Gruppe 3M2

Multiplayerspill til mobiltelefon
- med MIDP 2.0

et 10-poengs prosjekt i faget INF5260 Mobile Informasjonssystemer våren 2004

André Aubert
andreau@ifi.uio.no

Mats Bue
matsbu@ifi.uio.no

Martin Havnør
martihav@ifi.uio.no

1 Forord

”Multiplayerspill til mobiltelefon – med MIDP 2.0” er et prosjekt gjennomført våren 2004 i faget INF5260 Mobile Informasjonssystemer ved Institutt for Informatikk, Universitetet i Oslo.

Prosjektet gikk ut på å se på problemer knyttet til multiplayerspill til mobiltelefon og hvorfor det pr dags dato er så lite av det – selv om teknologien er her. Vi skulle også lage et multiplayerspill basert på MIDP 2.0 og bruke mulighetene man har der i forhold til MIDP 1.0. Vi har dermed både sett på eksisterende litteratur og selv fått erfare utfordringer som er med slike spill.

Dette dokumentet er slutttrapporten på dette prosjektet. Vi vil her først beskrive den eksisterende teknologien, for så å drøfte det vi har funnet ut i forhold til våre undringer ved prosjektstart. Vi vil også beskrive spillet vi har laget, spesielt kommunikasjonsdelen, uten å gjøre det for teknisk.

2 Innholdsliste

3 Innledning

3.1 Prosjektets bakgrunn

Mobiltelefoner har lenge vært allemannseie og har for de fleste blitt en god følgesvenn. Muligheten for å telefonere er fremdeles viktig, men det har vist seg at andre funksjoner blir vel så mye brukt. Dagens mobiltelefoner har gjerne funksjoner som kalender, alarm, meldingstjenester, huskelapper, internett, kanskje kamera, adressebok og ikke minst det vi her skal konsentrere oss om; spill.

Tidligere var det ikke mulig å bytte ut spillene som fulgte med mobilen fra leverandør, men etter Javas inntog i mobilverdenen er dette lettvint. Så og si alle mobiltelefoner som selges i dag har støtte for Java. Dette gjør at det har blitt enkelt for alle med noe programmeringskunnskap å utvikle spill til mobiltelefon, så i dag er det mange spillutviklere som tilbyr sine spill gjennom portaler som for eksempel Inpoc [4]. Her kan hvem som helst enkelt laste spill til sin mobil mot en betaling på typisk 29 kroner.

La oss kaste et raskt blikk mot spillmarkedet for PC'er. Der er tendensen at man beveger seg fra singleplayer til multiplayerspill over internett. Både MMG-spill (massively multiplayer) som EverQuest, RTS-spill (real time strategy) som Warcraft 3 og FPS-spill (first person shooter) som Counter Strike har vist seg å bli veldig populært. Faktisk er EverQuest blant de PC-spillene som genererer mest penger – ikke fordi de selger så mange enheter, men pga abonnementet spillerne må betale [2]. Også enklere multiplayerspill, som de man finner på Yahoo [3], har alltid mange spillere. Etter en rask telling, klokken halv to natt til niende mai, kom jeg til rundt 150 tusen samtidige spillere av deres multiplayerspill. At slike spill blir populært har klar sammenheng med utbredelsen av bredbåndforbindelse til internett for folk flest, for i PC-verdenen har man måttet vente på at enklere forbindelse mellom PC'er skulle bli vanlig før multiplayerspill virkelig har kunnet slå an.

Mobiltelefoner, derimot, er laget for kommunikasjon og er av den grunn alltid koblet sammen i nettverk. Dette gjør at de implisitt er bedre egnet for multiplayerspill enn tilsvarende en-spiller varianter, noe også Justin Hall mener når han slår fast at "Single-player games are a waste of devices built for human communication" [6]. Likevel er nesten samtlige spill som tilbys til mobiltelefon fremdeles kun for én spiller. Hvorfor har ikke multiplayerspill slått an? Tiltaler det ikke forbrukerne? Er det teknologien som setter begrensningen? Er det for dyrt for tilbyder? For forbruker? Dette skal vi prøve å gi svar på.

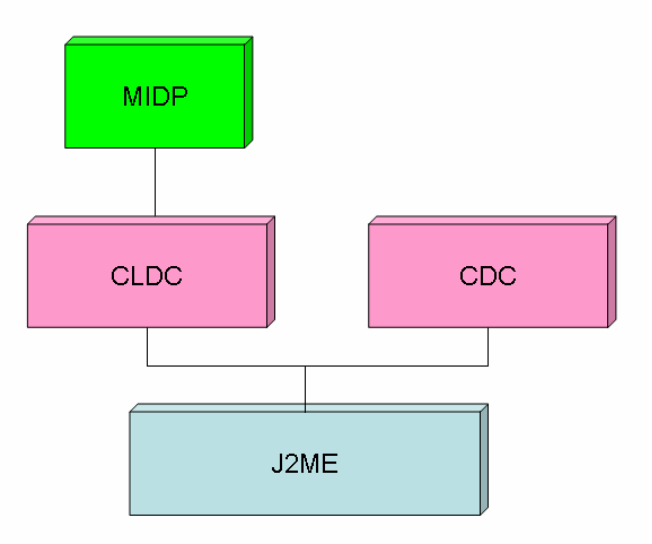
Før vi fortsetter er det viktig å få klarhet i noen begreper.

3.2 Java til mobiltelefon (J2ME) og MIDP

3.2.1 Hva er det?

Java 2 Micro Edition er en Java-versjon som er laget for enheter med begrenset minne, skjerm og prosessorkraft [5]. Den er basert på Java 2, så programmerere som er vant med J2SE vil fort finne seg til rette, men den er naturligvis redusert for å tilpasse den til små enheter.

J2ME er igjen delt i to konfigurasjoner; Connected Device Configuration (CDC) og Connected Limited Device Configuration (CLDC). CDC inneholder en Java Virtual Machine og er laget for større enheter som avanserte PDAer og datamaskiner i biler, mens CLDC er ment for mobiltelefoner og PDAer. CLDC inneholder også en Virtual Machine, men denne er kalt KVM der K gjerne blir tolket som Kilo for å symbolisere dens beskjedne størrelse.



