Opinnäytetyö (AMK)

Tietotekniikka

Sulautetut järjestelmät

2013

Joonas Kuusela

# VIDEOVALVONTASOVELLUS MOBIILILAITTEILLE

## –järjestelmäriippumaton mobiiliohjelmointi

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietotekniikan koulutusohjelma | Sulautetut järjestelmät

Huhtikuu 2013 | 33

Ohjaajat: TkL Jari-Pekka Paalassalo, TkK Lauri Nurmi

Joonas Kuusela

# VIDEOVALVONTASOVELLUS MOBIILILAITTEILLE

Tämän työn tarkoituksena oli toteuttaa mobiilivideovalvontasovellus Valvova Oy:n kehittämään Ksenos-videovalvontatallentimeen. Tarkoituksena oli mahdollistaa tallentimen etäkäyttö mobiililaitteella. Vaatimuksena oli mahdollistaa live-kuvan katseleminen, sekä lisäominaisuutena toivottiin Pan/Tilt/Zoom-ohjausta. Lisäksi toivottiin, että sovellus toimisi mahdollisimman monella eri laitteella.

Ohjelmisto toteutettiin käyttämällä Qt-sovelluskehitysympäristöä, joka mahdollistaa ohjelman kääntämisen monille eri ympäristöille, mm. Android-, Windows- ja MeeGo-ympäristöille. Vaikka Qt:lla tehty ohjelmakoodi toimii hienosti eri laitteissa, erilaisten mobiililaitteiden suuri lukumäärä voi aiheuttaa ongelmia. Erilaiset näyttökoot, laitteiden tehoerot sekä käyttöjärjestelmien käyttöliittymäohjeet ovat kaikki asioita, jotka tulee huomioida kehittäessä laitteistoriippumatonta mobiilisovellusta.

Toteutettu Ksenos Mobile -sovellus täyttää kaikki sille asetetut vaatimukset. Myös P/T/Z-ohjaus saatiin toteutettua. Ohjelmisto toimii monilla eri laitteilla ja ympäristöillä (Android, MeeGo, iOS) sekä noudattaa pääpiirteittäin näiden ympäristöjen käyttöliittymäohjeistuksia.

ASIASANAT:

C++, laitteistoriippumaton, Qt, Android, iOS, mobiili, OpenGL, video

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Embedded systems

2013 | 33

Instructors Jari-Pekka Paalassalo Lic.Tech., Lauri Nurmi B.Sc. (Tech.)

Joonas Kuusela

# MOBILE APPLICATION FOR VIDEO SURVEILLANCE

The goal of this thesis was to develop mobile application for Ksenos digital video recorder. Ksenos is developed by Valvova Oy. The main requirement was the possibility to view live feed from cameras. Another minor requirement was to enable Pan/Tilt/Zoom control. In addition, it would be desired that the mobile application would work in as many different devices and operating systems as possible.

Ksenos Mobile was developed with the Qt application framework. Qt is a cross-platform framework that can build applications for many different targets for example Android, MeeGo and Windows. Although applications developed with Qt will work on many different mobile targets, the number of different mobile devices could cause problems. Different screen sizes, hardware of the devices and different user interface guidelines are all small details that programmers must pay attention to while developing cross-platform mobile applications.

Ksenos Mobile meets all the requirements; it works in many different devices and operating systems (Android, MeeGo, iOS) and it also follows the general lines of different user interface guidelines.

KEYWORDS:

C++, cross-platform, mobile, Android, OpenGL, video

# SISÄLTÖ

# KUVAT

# SANASTO

| | |
|---|---|
| C++ | Yksi yleisemmin käytetyistä ohjelmointikielistä. |
| Qt | Laitteistoriippumattomien ohjelmistojen kehittämiseen tarkoitettu ohjelmointiympäristö. |
| Signal/Slot | Qt-ympäristön käyttämä viestimekanismi. Observer-ohjelmointimallin ilmentymä. |
| MOC | Meta-Object Compiler, esikääntäjä, joka luo Qt:n C++-laajennoksia sisältävistä luokista standardia C++-koodia. |
| QML | Qt-ympäristössä käyttöliittymien ohjelmointiin tarkoitettu kuvauskieli. |
| MJPEG | Motion JPEG -videonpakkausformaatti. Videon jokainen kuva pakataan käyttämällä JPEG-pakkausta. |
| P/T/Z | Pan/Tilt/Zoom, kamera, jota on mahdollista kääntää, nostaa sekä zoomata. |

# 1 JOHDANTO

Tämän työn tarkoituksena on suunnitella ja toteuttaa mobiiliasiakasohjelma Valvova Oy:lle. Valvova Oy on turkulainen turvallisuusalan yritys, jonka toimialaan kuuluu kameravalvonta-, kulunvalvonta- sekä rikosilmoitinjärjestelmien myynti ja asennus. Valvova Oy kehittää myös Ksenos-digitaalitallenninta.

Ksenos on videovalvontasovellus, jolla voidaan katsella ja tallentaa digitaalisten sekä analogisten että kameroiden kuvaa sekä tarkkailla ja ohjata erilaisia I/O-laitteita. Ksenosiin on saatavilla myös ohjelmistolisäkkeenä rekisterikilventunnistus.

Asiakasohjelman tarkoituksena on mahdollistaa Ksenos-videovalvontajärjestelmän käyttö mobiililaitteella. Ohjelmiston vaatimuksena oli mahdollistaa live-kuvan katseleminen. Lisätoiveena esitettiin PTZ-ohjausta sekä tukea mahdollisimman monelle puhelinmallille.

Työssä käydään läpi projektiin liittyvät tekniikat ja käytetyt menetelmät. Lisäksi tarkastellaan, mitä haasteita järjestelmäriippumattoman mobiilisovelluksen kehittämiseen liittyy.

# 1 KÄYTETYT VÄLINEET JA TEKNIIKAT

Työssä käytettiin monia erilaisia tekniikoita sekä välineitä. Tärkeimmät työvälineet olivat Qt-sovelluskehitin sekä sen laajennos Necessitas.

## 1.1 C++-ohjelmointikieli

C++ lienee tällä hetkellä käytetyin olio-ohjelmointikieli. Bjarne Stroustrup kehitti vuonna 1980 kielen nimeltä C with Classes, joka tunnetaan C++-kielen versiona 1.0. C++-kieli on ns. hybridikieli, joka tarkoittaa sitä, että se on tavallaan laajennos toisesta ohjelmointikielestä. [1]

C++-kielen perustana on C-kieli. C-kieli valittiin C++-kielen perustaksi, koska se on monipuolinen, selkeä sekä matalantason ohjelmointikieli, joka sopii useimpiin ohjelmointitehtäviin. C-kieli on myös käytettävissä lähes missä ympäristössä tahansa. [1]

C++ kieli laajentaa C-kieltä mm. seuraavilla olio-ohjelmointimekanismeilla: [1]

- kapselointi (encapsulation)
- olioiden luonti, tuhoaminen sekä viestinvälitys (class)
- periytyminen (inheritance)
- monimuotoisuus (polymorphism)
- aliohjelma- ja luokkamallit (templates)
- STL-kirjasto (Standard Template Library)
- viittausmuuttujat (references)
- aliohjelmien kuormitus (overloading)
- aliohjelmaparametrien oletusarvot
- poikkeuskäsittely (exception handling).

