Mikko Tillikainen

KUVANLÄHETYSSOVELLUS SYMBIAN-KÄYTTÖJÄRJESTELMÄLLE

Opinnäytetyö
KESKI-POHJANMAAN AMMATTIKORKEAKOULU
Tietojenkäsittelyn koulutusohjelma
Joulukuu 2005

OPINNÄYTETYÖN SUOMENKIELINEN TIIVISTELMÄ

KESKIPOHJANMAAN AMMATTIKORKEAKOULU

Koulutusohjelma: Tietojenkäsittely

Tekijä: Mikko Tillikainen
Opinnäytetyön nimi: Kuvanlähetyssovellus Symbian-käyttöjärjestelmälle
Työn ohjaaja: Petri Saviranta
Työn tarkastaja: Grzegorz Szewczyk
Opintojen aloitusvuosi: 2002
Valmistumisvuosi: 2005 **Sivumäärä:** 35

TIIVISTELMÄ

Työn tarkoituksena on perehtyä Symbian-ohjelmistokehitykseen yleisellä tasolla ja rakentaa alustalle matkapuhelimen kameraa ja yhteysmahdollisuuksia hyödyntävä kuvanlähetyssovellus.

Sovelluksella käyttäjä voi ottaa kuvia haluamistaan kohteista ja lähettää ne Internet-yhteyden välityksellä tietojärjestelmään.

Avainsanat: Symbian, matkapuhelimet, C++ -ohjelmointi, langaton viestintä

OPINNÄYTETYÖN ENGLANNINKIELINEN TIIVISTELMÄ

CENTRAL OSTROBOTHNIA POLYTECHNIC

Degree programme: Information Technology

Author: Mikko Tillikainen
Name of thesis: Picture Sending Application for Symbian OS
Supervisor: Petri Saviranta
Inspector: Grzegorz Szewczyk
Starting year of studies: 2002
Year of graduation: 2005 **Number of pages:** 35

ABSTRACT

The aim of this thesis is to get familiar with Symbian software development in general level and to produce picture sending application utilizing mobile phone's built-in camera and GPRS connection capabilities.

With application, the user can take pictures and send them to system over Internet-connection.

Key words: Symbian, mobile phones, C++ -programming, wireless communication

LYHENTEET

API	Application Programming Interface
ARM	Acorn RISC Machine
GPRS	General Packet Radio Service
GUI	Graphical User Interface
HTTP	HyperText Transfer Protocol
OS	Operating System
RAD	Rapid Application Development
RFC	Request for Comments
RISC	Reduced Instruction Set Computer
SDK	Software Development Kit
UIQ	User Interface Quartz

SISÄLLYSLUETTELO

1 JOHDANTO

1.1 Yleiskuvaus

Tämän opinnäytetyön tarkoituksena on valottaa Symbian-ohjelmointia yksinkertaisen esimerkkiohjelman kautta. Vaikka ohjelma ei olekaan mitenkään erityisen laaja, olen sitä tehdessäni joutunut perehtymään kattavasti Symbian-ohjelmistokehityksen ominaispiirteisiin ja erikoisuuksiin.

Asiat tullaan esittämään varsin pintapuolisesti ja hyvin tiivistetyssä muodossa, sillä syvempään paneutumiseen eivät yksinkertaisesti kirjasen sivut riitä, eikä se liene tarkoituksenmukaistakaan opinnäytetyön luonteen huomioon ottaen. Jos lukija todella haluaa oppia aiheesta, hän voi tutustua joihinkin markkinoilla olevista useista aiheeseen keskittyvistä kirjoista.

Teksti on jaettu pääpiirteissään kahteen osaan. Alkuosa kuvaa Symbian-käyttöjärjestelmän yleisiä ohjelmistokehityksen piirteitä, ja loppuosa keskittyy enemmän tämän kyseessä olevan esimerkkisovelluksen toteutukseen ja ominaisuuksiin.

1.2 Kuvanlähetyssovellus

Valitulla esimerkkisovelluksella käyttäjä pystyy ottamaan kuvia kiinnostavista kohteista ja lähettämään kuvat suoraan tietojärjestelmään Internet-yhteyden kautta. Sovellus hyödyntää kuvauksessa matkapuhelimen sisäänrakennettua kameraa, mikä useimmissa uudehkoissa malleissa on. Ohjelmaa voi käyttää esim. tapauksissa, joissa henkilö haluaa välittää kokemuksiaan Internetin kautta muille käyttäjille. Kuvat siirtyvät henkilön kotisivuille, joista ne ovat muiden Internetin selaajien nähtävillä. Tällaista toimintaa kutsutaan joskus englanninkielisen termin mukaan ns. ”bloggaamiseksi”. Myös muita sovelluskohteita on myös mahdollista keksiä. Kiinteistönvälittäjä voisi ottaa välittämistään asunnoista kuvia ja siirtää ne saman tien yrityksen Internet-sivuille asiakkaiden katseltaviksi. Toisessa esimerkkiske-

naariossa pysäköinninvalvoja voisi ottaa kuvan väärin pysäköidystä autosta ja siirtää sen laitoksen tietojärjestelmään. Näin kiistatapauksissa olisi mahdollista esittää todiste tapahtuneesta rikkeestä.

Ohjelman projektinimi on Picsender, jota tullaan jatkossa käyttämään tekstissä käsiteltäessä tätä ohjelmaa.

2 SYMBIAN-KÄYTTÖJÄRJESTELMÄ

2.1 Yleistä

Symbian on samannimisen yhtiön kehittämä käyttöjärjestelmä, joka on suunnattu kehittyneiden matkapuhelimien eli ns. älypuhelimien käyttöjärjestelmäksi. Yrityksen omistavat yhdessä suuret matkapuhelinliiketoiminnassa mukana olevat yhtiöt. Omistajayhtiöihin kuuluvat Nokia, Ericsson, Panasonic, Samsung, Siemens ja Sony Ericsson. Teknologiaa ovat myös monet muut yritykset lisensoineet käytettäväksi omissa tuotteissaan. Symbian OS on selvästi suosituin älypuhelinten käyttöjärjestelmä, koska noin kaksi kolmasosaa kyseisistä laitteista sisältää Symbian OS:n. (Symbian 2005a.)

Miksi sitten kehittää kokonaan uusi käyttöjärjestelmä ainoastaan puhelimia varten eikä käyttää esimerkiksi jonkin jo olemassa olevan järjestelmän muokattua versiota? Symbian on esittänyt viisi seikkaa, joiden takia uuden luominen on ollut perusteltua (Kuvio 1).

- Matkapuhelimet ovat pieniä ja mobiileja.
- Matkapuhelimet on kohdennettu kaikille käyttäjäryhmille.
- Matkapuhelimet eivät ole koko aikaa yhdistettynä verkkoon.
- Valmistajien täytyy pystyä räätälöimään tuotteitaan.
- Järjestelmän pitää olla avoin ulkopuolisille ohjelmistokehittäjille.

KUVIO 1. Matkapuhelinmarkkinoiden viisi erityispiirrettä (Mery 2003)

Mukauttamalla tietokoneisiin suunniteltu käyttöjärjestelmä toimimaan puhelimessa tai liittämällä kommunikaatiovalmiudet jo valmiina olevaan kevyeen järjestelmään, aiheutuu liian monia perustavan laatuisia kompromisseja (Mery 2003).

Matkapuhelimien ja pöytäkoneiden välillä on joitakin tärkeitä eroavaisuuksia. Puhelinten resurssit ovat erittäin rajatut: prosessorit ovat hitaita, ja muistia on yleensä paljon vähemmän. Muistia ei voi myöskään jatkaa kovalevylle tarpeen vaatiessa, koska kovalevyä ei ole. Näin ohjelma ei voi yksinkertaisesti luottaa, että sen käsittelemille tiedoille olisi rajatto-

masti tilaa. Myös virrankulutuksen huomiointi on tärkeää. Uusien laitteiden odotetaan toimivan pitkiä aikoja ilman akun uudelleenlatausta, ja usein järjestelmässä on myös monia muita virrankulutusta aiheuttavia tekijöitä ohjelman lisäksi. Ohjelmat täytyy pitää tiiviinä ja kaikki mahdolliset virhetilanteet kuuluu käsitellä, koska matkapuhelimia harvoin käynnistellään uudelleen. (Harrison 2003.)

Puhelimilta odotetaan huomattavasti suurempaa toimintavarmuutta ja virheidensietokykyä kuin tavallisilta tietokoneilta. Tämän vaatimuksen täytyy heijastua myös itse käyttöjärjestelmän perusrakenteisiin asti eikä vain sen päällä toimiviin ohjelmistoihin, jotta sillä olisi käytännön vaikutusta. Symbian OS on suunniteltu alusta lähtien auttamaan ohjelmistokehittäjää edellä esiteltyjen ongelmien kohtaamisessa, mutta se vaatii tiettyjen kurinalaisten sääntöjen opettelemista ja noudattamista (Harrison 2003). Luultavasti juuri nämä säännöt ovat tuoneet Symbian-ohjelmistokehitykselle ikävän maineen ja nostaneet monien kohdalla kynnystä ryhtyä tekemään ohjelmistoja Symbian-alustalle.

2.2 Historia ja versiot

Symbian-käyttöjärjestelmän perusta on Psionin kehittämässä EPOC-käyttöjärjestelmässä. EPOC oli modulaarinen 32-bittinen moniajettava, erityisesti mobiililaitteille suunniteltu järjestelmä, joka kehitettiin 1990-luvun puolivälissä. Se oli vahvasti oliopohjainen ja toteutettu enimmäkseen C++ -ohjelmointikielellä. Järjestelmä oli niin hyvä, että se oli sopiva valinta matkapuhelinvalmistajille tulevaisuuden kommunikaatiolaitteita varten. Kun Symbian perustettiin, EPOC-käyttöjärjestelmää kehitettiin eteenpäin sopimaan laajalle valikoimalle matkapuhelimia. EPOC release 5 oli viimeinen versio vanhalla nimellä, minkä jälkeen se muutettiin muotoon Symbian OS. (Digia Inc 2003, 6–7.)

Symbian-käyttöjärjestelmästä on tullut monia versioita vuosien mittaan. Ensimmäisen uudella nimellä julkistetun version numero oli 6. Nykyään mennään jo versiossa 9.1. (Symbian 2005b.)