På toppen av CLDC ligger profiler, hvor den som brukes for mobiltelefoner kalles Mobile Information Device Profile (MIDP). Gjennom MIDP får programmereren tilgang til et API som tilbyr nettverksmuligheter, brukergrensesnitt, tegning på skjerm, enkel beregning og andre ting for å lage applikasjoner til enheten. En applikasjon basert på MIDP kalles en MIDlet.

MIDP 1.0 ble lansert i år 2000 og er idag den klart mest utbredte versjonen på mobiltelefonene på markedet. Denne standarden gir en minimumsspesifikasjon på hvilken funksjonalitet mobilprodusentene må implementere, men produsentene står fritt til å implementere mer om de ønsker. For eksempel er det i følge MIDP 1.0 et minimum at rutiner for http-kommunikasjon er implementert, mens socket- og datagram-forbindelse er valgfritt. Siemens har på de fleste av sine mobiler valgt å støtte dette, mens Nokia som regel kun har støtte for http.

De nyeste telefonene på markedet begynner nå å ha støtte for MIDP 2.0, som er en oppgradering av MIDP 1.0. Her har man fått utvidete grafikkmuligheter og spillfunksjonalitet, støtte for lyd, større nettverksmuligheter, sikker http-forbindelse (https) og andre forbedringer.

For å utvide mulighetene til MIDP 1.0 har de forskjellige mobiltelefonprodusentene utviklet ekstra pakker som tilbød enhetsspesifikke muligheter som for eksempel vibrasjon, bluetooth, SMS, tilgang til adressebok, bedre grafikkmuligheter, lyd etc. Problemene med disse pakkene er at en av Javas grunntanker forsvant, nemlig plattformuanhengighet. Nokia laget sin egen pakke som kun fungerte på sine telefoner, Siemens laget en annen osv. Mye av dette er nå standardisert i MIDP 2.0 slik at det blir enklere og billigere å utvikle applikasjoner som støttes av alle mobiltelefoner, og dermed øke den potensielle kundemassen.

3.2.2 Er det noe å satse på?

Så godt som alle mobiltelefoner som selges i Norge i dag har støtte for Java, og det ser ut som om J2ME har kommet for å bli.

En grunn til at det har slått an kan være at som Java ellers er også J2ME portabelt. En applikasjon skrevet i J2ME kan kjøre på alle mobiltelefoner som støtter Java uten forandring av koden. Dette er naturligvis en sannhet med noen modifikasjoner, som også nevnt i forrige avsnitt (mobilprodusenter kan legge med egne API'er og velge hva de vil implementere), men i utgangspunktet er det slik.

En annen ting som taler for J2ME er at det gjerne er flere erfarne Java-programmerere i mange IT-selskaper, og blant studenter er det også blitt et veldig vanlig programmeringsspråk. Dette gjør at det finnes svært mange potensielle utviklere av J2ME-applikasjoner.

3.3 Multiplayerspill til mobiltelefon

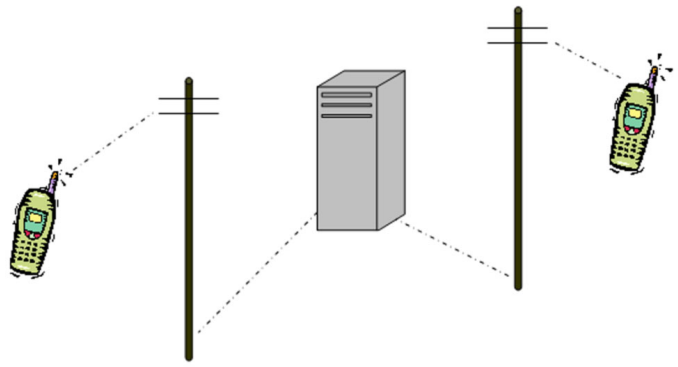
3.3.1 Hva er multiplayerspill?

Dette er naturlig nok spill med samhandling med en eller flere andre personer. Det er flere måter å gjøre dette på, men vi konsentrerer oss om spill hvor spillerne spiller samtidig og på forskjellig mobiltelefon. Spill-applikasjonene må dermed kommunisere med hverandre.

Én måte er å la mobiltelefonene koble seg direkte til hverandre med for eksempel bluetooth, kabel eller IR. Fordelen med denne type tilkobling er at man slipper treghetene i mobilnettet, og kan dermed få meget god responstid som gjør at også real-time spill går greit. Ulempen er at spillerne må være fysisk nær hverandre for å kunne spille.

Man kan også tenke seg et peer-to-peer system over mobilnettverket hvor en separat server på internett kun er til for å koble spillere sammen, og all datatrafikk går deretter direkte fra mobil til mobil. Dette er vanlig i PC-verdenen, spesielt i RTS-spill (f.eks Battle.net), men vil ikke fungere for mobiltelefoner i dag fordi mobiltelefoner ikke har egen IP-adresse [2].

En annen måte er å ha en sentral separat server på internett som mobilene (klientene) kobler seg til. All kommunikasjon går dermed gjennom denne serveren som styrer datatrafikken. Her må man tilpasse seg tregheten i mobilnettet, men fordelen med dette er at hvem-som-helst kan spille mot hvem-som-helst samme hvor de befinner seg, så utvalget av motstandere kan være veldig stort. Det er denne type system vi vil referere til som multiplayerspill i dette prosjektet med mindre vi spesifikt sier noe annet.



3.3.2 Hvorfor multiplayerspill?

”Knut’ern liker å spille. Han har flere spill til mobiltelefonen sin, og det hender han laster ned nye etterhvert som han går lei av de han har. Og det blir han! Knut’ern synes ikke spillene gir nok utfordring. Etter at han har ”rundet” et spill er det ikke like gøy å spille gjennom det en gang til. Det er også grenser for hvor mange ganger han synes det er gøy å vinne over (den relativt svake) mobiltelefonen sin i sjakk og reversi, og å stadig slå sine egne poengsummer i tetris ble han også fort lei. Knut er kronisk deprimert.”

I dette scenarioet ser vi tydelig mange av svakhetene til spill beregnet for én spiller. Mangel på variert utfordring gjør at hvert enkelt spill får en begrenset underholdningslivstid. En mobiltelefon har heller ikke kapasitet til å romme avanserte motorer for strategispill (sjakk etc), så utfordringen der vil forsvinne etterhvert som spilleren når et visst nivå. La oss ta en titt på et annet tenkt scenario som utspiller seg en tid etter det første.