## 1.2 Qt ohjelmistonkehitysympäristö

Qt on käyttöjärjestelmäriippumaton ohjelmistonkehitysympäristö, jota käytetään hyvin paljon käyttöliittymällisten ohjelmistojen tekemiseen. Qt käyttää C++-kieltä ja tarjoaa tätä varten monia hyödyllisiä kirjastoja tiedostojenkäsittelyä, tietokantoja, verkkoja sekä moniajoa varten. [2]

Qt:n on alun perin kehittänyt norjalainen Trolltech. Vuonna 2008 Nokia Oy osti Qt:n, ja vuonna 2011 Digia Oy osti Qt:n kaupalliset oikeudet Nokialta. Digia Oy hankki Qt:n kokonaisuudessaan Nokia Oy:ltä vuonna 2012 . Tällä hetkellä Digia Oy vastaa Qt:n kehityksestä, sen kaupallisesta lisensoinnista sekä avoimen lähdekoodin lisensoinnista yhdessä Qt-project-yhteisön kanssa. [2]

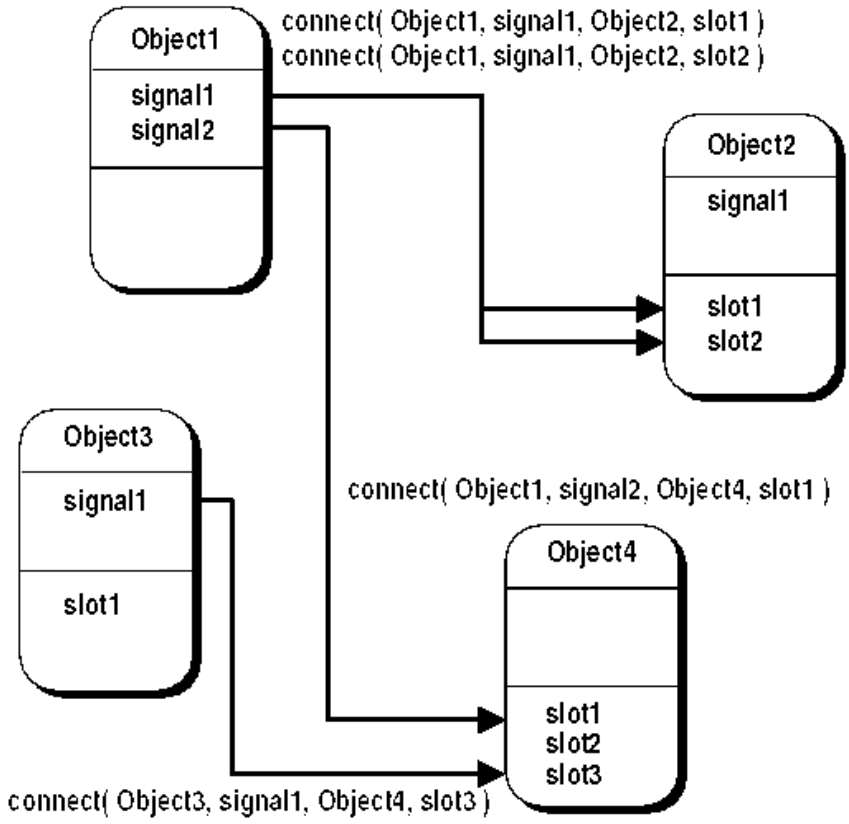
Qt 5.0 -ohjelmistokehitysympäristöllä voidaan tehdä sovelluksia mm. seuraaviin ympäristöihin: [3]

- Windows
- Windows CE,Mobile
- Symbian
- OS X
- X11
- Maemo, MeeGo
- Android
- iOS
- BlackBerry

## 1.1 Qt:n laajennokset C++-kieleen

Qt tarjoaa paljon erilaisia apukirjastoja. Se myös laajentaa C++-kieltä omilla ominaisuuksillaan, joista keskeisin on Signal/Slot-järjestelmä [4], joka on periaattessa Observer-ohjelmointimallin ilmentymä. Järjestelmää käytetään lähes jokaisessa Qt:n tarjoamassa luokassa.

Signal/Slot-järjestelmän ideana on, että olio voi lähettää signaalin johon voidaan kytkeä toisien olioiden metodeja eli slotteja. Kun olio lähettää signaalin, kaikkien tähän kytkettyjen slottien ohjelmakoodi suoritetaan [4]. Kuva 1 esittää Signal/Slot-järjestelmää neljän eri objektin välillä.




Kuva 1 Signal/Slot-järjestelmä [4].

Signal/Slot-järjestelmä sekä muut laajennokset on toteutettu ns. MOC-kääntäjän kautta.Tällä saavutetaan se, että Qt:n ei tarvitse tehdä omaa C++-toteutusta, vaikka käyttäjälle tämä näyttääkin siltä. Ennenkuin koodi menee oikealle kääntäjälle, se ajetaan MOC:n läpi, jolloin tuloksena on standardia C++-koodia, jota voidaan kääntää normaalisti. [4]

## 1.2 Qt:n tarjoamat käyttöliittymä ratkaisut

Qt tarjoaa käyttöliittymän tuottamiseen kaksi eri ratkaisua: QtWidgets sekä QtQuick. QtWidgets-ympäristö tarjoaa QtWidget-komponentteja, jotka on tar-

koitettu työpöytäympäristöihin (PC, MAC). Nämä komponentit käyttävät käyttöjärjestelmän omia käyttöliittymäkomponentteja. [5]

QtQuick on kokoelma erilaisia tekniikoita, joilla pystytään rakentamaan nykyaikaisia sekä interaktiivisia käyttöliittymäsovelluksia. Ympäristö on tarkoitettu järjestelmiin, joita käytetään kosketusnäytöllisissä laitteissa kuten mobiililaitteissa. QtQuick-käyttöliittymät rakennetaan käyttämällä QML-kuvauskieltä. QML perustuu JavaScript-kieleen, joten QML-komponenttien sekaan voidaan kirjoittaa logiikkaa JavaScriptillä. [6]

Koodilistauksessa 1 on esimerkki yksinkertaisesta QML-komponentista. Koodi luo punaisen neliön, jonka keskellä on teksti ”Hello”. Mikäli punaisen neliön aluetta klikkaa hiirellä, sen väri vaihtuu mustaksi.

Koodilistaus 1

```
Rectangle{
        id:main
        color:"red"
        width:100
        height:100
        Text{
                text:"Hello";
                anchors.verticalCenter:parent.verticalCenter;
                anchors.horizontalCenter:parent.horizontalCenter;
        }
        MouseArea{
                anchors.fill:parent
                onClicked:{
                        main.color="black";
                }
        }
}
```

Vaikka on mahdollista kirjoittaa koko sovellus käyttämällä vain QML- ja JavaScript-kieltä, suositeltu tapa on tehdä käyttöliittymä QML-kielellä ja sitten ohjelman ”business”-koodi C++-kielellä. [6]

## 1.3 Qt:n Android-versio Necessitas

Koska Qt:lla ei vielä ole virallista Android-tukea, romanialainen BogDan Varta on toteuttanut Qt-tuen Androidille. Tämän projektin nimi on Necessitas. Projekti

on saatavilla kaikille yleisimmille työpöytäkäyttöjärjestelmille: Windows, MAC sekä Linux. [7]

Necessitas käyttää Android-NDK-työkaluja. Koska Android-sovelluksien pääkehityskieli on Java, Qt-ohjelmat käännetään dynaamiseksi kirjastoksi, joka ladataan käyttämällä yksinkertaista Java-ohjelmaa. Java-ohjelman tarkoituksena on toimia Android-järjestelmän sekä Qt-ohjelmakoodin välikätenä. Käyttäjän ei tarvitse välittää Java-ohjelmasta, vaan Necessitas-ympäristö huolehtii siitä automaattisesti. [7]

Qt:n blogissa Tuukka Teronen kertoi, että Necessitas on lahjoitettu Qt-projektiin ja BogDan Varta jatkaa virallisen Android-version kehityksessä [8]. Necessitas tukee lähes kaikkia Qt:n ominaisuuksia. Joitakin multimediaominaisuuksia ei vielä tueta. Tällä hetkellä uusin Necessitaksen versio on Alpha 4. [7]

# 2 JÄRJESTELMÄRIIPPUMATTOMAN MOBIILISOVELLUKSEN KEHITYS

Tänä päivänä mobiilisovelluksen kehittäminen on mielekkäämpää kuin koskaan. Valmistajat tarjoavat valmiit kehitysympäristöt sekä hyvät dokumentaatiot ja laitteissa on paljon erilaisia ominaisuuksia, joiden pohjalta on hyvä tehdä mitä erilaisempia sovelluksia. Toisaalta uusia puhelinmalleja julkaistaan nopealla tahdilla ja uusia ominaisuuksia tulee koko ajan enemmän, joten laitteet ovat entistä monipuolisempia.