On huomattu, että yleensä suurimmat erot laitteiden välillä johtuvat käyttöliittymän toteutuksesta (Digia Inc 2005, 10). Tämän vuoksi järjestelmästä on tehty erilaisia käyttöliitty-

mäversioita erityyppisiin laitteisiin soveltuviksi. Näihin käyttöliittymätoteutuksiin kuuluvat Nokian Series 60 ja Series 80, NTT DoCoMo:n FOMA™ ja UIQ Technologyn kehittämä UIQ (Symbian Fast Facts 2005). Series 60 on tarkoitettu yhdellä kädellä käytettäviin älypuhelimiin, joissa on yleensä melko pieni näyttö. Series 80 on suunnattu näppäimistöllä ja suurella näytöllä varustettuihin malleihin. Pieniin, jo tietokoneen toiminnallisuutta lähenteleviin, kynällä ohjattaviin laitteisiin on UIQ-käyttöliittymätoteutus.

2.3 Työkalut

Ohjelmistojen kehitys Symbian-ympäristöön tapahtuu C++ -kielellä vahvasti olioita hyödyntäen. Markkinoilla on tarjolla useita eri valmistajien ohjelmistotyökaluja, joilla ohjelmien luominen on mahdollista. Valitettavasti useimmat kehitystyökalut ovat maksullisia, mutta esim. Borlandin C++BuilderX Mobile -ohjelmisto on saatavissa ilmaiseksi Internetistä. Muita vaihtoehtoja ovat Microsoftin Visual C++ 6 ja Visual C++ .NET -ohjelmistot ja MetroWerksin CodeWarrior. CodeWarrior-ohjelmasta on ladattavissa valmistajan kotisivuilta ilmainen rajoitettu kokeiluversio.

Borlandin ohjelmistolla on tietyssä määrin mahdollista suunnitella graafisesti ohjelmia, mikä muissa kokeilemissani työkaluissa ei ollut mahdollista. Tämä ei välttämättä aina ole hyvä asia, sillä C++BuilderX joutuu generoimaan ylimääräistä koodia vastaamaan hiiren avulla aseteltuja komponentteja. Automaattisesti luotua koodia ei aina ole kovin helppo ymmärtää, varsinkaan kun se ei ole kovin hyvin kommentoitua eikä ohjelman ohjeistus ole tässä suhteessa mitenkään hyvä. Tämän oman projektini aikana huomasin, että vaikka graafinen suunnittelu voi aluksi tuottaa nopeasti tuloksia, se voi kääntyä myöhemmin haitaksi, kun osa koodista jää epäselväksi ja siten vaikeaksi kehittää eteenpäin. Tilanne tulee varmasti korjaantumaan tulevaisuudessa, kun työkalut kehittyvät paremmiksi. Nokian lanseeraama uusi kehitysympäristö Carbide.c++ (Nokia 2005) on varmaan askel oikeaan suuntaan.

Tarvitaan vielä jokin Symbianin SDK, jotta ohjelmien kääntäminen olisi mahdollista. SDK sisältää tarvittavat otsikko- ja kirjastotiedostot, kääntäjän, emulaattorin ja kattavat ohjeet. Symbian-käyttöjärjestelmän sisältävissä matkapuhelimissa on ARM-prosessori, joka tar-

vitsee oman kääntäjänsä. Paketissa mukana tulevat ohjeet ovat erittäin kattavat sisältäen mm. selitykset yleisistä ohjelmointimenetelmistä sekä kuvaukset kaikista luokista ja niiden sisältämistä metodeista. Emulaattorissa voi testata tekemiään ohjelmia, jos ei omista tarvittavaa puhelinta. SDK:t ovat ilmaiseksi ladattavissa Internetistä Symbianin kotisivuilta (Symbian 2005c).

3 OHJELMOINTI SYMBIAN OS:SSA

3.1 Nimeämiskäytännöt

Kuten mikä tahansa järjestelmä, myös Symbian OS käyttää nimeämiskäytäntöjä osoittamaan huomionarvoisia seikkoja ja helpottamaan lähdekoodin ymmärtämistä (Harrison 2004, 19). Järjestelmän sisältämät C++ -luokat jaotellaan eri kategorioihin niiden nimen alkukirjaimen mukaan (Taulukko 1). Nämä käytännöt on hyvä muistaa myös omia luokkia määriteltäessä.

TAULUKKO 1. Luokkien nimeämiskäytäntö (Harrison 2004, 19–20)

Kategoria	**Esimerkki**
T-luokat	TDesC, TPoint, TFileName
C-luokat	CActive, CBase
R-luokat	RFile, RTimer, RWindow
M-luokat	MEikMenuObserver
Staattiset luokat	User, Math, Mem
Struktuurit	SEikControlInfo

Myös joitakin muita etumerkkejä on käytössä, mutta ne ovat harvinaisempia. Ero T-, C- ja R- tyyppisten luokkien välillä on erittäin tärkeä resurssien varauksen ja vapauttamisen suhteen. T-luokat eivät tarvitse ”siivousta”, koska ne eivät varaa resursseja. R-luokat varaavat resursseja, jotka täytyy käytön jälkeen sulkea tai vapauttaa. C-luokat täytyy tuhota asianmukaisesti, kun niitä ei enää tarvita. (Harrison 2004, 21.)

M-luokat ovat rajapintoja, joita ei voi suoraan luoda, vaan muiden luokkien täytyy periytyä niistä ja toteuttaa rajapintojen sisältämät virtuaalifunktiot. Ainoa keino hyödyntää moniperiytymistä Symbian OS:ssa on käyttää näitä luokkia. (Harrison 2004, 20.)

Staattiset luokat sisältävät ainoastaan staattisia funktioita, jonka vuoksi niistä ei pysty suoraan luomaan ilmentymiä. Nämä luokat ovat yleensä hyödyllisiä kirjastorutiinien tarjoajia.

Struktuurit ovat C-kielestä tuttuja tietotyyppejä, jotka eivät sisällä jäsenfunktioita (Harrison 2004, 20.)

C-kirjain luokan nimen perässä tulee sanasta const, mikä tarkoittaa, ettei tällaisen luokan tietoja pysty muuttamaan.

Myös muuttujien nimeämisessä on käytössä tiettyjä tapoja (Taulukko 2). Näidenkin tapojen noudattamisesta on hyötyä, kun koodista tulee vähän selkeämpää ja helpommin ymmärrettävää.

TAULUKKO 2. Muuttujien nimeämiskäytänteet (Harrison 2004, 21)

Muuttujan tyyppi	**Esimerkki**
Lueteltu vakio	ESolidBrush
Vakio	KMaxFileName
Funktion argumentti	aPoint
Jäsenmuuttuja	iCamera

Metodien nimeämiseen on eräs erittäin tärkeä sääntö olemassa: jos metodi voi aiheuttaa poikkeuksen, on sen nimen perässä oltava iso L-kirjain, esimerkiksi TBufC-luokan metodi AllocL. Jos jäsenfunktio jättää palauttamansa olion siivouspinoon, nimen perässä on kirjainyhdistelmä LC. Siivouspinosta kerrotaan myöhemmissä luvuissa enemmän.

3.2 Perustietotyypit

Symbian OS:ssa käytetään standardista C++ -kielestä poikkeavia tietotyyppejä (Taulukko 3). Tyypit ovat pääasiassa määritelty otsikkotiedostossa e32defs.h, ja niillä kaikilla on T-tyyppisen luokan rajapinta, eli ne eivät tarvitse monimutkaista luontiprosessia tai erikseen määriteltyjä tuhoajia. Oikeastaan jotkut tyypeistä on muodostettu kääntäjäkohtaisilla typedef-lauseilla standardeista C++ -tyypeistä ja jotkut oikeilla yksinkertaisilla luokilla. Symbian OS-tyyppejä tulisi käyttää aina normaalien C++ -tyyppien sijasta, näin turvataan

käännettyjen ohjelmien arkkitehtuuririippumattomuus. Esimerkiksi TInt-tyyppiä tulee käyttää int-tyypin sijaan. (Edwards & Barker 2004, 82.)

TAULUKKO 3. Perustietotyypit (Edwards & Barker 2004, 83)

Tyyppi	Erikoistapaus	Tyyppi
TInt	TInt8, TInt16, TInt32, TInt64	Kokonaisluku
TUInt	TUInt8, TUInt16, TUInt32	Etumerkitön kokonaisluku
TReal	TReal32, TReal64, TRealX	Liukuluku
TText	TText8, TText16	Merkki
TChar		Laajennettu Unicode-merkki
TBool		Boolean
TAny		Mikä tahansa (void)

Joissakin tapauksissa voi tietotyypin koon määrittää tarkasti lisäämällä tyypin nimen perään varattujen bittien lukumäärää ilmaisevan arvon. TReal-tyypin perässä oleva X-kirjan merkitsee laajennettua liukulukutyyppiä, jonka suuruus on 12 bittiä.

3.3 Deskriptorit

Deskriptorit ovat lähes joka paikassa käytettäviä tärkeitä olioita satunnaisen datan säilyttämiseen. Useimmiten niitä käytetään tekstin tallentamiseen, mutta ne voivat sisältää myös binääristä tietoa ja jopa muita serialisoituja olioita. Tämä on mahdollista, koska ne eivät käytä nolla-lopetusmerkkiä datan koon määrittämisessä, kuten normaalit C-kielen merkkijonot. Deskriptorit on suunniteltu mahdollisimman tehokkaiksi, ja siksi ne välttävät käyttämästä virtuaalifunktioita muistin käsittelyyn. (Edwards & Barker 2004, 97.)