”Knut liker å spille. I det siste har det gått spesielt mye tid på sengekanten til ”Yahoo! Mobile Games” hvor han stadig tester sine evner i sjakk og reversi mot tusener av andre mennesker over hele kloden. Han klatrer stadig på rangeringslisten og er nå rangert som #2507 i sjakk med en poengsum på 1683. Rangeringslisten sjekker han ofte på wap. Knut sier det gir en mye bedre følelse å vinne over et menneske enn en maskin som ikke bryr seg om den vinner eller taper, dessuten blir vanskelighetsgraden svært variert etter hvem man spiller mot.

Det er ikke bare brettspill Knut bruker tid på. Han liker også mer action-pregede spill, som Mobiluz, hvor spillerne utfører handlinger samtidig og kontinuerlig. Her kjemper Knut i teten på rangeringslisten og håper å være blant de fem beste ved slutten av måneden, for de får hver sin t-skjorte. Knut er ikke lenger deprimert.”

Etter å ha sett på disse to fiktive, men fullstendig realistiske scenarioene, ser vi at multiplayerspill virkelig har mulighet til å gi et underholdende løft for brukere i forhold til singleplayerspill. Også for mobiloperatører er det grunn til å glede seg over multiplayerspill. Slike spill er nødt til å kommunisere, og det gjør at de skaper datatrafikk på mobilnettet som igjen operatørene tar betalt for. Vi vil komme nærmere inn på dette senere.

3.3.3 Hvordan ser markedet ut?

Spill til mobiltelefon er populært, og tendensen viser også sterkt oppadgående kurve. I følge det amerikanske selskapet Frost & Sullivan vil markedet for mobilspill i Europa vokse fra $801 millioner i 2003 til nesten $7 milliarder i 2006, mens andre analytikere er forsiktigere og spår $3.54 milliarder i 2008 [1]. Uansett er det utvilsomt et marked i stor vekst.

Fra brukerens side er utfordringen, og dermed intensivet til å spille langt høyere i et spill som tilbyr menneskelige motstandere. Dette fører igjen til mulighet for større inntjening for tilbydere, men ikke bare i form av flere solgte enheter; månedlige abonnements-avgifter på spillene samt betaling for online tid/datamengde skaper inntekt.

Som vi tidligere så har multiplayer PC-spill fått en stadig større markedsandel, så vi regner med at lysten på multiplayerspill er generell, og dermed også er gjeldende i mobilverdenen.

4 Utfordringer og løsninger

4.1 Multiplayerspill

Som nevnt finnes det minimalt med multiplayerspill til mobiltelefon på markedet i dag. Vi skal her se på endel problemer knyttet til slike tjenester og hvordan man kan minimalisere disse.

4.1.1 Mange spillere er nødvendig

”Knut’ern følger med på det som skjer i mobilverdenen. Med en gang multiplayerspillet Mobiluz ble lansert, var han snar med å laste ned en gratisversjon til mobiltelefonen sin. I stor iver startet han MIDlet’en og koblet seg til spillserveren hvor han fikk tildelt kallenavnet guest1023 som uregistrert spiller. I spill-lobby’en (der man velger motstander) var det en håndfull andre som også tydeligvis hadde fått med seg lanseringen. Knut utfordret en av dem og han syntes spillet var riktig så festlig. Han vurderte med en gang sterkt å betale litt for å få registrert et eget kallenavn og få tilgang til rangeringslisten.

Utpå kvelden, da han hadde lagt seg, koblet han seg på igjen for å spille et par kamper før han sovnet. Men det var ingen andre enn Knut i lobby’en! Det samme skjedde dagen etter. Knut la bort spillet, og det tok lang tid før han kom tilbake. Istedet spilte han på Yahoo! Mobile Games, som også var helt nytt på den tiden. Der var det alltid noen tilgjengelig for spilling. Om ingen andre var tilstede, kunne han alltid stole på MasterAI og WizBot. De må virkelig være avhengig, mente Knut.”

Hva er vel et multiplayerspill om man ikke har noen å spille mot? Knut har forstått dette, og det må også spilltilbydere gjøre. En slik spilltjeneste er nødt til å ha mange spillere om den skal være levedyktig. Om potensielle spillere stadig kommer til en tom lobby, vil nok mange av dem gjøre som Knut og finne andre steder å spille. Ved lansering av en slik tjeneste er derfor bred markedsføring absolutt nødvendig i startfasen for å tiltrekke spillere.

Noe utviklere også kan gjøre for å unngå at folk møter på en tom lobby er alltid å ha et par AI-spillere i lobby’en man kan spille mot om det ikke er andre pålogget. Eventuelt kan de dukke opp kun når aktiviteten ellers er liten. AI-spillerne kan bli styrt av serveren eller mobilen selv.

4.1.2 Tregt mobilnettverk

Et annet viktig moment som setter begrensninger på muligheter med multiplayerspill er dagens GSM mobilnettverk med bruk av GPRS. Det har en forholdsvis treg overføringshastighet, men spesielt er det den høye responstiden, som typisk er i underkant av halvannet sekund ved TCP [8], som lammer (men dette kan variere noe fra nettverk til nettverk). GPRS er heller ikke prioritert i mobilnettverket og vil kun få benyttet ledig kapasitet, noe som gjør det til en upålitelig kommunikasjonsbærer.

Mobilnettverket er det lite å gjøre noe med fra spillutviklers side, så løsningen vil være å tilpasse spillene og kommunikasjonen til nettverket som er gjeldende. Inngående tester [7, 8]

har vist at GPRS er uegnet til realtime spill (”action games”) grunnet den høye responstiden, men at andre typer spill går greit.

Det er også et poeng for utvikler å kamuflere høy responstid så brukeren skal oppleve minst mulig venting. En måte å gjøre dette på er at det ved venting på respons er ”noe som skjer” som tar oppmerksomheten til brukeren, for eksempel en splashscreen, skifting av kameravinkel, animasjoner eller lignende. Et annet triks, som blir brukt i FPS-spill til PC, er å la spillet ”gjette” hvor motstandere befinner seg i neste øyeblikk utfra bevegelsesbane og fart, så bevegelsene deres ser jevne ut.