Ohjelmisto olisi hyvä saada toimimaan mahdollisimman vähällä vaivalla monella eri alustalla (esim. Androidilla sekä iOS:llä). Mobiilisovelluksien tekemiseen on yleensä saatavilla kahta erityyppistä kehitystyökalua: natiivikehitystyökalut sekä järjestelmäriippumattomat työkalut [9] .

## 2.1 Natiivikehitystyökalut

Natiivikehitystyökaluilla kehitetään ohjelmistoja, jotka toimivat vain sille tarkoitetulla alustalla, esim. Applen iOS, Android ja Microsoft Windows Phone [9]. Natiivityökaluilla tehdyt ohjelmat näyttävät ja tuntuvat samoilta kuin muutkin sovellukset kyseisessä järjestelmässä. Lisäksi ne voivat tarjota joitain ominaisuuksia, mitä ei muista järjestelmistä löydy.

## 2.2 Järjestelmäriippumattomat kehitystyökalut

Järjestelmäriippumattomilla kehitystyökaluilla tehdään järjestelmäriippumattomia sovelluksia. Tämä tarkoittaa sitä, että sovellus toimii useammalla kuin yhdellä alustalla. [10]

Järjestelmäriippumaton sovellus voi koostua monista eri lähdekoodeista. Ohjelmasta voi esimerkiksi olla erikseen lähdekoodit iOS-käyttöjärjestelmälle sekä Android-käyttöjärjestelmälle. Sovellus voidaan kehittää myös työkalulla, joka

piilottaa erilaisten järjestelmien eroavaisuudet. Tälläinen on esimerkiksi tässä työssä käytetty Qt.

Järjestelmäriippumattomien mobiiliohjelmistojen tekemiseen on tarjolla monenlaisia ympäristöjä. Qt:n lisäksi on myös monia muita kehitysympäristöjä kuten [11]

- Sencha Touch 2
- jQuery Mobile
- Tiggzi
- PhoneGap.

Suurin osa näistä käyttää kehityskielenä HTML 2:sta. [11]

## 2.3 Testaaminen

PC/MAC-sovelluksia tai internetsivuja tehdessä tuotosta pystyy helposti testaamaan omalla työasemalla, eri käyttöjärjestelmällä sekä eri kokoonpanoilla. Testitapauksia on suhteellisen vähän ja yleensä myös tiedetään minkälaiseen ympäristöön sovellus tulee, joten voidaan varmistua siitä, että se toimii niin kuin pitääkin. [12] Testaamisen voi jakaa kahteen eri kategoriaan: ohjelmakoodin testaamiseen, jolla varmistetaan, että ohjelman logiikka toimii niin kuin pitää (esim. yksikkötestaus), sekä integrointi- ja käyttöliittymätestaukseen, jolla varmistetaan ohjelman sujuva toiminta erilaisissa laitteisteissa.

Mobiilisovelluksissa ongelmaksi muodostuvat mahdollisten kohdelaitteiden määrä ja näiden pienet eroavaisuudet. Sovelluksien sujuva toiminta mahdollisimman monella laitteella voi osoittautua hyvinkin hankalaksi. [12]

### 2.3.1 Yksikkötestaus

Yksikkötestaus tarkoittaa lähdekoodin yksittäisen osan testaamista. Yksittäisellä osalla tarkoitetaan esimerkiksi jonkin luokan yhtä metodia. Yksikkötestauksessa on tavoitteena jakaa ohjelmisto yksittäisiin osiin ja varmistua jokaisen ohjel-

misto-osan toimivuudesta [13]. Koodilistauksessa 2 on esitetty yksinkertainen yksikkötesti. Koodi ei ole mitään tiettyä ohjelmointikieltä vaan tarkoituksena on näyttää yksikkötestin idea.

Koodilistaus 2

```
class Laskin {
public:
        int summaa(int _lukua, int _lukub){
                return _lukua+lukub;
        }
}

class laskuTest{
public:
        void testaaSumma() {
                Laskin l;
                //Testataan menevätkö summat oikein
                IS_TRUE(l.summaa(4,4),8);
                IS_TRUE(l.summaa(-4,4),0);
                IS_TRUE(l.summaa(-4,-4),-8);
                //Testataan liukuluvuilla
                IS_TRUE(l.summaa(4.25,4.25),8.5);
                IS_TRUE(l.summaa(-4.01,4.0),-0.1);
                IS_TRUE(l.summaa(-4,-4.88),-8.88);
        }
}
```

Yksikkötestaus tapahtuu yleensä testausohjelmistokehyksen avulla, mikä mahdollistaa testien automaattisen ajamisen. Testausohjelmistokehyksiä löytyy lähes jokaiselle ohjelmointikielelle, esimerkkinä mainittakoon Junit-testauskehys Java-ohjelmointikielelle. [13]

Yksikkötestausta voidaan käyttää myös ohjelmiston suunnittelussa. Tätä kutsutaan testivetoiseksi kehitykseksi (engl. TDD, Test-Driven Development). Ideana on ensin kirjoittaa ohjelmisto-osalle testi ja tämän jälkeen tehdä ohjelmakoodi, joka läpäisee testin. Tätä toistetaan kunnes ohjelmisto-osa on valmis. Kun ohjelmisto-osa on valmis, sille löytyy valmiina monia testejä. Mikäli ne ovat hyvät, myös ohjelmisto-osa on toimiva. [13]

### 2.3.2 Integrointi- ja käyttöliittymätestaus

Mobiilisovellusta tulisi testata mahdollisimman paljon projektin eri vaiheissa, mieluumin mahdollisimman monella laitteella. Tämä on tärkeää varsinkin käyttöliittymän teon aikana, sillä mobiililaitetta käytetään hyvin eri tavalla verrattuna vaikka PC/MAC-ympäristöön. [12] Mobiililaitetta voidaan käyttää yhdellä kädellä, pimeässä tai vaikka erittäin kirkkaassa valossa. Mobiililaitetta voidaan myös pitää pysty- tai vaakasuunnassa .