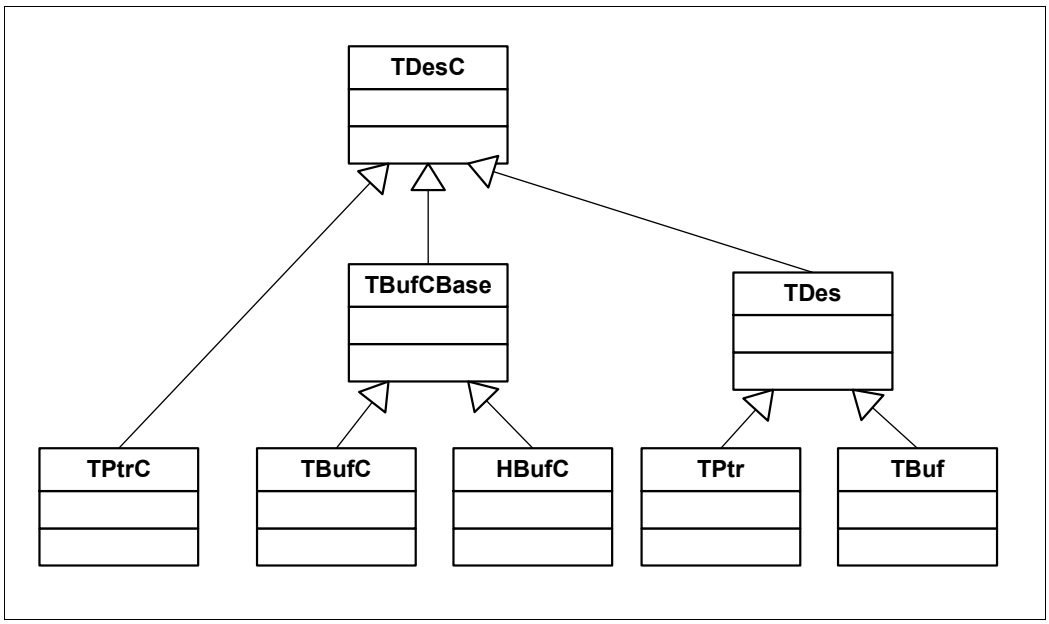
Deskriptorit tarjoavat ohjelmoijalle

- suorituksen aikaisen muistin rajojen tarkastuksen
- kattavan ohjelmointirajapinnan datan ja tekstin käsittelyyn
- tiukan integroinnin Symbian OS:n resurssienhallinta-arkkitehtuuriin.

Uudet ohjelmoijat haluavat joskus välttää käyttämästä deskriptoreita ja yrittävät sen sijaan luoda omia versioita näistä tietotyypeistä. Tämä johtaa usein vaarallisen koodin muodostumiseen, ja koska monet Symbian OS:n sisäiset rajapinnat käyttävät deskriptoreita, niitä ei voi jonkin tietyn pisteen jälkeen enää välttää kohtaamasta. (Edwards & Barker 2004, 97.)

Deskriptorit ovat melko monimutkaisia tietorakenteita käyttää, enkä voi vielä tunnustaa ymmärtäväni täysin kaikkia niiden hienouksia.

Deskriptori-luokkiin lasketaan yleensä kuuluvaksi kahdeksan eri luokkaa (Kuvio 2).



KUVIO 2. Deskriptori-luokat (Harrison 2004, 29)

TDesC on abstrakti luokka, mistä kaikki muut deskriptori-luokat ovat periytyneet. TBufC-Base- ja TDes-luokat ovat harvemmin käytössä olevia kantaluokkia. Loppuja luokkia käytetään yleisesti.

Aiemmin esiteltyjen nimeämiskäytäntöjen avulla voidaan nähdä, että osa luokista on muuttamisen sallivia ja osa ei-sallivia. Ero voidaan nähdä luokan nimen perässä olevasta C-kirjaimesta. Muuttamisen salliminen tarkoittaa tässä tapauksessa, että luokka sisältää metodeja, joilla pystyy muuttamaan olion sisältämää tietoa, esim. lisäämään uusia merkkejä tekstin sekaan.

3.4 Literaalit

Literaalit ovat yleinen tapa esittää tekstejä lähdekoodissa Symbian OS:ssa. Jos ohjelman markkina-alue kuitenkin kattaa useamman kuin yhden kielialueen, on järkevämpää käyttää jotain muuta menetelmää tekstien määrittämisessä.

```
_LIT(KTxtHelloWorld, ”Hello World!”);
```

KUVIO 3. Esimerkki literaalin määrityksestä

Literaali määritellään käyttämällä _LIT-makroa, joka tuottaa merkkijonoon viittaavan TLitC-tyyppisen olion (Kuvio 3). Tämä tyyppi käyttäytyy samalla tavalla kuin TDesC-tyyppi. (Harrison 2004, 32.) Sopivien tyyppimuunnosten kera literaali voidaan välittää merkkijonofunktioille, joiden parametreiksi käyvät deskriptori-tyyppiset muuttujat.

3.5 Poikkeukset ja siivouspino

Poikkeus on virhetilanne, joka ei välttämättä johdu ohjelman koodissa olevasta ohjelmointivirheestä, vaan järjestelmän poikkeuksellisesti muuttuneesta tilasta. Yleisiä poikkeusten aiheuttajia ovat mm. muistin loppuminen tai olemattoman tiedoston avausyritys.

C++ -kielen sisäiseen try-catch-poikkeuskäsittelyyn tutustuneet voivat ihmetellä, miksi Symbian OS ei hyödynnä tätä, jo valmiina kielessä olevaa laajalti ymmärrettyä ominaisuutta. Syitä on pääasiassa kaksi: Symbian OS:n oma poikkeuskäsittelymekanismi on kevyempi ja siten paremmin soveltuva pieniin laitteisiin, eikä C++ -kielen oma menetelmä ollut

vielä tuettuna ARM-kääntäjässä, kun Symbian OS:n ensimmäiset versiot tulivat. (Edwards & Barker 2004, 84.)

Symbian OS:ssa on käytössä ansahaarniska (trap harness), joka ottaa kiinni ohjelman heittämät poikkeukset ja käsittelee ne mahdollisuuksiensa mukaan. Ohjelma voi määrittää haarniskan käyttämällä TRAP-makroa. Kaikissa ohjelmissa ei tarvitse välttämättä käyttää TRAP-komentoa, koska järjestelmä luo oman oletuspoikkeuskäsittelijän jokaiselle graafisen käyttöliittymän sisältävälle ohjelmalle. Tämä käsittelijä tosin ei tee muuta kuin lopettaa ohjelman ja tulostaa virheilmoituksen näytölle.

Kun poikkeus tapahtuu, on aina vaarana, että muistia joutuu hukkaan. Jos funktiossa varatun muistin osoitteet on sijoitettu paikallisiin muuttujiin, siirryttäessä poikkeuskäsittelijään nämä muuttujat eivät enää ole käytettävissä. Kun osoitteita ei enää tiedetä, ei muistiakaan voi vapauttaa, ja näin ohjelman varaamat muistialueet jäävät ”orvoiksi” kuluttamaan järjestelmän vapaata muistia. Tämän ongelman ratkaisemiseksi Symbian OS:ssa on käytössä siivouspino, joka on määritelty luokassa CleanupStack. Graafisen sovelluksen käynnistyessä järjestelmä luo automaattisesti myös siivouspinon ohjelman käyttöön.

CleanupStack-luokassa on PushL-metodi olion siirtämiseksi siivouspinoon ja Pop-metodin erityyppisiä versioita, joilla voi poistaa olion pinosta (Symbian Developer Library 2002). Poikkeuksen tapahtuessa kaikki pinossa olevat oliot tuhotaan automaattisesti, eikä muistia pääse näin häviämään. Pinoa ei kannata käyttää tarpeettomasti, eikä esim. luokan jäsenmuuttujia saa laittaa siivouspinoon ollenkaan, koska luokan tuhoajan tehtävänä on huolehtia jäsenmuuttujien varaaman muistin vapauttamisesta (Edwards & Barker 2004, 90). Jos jäsenmuuttujan vahingossa laittaa pinoon, voi seurauksena olla kaksinkertainen tuhoaminen, kun sekä siivouspino että luokan tuhoaja vapauttavat muuttujalle varatun saman muistialueen.

Muistin hukkaamisen estämiseksi on Symbian OS:ssa lisäksi käytössä olioiden kaksivaiheinen rakennus. Varsinaiseen rakentajaan sijoitetaan sellainen koodi, mikä ei voi aiheuttaa poikkeuksia, ja loput laitetaan omaan jäsenfunktioonsa, jonka nimi on yleensä ConstructL. Myöskään luokan tuhoaja ei saa aiheuttaa poikkeuksia, eikä tuhoaja saa olettaa olion luonnin onnistuneen täydellisesti (Edwards & Barker 2004, 92). Ohjelmoijan työn helpottami-

seksi järjestelmän luokissa on yleensä määritelty metodi NewL, joka huolehtii olion luonnista ja palauttaa sen kutsujalle.

C++ -kielessä muistia voidaan varata new-operaattoria käyttämällä. ”New (ELeave)” on new-operaattorin kuormitettu versio, joka aiheuttaa poikkeuksen, jos muisti ei riitä olion sisältämien tietojen varaukseen (Symbian Developer Library 2002). Tätä muotoa tulisi käyttää useimmissa tapauksissa pelkän normaalin new-operaattorin kutsun sijaan.

Kuviossa 4 on esitelty siivouspinon käyttöä ja kaksivaiheista rakentamista käytännössä lähdekoodin avulla. Ensimmäisenä toimenaan funktio luo omasta luokastaan uuden ilmentymän. Juuri luotu olio asetetaan siivouspinoon ja sen ConstructL-metodia kutsutaan tekemään alustustoimenpiteet, jotka voivat mahdollisesti aiheuttaa poikkeuksen. Jos kaikki sujuu hyvin, olio otetaan pois pinosta ja palautetaan metodin kutsujalle.

```
static CHTTPSender* NewL()
{
    CHTTPSender* self=new (ELeave) CHTTPSender();
    CleanupStack::PushL(self);
    self->ConstructL();
    CleanupStack::Pop();
    return self;

}
```

KUVIO 4. Luokan CHTTPSender NewL-metodi

3.6 Aktiiviset oliot

Monissa järjestelmissä yleisin tapa hyödyntää moniajoa on käyttää säikeitä, mutta Symbian OS:ssa suositeltu tapa on käyttää aktiivisia olioita (Harrison 2004, 41). Aktiiviset oliot mahdollistavat säikeitä kevyemmän toteutuksen, joka sopii hyvin heikkotehoisille laitteille.

Yksi säie palvelee aktiivisen vuorontajan (active scheduler) avulla yhtä tai useampaa rinnakkaista aktiivista oliota. Symbian-sovellusarkkitehtuuri sisältää automaattisesti aktiivi-

sen vuorontajan, jota myös sovellusta kehittävä ohjelmoija voi hyödyntää. (Mikkonen 2004, 175.)

