.IOException;
```

```
import javax.microedition.media.*;
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;

public class Bakgrund
    extends GameCanvas
    implements Runnable {
  private Animering ani;
  int w = getWidth();
  int h = getHeight();
  int rorelse = 0;
  int hall = 3;
  private LayerManager lM;

  public Bakgrund() throws IOException {
    super(true);
   ani = skapaAnimering();
   ani.setPosition(20, 20);
   lM = new LayerManager();
   lM.append(ani);
  }
  private Animering skapaAnimering() throws IOException {
    Image image = Image.createImage("/animering.png");
    return new Animering(image, 85, 130);
  }

                public void start() {
    Thread t = new Thread(this);
    t.start();
  }

  public void run() {
    Graphics g = getGraphics();

    while (true) {
                                ani.dans();
                                rorelse++;
                                if (rorelse> 20){
                                                hall=hall*-1;
                                                rorelse=0;
                                }
                                ani.move(hall, 0);

                                 try { Thread.sleep(100); }
       catch (InterruptedException ie) {}
      render(g);
    }
  }
  private void render(Graphics g) {
                g.setColor(0x00FF33);
```

```
    g.fillRect(0, 0, w, h);
    lM.paint(g, 0, 0);
    flushGraphics();
  }
}
```

Nätverksstöd

Klassen HttpConnect

```
import java.io.*;
import javax.microedition.io.*;
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;

public class HttpConnect extends MIDlet {
   private Display display;
                TextBox tb = null;
                Kontakt k = new Kontakt();
   public HttpConnect() {
     display = Display.getDisplay(this);
   }
   public void startApp() {
                try {
                tb=k.getUrlText("http://www.ebson.se/fredrik/j2me/urltext.txt");
                }
                catch (IOException e) {
          System.out.println("IOException " + e);
          e.printStackTrace();
                }
                display.setCurrent(tb);
   }
   public void pauseApp() {
   }
   public void destroyApp(boolean unconditional) {
   }
}
```

Klassen Kontakt

```
import java.io.*;
import javax.microedition.io.*;
import javax.microedition.lcdui.*;
public class Kontakt {
   public Kontakt() {
   }

                TextBox getUrlText(String url) throws IOException {
      StreamConnection c = null;
      InputStream s = null;
                  String b ="";
      TextBox t = null;
      try {
```

```
            c = (StreamConnection)Connector.open(url);
            s = c.openInputStream();
            int ch;
            while((ch = s.read()) != -1) {

                                                    b=b+(char)ch;
            }
            System.out.println(b.toString());
            t = new TextBox("Url Texten", b, 1024, 0);
          }
                                    catch (IOException e) {
                                    }
          return t;
        }

}
```