Et annen stor utfordring en må ta hensyn til er at en spiller når som helst kan miste forbindelse til spillserver! Om han sitter på toget kan toget kjøre inn i en tunnel så han mister dekning, batteriet kan gå ut, det kan komme telefonsamtaler etc. Slike hendelser vil gjerne være svært irriterende for en motspiller. Det er opp til spillutvikleren å utformet spillet så slike problemer vil bli behandlet på en smidig måte. Hvordan det skal gjøres avhenger av type spill, men uansett bør en spiller skånes for lengre tids inaktivitet. For eksempel kan spilleren få beskjed om at serveren har mistet kontakt med motspilleren, så han får valget mellom å vente på motspillerens tilbakekomst, eller han kan forlate spillet.

4.1.3 Kostnader for spiller

Et multiplayerspill vil helt sikkert ikke slå an hvis det er dyrt å spille det. Prisen på selve applikasjonen (og et eventuelt abonnement) er en ting, men noe som tilbyderen ikke styrer er prisen på datatrafikk over mobilnettverket. Her er det operatøren som setter prisen.

Japan har hatt teknologi for multiplayerspill i lang tid, men det har ikke tatt av pga svært høye priser på mobil datatrafikk [11]. I Sør-Korea derimot har de veldig raskt mobilnettverk, og prisene langt lavere enn i Japan og Norge, så multiplayerspill til mobil har blitt veldig populært [11, 12, 13].

På dagens GPRS-nettverk i Norge betaler man pr kilobyte som overføres. Pr 11.05.2004 koster det på Telenor sitt nett 10 øre pr kB ved datatrafikk opp til en halv megabyte på en måned [9]. På trafikk mellom 0,5 megabyte og opp til 20 megabyte er prisen 1,5 øre pr kB, og på alt over 20 megabyte koster hver kB 0,1 øre. En halv megabyte koster altså 50 kroner, en hel megabyte koster 57,50 og fem megabyte koster 117,50. Som vi ser går 50 kroner forholdsvis fort, men etter det blir det mye mer trafikk for pengene.

Inntil mobiloperatørene tilbyr en fast avgift som ikke påvirkes av datamengden som blir overført, er det et poeng å forsøke å minimere nødvendig datatrafikk mellom klient og server.

Http-forbindelse, som jo er standard på MIDP 1.0, er svært dårlig egnet til dette. Hver http-spørring inneholder en stor nødvendig header som i de fleste tilfeller vil være mange ganger så stor som selve datamengden vi ønsker å overføre for spillets del. I tillegg er ikke http egnet for toveiskommunikasjon. Det er ingen måte for serveren å fortelle klienten noe før klienten har spurt. Dette gjør at klienten blir nødt til å polle server jevnlig for å spørre om det er noe nytt. Det sier seg selv at dette er unødvendig datatrafikk som man bør unngå om man kan. Og det kan man.

Heldigvis finnes det også andre protokoller, som den vi har basert oss på i vårt spill; socket-forbindelse (TCP). Her har man en åpen forbindelse mellom server og klient hvor begge kan

sende beskjeder til hverandre. Headeren er også svært beskjeden i forhold til http. TCP er en sikker og pålitelig protokoll som garanterer at hver pakke kommer frem og at de kommer frem i riktig rekkefølge [8].

UDP (datagram) er en annen protokoll som har enda mindre overhead enn TCP og er gjerne den mest effektive. Problemet er at UDP er en upålitelig protokoll [8], så den bør bare brukes på pakker som ikke er kritiske, spesielt på et så upålitelig nettverk som dagens mobilnett.

I tillegg til å velge en passende protokoll bør utvikler naturligvis tenke på å overføre så lite som mulig i selve spillkommunikasjonen. Bytekode tar mindre plass enn tekst, og forandringer i spillet (deltaer) bør overføres istedenfor komplette spilltilstander.

4.1.4 Kostnader for tilbyder – er det noe å tjene?

Også for tilbyderen av multiplayerspill er det kostnader. I tillegg til kostnadene for selve utviklingen av spillet, kommer det løpende kostnader for opprettholdelse og vedlikehold av server samt båndbredden. Kostnadene er likevel svært beskjedne i forhold til endel multiplayerspill til PC av type EverQuest [8].

Mange potensielle tilbydere holder trolig igjen om de frykter at kostnadene vil bli større enn inntektene, og vi har inntrykk av at det er en viss pessimisme mht mulige inntekter. Vi kontaktet Plutonium Software [10], som vi anser å være blant de fremste i Norge når det gjelder spill til mobiltelefon, på mail for å høre deres tanker om multiplayerspill til mobil. Svaret vi fikk var at problemet er når man tenker internasjonalt og tar høyde for alle mobilterminaler som finnes på markedet. Kombinasjoner av vanskelige terminaler/operatører kan skape problemer for slike spill, mente de. Videre kom det på det rene at å lage multiplayerspill som fungerer på ”enkle” terminaler i ”enkle” markeder faktisk er uproblematisk, men de la til at det ikke blir penger uten volum – så det er tydeligvis kun liten tro på stor fortjeneste som stopper Plutonium Software.

Så hvilke muligheter har tilbydere til å tjene penger?

Et multiplayerspill genererer som nevnt datatrafikk som spilleren må betale for til mobiloperatøren sin, som for eksempel Telenor. Dette tjener Telenor på, så det kunne vært mulig å få til en avtale mellom mobiloperatøren og spill-leverandøren om å dele inntektene fra trafikken generert av spillet. Å få til dette i praksis kan være vanskelig for det spørs om mobiloperatørene er villige til å dele på inntekten sin.

En annen løsning er at spillerne betaler en engangskostnad for å laste ned spillet, slik singleplayerspill blir solgt i dag. Eventuelt at man i tillegg har valget om å registrere seg, og dermed få økte rettigheter og goder, mot en liten sum hver måned. Dette kan være i form av at man gjennom spilling over tid utvikler en profil/karakter. Denne fungerer i spill-verdenen som en identitet for spilleren, og blir et intensiv til å faktisk betale for kontinuerlig spilling, fremfor å utnytte for eksempel en første gratis uke om og om igjen.

4.2 Utvikling av applikasjoner til mobiltelefon

Mobiltelefoner har fremdeles svært beskjeden kapasitet, så en programmerer har en rekke begrensninger å forholde seg til ved utvikling av slike applikasjoner.