Aktiiviset oliot mahdollistavat sovellukselle sen, että se voi kutsua käyttöjärjestelmän palveluja asynkronisesti. Sovellus lähettää pyynnön palvelimelle, ja ohjelma palaa heti jatkamaan muiden tehtävien suoritusta. Palvelin käsittelee ohjelman lähettämän pyynnön, ja käsittelyn valmistuttua, se kutsuu ohjelman CActive-luokasta periytetyn aktiiviolion RunL-metodia. Eli asynkronisuus tarkoittaa käytännössä, että ohjelma ei jää odottamaan palvelimen vastausta, vaan jatkaa muiden toimien suorittamista. Lisäksi on mahdollista määritellä uudelleen metodi RunError, jota kutsutaan poikkeuksen sattuessa RunL:n suorituksen aikana. (Mikkonen 2004, 176.)

3.7 Client-server-arkkitehtuuri

Asiakas (client) on ohjelma, joka käyttää tiettyä palvelimen (server) tarjoamaa palvelua. Palvelin on myös ohjelma, joka vastaa asiakkaiden pyyntöihin jollain tavalla. (Edwards & Barker 2004, 165.)

Client-server-arkkitehtuurin etuihin kuuluu (Edwards & Barker 2004, 165)

- Laajennettavuus (uusia moduuleja voidaan lisätä tarjoamaan palveluja jopa suorituksen aikana)
- Tehokkuus (yksi palvelin voi palvella useita asiakkaita samaan aikaan)
- Turvallisuus (väärin käyttäytyvä asiakas ei voi tuoda alas palvelinta)
- Asynkronisuus (aktiiviset oliot mahdollistavat palvelimen käytön asynkronisesti)

Useat ohjelmien käytössä olevat palvelut on toteutettu client-server-mallin pohjalta. Ohjelman hyödynnettävissä oleviin palvelimiin kuuluvat mm. tiedosto-, ikkuna-, kommunikaatio-, fontti- ja bittikarttapalvelimet.

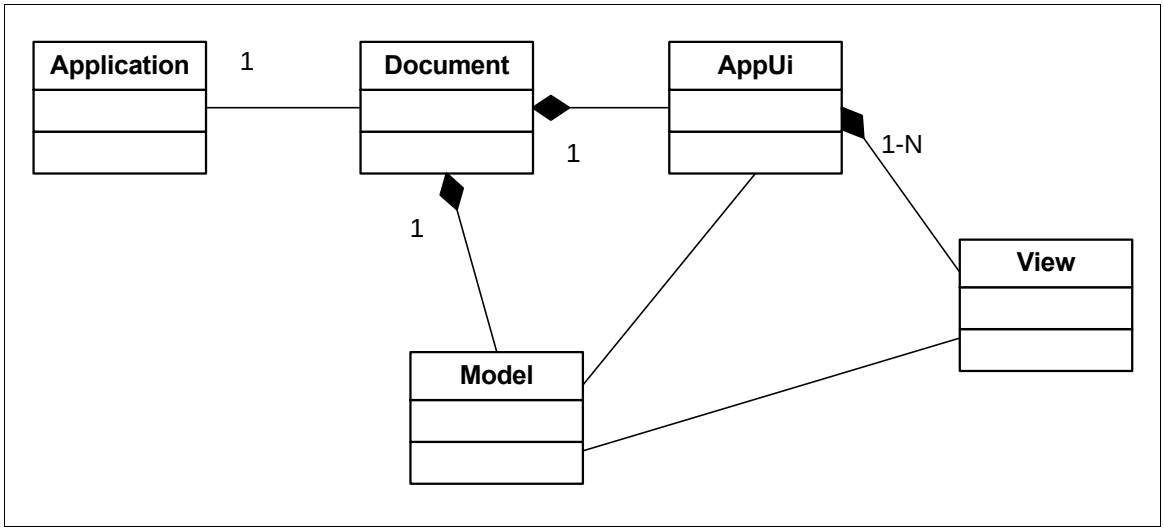
3.8 Sovelluksen perusrakenne

MVC-mallin (model-view-controller) avulla voidaan erottaa käyttöliittymä ohjelman muusta toiminnallisuudesta. MVC-malli tarkoittaa sovelluksen jakamista osiin siten, että voidaan erottaa kolme komponenttia: malli (model), view (näkymä) ja ohjain (controller). Näistä malli pitää sisällään kaiken sovellukseen liittyvän tiedon, näkymä on vastuussa sen esittämisestä eri tilanteissa ja ohjain tarjoaa käyttäjälle mahdollisuuden muokata tietoa. (Mikkonen 2004, 169.)

MVC-mallin mukaiseen sovellusarkkitehtuuriin ohjaa Symbian-ympäristössä sovelluskehys, joka vaatii ohjelmoijaa kirjoittamaan koodinsa tiettyihin luokkien metodeihin. Tästä syystä sovellukset muistuttavat toisiaan rakenteeltaan, mikä helpottaa uuden koodin kirjoittamista ja valmiin lähdekoodin ymmärtämistä. (Mikkonen 2004, 169.)

Yleensä Symbian-käyttöjärjestelmässä toimivat graafiset ohjelmat hiukan soveltavat MVC-mallia omissa toteutuksissaan (Kuvio 5). MVC-malliin verrattuna Symbian-ohjelmistoarkkitehtuuriin on lisätty komponentit Document ja Application. AppUi vastaa toiminnallisuudeltaan MVC-mallin ohjainta.

Application on yksinkertainen luokka, joka periytytetään Series 60-ohjelmissa luokasta CAknApplication. Application-luokan vastuulla on lähinnä dokumentin luominen ja erilaisten apumetodien tarjoaminen.



KUVIO 5. Sovelluksen peruskomponentit (Edwards & Barker 2004, 179)

Dokumenttia käytetään usein luomaan ja yhdistämään malli ja siihen liittyvä ohjain (AppUi). Dokumentti-luokka periytytetään luokasta CAknDocument. Dokumenttia edustavaa luokkaa voi jossakin mielessä ajatella myös mallina, sillä sen tehtävät ovat samansuuntaisia mallin kanssa (Mikkonen 2004, 171).

Ohjaimen tehtävänä on ottaa vastaan käyttäjän antamat komennot. Komentojen käsittelyssä voidaan hyödyntää mallin sisältämiä tietoja. Lisäksi ohjain yleensä luo sovelluksen näkymän. (Mikkonen 2004, 173.) Ohjainluokan kantaluokkana toimii Series 60 -ohjelmissa CAknAppUi-luokka.

Näkymä voidaan toteuttaa useammalla eri tavalla Series 60 -ympäristössä. Näkymäluokka voidaan esim. periyttää CAknDialog- tai CCoeControl-luokista. Mallille ei ole määritelty mitään tiettyä luokkaa Symbian-järjestelmässä.

Käytännössä sovellus rakennetaan ylikirjoittamalla näiden edellä esiteltyjen luokkien metodeja, joita sovelluskehys kutsuu tarvittaessa. Esim. ohjelmoija voi määritellä uudelleen AppUi-luokan HandleKeyEventL-metodin, jos hän haluaa käsitellä käyttäjän tuottamat näppäimistönpainallukset.

3.9 Joitakin yleisiä käyttöliittymäkomponentteja

Keskityn kuvaamaan vain Series 60 -käyttöliittymätoteutuksen komponentteja, koska Picsender-ohjelma on suunniteltu toimimaan ainoastaan Series 60 -laitteissa.

Series 60 on rakennettu UIKON (common UI framework) -kirjaston päälle, mikä on yhteinen kaikille käyttöliittymäversioille.

Kontrollien luonnin apuna käytetään usein tekstimuotoisia käsin kirjoitettuja tai ohjelmointityökalun tuottamia resurssitiedostoja. Tapa on tuttu jo entuudestaan Windows-ohjelmoinnista, jossa käyttöliittymä on määritelty samalla tavalla resurssitiedostoissa. Joissakin RAD-ohjelmointityökaluissa ne on vain piilotettu ja resurssitiedostojen käsittely tapahtuu implisiittisesti.

Muissa käyttöjärjestelmissä usein esiintyviä komentopainikkeita (button) ei ole Series 60:ssa käytettävissä, mikä johtuu varsinaisen osoitinjärjestelmän puutteesta.

3.9.1 Valikot

Valikko (menu) on ponnahdusikkuna, joka esittää listan mahdollisista valinnoista käyttäjälle. Valikot ovat usein tärkeä komponentti matkapuhelimissa toimivien sovellusten käyttöliittymissä. Menuja voi liittää ohjelman pääikkunaan, näkymiin, dialogeihin ja lisäksi joihinkin muihin kontrolleihin. (Edwards & Barker 2004.)

```
RESOURCE MENU_BAR r_picsender_menubar
  {
  titles =
    {
    MENU_TITLE
      {
      menu_pane = r_picsender_menupane;
      txt = "";
      }
    };
  }

RESOURCE MENU_PANE r_picsender_menupane
  {
  items =
    {
    MENU_ITEM
      {
      command = EOtaKuvaCmd;
      txt = "Ota kuva";
      },
      MENU_ITEM
      {
      command = ELahetaKuvaCmd;
      txt = "Lähetä kuva";
      }
    };
  }
```

KUVIO 6. Esimerkki menun määrittelystä resurssitiedostossa

Valikot määritellään resurssitiedoston avulla. Kuviossa 5 on Picsender-ohjelman valikon määritelmä. Vaikka menussa on vain kaksi valintaa, joutuu silti aika monta riviä kirjoittamaan. Määritelmä on jaettu kahteen osaan: MENU_BAR ja MENU_PANE -osioihin. MENU_BAR-osa on varsinainen valikon määritys, ja MENU_PANE sisältää valikon alkiot.

3.9.2 Dialogit

Yliluokka lähes kaikille dialogeille Series 60 -ympäristössä on CAknDialog (Edwards & Barker 2004, 291). Periyttämällä luokan CAknDialog-luokasta voi luoda yksinkertaisen perusikkunan. Tämä ikkuna ei sisällä oletuksena muita komponentteja ja on tyhjä. On myös olemassa muita erikoistuneempia luokkia eri käyttötarkoituksiin, kuten esim. CAknForm lomakkeiden esittämiseksi ja CAknTextQueryDialog tekstimuotoisen tiedon kysymiseksi käyttäjältä. Dialogi voi olla myös monisivuinen, jolloin nuolinäppäimillä pystyy vaihtamaan sivua.