Sovellusta on testattava siis mahdollisimman monella laitteella, joten työkavereilta sekä tutuilta voi lainata niitä, jolloin saa oikeaa tietoa miten sovellus toimii kussakin laitteessa. Koko sovelluksen täydelliseen testaukseen tämmöinen ei tietenkään sovi, mutta on se parempi kuin ei mitään. [12]

Mobiililaitevalmistajat tarjoavat myös laitteidensa emulaattoreita testivälineiksi. Emulaattorikaan ei kuitenkaan ole mikään täydellinen ratkaisu, sillä siinä voi olla ohjelmointivirheitä tai emulaattori ei vain toimi täsmälleen samalla tavalla kuin kohdelaite. Esimerkiksi tietokoneen näytöllä näkyvä emulaattori ei ole välttämättä samankokoinen kuin oikea laite. [12]

Maksullisista testauspalveluista ehkä mielenkiintoisin on ns. etätestauspalvelu. Ideana siinä on, että paljon erilaisia mobiililaitteita on kytketty internettiin sekä eräänlaisiin ohjauslaatikoihin. Näiden ohjauslaatikoiden avulla mobiililaitteiden perustoimintoja pystytään käyttämään esimerkiksi www-käyttöliittymän avulla, josta voidaan myös käynnistää testattava sovellus. [12]

## 2.4 Mobiilialustojen käyttöliittymien haasteet

Mikäli käytettävä sovelluskehitin ei tarjoa rajapintaa natiivikäyttöliittymäkomponentteihin, tästä voi tulla ongelma, koska käyttäjät ovat luultavasti tottuneet siihen, että ohjelmistot toimivat tietyllä tavalla tietyissä ympäristöissä. Esimerkiksi Windows-ohjelmistoissa sulkemispainike on oikeassa ylänurkassa, kun taas OS X -ohjelmistoissa se on vasemmassa ylänurkassa.

Mobiilialustoilla on kaikilla järjestelmillä omat käyttöliittymäohjeistuksensa [14], joissa määritellään ohjelman yleiset ulkoasumallit, värimaailma, fyysisten nappien toiminta sekä yleisesti se, miten ohjelmien tulisi toimia. Ongelmaksi voi myös muodostua eri laitteiden näyttöresoluutiot. iOS-laitteita on viidellä eri resoluutiolla [15], mutta Android-laitteiden näytöissä on yli kymmenen eri resoluutiovaihtoehtoja. [16] Ohjelman toimimaan saaminen näillä kaikilla on hankalaa.

iOS:n ja Androidin tapauksessa käyttöliittymäohjeet sekä komponentit ovat suhteellisen samanlaisia, mutta joitain eroavaisuuksia löytyy. Mikäli päätyy tekemään mukautetun käyttöliittymän, on erittäin suotavaa tehdä kaikista komponenteista kohdealustansa näköisiä.

# 3 KSENOS MOBILE

Ksenos Mobile on Valvova Oy:lle tuotettu mobiiliohjelmisto, jonka tulisi toimia mahdollisimman monella mobiilialustalla. Ksenos Mobile tuotettiin käyttämällä seuraavia ohjelmistoversioita:

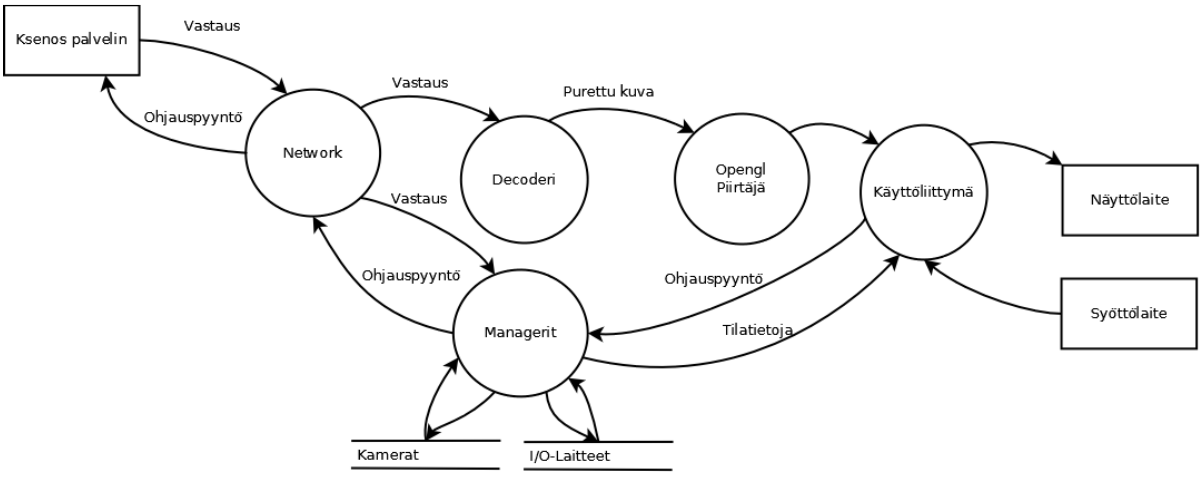
- Necessitas Alpha 4
- Qt 4.8
- QML 1.0
- Qt Creator 2.4.1.

## 3.1 Tavoite

Ksenos Mobilen tavoitteena on mahdollistaa live-kuvan katseleminen Ksenos-tallentimen kautta. Mahdollisuuksien mukaan ohjelmistoon tulisi lisätä lisäominaisuuksia, kuten P/T/Z-ohjaus, monen kameran kuvan katseleminen yhtäaikaa sekä digitaalisten I/O-laitteiden tarkkailu. Lisäksi toivottiin, että ohjelmisto toimisi mahdollisimman monella mobiililaitteella.

## 3.2 Arkkitehtuuri

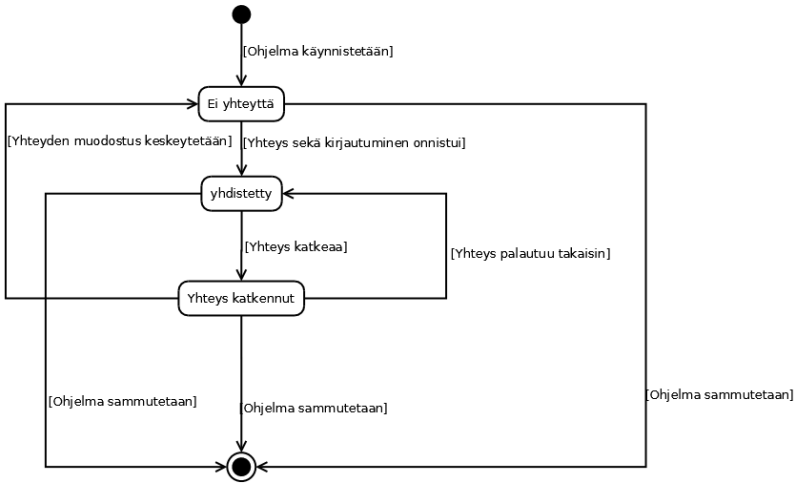
Ohjelmisto jakautuu viiteen eri osioon, joista jokaisella on omat tehtävänsä. Tarkoituksena on, että osiot eivät tiedä toisistaan, vaan ne suorittavat oman tehtävänsä ja tämän jälkeen antavat siitä viestin käyttämällä Qt:n Signal/Slot-mekanismia. Näiden osioiden lisäksi ohjelmistossa on pääluokka, jonka tehtävänä on luoda sekä alustaa ohjelman muut osiot. Kuvassa 2 on esitetty Ksenos Mobilen pääpiirteinen toiminta Dataflow-kaavion avulla.




Kuva 2 Dataflow-kaavio Ksenos Mobilen toiminnasta.

### 3.2.1 Pääohjelma

Korkeantason toimintana Ksenos Mobile on tilakone, jolla on kolme tilaa: Ei yhteyttä, Yhdistetty, sekä Yhteys katkennut -tila. Ei yhdistetty -tila on tila, jossa ohjelmisto vain odottaa yhteydenmuodostuskäskyä. Mikäli yhdistämiskäsky suoritetaan ja se onnistuu, ohjelmisto siirtyy ”Yhdistetty”-tilaan. ”Yhdistetty”-tila on tila, jossa ohjelmiston päätoiminnot tapahtuvat. Mikäli yhteys Ksenos-tallentimeen katkeaa, ohjelmisto siirtyy Yhteys katkennut -tilaan, jossa se pysyy niin kauan, että yhteys palautuu tai käyttäjä siirtää ohjelman Ei yhteyttä -tilaan. Kuvassa 3 on esitelty Ksenos Mobilen tilat sekä siirtymäehdot.