4.2.1 Minne

Dagens mobiltelefoner har betydelig mindre internminne enn en vanlig PC, typisk rundt en tusendel. Dette gjør at en mobil ikke kan prosessesere mye data på en gang. I tillegg er lagringskapasiteten for applikasjoner svært begrenset, slik at det f eks er vanskelig å lagre mye grafikk og lyd (som tar mye plass) .

4.2.2 Prosessorkraft

Mobiltelefoner har begrenset prosessorytelse grunnet krav til lavt strømforbruk og mobilitet. Derfor er det begrenset hvor mye man kan bruke mobiltelefonen til å prosessere data.

4.2.3 Skjerm

En mobiltelefon har en skjerm som er betydelig mindre enn det en vanlig PC benytter, gjerne ca 2% av skjermområdet. Siden vi dermed ikke kan ha så mange elementer på en gang, må data presenteres på en enkel måte, samtidig som at den må være lett å oppfatte siden skjermen er liten fysisk og dermed kan være vanskelig å lese.

4.2.4 Brukerinput

De fleste mobiltelefoner benytter et numerisk tastatur hvor det er begrenset antall taster og styremuligheter. Applikasjonene må dermed ikke basere seg på mye og rask brukerinput.

4.2.5 Begrensninger i J2ME

Selv om J2ME gjør at man kan programmere på et høyt nivå uten å bry seg om de spesifikke detaljene om hvordan den enkelte mobiltelefon er bygget opp, er det likevel en del begrensninger som kan komplisere ting.

Blant annet er det ikke støtte for flyttall slik at desimalregning må gjøres på annen måte. I forhold til J2SE mangler det også geometriske funksjoner som sinus og cosinus, så disse må erstattes med prekalkulerte tabeller. Serialisering av objekter for f eks overføring over nettverk støttes heller ikke.

5 Erfaringer gjennom praksis

5.1 Hvordan er produktet vårt bygget opp?

5.1.1 Kort om spillet

Spillet Morps (Multiplayer Online Rock Paper Scissor) er et multiplayer spill for mobiltelefoner som vi har implementert som en del av oppgaven. Idéen baserer seg på den kjente leken stein-saks-papir.

Multiplayerfunksjonaliteten fungerer slik at en spiller logger seg inn med et valgt brukernavn, og kommer deretter inn i lobbyen. Der får man oversikt over hvilke spillere som venter på motstandere. Disse kan man velge å spille sammen med, eller man kan starte sitt eget spill som andre kan bli med på. Når to spillere har avtalt å spille sammen starter spillet og selve spillskjermen vises. Her får man 30 sekunder på å velge sine trekk, deretter sendes de til den andre spilleren. Når begge spillere har sendt sine trekk til hverandre beregnes det hvem som har vunnet, og dette vises på skjermen. Deretter går begge spillerne tilbake til lobbyen, hvor man kan spille på nytt.

Det er tatt hensyn til at kommunikasjonen mellom mobiltelefonene og serveren kan forsvinne, og i slike tilfeller ordner serveren opp og sender motspilleren tilbake til lobbyen med beskjed om at forbindelsen til den andre spilleren ble brutt.

3D-motoren har blitt tatt i bruk flere steder i spillet. Den er meget fleksibel og kan enkelt brukes til mange forskjellige oppgaver.

5.1.2 Abstraksjon

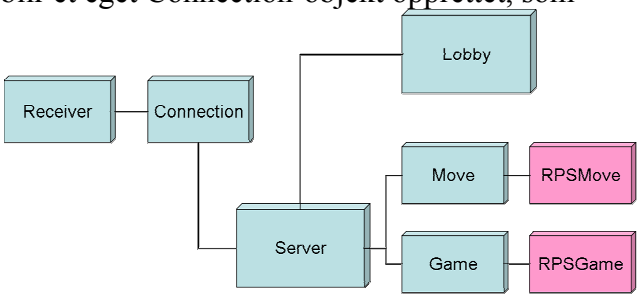
Både tjener (server) og klient (mobilapplikasjon) er konstruert for å være fleksible med hensyn til hvilke spill som kan kjøres. Ved å abstrahere seg fra hvordan hvert enkelt spill vil fungere i detalj og isteden se på hvilke tjenester et generelt spill trenger, har det vært mulig å bygge et fundament som nær sagt alle typer spill vil fungere på. Det vil igjen si at spillutvikleren bare behøver å forholde seg til hvordan nettopp det spillet han selv utvikler skal fungere, og deretter benytte de ferdige rutinene til å realisere dette. Dette gjør terskelen for å utvikle spill til mobiltelefon lavere, siden utvikleren ikke behøver å tenke på de spesielle rutinene for kommunikasjon som gjelder for mobiltelefoner. Dette vil igjen kunne føre til at flere utviklere kommer på banen når det gjelder mobiltelefonspill, og at selve utviklingen av et spill går hurtigere.

5.1.3 Oppbygging av systemet

Konkret i vårt tilfelle har vi laget en server som tar seg av kommunikasjon mellom klientene og en abstrakt superklasse som spill kan utvikles på toppen av. Deretter har vi laget et konkret spill (basert på Stein-Saks-Papir) som benytter denne plattformen (i tillegg har vi laget et chatte-”spill” som vi brukte under utvikling og testing av plattformen). Oppbygningen er skissert nedenfor.

Tjener

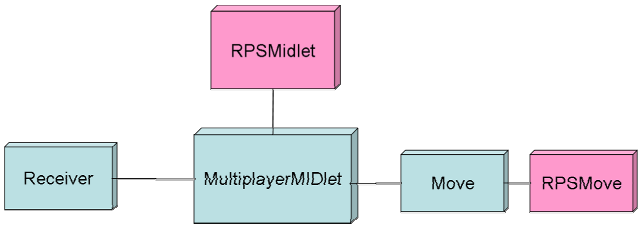
Tjeneren er implementert i pakken ”server” og består av nødvendige klasser for å støtte den underliggende nettverkskommunikasjonen. Server-klassen lytter på en gitt port etter klienter som vil koble seg til. Når en klient tar kontakt blir et eget Connection-objekt opprettet, som vil ha ansvaret for all videre kommunikasjon med den ene klienten. Hvert Connection-objekt har også et eget Receiver-objekt som kjører i en egen tråd. Denne lytter etter innkommende beskjeder fra klienten og sender så beskjed til Connection-objektet som behandler forespørselen. Tjeneren har også en lobby, som inneholder en liste over spillere som er logget på tjenesten og hvilke spill som spilles for øyeblikket. Game-klassen er en abstrakt klasse som representerer felles logikk som hvert type spill trenger. Av denne lages det spesifikke klasser for hvert spill, som for vårt spill Morps er RPSGame. Denne inneholder spesifikke instruksjoner om hva som skal skje når dette spillet mottar et trekk fra klientene (som jo kan ønskes å behandles forskjellig avhengig av hvilket type spill som drives på tjenesten). Grensesnittet Move inneholder metoder for å serialisere og de-serialisere, og må implementeres av de konkrete Move-klassene for hvert spill (som inneholder de trekkene spillerene sender til hverandre, og kan bygges opp akkurat slik utvikleren ønsker). Grensesnittet Traffic består av konstanter som både tjener og klient bruker for å snakke sammen.