Dialogit määritellään yleensä resurssitiedostojen avulla. Määrittelyn voi suorittaa myös dynaamisesti koodin avulla, mutta tämä on kannattavaa vain yksinkertaisten dialogien kohdalla.

Lomake (form)

Lomakkeet soveltuvat hyvin sellaisiin tapauksiin, joissa käyttäjältä halutaan kysyä samaan aikaan useata toisiinsa liittyvää tietoa. Lomake sisältää kenttiä, joihin käyttäjä pystyy syöttämään tiedot. Kentät voivat olla mm. tekstilaatikkoja, vierityspalkkeja tai valintaruutuja. Jos kaikki kentät eivät mahdu kerralla näytölle, on myös vieritys mahdollista.

Lomakkeita käytetään ohjelmassa periyttämällä oma lomakeluokka CAknForm-luokasta. Periytetyssä luokassa joudutaan useimmissa tapauksissa määrittelemään uudelleen muutama jäsenfunktio, joita ohjelmakehys kutsuu lomakkeen elinkaaren aikana. Esim. metodia OkToExitL kutsutaan, kun käyttäjä haluaa sulkea lomakkeen. Tähän metodiin kannattaa laittaa ohjelmalogiikka, joka vastaa käyttäjän syöttämien tietojen tallentamisesta. Pre-

LayoutDynInitL-jäsenfunktiossa voidaan alustaa kentät joihinkin tiettyihin arvoihin ennen lomakkeen näyttämistä.

Muistutus (note)

Series 60 käyttää muistutuksia esittämään huomionarvoisia tietoja käyttäjälle ohjelman suorituksen kulusta tai muista tapahtumista. Esim. erilaisista virhetilanteista ilmoitetaan yleensä muistutusten avulla. Normaalisti muistutus näkyy ruudulla muutaman sekunnin ajan, mutta käyttäjä voi halutessaan sulkea sen aikaisemminkin.

Useita erityyppisiä muistutuksia on valmiiksi määritelty ja ohjelmoija voi tarpeen vaatiessa tehdä myös oman tyyppisensä. Valmiita muistutuksia on mm. virheen, vahvistuksen, varoituksen, viiveen ja edistymisen ilmaisemiseen. Nämä eroavat toisistaan lähinnä vain siinä, millaisen kuvakkeen ne sisältävät ja millaisen äänimerkin ne soittavat käyttäjälle.

Kuviossa 6 on esitetty yksinkertaisin mahdollinen tapa esittää muistutus. Jotkut muut muistutustyypit ovat vaikeampia toteuttaa, kuten jos halutaan osoittaa tehtävän suorituksen eteneminen käyttäjälle. Tämä vaatii aktiivisen olion käyttöä ja sitä, että tehtävä voidaan jakaa osiin, jotta muistutusta pystytään päivittämään sen näyttämisen aikana.

Aluksi esitellään KTxtLahetysValmis-niminen literaali, mikä sisältää tekstin ”Kuvan lähetys valmis” (Kuvio 6). Seuraavaksi note-muuttujaan luodaan CAknInformationNote-tyyppinen olio ja kutsutaan olion ExecuteLD-metodia, joka näyttää muistutuksen. Muistutus tuhoaa itse itsensä käytön jälkeen, joten sitä ei tarvitse eksplisiittisesti tuhota.

```
_LIT(KTxtLahetysValmis, "Kuvan lähetys valmis");
CAknInformationNote* note=new (ELeave)CAknInformationNote();
note->ExecuteLD(KTxtLahetysValmis);
```

KUVIO 7. Esimerkki muistutuksen esittämisestä käyttäjälle

Kysely (query)

Series 60 -ohjelmat voivat hyödyntää erilaisia kyselyjä (query) tiedustelemaan käyttäjältä ohjelman haluamia asioita. Käytännössä kyselyt esittävät käyttäjälle ponnahdusikkunan, johon käyttäjä voi syöttää tarvittavat tiedot. Käyttäjä voi vastata kirjoittamalla arvon tekstilaatikkoon, valitsemalla yhden tai useamman kohdan listasta tai yksinkertaisesti vahvistamalla kysytyn toimenpiteen. Kyselyiden perustyypit ovat vahvistus-, tieto- ja listakyselyt.

Vahvistus (confirmation) -kyselyä voidaan käyttää vahvistamaan jonkin tietyn toimenpiteen suorittaminen, esim. voidaan varmistaa, saako tallennetun pelitilanteen tuhota. (Edwards & Barker 2004, 326.)

Tieto (data) -kyselyn avulla voidaan pyytää käyttäjää syöttämään tekstimuotoista tietoa, kuten käyttäjän nimi. Syötetty teksti voi olla tyypiltään tavallista tekstiä tai se voi olla salasana-, PIN-koodi-, puhelinnumero-, päivämäärä-, aika-, kesto- tai liukulukumuodossa. Esim. CAknTextQueryDialog-luokan avulla pystytään luomaan kysely pyytämään käyttäjältä tekstimuotoista tietoa.

Listakyselyt (list query) vastaavat toiminnallisuudeltaan seuraavassa osiossa esiteltäviä normaaleja listakomponentteja, mutta ne toimivat ponnahdusikkunan lailla ja ovat yksinkertaisempia tehdä ja määritellä. Listakyselyt luodaan CAknListQueryDialog-luokan avulla.

3.9.3 Listat

Listat ovat käyttöliittymäkomponentteja, jotka yksinkertaisesti näyttävät käyttäjälle listan ohjelman määräämistä eri vaihtoehdoista. Listan alkiot voivat sisältää tekstiä ja kuvia.

Pystysuorat listat (vertical lists)

Pystysuora listakomponentti sisältää yksiulotteisen listan alkioista, joista käyttäjä voi nuolinäppäimillä liikuttaen valita haluamansa. On myös mahdollista sallia useamman alkion valitseminen tai alkioiden merkitseminen. Merkitsemisellä mm. voidaan antaa käyttäjälle mahdollisuus suorittaa jokin tietty toimenpide useammalle listan alkiolle samalla kertaa. Listan luomiseksi on käytössä useita eri luokkia halutusta toiminnallisuudesta ja ulkonäöstä riippuen, kuten CAknSingleStyleListBox yksinkertaista listaa ja CAknSingleGraphicStyleListBox ikonilla varustettua listaa varten.

Kuviossa 8 on esitelty esimerkkikoodi listan luomista varten.

```
CAknColumnListBox* iSavedGamesListBox;
iSavedGamesListBox=new (ELeave) CAknSingleGraphic-
StyleListBox();
iSavedGamesListBox->SetContainerWindowL(*this);

TResourceReader reader;
iEikonEnv->CreateResourceReaderLC(reader,
R_SIMPLELIST_SAVED_GAMES_LISTBOX);
iSavedGamesListBox->ConstructFromResourceL(reader);
```

KUVIO 8. Esimerkkikoodi listan luomiseksi (Edwards & Barker 2004, 349)

Ruudukot (grid)

Ruudukkoja (grid) voidaan käyttää näyttämään käyttäjälle kaksiulotteinen, useimmiten graafinen valintataulukko. Käyttäjä voi liikkua ruudukossa nuolinäppäimiä käyttäen ja alkioita valiten. Esimerkiksi kalenterin kuukausinäkymä voitaisiin tehdä suhteellisen helposti ruudukon avulla. Toteuttaakseen ruudukon ohjelmoija joutuu periyttämään luokan CAknGrid-luokasta. Erityisesti kalenterin toteuttamista varten Symbian SDK:ssa on myös valmiina CAknCaleMonthlyStyleGrid-luokka.

Asetuslistat (settings lists)

Asetuslistat voivat koota yhteen ohjelman toimintaan liittyvät asetukset. Muista listoista poiketen asetuslistan alkioissa näkyy nimen lisäksi myös sen arvo. Ohjelmistokehittäjä voi hyödyntää asetuslistoja käyttämällä kantaluokkana CAknSettingItemList-luokkaa.

3.9.4 Editorit

Useimmissa ohjelmissa joudutaan jossain vaiheessa kysymään käyttäjältä tietoja ja tulostamaan niitä. Editorit ovat komponentteja, joilla tämä on mahdollista. Ne vastaavat tavallaan Windows-ympäristöstä tuttuja tekstilaatikkokomponentteja.

Editorit voivat sisältää useita rivejä tekstiä. Erilaisten visuaalisten tehokeinojen, kuten alleviivauksen, lihavoinnin tai kuvien, käyttö on myös mahdollista CRichTextEditor-luokan avulla. Editorit voidaan määrittää vain lukutyyppisiksi, ja niiden kursorit on mahdollista piilottaa, jolloin editoreja voi käyttää informaation esittämisessä.

Erimuotoisten tietojen syöttämiseen ja esittämiseen on tarjolla omat erikoistuneet luokkansa, esim. CAknIntegerEdwin kokonaislukujen ja CEikFloatingPointEditor liukulukujen syöttämiseen.

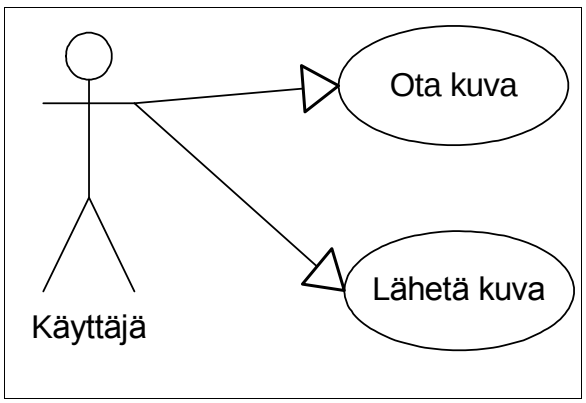
4 PICSENDER

4.1 Yleiskuvaus

Picsender-ohjelma on toiminnallisuudeltaan melko yksinkertainen. Ohjelmassa on oikeastaan vain kaksi mahdollista käyttötapausta: ”Ota kuva” ja ”Lähetä kuva” (Kuvio 9). Ohjelmaan olisi ollut helppo keksiä lisäominaisuuksia, mutta Symbian-alustaan tutustumiseen kulunut suuri aika pakotti pitämään itse sovelluksen yksinkertaisena.