Kuva 3 Tilakonekaavio Ksenos Mobilesta.

### 3.2.2 Verkkoprotokolla

Ksenos-tallentimella on oma etäkäyttöprotokolla, jonka avulla Ksenos-tallentimia voidaan ohjata. Yhteys avataan tallentimessa määriteltyyn TCP-porttiin. Kun yhteys on avattu, tallennin ei ilmoita mitään ennen kuin sisäänkirjauminen on suoritettu. Etäkäyttöprokolla käyttää UTF-8-muotoista tekstiliikennettä ja komentorivit erotellaan CR+LF-merkkijonolla (carriage return + line feed).

Suurin osa protokollan viesteistä on pyyntöjä, joihin Ksenos-tallennin luo yleensä jonkinlaisen vastauksen. Pyynnöt ovat muotoa:

*”request <tunnistenumero> <toistuvuusväli> <koko> <CRLF> <pyyntö>”.*

Tunnistenumeron perusteella voidaan päätellä, mikä vastaus liittyy mihinkin pyyntöön. Toistuvuusväli määrittää kuinka usein tallennin suorittaa pyynnön. Koko parametri kertoo, kuinka monta tavua pyyntökomentoa seuraava tietomäärä on. Pyyntö-parametri on jokin ennalta määrätyistä pyynnöistä. Jokaiseen pyyntöön annetaan vastaus, kun se on käsitelty. Vastaus on muotoa:

”*response <pyynnnön tunnistenumero> <koko> <CRLF> <vastaus>”.*

Kuvassa 4 on esimerkki asiakkaan ja Ksenos-tallentimen välisestä viestiliikenteestä.

| Asiakas | Palvelin |
|---|---|
| request 1 -1 29[CRLF]login admin ]96939a9e99 15 60 | -> |
| <- | response 1 16[CRLF]login successful |
| request 2 -1 11[CRLF]listcameras | -> |
| <- | response 2 **aaaaa**[CRLF]4[CRLF] |
| <- | 0 352 288 0 XeCap-1 **xx**[CRLF](asetukset, **xx** tavua) |
| <- | 1 352 288 0 XeCap-2 **yy**[CRLF](asetukset, **yy** tavua) |
| <- | 2 704 288 0 XeCap-3 **zz**[CRLF](asetukset, **zz** tavua) |
| <- | 3 352 288 0 XeCap-4 **åå**[CRLF](asetukset, **åå** tavua) |

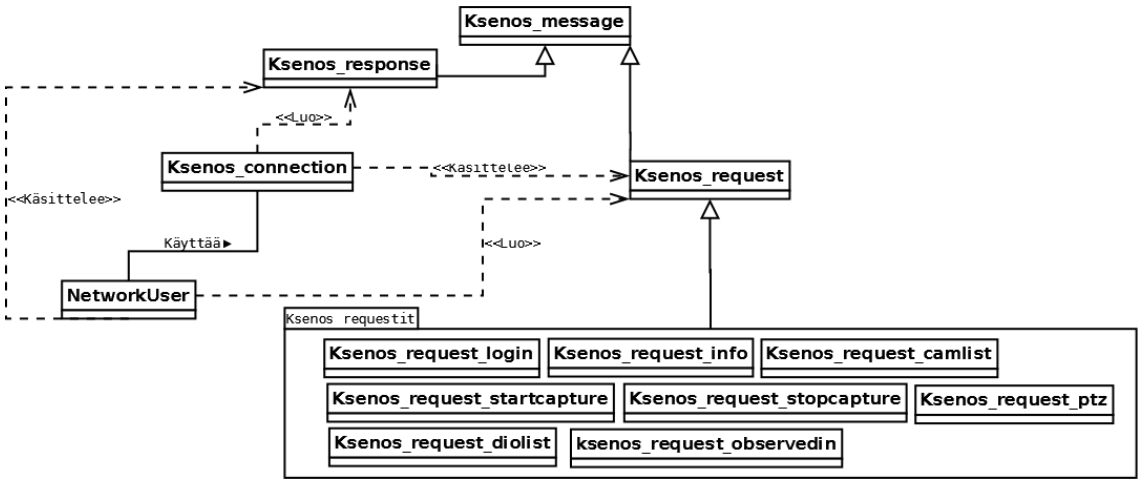
Kuva 4 Sisäänkirjautuminen ja asetusten haku.

Tallentimelle on myös mahdollista lähettää komentoja. Nämä eroavat pyynnöistä siten, että näihin ei aina luoda vastausta. Esimerkkinä komennosta on esimerkiksi kameran kirkkauden säätö.

### 3.2.3 Verkkototeutus

Ksenos Mobilen verkko-osio koostuu pääluokasta (Ksenos_connection) sekä pienistä apuluokista (Ksenos_message,Ksenos_request,Ksenos_response). Tarkoituksena on, että Ksenos_connection-luokalle annetaan pyyntö (Ksenos_request), jonka luokka lähettää tallentimelle. Pyynnön saatuaan tallennin lähettää vastaustiedon. Ksenos_connection lukee vastaustiedon sekä luo tästä vastausluokan (Ksenos_response) ja lähettää sen käyttämällä Qt:n Signal/Slot-mekanismia.

Kuvassa 5 on esitelty Ksenos Mobilen verkko-osian luokat sekä luokkien väliset yhteydet.




Kuva 5 Verkko-osion luokat sekä niiden yhteydet.

### 3.2.4 Kamerat

Yhtä kameraa kuvataan Camera-luokan avulla. Tämä luokka sisältää kamerasta joitain tietoja (mm. nimi, kuvanlaatu). Camera-luokan tärkein toiminto on

käytettävän decoderin käskyttäminen startCapture()- sekä stopCapture()-metodeilla.

Camera_manager on kameroiden hallintaluokka, joka periytyy QAbstractList-Model-luokasta. Tämä tarkoittaa, että luokka on eräänlainen tietomalli, jota pystytään helposti käyttämään esimerkiksi käyttöliittymässä. Mallin käyttäminen käyttöliittymässä on esitelty koodilistauksessa 3. Listauksen yläosassa malli paljastetaan C++-ohjelmakoodinpuolella QDeclarativeViewille . Alaosassa on QML-ohjelmakoodia, jossa mallin avulla luodaan lista, joka näyttää kameroiden nimet.

Koodilistaus 3

```
/**
 * Paljastetaan cameraManager-luokan ilmentymä m_cameras
 * QDeclarativeViewille nimellä "cameraManager".
 */
view->rootContext()->setContextProperty("cameraManager",&m_cameras);
```

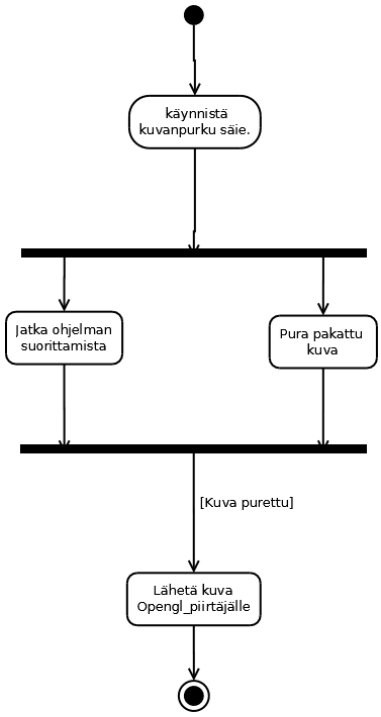
```
import QtQuick 1.1
ListView{
    id:kameralista

    model:cameraManager
    delegate: Text{
        text:name//name on cameraManagerin "tieto". Tässä tapauksessa se antaa kameran nimen
    }
}
```

### 3.2.5 Kuvan purkaminen

Ksenos_decoder-luokan tarkoituksena on tarjota rajapinta kuvanpurkutoiminnolle. Idea on yksinkertaisesti se, että Ksenos_decoder-luokalle annetaan pakattua kuvatietoa, jonka se purkaa ja lähettää tämän puretun kuvatiedon eteenpäin. Ksenos_decoder-luokka ei ota kantaa siihen, miten pakattu kuvatieto puretaan. Se tarjoaa virtuaalisen processImage(QSharedPointer<QByteArray>)-metodin sekä imageReady(QSharedPointer<QImage>)-signaalin, joiden avulla voidaan luoda omia Ksenos_decoder-toteutuksia.