Klient

Klienten behøver mange funksjoner for å ta seg av kommunikasjon og annen logikk som kreves for å samhandle med tjeneren. For å skjule denne logikken fra spillutvikleren har dette blitt samlet i en felles superklasse, MultiplayerMIDlet, i pakken client. Denne klassen er igjen en subklasse av MIDlet, som alle Java-applikasjoner til mobiltelefon må arve fra. Her er all logikk som ikke er direkte relatert til det konkrete spillet samlet. I tillegg inneholder klientpakken en Receiver-klasse tilsvarende tjeneren (for å lytte på forespørsler fra tjeneren) og Move og Traffic (de to siste er identiske med tjener-versjonene, nettopp for å ha en felles kommunikasjonsbasis).

Vårt konkrete spill Morps, implementert i klassen RPSGame, arver igjen fra MultiplayerMIDlet, og inneholder logikk spesielt for dette spillet. Når spillet vil kommunisere kaller den enkelt ferdig implementerte metoder i MultiplayerMIDlet. Når klienten mottar anrop fra tjeneren blir disse behandlet i MultiplayerMIDlet og deretter sendt til metoder i det konkrete spillet (RPSGame), slik at spillutvikleren kan velge å presentere dataene slik man ønsker.



Et eksempel

Som et eksemepel, la oss tenke oss av spillet vårt (RPSGame) vil ha tak i de personene som venter på motstandere i lobbyen. Den kaller da opp en ferdig metode i MultiplayerMIDlet som videre sender en forespørsel til tjeneren. Receiver-klassen til Connection-objektet som

svarer til klienten mottar beskjeden, dekoder den (ved hjelp av Traffic) og leter deretter gjennom Lobby-objektet etter spillere. Deretter sendes dette tilbake til klienten, som mottar data via Receiver-objektet sitt og kaller opp riktig metode i MultiplayerMIDlet. Så kalles en metode opp i RPSGame, og dataene presenteres på valgt måte. I dette eksempelet er det eneste de forskjellige spillene trenger å implementere selv måten de ventende spillerene presenteres på, alt annet er ferdig implementert og uavhengig av hvilket spill som kjøres.

5.1.4 Bruk av nye nettverksmuligheter i MIDP 2.0

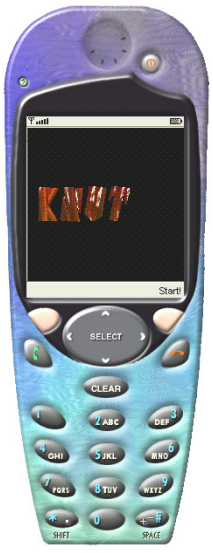
Noe av prosjektets mål var å effektivisere kommunikasjonen i forhold det som var mulig med tidligere versjoner av MIDP og http. Derfor valgte vi å bruke Sockets som nettverksprotokoll under overføring av data mellom klient og tjener. Tjeneren lytter på en bestemt port etter klienter som vil koble seg til, og når en klient tar kontakt får denne tildelt en socket, som fungerer som det ene endepunktet for kommunikasjonen med serveren. Samtidig opprettes en annen socket i andre enden av kommunikasjonslinjen, hos klienten. Deretter kan tjener og klient snakke sammen uforstyrret.

De to største fordelene med denne formen for kommunikasjon er at veldig lite annet enn de nødvendige dataene overføres i motsetning til http, samt at begge parter kan ta kontakt og sende data asynkront og uavhengig av hverandre. Dette fører i første rekke til enklere, raskere og billigere dataoverføring. Dette merket vi klart når vi programmerte og testet ut systemet vårt. I forhold til tidligere arbeider med MIDP 1.0 og http var det enklere å få klient og tjener til å samarbeide, og informasjonen ble sendt mye kjappere. Vi vil anta at de nye nettverksmulighetene vil gjøre at flere multiplayer mobiltelefonspill vil dukke opp etterhvert, siden kravet for å få en god nok kommunikasjon er betydelig lettere å oppnå nå enn tidligere.

Socket-kommunikasjon gir god responstid. Vi utførte en ”real-life” test ved å måle tiden det tok fra en ekte mobiltelefon sendte data til tjeneren fanget opp dette. Dette viste seg å skje nærmest øyeblikkelig, og er en stor forbedring fra http, der responstiden kunne være på langt over sekundet. Når det gjelder hvor mye datamenger spillet genererer forteller tallene oss at sockets har hatt en særdeles god effekt også her. Vi utførte en test ved hjelp av emulatorens innebygde nettverksmåler. Ved en normal spilleomgang, inkludert pålogging, sender klienten 44 byte og mottar 58 byte, tilsammen 102 bytes. Dvs ca 0,1 KB for et spill (som med de gjeldene prisene ikke koster mer enn 1 øre)! Dette er en betydelig forbedring for en tilsvarende spill basert på http, der vi opplevde tall på 15-20 KB for en ganske tilsvarende spilleomgang.