”Ota kuva”-käyttötapauksen mukaan käyttäjä kohdistaa puhelimen kameran haluamaansa kohteeseen ja valitsee ohjelman valikosta kohdan ”Ota kuva”. Tällöin Picsender ottaa kuvan ja tallentaa sen muistiin jatkokäsittelyä varten.

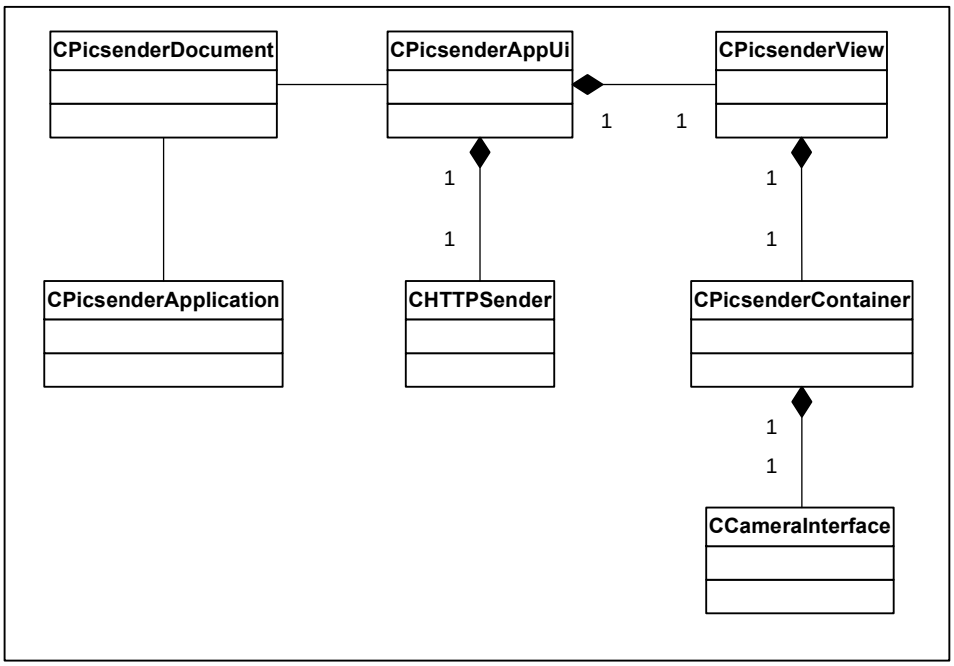
”Lähetä kuva”-käyttötapauksessa ohjelma lähettää käyttäjän ottaman kuvan GPRS-yhteyden kautta palvelimelle, samalla kysyen kuvan identifioivaa tunnistetta. Palvelin palauttaa toimenpiteen onnistumisesta vastauksen, joka näytetään käyttäjälle muistutuksen avulla.



KUVIO 9. Picsender-käyttötapauskaavio

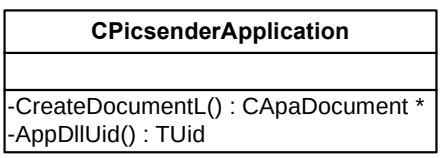
4.2 Luokat

Tässä osiossa esitellään lyhyesti ohjelman sisältämät luokat. CCameraInterface- ja CHTTPSender-luokat ovat alusta asti itse toteuttamiani, loput on muokattu C++BuilderX-ohjelmiston luomien tiedostojen pohjalta.



KUVIO 10. Picsender-ohjelman luokkakaavio

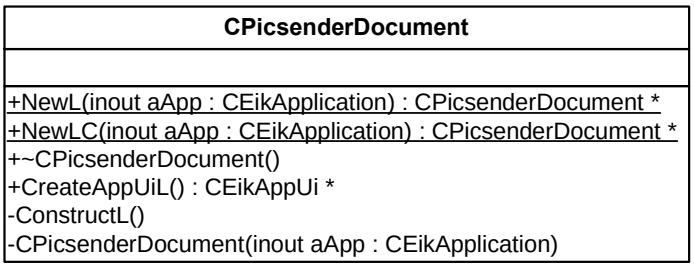
4.2.1 CPicsenderApplication



KUVIO 11. CPicsenderApplication-luokka

Symbian OS käyttää CPicsenderApplication-luokan CreateDocumentL-metodia luomaan sovelluksen dokumenttiolion. Luokka periytyy CAknApplication-luokasta.

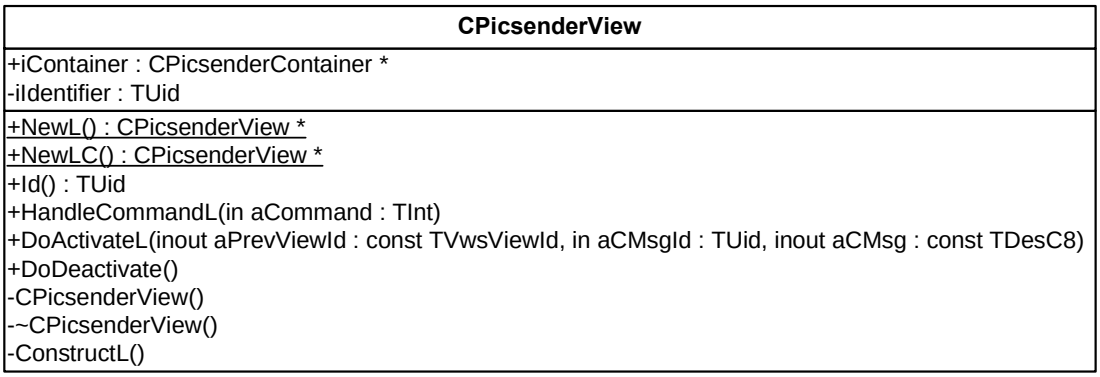
4.2.2 CPicsenderDocument



KUVIO 12. CPicsenderDocument-luokka

Dokumenttiluokkaa ei tarvita Picsender-ohjelmassa muuhun kuin edelleen ohjainluokan luomiseen. Luokka periytyy luokasta CAknDocument.

4.2.3 CPicsenderView



KUVIO 13. CPicsenderView-luokka

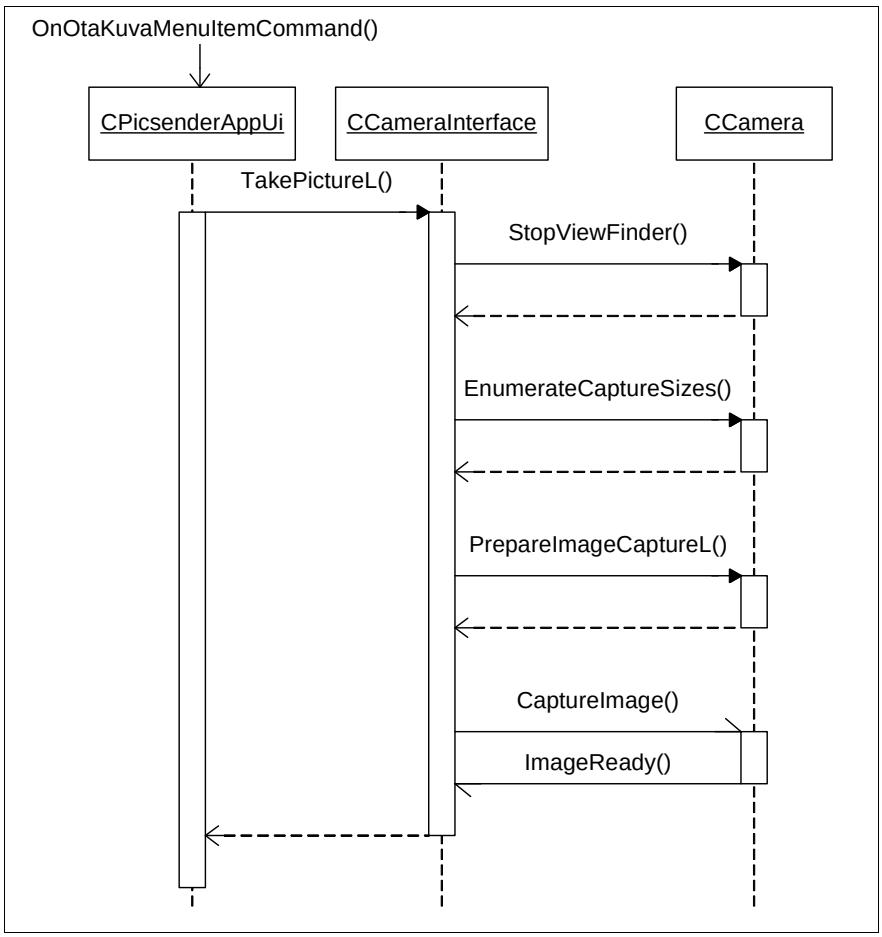
CPicsenderView on ohjelman näkymäluokka, joka omistaa CPicsenderContainer-säilöluokan. Normaalisti näkymäluokka huolehtisi tarvittaessa eri näkymien vaihtamisesta, mutta nyt tätä toiminnallisuutta ei käytetä, koska Picsender sisältää vain yhden näkymän. Näkymän kantaluokkana toimii CAknView.