Ksenos_decoder_MJPEG on luokka, joka on periytetty Ksenos_decoder-luokasta ja tälle on kirjoitettu MJPEG-kuvan purkava processImage(QSharedPointer<QByteArray>)-metodi. Koska MJPEG-kuvan purku voi kes-

tää jonkin aikaa, tämä operaatio ajetaan erillisessä säikeessä. Qt tarjoaa tätä varten QtConcurrent-luokan, joka mahdollistaa yksittäisen metodin ajamisen omassa säikeessään. Kuvassa 6 on esitetty Ksenos_decoder-luokan toiminta UML-kaavion avulla. Koodilistauksessa 4 on esitetty MJPEG-kuvan purkamiseen tarkoitettu ohjelmakoodi.




Kuva 6 Kuvanpurku UML-kaaviona.

Koodilistaus 4

```
void  Ksenos_decoder_MJPEG::processImage(QSharedPointer<QByteArray> m_rawdata){
    if(m_future.isFinished()){
        m_futurewatcher.setFuture(m_future);
        /**
         *ajetaan creatImageFromData metodi erillisessä säikeessä.
         *m_future on QFuture luokan ilmentymä, tämä lähettää signaalin kun metodi on saatu suoritettua.
         *metodin palauttamaan tietoon päästään käsiksi m_futurewatcherin kautta, joka on
         *QFutureWatcher luokan ilmentymä.
         */
        m_future = QtConcurrent::run(this,&Ksenos_decoder_MJPEG::createImageFromData,m_rawdata);
    }
}

/**
 * metodi luo parametrina annetusta tiedosta kuvan,
 * ja palauttaa tämän kuvan QImage luokassa.
 */
QSharedPointer<QImage>  Ksenos_decoder_MJPEG::createImageFromData(QSharedPointer<QByteArray> data){
    int datastartidx=0;
    for(int i = 0;i<data->count();i++){
        if(data->at(i)=='\015' &&data->at(i+1) == '\012'){
            datastartidx=i+1;
            break;
        }
    }
    data->remove(0,datastartidx+1);
    QSharedPointer<QImage> img =  QSharedPointer<QImage>(new QImage());
    img->loadFromData(*data.data());
    return img;
}
```

## 3.3 Käyttöliittymä

Ohjelman käyttöliittymä on tehty käyttämällä QML-kuvauskieltä. Tämä tarkoittaa sitä, että käyttöjärjestelmän omat käyttöliittymäkomponentit eivät ole käytettävissä vaan ohjelmaa varten jouduttiin tekemään omia komponentteja kuten nappi, liukuvalitsin sekä palkki. Komponenttien suunnittelu oli hankalaa. Aluksi piti tutkia, miltä kyseiset komponentit näyttävät eri järjestelmillä, minkä jälkeen ne piti toteuttaa QML-kielellä siten, että ne näyttävät ja toimivat suurinpiirtein samalla tavalla kuin alustan omat komponentit.

### 3.3.1 Käyttöliittymäkomponentit

Liukuvalitsinkomponentilla on mahdollista valita arvoja tiettyjen raja-arvojen väliltä. Käyttäjä liikuttaa valitsimen keskellä olevaa palloa ja käyttäjän lopettaessa liikuttamisen valitsin lähettää OnCurrentValueChanged-signaalin. Kuvassa 7 on ohjelmaa varten tehty liukuvalitsin arvoilla 0–100.


Kuva 7 Liukuvalitsin arvoilla 0–100.

Palkkikomponentit ovat ruudun ylä- ja alaosassa. Yläpalkissa näytetään sivun otsikko ja alapalkista löytyvät päänavigointinappulat.

Nappikomponenttia painettaessa sen väri vaihtuu. Käyttäjän lopettaessa painalluksen nappi palautuu alkuperäiseksi. Napille voidaan määritellä teksti, jota se näyttää sekä lisäksi voidaan määritellä ikonit oikeaan ja vasempaan reunaan. Nappi lähettää painalluksesta erilaisia singlaaleja riippuen napin tilasta: OnClicked-signaali lähetetään, kun nappia on klikattu, OnPressed-signaali lähetetään, kun nappi painetaan pohjaan sekä OnReleased-signaali, kun nappi vapautetaan. Kuvassa 8 on nappi, jonka tekstiksi on määritelty ”Login”.


Kuva 8 Nappi jonka tekstiksi on määritelty ”Login”.

### 3.3.2 Ulkoasu

Käyttöliittymästä pyrittiin tekemään mahdollisimman neutraali, jotta eri valmistajien puhelimien käyttäjät osaisivat käyttää ohjelmaa. Ylhäällä on yläpalkki, jossa lukee ohjelmanosio jossa ohjelma on. Keskellä on ohjelman pääosio, jossa on tähän ohjelmaosioon liittyvät toiminnot ja alhaalla alapalkki, josta löytyy ohjelman päänavigointi.  Edellä kuvattu käyttöliittymä näkyy kuvassa 9.


Kuva 9 Ksenos Mobilen käyttöliittymä.

Navigoinnin teki hankalaksi ns. takaisin-nappi. Android-laitteissa on fyysinen takaisin-nappi, jolla ohjelmiston olisi tarkoitus mennä edelliseen tilaan tai sammua. Applen iOS-laitteissa ei ole tällaista nappia, mutta sovelluksen vasemmassa ylänurkassa on nappi, jolle voidaan ohjelmoida takaisin-toiminto. Mee-

Go-laitteissa takaisin-nappi on taas sovelluksen alaosassa, pohjapalkin yläpuolella.

Takaisin-napin tuomat ongelmat ratkaistiin siten, että ohjelmassa ei ole muuta navigointia kuin alapalkin navigointinapit. Näin ei tule tilannetta, jossa takaisintoimintoa tarvittaisiin. Ratkaisu voi jatkokehityksen kannalta olla huono, mikäli halutaan lisätä päänavigoinnin lisäksi jonkinlainen alavalikko. Silloin takaisintoiminnolle täytyy kehittää jokin muu ratkaisu.

Pääosio koostuu erilaisista sivuista, joita piilotetaan ja näytetään riippuen siitä, millä sivulla käyttäjä on. Ohjelmassa on seuraavat sivut: sisäänkirjautuminen, monikamera, kamera ja I/O-laite.