5.1.5 Utvikling av 3D-motor

MIDP 2.0 muliggjør 3D-grafikk



MIDP 2.0 tilbyr en rekke nye muligheter for grafikk. Av disse er vår grafiske fremstilling helt avhengig av to nye metoder: ”Graphics.getRGB()” og ”Graphics.drawRGB() ”. Førstnevnte transformerer et bilde, dvs. et Image-objekt, til en liste av 32-bits heltall verdier. Denne kan så brukes i diverse utregninger for å generere neste bilde som skal tegnes på den mobile enhetens skjerm. Til dette brukes den andre metoden; drawRGB(). Denne gjør det motsatte; tar en liste av 32-bits heltall verdier og lager et Image objekt, som så skrives til skjerm. Tidligere versjoner av MIDP tilbød ikke disse metodene, og utviklere

hadde ingen andre muligheter til grafikk enn de eksisterende metodene for tegning av linjer, tekst og lignende. Med de nye metodene er det nå mulig å konstruere 3-dimensjonal vektor/polygonbasert grafikk, som åpner helt nye muligheter. I tillegg kan slik grafikk redusere nettverkstrafikk, ettersom kun hjørnepunkter og objekt-lokasjon behøver transmisjon.

Begrensninger vil selvfølgelig alltid eksistere, men de er nå flyttet over på hardware, i form av prosessorkraft og tilgjengelig minne. Utviklere kan dermed til en langt høyere grad lage produkter som gir varierende bildekvalitet basert på gjeldende enhet.

Egenskaper ved 3D-motoren

Genereringen av 3D-grafikk er basert på en rekke objekter i 3D. Hvert objekt har en modell som ligger i egne filer under applikasjonen. Modellene lagres i et proprietært format som er basert på ascii formatet brukt av 3D-studio [17] 4.0. Når objektet skal tegnes gis modellen enten en fast farge, den kan skyggelegges eller dekkes av et bilde – også kalt en ”texture.” Hva som vises bestemmes av motorens kamera, som kan flyttes og roteres på samme måte som andre objekter i 3D-verdenen.

5.1.6 Løsning på J2ME-spesifikke problemer

J2ME tilbyr kun et begrenset utvalg metoder, spesifisert gjennom MIDP. En del av manglene skapte problemer for oss, som vi i det store og hele lyktes godt i å overkomme.

Desimaltall

For å komme rundt dette foregår all utregning opphøyd i en toerpotens. Når resultater så skal brukes; tegnes på skjerm eller annet, skiftes det tilsvarende til høyre.

Sin() og Cos()

Vi inkluderer en prekalkulert tabell av sin() fra og med 0 til og med 449 ganget med 2^10. 0-359 er verdiene for sin(), 89-449 for cos(). Skal for eksempel utregningen ”y = y * sin(x)” utføres, vil først verdien til sin(x) hentes ut av tabellen, så ganges med y. Resultatet av dette vil skiftes ti ganger til høyre.

Fil- og ”stringparsing”

Også innlesing og tolkning av filer er utelatt i MIDP. Vi lette rundt etter tidligere løsninger på problemet, og fant to open-source prosjekter som omhandler innlesing og tolkning av xml-filer; Kxml [14] og Nanoxml [15]. Begge er lettvektsapplikasjoner, men introduserer allikevel betydelig ekstra minneforbruk. Jar-arkivet med spillet blir også større, både fordi XML-parseren tar plass, og fordi objektene må lagres i XML-format som gjør dem større. Vi valgte derfor å skrive noen få metoder selv, som tar seg av det vi behøver.

Serialisering

I motsetning til standardutgaven av Java, hvor det er mulig å enkelt sende nær sagt alle typer objekter over en nettverksforbindelse, støttes dette ikke i J2ME. Når klientene skal utveksle sine trekk i spillet hadde det vært ønskelig å samle alle de nødvendige dataene i et objekt, for så å sende dette til den andre klienten. Dette er ikke mulig uten videre, isteden har vi valgt å bryte hvert objekt ned i primitive datatyper som int, byte, char osv, sende disse over linjen, for deretter å bygge objektet opp igjen trinn for trinn i den andre enden.

5.1.7 Maskinvarebegrensninger

Et annet sett med begrensninger vi har måttet ta hensyn til, gis implisitt i maskinvaren vi jobber mot.

Minne

Vi ønsket å bruke litt avansert grafikk i spillet for bedre å vise mulighetene ved MIDP 2.0. Slik grafikk er minneintensivt, så det har vært nødvendig å legge mye vekt på reduksjon av minneforbruk. Skulle spillet utgis, måtte vi ha laget en langt enklere utgave som kunne kjøres dersom enheten skulle vise seg å ha for lite minne til den grafiske. Dette har vi ikke hatt tid til å implementere, men det er en problemstilling vi er klar over og ser viktigheten av.

Prosessor

Grafikk av typen vi bruker har eksistert siden 386-maskiner var vanlige, og fungerte til en viss grad allerede da. Mange av dagens telefoner er langt kraftigere enn dette. Det er allikevel en reell sjanse for at prosessorene til MIDP 2.0 telefoner kan gi dårlig ”frame-rate” på spillet vårt, ettersom vi kun har fått testet det på emulator hittil.

5.2 Tjenesten i praksis

Foreløpig er det bare et fåtalls mobiltelefoner på markedet som støtter MIDP 2.0, så ”real-life” testing har ikke vært så lett å utføre. Tidligere erfaringer med Kattekvinnan Spillserver [16] tilsier at det som fungerer på emulatorene ikke nødvendigvis fungerer like glatt på de virkelige mobilene. Derfor ønsker vi å gjøre omfattende tester når dette blir en mulighet. Vi vil gjette at det ikke tar lang tid (6-12 mnd) før telefoner med MIDP 2.0-støtte er blitt vanlig i markedet, og det er først da tjenesten vår virkelig vil kunne settes ut i praksis.

5.3 Problemer med multiplayerspill?

Vi støtte ikke på nevneverdige problemer under utviklingsarbeidet. Med sockets går nettverkskommunikasjonen ganske smertefritt, og å implementere multiplayerspill krever ikke nødvendigvis mye mer arbeid enn et singleplayerspill. Når brukermassen blir stor trenger man etter hvert bedre båndbredde og større servere som trenger vedikehold, men det skal mange spillere til før dette blir et aktuelt problem.

Et mulig ankepunkt mot multiplayerspill på mobiltelefoner kan være at brukerene ofte ikke har god tid til å spille, at spillingen blir et tidsfordriv i korte perioder når man venter på noe annet. Dermed vil ofte en spiller hoppe ut av spillet når ventetiden er over, og således vil den andre spilleren oppleve å ikke kunne spille ferdig. Dette problemet kan minimaliseres ved å gjøre hver spillomgang kortere, og å tilby rangeringslister slik at spillere gjerne vil fullføre påbegynte spill for å kunne klatre på listen.