4.2.4 CPicsenderAppUi

CPicsenderAppUi
-iNaviPane : CAknNavigationControlContainer * -iTabGroup : CAknTabGroup * -iDecoratedTabGroup : CAknNavigationDecorator * -iPicsenderView : CPicsenderView * -iHTTPSender : CHTTPSender *
-InitViewsL() +ConstructL() +~CPicsenderAppUi() +HandleCommandL(in aCommand : TInt) +DispatchAppUICommandEvents(in aCommand : TInt) : bool +HandleKeyEventL(inout aKeyEvent : const TKeyEvent, in aType : TEventCode) : TKeyResponse -OnOtaKuvaMenuItemCommand(in aCommand : TInt) -OnLahetaKuvaMenuItemCommand(in aCommand : TInt)

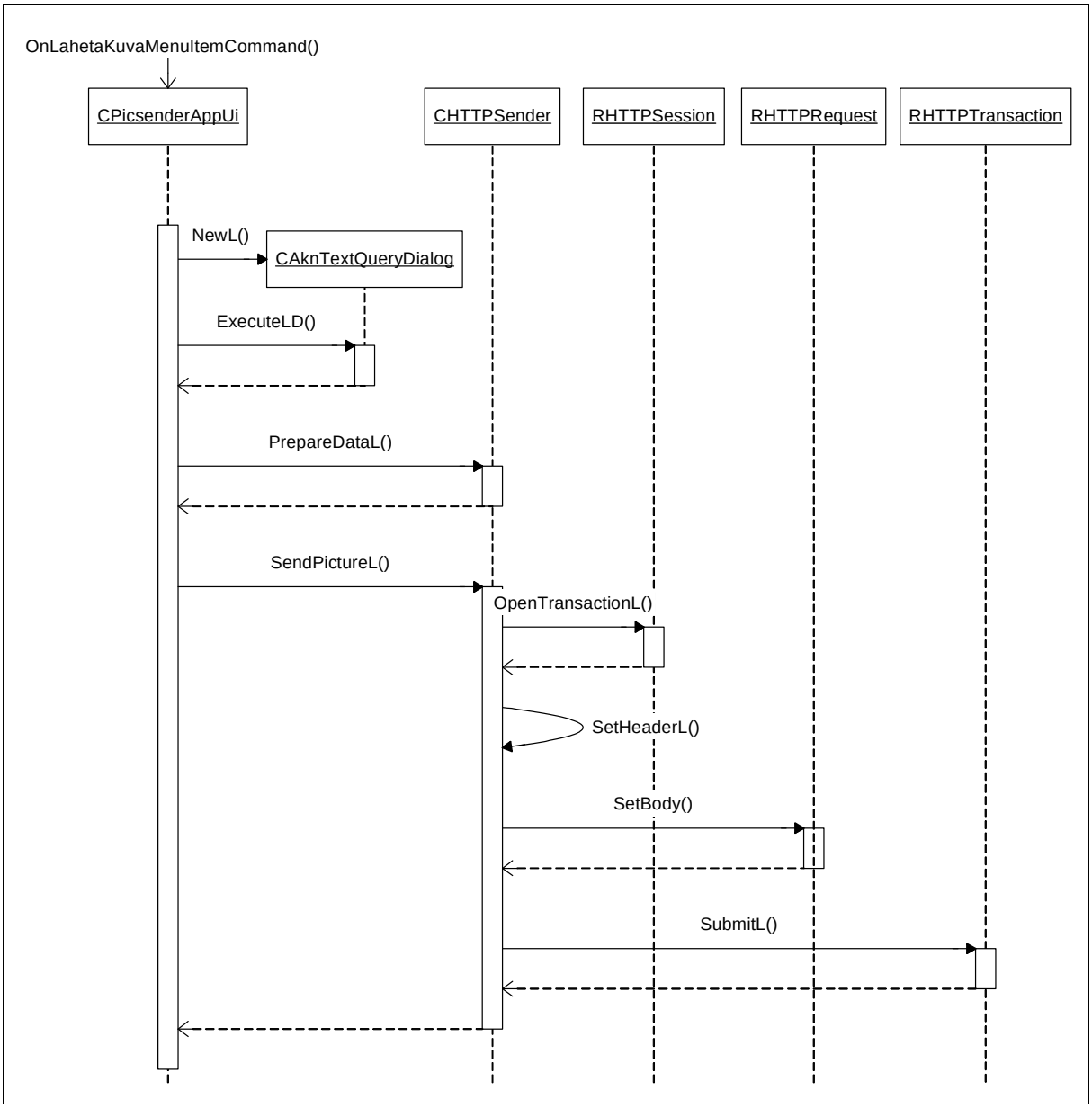
KUVIO 14. CPicsenderAppUi-luokka

CPicsenderAppUi on sovelluksen ohjainluokka. Metodia OnOtaKuvaMenuItemCommand kutsutaan, kun käyttäjä valitsee ohjelman valikosta kohdan ”Ota Kuva”. Vastaavasti järjestelmä kutsuu jäsenfunktiota OnLahetaKuvaMenuItemCommand, kun käyttäjä valitsee ”Lähetä Kuva”. Luokka periytyy luokasta CAknViewAppUi.



KUVIO 15. OnOtaKuvaMenuItemCommand-metodin viestiyhteyskaavio

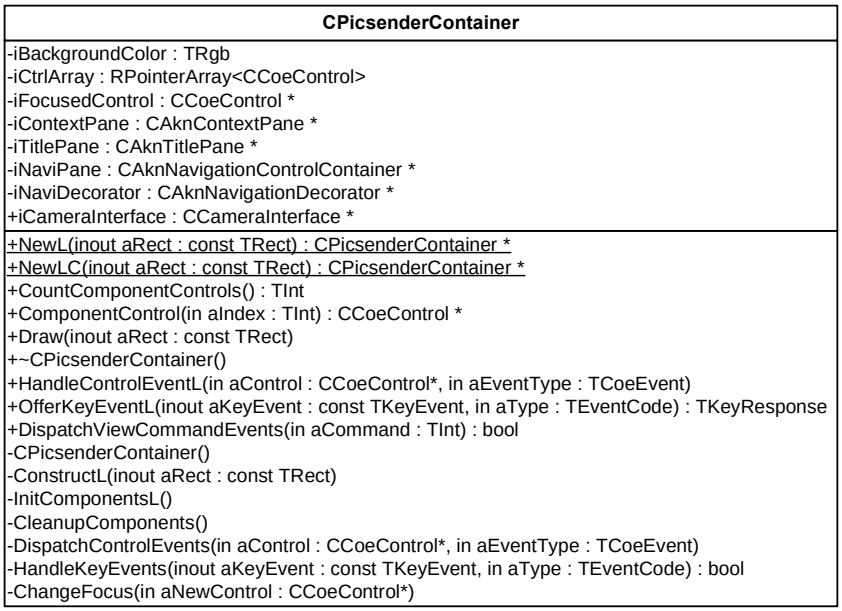
Kuviossa 15 esitetään OnOtaKuvaMenuItemCommand-metodin toimintaa viestiyhteyskaavion (sequence diagram) avulla.



KUVIO 16. OnLahetaKuvaMenuItemCommand-metodin viestiyhteyskaavio

Kuviossa 16 havainnollistetaan mitä tapahtuu, kun käyttäjän ottama kuva lähetetään palvelimelle.

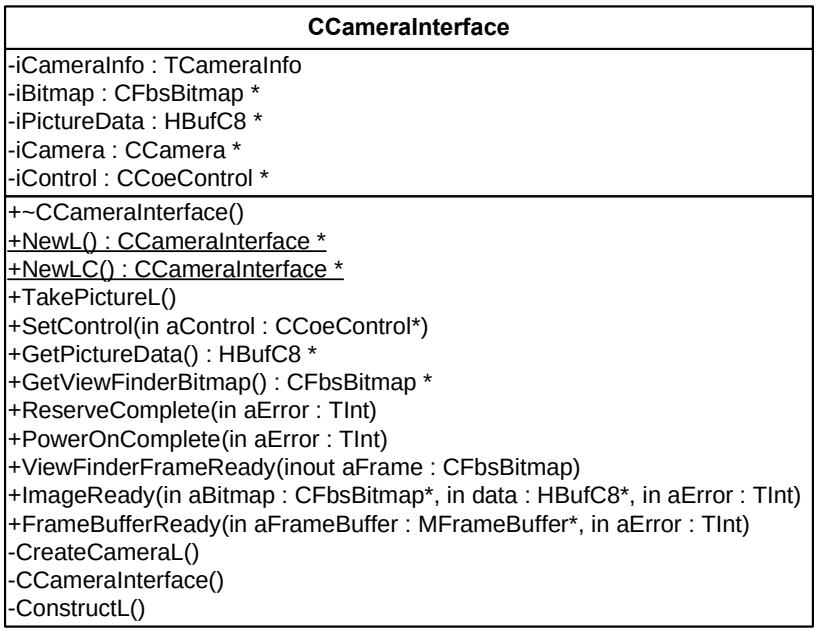
4.2.5 CPicsenderContainer



KUVIO 17. CPicsenderContainer-luokka

CPicsenderContainer-luokka sisältää sovelluksen varsinaiset käyttöliittymäkomponentit. Luokan kantaluokkina toimivat luokat CCoeControl ja MCoeControlObserver.

4.2.6 CCameraInterface



KUVIO 18. CCameraInterface-luokka

CCameraInterface-luokka vastaa puhelimen kameran käsittelystä. Luokka periytyy luokista CBase ja MCameraObserver. Luokasta voidaan luoda ilmentymä normaaliin tapaan NewL tai NewLC-metodeilla. SetControl-jäsenfunktiolla voi asettaa kontrollin, mikä tulee sisältämään etsimen kuvan. Kuvan ottaminen onnistuu kutsumalla metodia TakePictureL.

4.2.7 CHTTPSender

CHTTPSender
-iData : HBufC8 * -iSession : RHTTPSession -iTransaction : RHTTPTransaction
+PrepareDataL(in aPicture : HBufC8*, inout aDescription : TDesC) +GetNextDataPart(inout aDataPart : TPtrC8) : TBool +ReleaseData() +OverallDataSize() : TInt +Reset() : TInt +MHFRunError(in aError : TInt, in Parameter1 : RHTTPTransaction, inout Parameter2 : const THTTPEvent) : TInt +NewL() : CHTTPSender * +SetHeaderL(in aHeaders : RHTTPHeaders, in aHdrField : TInt, inout aHdrValue : const TDesC8) +GetL() +SendPictureL() +MHFRunL(in aTransaction : RHTTPTransaction, inout aEvent : const THTTPEvent) +~CHTTPSender() -ConstructL()

KUVIO 19. CHTTPSender-luokka

CHTTPSender-luokka vastaa tiedon lähettämisestä ja vastaanottamisesta GPRS-yhteyden kautta. GetL-metodilla voi ladata Internet-sivun, ja SendPictureL-metodi lähettää kuvan palvelimelle. PrepareDataL ottaa parametreikseen kuvan sisältämän datan ja tunnisteen, jotka se sitten muotoilee sopivaan formaattiin lähetystä varten iData-jäsenmuuttujaan. CHTTPSender-luokka periytyy luokista MHTTPDataSupplier, MHTTPTransactionCallback ja CBase.