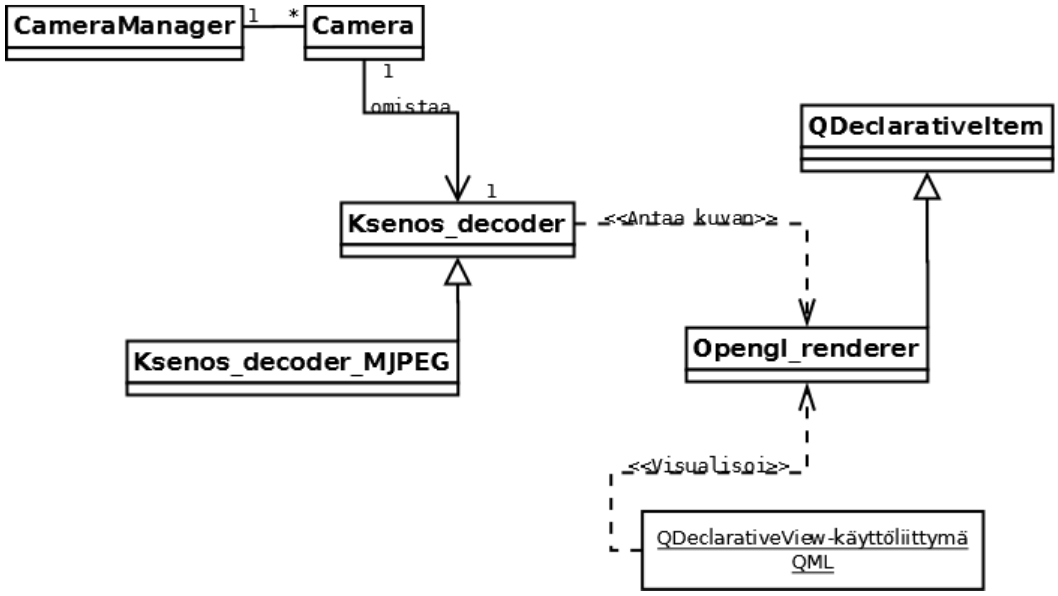
### 3.3.3 Kuvan esitys

Videokuvassa on oleellista, että kuvia voidaan toistaa tehokkaasti ja nopeasti. Tätä varten ohjelmoitiin Opengl_renderer-luokka. Luokan tehtävänä on saada Ksenos_decoder-luokalta kuva, skaalata se oikean kokoiseksi ja esittää se.

Opengl_renderer periytyy QDeclarativeItem-luokasta. QDeclarativeItem on QML-elementtien pääluokka, josta kaikki QtDeclarative-komponentit periytyvät [17]. Käytännössä siis Opengl_renderer on QML-komponentti.

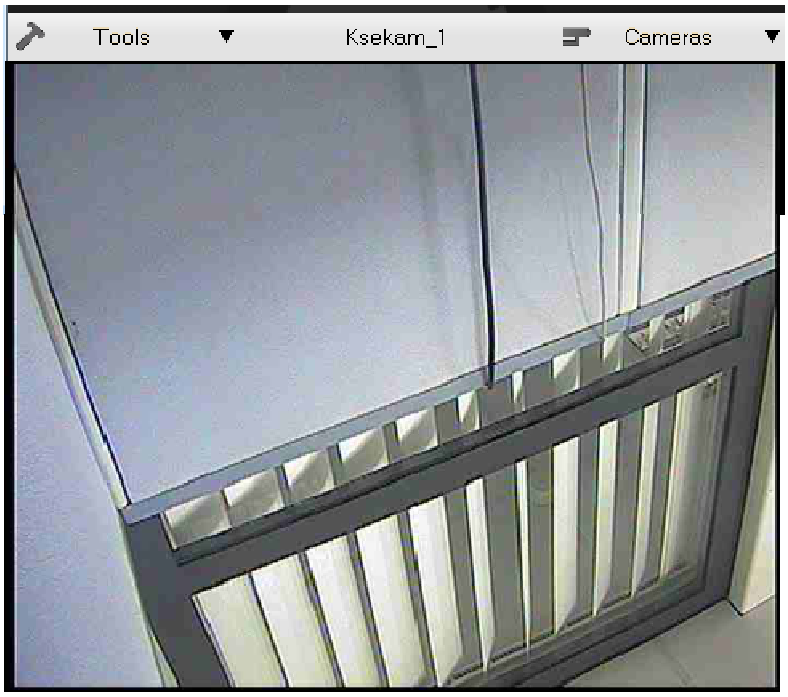
Opengl_rendererille annetaan osoitin Ksenos_decoder-luokkaan setDecoder(Ksenos_decoder *)-metodilla. Opengl_renderer alkaa kuunnella Ksenos_Decoder-luokan signaalia imageReady(QSharedPointer<QImage>). Kun Opengl_renderer saa tämän signaalin, se aiheuttaa slotin updateImage(QSharedPointer<QImage>) suorittamisen. Metodissa otetaan talteen kuvatieto ja kutsutaan metodia update(), joka kertoo grafiikkajärjestelmälle, että tämän olion paint()-metodia pitäisi kutsua. Tämä metodi on ylikuormitettu piirtämään kuva Opengl_renderer-komponentin keskelle.

Opengl_renderer piirtää kuvan käyttämällä OpenGL-rajapintaa. Tällä saavutetaan tehokas kuvanpiirto sekä skaalaus. Kuvassa 10 esitetään kuvan esittämiseen ja käsittelyyn liittyvät luokat sekä niiden yhteydet.




Kuva 10 Videokuvan esittämiseen sekä piirtoon tarkoitetut luokat.

Opengl_renderer piirtää videokuvan mustan alueen keskelle. Sen ympärille tehtiin QML-komponentti nimeltä CameraContext, joka on täysin QML-kielellä määritelty komponentti tarkoituksenaan olla visuaalinen säiliö videokuvalle. Se lisää videokuvan päälle otsikon sekä reunukset. CameraContex tarjoaa myös käyttöliittymän P/T/Z-ohjaukseen sekä käytettävän kameran valitsemiseen. Otsikon vieressä on vasemmalla puolella Tools-valikko, josta voi aktivoida esimerkiksi P/T/Z-ohjauksen. Oikealla puolella on Cameras-valikko, josta voidaan valita, minkä kameran kuvaa näytetään. Kuvassa 11 on yksi CameraContext-QML-komponentti, jossa näkyvät kameran nimi, videokuva sekä yläpalkin napit.




Kuva 11 CameraContext-QML-komponentti.

## 3.4 Testaus

Ksenos Mobilen ohjelmakoodia varten kirjoitettiin yksikkötestejä, jotta voidaan varmistua ohjelmisto-osioiden toimivuudesta. Yksikkötestejä kirjoitettiin metodeille, jotka suorittavat tiedonpurkua, tiedonparsimista sekä laskutoimituksia.

Ohjelmiston toimivuutta testattiin erilaisilla mobiililaitteilla. Ohjelmisto käännettiin tasaisin väliajoin PC-, MAC-, Android- sekä MeeGo-ympäristöille, joissa ohjelmiston toiminnot testattiin. Testilaitteina käytettiin seuraavia laitteita: Windows 8 PC, MACMini, Nokia n950 sekä HTC Desire HD.

Käytettävyystestaus suoritettiin sokkotestinä. Käytännössä ohjelmiston sisältävä laite annettiin henkilölle, joka ei tiennyt ohjelmasta entuudestaan mitään ja kerrottiin mitä hänen tulisi tehdä antamatta varsinaisia käyttöohjeita. Tästä saatiin hyvin paljon informaatiota siitä, miten ihmiset olettavat ohjelman toimivan ja tämän perusteella siihen tehtiin joitain muutoksia.

Esimerkiksi P/T/Z-ohjauksessa jotkin testihenkilöt etsivät zoomaus-nappeja, kun taas enemmän mobiililaitteita käyttäneet henkilöt koittivat zoomata käyttämällä

kahden sormen nipistystä. Tämän testiin perusteella ohjelmaan lisättiin mahdollisuus zoomata kummallakin tekniikalla.