Dagens GSM-nettverk i Norge begynner å trekke på årene, og gir ikke spesielt god overføringshastighet eller responstid. Men tilpasser man spillet til dette, ved å forsøke å minimalisere datamengden som sendes og å ikke gjøre seg avhengig av hurtig respons, er disse problemene greie å unngå.

6 Hva skjer videre?

6.1 Oppsummering

Vi har med denne oppgaven sett på multiplayerspill for mobiltelefoner basert på MIDP 2.0, og kan ikke se noen store hindringer for at en tjeneste liknende vår skal kunne settes ut i drift. Det er ikke spesielt dyrt å lansere det (man trenger stort sett bare en maskin med fast linje til internett for å kjøre tjeneren). Vi kunne enkelt satt opp en fungerende tjeneste, noe vi for så vidt også har gjort (bare ikke lansert den offentlig). Derfor lurer vi fortsatt på hvorfor ingen andre slike tjenester er lansert her til lands.

Plutonium Software [10] mener at en slik tjeneste ikke ville være lønnsom fordi markedet ikke er stort nok. Dette stiller vi oss kritiske til. Riktignok vil antall brukere som har muligheten til å benytte tjenesten den nærmeste tiden være relativt liten, men det tar nok ikke lang tid før kundemassen har økt betraktelig (pga hyppig utbytting av mobiltelefoner).

Når man sammenlikner med populariteten til multiplayerspill på PC-er er det viktig å få med at de har hatt lang tid å utvikle seg på, og populariteten her mye skyldes at kvaliteten på spillene etter hvert har blitt meget god. Disse spillene har også mulighet til å tilby funksjonalitet (og underholdningsverdi!) langt forbi hva dagens mobiltelefoner er i stand til. Derfor er det rimelig å anta at man ikke vil se den samme oppslutningen rundt mobiltelefonvariantene i umiddelbar fremtid. Dessuten er motivasjonsfaktoren for å spille PC-spill og mobiltelefonspill ofte forskjellig. Mens de fleste PC-brukere spiller for underholdningens skyld og ofte bruker lang tid når de først begynner, blir gjerne mobilspillene brukt til tidsfordriv når man bare har noen minutters ledig tid (og ikke har tilgang til PC…) [1]. Sagt på en annen måte, for de som virkelig vil spille multiplayerspill er ikke mobiltelefonen førstevalget som medium. Markedet er heller de som søker et morsomt tidsfordriv når de er på farten. Dette må spillene tilpasse seg til for å kunne overleve, og noen viktige momenter her er at spillet er enkelt, raskt å sette seg inn i, går raskt å spille en omgang og selvfølgelig underholdende. Å kunne kjempe på rangeringslister er nok også en viktig motivasjonsfaktor.

6.2 Fremtiden

Vi mener det er rimelig å anta at multiplayerspill til mobiltelefon blir mer og mer utbredt, spesielt når vi får tilgang til raskere og bedre nettverk og telefoner. Etter populariteten til singleplayer mobilspill å dømme er folk meget interessert i å spille spill på mobiltelefonene sine, og den ekstra dimensjonen ved å spille mot andre vil etter hva vi kan se bare øke populariteten, siden vi har kunnet følge den samme utviklingen når det gjelder PC-spill. Spesielt når en tenker på at mobiltelefoner fra grunnen av er et kommunikasjonsmiddel. Men lave priser vil være essensielt for at folk skal fortsette med å spille. Det samme kan sies om viktigheten av at alle kan bruke tjenesten, og her hjelper standarder som MIDP 2.0 godt på vei.

I denne sammenhengen kan det også være på sin plass å se mot øst, nemlig hvordan utviklingen har vært innen dette området i Korea. Her har raskt mobilnett (3G) eksistert siden 2002 [13] (et tilsvarende nett er på trappene til å lanseres for vanlige brukere i Norge). I tillegg har utviklingen av mobiltelefonene kommet betydelig lenger der, telefoner med

videokamera og fargeskjermer har vært vanlige i lang tid. Dette har ført til at multiplayerspill har blitt både vanlig og populært [12]. Det er sannsynlig å anta at en lignende utvikling vil skje når Norge og Norden får tilsvarende nettverk. Spørsmålet er bare om vi er like vant med eller villige til å ta i bruk teknologi for å underholde oss som den gjennomsnittlige koreaneren (de er nemlig glade i å spille!).

Angående kildekode:
Vi har valgt å ikke legge ved kildekode til tjenesten vår fordi vi ikke ønsker å gjøre den offentlig tilgjelgelig.

7 Referanser

Forum Nokia

[2] Multiplayer Mobile Games: Business Challenges And Opportunities, 16 April 2004
[7] Multiplayer Game Performance over Cellular Networks, 20 Januar 2004
[8] Multi-Player MIDP Game Programming, 29 Oktober 2003

Andre artikler

[1] Mobile Gaming – Today and Tomorrow, Business Briefing: Wireless Technology 2004, Atte Miettinen, Chief Marketing Officer, End2End
[6] Multiplayer – the Only Mobile Game, Justin Hall, TheFeature
[11] Packets, Prices and Multiplayer, Mobile Entertainment Analyst, 03 oktober 2003, Matthew Bellows
[12] Korea's Mobile Multiplayer, 27 januar 2004, Justin Hall, TheFeature
[13] Framtida kommer fra øst, 4 mai 2004, Dagbladet – http://www.dagbladet.no/dinside/2004/05/04/397470.html
[16] KSS – Kattekvinnan Spillserver, Hovedprosjekt IU/HIO våren 2003, André Aubert, Mats Bue, Harald Børresen, Vegard Dehlen http://student.iu.hio.no/hovedprosjekter/2003/data/07/

Nettsider

[3] Yahoo! Games – http://games.yahoo.com
[4] Inpoc – http://www.inpoc.no
[5] J2ME – http://java.sun.com/j2me
[9] GPRS-priser hos Telenor – http://telenormobil.no/priser/tjenester/gprs/
[10] Plutonium Software – http://www.plutoniumsoftware.com
[14] Kxml – http://kxml.enhydra.org/index.html
[15] Nanoxml – http://sourceforge.net/projects/nanoxml
[17] Discreet 3d-STUDIO – http://www.discreet.com/