4.3 ECam API

ECam-kamera API on Symbian OS:n valinnainen multimediakomponentti, jonka matkapuhelinvalmistaja voi halutessaan toteuttaa. ECam antaa sovellukselle mahdollisuuden

kontrolloida mitä tahansa puhelimen sisältämää kameralaitteistoa. Se tarjoaa funktioita selvittämään kameran tilaa, määrittämään kameran asetuksia, kontrolloimaan etsintä ja ottamaan kuvia. Myös videon tallennus on mahdollista, vaikka tähän yleensä käytetään VideoRecorderUtility-luokkaa. ECam on määritelty otsikkotiedostossa ECam.h. (Harrison 2004, 309.)

MCameraObserver on ”havannoija” (observer) -luokka, joka sisältää seuraavat abstraktit metodit:

- ReserveComplete
- PowerOnComplete
- ViewFinderFrameReady
- ImageReady
- FrameBufferReady

Käyttääkseen ECam rajapintaa ohjelmoija joutuu periyttämään luokan MCameraObserver-luokasta ja toteuttamaan sen sisältämät abstraktit metodit.

Varsinainen yhteys ECam API:iin saadaan CCamera-luokan kautta, joka kapseloi matkapuhelimen sisältämän kameran. CCamera-olio luodaan NewL-metodin avulla, joka ottaa parametreikseen observer-luokan ja indeksin haluttuun kameraan (Kuvio 20).

```
if(!CCamera::CamerasAvailable())
{
        User::Leave(KErrNotSupported);
}
iCamera = CCamera::NewL(*this,0);
```

KUVIO 19. Esimerkki CCamera-olion luomisesta

Aluksi ohjelma joutuu varaamaan kameran käyttöönsä kutsumalla Reserve-jäsenfunktiota. Kun varaamisprosessi on valmis, järjestelmä kutsuu observer-luokan ReserveComplete-metodia. Kamera joudutaan vielä laittamaan päälle kutsumalla PowerOn-metodia. Tässäkin

tapauksessa järjestelmä kutsuu observer-luokan jäsenfunktiota, tällä kertaa PowerOnComplete-metodia.

ECam API tarjoaa kaksi eri tapaa näyttää etsin puhelimen ruudulla. Ensimmäinen on suora näyttöön piirto, jossa sovellus kertoo järjestelmälle, missä osassa näyttöä haluaa etsimen näkyvän. Tämän jälkeen ECam päivittää kuvan ruudulle. Toisessa tavassa ECam välittää ohjelmalle bittikartat ja sovellus suorittaa itse niiden piirtämisen näytölle. Huomasin, että testauksessa käyttämäni Nokian 6670 -puhelin ei tukenut etsimen suoraa piirtoa näytölle, joten jouduin käyttämään bittikarttoja hyödyntävää, useimmiten hitaampaa tapaa. StartViewFinderBitmapsL-metodia kutsumalla aloitetaan etsimen näyttäminen ruudulle. ECam välittää kuvat määrätyn observer-luokan ViewFinderFrameReady-metodille, joka edelleen välittää ne piirrettäväksi näytölle.

Kuvan ottaminen suoritetaan PrepareImageCaptureL- ja CaptureImage-metodien avulla (Kuvio 21). PrepareImageCaptureL-jäsenfunktion avulla määritetään kuvan haluttu koko ja formaatti. CaptureImage ottaa varsinaisen kuvan ja palauttaa sen ImageReady-metodille.

```
TInt index=0;
TSize size;
CCamera::TFormat format;
format=CCamera::EFormatJpeg;
iCamera->StopViewFinder();
iCamera->EnumerateCaptureSizes(size,index,format);
iCamera->PrepareImageCaptureL(format,index);
iCamera->CaptureImage();
```

KUVIO 20. Esimerkkikoodi kuvan ottamiseen

4.4 HTTP-lähetys

Series 60 2.x sisältää Internet-liikennöintiä varten HTTP Client API:n. Sovellus voi tämän rajapinnan kautta kommunikoida Internetin palvelinten kanssa.

Rajapinta sisältää useita luokkia, joita käytetään datan välityksessä Internetiin. Sessio (RHTTPSession-luokka) sisältää yhden tai useampia, saman yhteyden jakavia transaktioita. Transaktion (RHTTPTransaction-luokka) voi ajatella olevan yksi viestienvälitystoimenpide asiakkaan ja palvelimen välillä. Se sisältää tiedot asiakkaan pyynnöstä (request) ja palvelimen palauttaman vastauksen (response). MHTTPDataSupplier-luokan avulla määritellään joko pyynnön tai vastauksen varsinainen sisältö.

4.5 Kuvan lähetyksen formaatti

RFC 1867 määrittää multipart/form-data – formaatin, jota voi käyttää tiedostojen välittämiseen Internetissä koneiden kesken. Useimmat palvelinsovellukset ja skriptikielet (kuten PHP) tukevat tätä formaattia, joten se oli luonnollinen valinta kuvan lähettämiseen.

```
Content-type: multipart/form-data, boundary=123456789

--123456789
content-disposition: form-data; name="kenttä1"

arvo
--123456789
content-disposition: form-data; name="kuva"; filename="kuva.jpg"
Content-Type: image/jpg

[...tiedoston kuva.jpg sisältö...]
--123456789--
```

KUVIO 21. Esimerkki multipart/form-data-formaatissa olevasta tiedosta

4.6 Kuvan lisäys tietokantaan

Koska kuva välitetään laajalti käytössä olevan standardin mukaan muotoiltuna, on lähetettyjä tietoja helppo hyödyntää esim. PHP-kielellä toteutetussa järjestelmässä. PHP hoitaa datan parseroinnin, joten ohjelmoijan ei tarvitse kirjoittaa sitä varten omaa koodia. Kuvios-

sa 23 näytetään esimerkkikoodi kuvan lisäämiseksi tietokantaan. Kuva on tämän jälkeen tietokannassa muiden sovellusten käytettävissä.

```
<?php

//Luodaan yhteys ja valitaan tietokanta
$link = mysql_connect(null,'user','pass')
   or die('Could not connect: '.mysql_error());
mysql_select_db('picsender')
   or die('Could not select database'.mysql_error());

//Luetaan data väliaikaistiedostosta ja koodataan se
//Base64-algoritmilla
$file=fopen($_FILES['kuva']['tmp_name'],"r");
$data=fread($file,filesize($_FILES['kuva']['tmp_name']));
$data=base64_encode($data);

$query = "INSERT INTO kuva
VALUES(null,'".$_REQUEST['kuvaus']."','$data')";

//Suoritetaan SQL-kysely
$result = mysql_query($query)
   or die('Query failed: ' . mysql_error());

//Suljetaan yhteys
mysql_close($link);

?>
```

KUVIO 22. Esimerkki kuvan lisäyksestä tietokantaan PHP-kielellä

YHTEENVETO

Kartutin paljon tietojani ja taitojani tämän projektin aikana, joten opinnäytetyöni tekemisestä on ollut selvästi hyötyä minulle ja tämänhetkiselle työnantajalleni. Työnantajallani löytyy luultavasti käyttöä lisääntyneestä tietämyksestäni Symbian-ohjelmistokehityksestä uusissa projekteissa. Kokemuksistani on myös itselleni hyötyä, koska aihe on vielä melko uusi ja siten saattaa tarjota työmahdollisuuksia tulevaisuudessa. Kirjoitusprosessi vaati paljon suunnittelua ja organisointia, joten myös nämä taidot harjaantuivat.

Ammattikunnalle työstäni ei välttämättä suoraan kovin suurta hyötyä ole, sillä asiat ovat melko suppeasti esitetty ja ne voivat näin jäädä epäselviksi ilman lisämateriaaliin tutustumista. Työni voi ajatella olevan ainoastaan eräänlainen läpileikkaus Symbian-ohjelmistokehitykseen, esitellen joitakin asioita, jotka ovat mahdollisia toteuttaa järjestelmässä.

Opinnäytetyön tekemistä haittasi, että yhteisö, jolle sovellus oli alun perin tarkoitus tehdä, luopui siitä kustannussyistä projektin alkuvaiheessa. Olisi ollut mielekkäämpää työskennellä, jos järjestelmä olisi tullut ratkaisuksi johonkin olemassa olevaan ongelmaan.

Olisi ollut hyvä, jos meille olisi pidetty jokin opinnäytetyön tekemiseen perehdyttävä opintojakso koulussa. Eräässä kurssissa aihetta sivuttiin, mutta käsittely jäi melko pintapuoliseksi ja silloinkin keskityttiin vain opinnäytetyön aiheen valintaan. Saatavilla olevasta ohjeistuksesta ja muiden tekemien töiden lueskelusta oli apua, mutta silti jotkin asiat jäivät vähän epäselviksi ja aiheuttivat ylimääräisiä ongelmia.

LÄHTEET

Digia Inc. Programming for the Series 60 Platform and Symbian OS. 2003. Chichester: John Wiley & Sons Ltd.

Edwards, Leigh & Barker, Richard. 2004. Developing Series 60 Applications: A Guide for Symbian C++ Developers. Boston: Addison-Wesley.

Harrison, Richard. 2003. Symbian OS C++ for Mobile Phones. John Wiley & Sons Ltd.

Harrison, Richard. 2004. Symbian OS C++ for Mobile Phones, Volume 2. Chichester: John Wiley & Sons Ltd.

Mery, David. 2003. Why is a different operating system needed? WWW-dokumentti. Saatavissa: http://www.symbian.com/technology/why-diff-os.html. Luettu 17.10.2005.

Mikkonen, Tommi. 2004. Mobiili-ohjelmointi. Toinen painos. Helsinki: Talentum Media Oy.

Nokia. 2005. Nokia Introduces Carbide.c++, New Family of Development Tools for Symbian OS. Lehdistötiedote. Saatavissa: http://press.nokia.com/PR/200510/1014976_5.html. 11.10.2005.

Symbian. 2005a. Symbian Fast Facts. WWW-dokumentti. Saatavissa: http://www.symbian.com/about/fastfaqs.html. Luettu 17.10.2005.

Symbian. 2005b. Symbian OS releases. WWW-dokumentti. Saatavissa: http://www.symbian.com/technology/product_descriptions.html. Luettu 18.10.2005.

Symbian. 2005c. Symbian OS software development kits. WWW-dokumentti. http://www.symbian.com/developer/sdks.asp. Luettu 19.10.2005.

Symbian Developer Library. 2002. Saatavissa: http://www.symbian.com/developer/techlib/v70sdocs/doc_source/index.html. Luettu 20.10.2005.