## 3.5 Tietoturva

Ksenos-tallentimen etäkäyttöön liittyvää verkkoliikennettä ei salata mitenkään. Käytännössä tämä tarkoittaa sitä, että mikäli Ksenos Mobilen avulla otetaan yhteys internetin läpi johonkin tallentimeen, teoriassa joku voisi kaapata liikenteestä esim. käyttäjätunnukset. Onneksi videovalvontajärjestelmät eivät yleensä ole suoraan kiinni internetissä. Mikäli tallenninta käsitellään internetin yli, tämä on yleensä suojattu käyttämällä jonkinlaisia VPN-yhteyksiä. Ksenos Mobile ei myöskään käsittele puhelimeen tallennettuja tietoja (yhteystietoja, viestejä), joten ohjelmiston kanssa ei ole vaaraa, että ne päätyisivät vääriin käsiin.

# 4 YHTEENVETO

Tässä työssä oli tarkoituksena suunnitella sekä kehittää mobiiliasiakassovellus Valvova Oy:n kehittämään Ksenos-tallentimeen. Sovelluksen tarkoituksena oli mahdollistaa Ksenos-tallentimeen kytkettyjen kameroiden live-kuvan tarkkaillu mobiililaitteella.

Ohjelmisto jaettiin loogisiin osioihin, jotka ohjelmoitiin ja testattiin. Lopuksi nämä osiot liitettiin yhteen. Osioita oli viisi: verkko, videokuvan purku, sisäiset rakenteet, käyttöliittymä sekä videokuvan esitys.

Ksenos-verkkoprotokollan avulla oli helppo toteuttaa viestiliikenne mobiililaitteen sekä tallentimen välille. Tätä varten tehdyt verkkoluokat toimivat hyvin.

MJPEG-videopakkaus oli positiivinen yllätys. Aluksi kuvavirran pelättiin oleva liian raskasta siirrettäväksi internetyhteyden yli, mutta videokuva toimi kohtalaisesti jopa 1 Mb/s -nopeuksisella internetyhteydellä. Vaikka MJPEG-pakkaus ei ole lähelläkään MPEG-4- tai h.264-pakkausta, on se tähän työhön varsin riittävä. Kuvien skaalaus ja piirtäminen oli aluksi erittäin hidasta, minkä takia kuvien piirtoa varten tehtiin OpenGL-piirtäjä, joka hoiti edellä mainitut tehtävät noin kahdeksan kertaa nopeammin kuin alkuperäinen ratkaisu.

Työn aikana huomattiin, että muilla kuin natiivikehitystyökaluilla mobiilisovelluksen tekeminen ei ole niin yksinkertaista kuin miltä se näyttää. Vaikka Qt:lla tehty ohjelmakoodi toimii hienosti eri ympäristöissä, suurimmat ongelmat tulevat käyttöliittymästä. Käyttöliittymän pitäisi tyydyttää jokaisen mobiiliympäristön käyttäjiä. Käyttäjätestauksen perusteella Ksenos Mobile on onnistunut tässä hyvin. Niin Android-, MeeGo- kuin iOS-laitteiden omistajat osasivat käyttää sovellusta. Vaikka sovellus ei näytä edellä mainittujen ympäristöjen natiivisovellukselta, käyttäjätestauksen perusteella ohjelmistossa on riittävän paljon käyttäjille tuttuja rakenteita.

Lopuksi voisi siis todeta, että ohjelmisto täyttää sille asetetut vaatimukset. Seuraavaksi ohjelmisto tullaan julkaisemaan Googlen Play-kaupassa ja Nokian Ovi-

kaupassa. Applen iOS-laitteille sovellus pyritään julkaisemaan vuoden 2013 loppuun mennessä.

# 5 LÄHTEET


[1] P. Hietanen, C++ ja olio-ohjelmointi, Docendo Finland Oy, 2004.

[2] Digia Oyj, ”Signals and Slots,” Digia Oyj, [www-dokumentti]. Saatavilla: http://qt-project.org/doc/qt-4.8/signalsandslots.html. Luettu: 3.4.2013.

[3] Digia Oyj, ”About us,” Digia Oyj, [www-dokumentti]. Saatavilla: http://qt.digia.com/About-us/. Luettu: 3.4.2013.

[4] Digia Oyj, ”Supported platforms,” Digia Oyj, [www-dokumentti]. Saatavilla: http://qt-project.org/doc/qt-5.0/qtdoc/supported-platforms.html. Luettu: 3.4.2013.

[5] Digia Oyj, ”Getting started,” Digia Oyj, [www-dokumentti]. Saatavilla: http://qt-project.org/doc/qt-5.0/qtdoc/gettingstarted.html. Luettu: 3.4.2013.

[6] Digia Oyj, ”QtQuick Best Practices,” Digia Oyj, [www-dokumentti]. Saatavilla: http://qt-project.org/wiki/Qt_Quick_Best_Practices. Luettu: 3.4.2013.

[7] ”KDE Necessitas project,” necessitas.kde.org, [www-dokumentti]. Saatavilla: http://necessitas.kde.org/. Luettu: 3.4.2013.

[8] T. Turunen, ”Necessitas android port contributed to the Qt-project,” 8 11 2012. [www-dokumentti]. Saatavilla: http://blog.qt.digia.com/blog/2012/11/08/necessitas-android-port-contributed-to-the-qt-project/. Luettu: 3.4.2013.

[9] Accenture, ”Mobile Application Development: Challenges and Best Practices,” [www-dokumentti]. Saatavilla: http://www.accenture.com/SiteCollectionDocuments/PDF/Accenture-Mobile-Application-Development-Challenges-Best-Practices.pdf. Luettu: 3 4 2013.

[10] TechTerms.com, ”Crossplatform,” [www-dokumentti]. Saatavilla: http://www.techterms.com/definition/crossplatform. Luettu: 3.4.2013.

[11] D. Hay, ”10 Solutions for Creating Cross-Platform Mobile Apps,” sixrevisions, [www-dokumentti]. Saatavilla: http://sixrevisions.com/mobile/cross-platform-mobile-apps. Luettu: 3.4.2013.

[12] C. Assire, ”Designing and testing cross-platforms mobile applications,” Smile Benelux, [www-dokumentti]. Saatavilla: http://blog.smile-benelux.com/Designing-and-testing-cross-platforms-mobile-applications. Luettu: 3.4.2013.

[13] Helsingin yliopisto, ”Yksikkötestaamisesta”, [www-dokumentti]. Saatavilla: http://www.cs.helsinki.fi/u/avihavai/edutainment/2011/ohma/7-yksikkotestaamisesta.pdf. Luettu: 3.4.2013.

[14] S. Whatley, ”User interface guidelines for mobile and tablet devices,” [www-dokumentti]. Saatavilla: http://www.simonwhatley.co.uk/user-interface-guidelines-for-mobile-and-tablet-devices. Luettu: 3.4.2013.

[15] M. Maher, ”iOS Device Resolutions Diagram,” [www-dokumentti]. Saatavilla: http://metal-sole.com/2012/01/13/ios-device-resolutions-diagram-ipad-vs-iphone-vs-retina-display.

Luettu: 3.4.2013.

[16] Tek Eye, ”Example List of Android Device Screen Resolutions and Sizes”. [www-dokumentti]. Saatavilla: http://tekeye.biz/2012/example-list-of-android-device-screen-resolutions-and-sizes. Luettu: 3.4.2013.

[17] Digia Oyj, ”QDeclarativeItem Class Reference,” [www-dokumentti]. Saatavilla: http://qt-project.org/doc/qt-4.8/qdeclarativeitem.html. Luettu: 3.4.2013.