OULUN AMMATTIKORKEAKOULU

Ville Viitala

AVOIMEN LÄHDEKOODIN SOVELLUS OHJELMOINNIN OPETUSKÄYTTÖÖN

AVOIMEN LÄHDEKOODIN SOVELLUS OHJELMOINNIN OPETUSKÄYTTÖÖN

Ville Viitala
Opinnäytetyö
Syksy 2014
Tietojenkäsittelyn koulutusohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Web-sovelluskehitys

Tekijä: Ville Viitala
Opinnäytetyön nimi: Avoimen lähdekoodin sovellus ohjelmoinnin opetuskäyttöön
Työn ohjaaja: Sinikka Suutari
Työn valmistumislukukausi- ja vuosi: Syksy 2014 Sivumäärä: 34 + 3

Tämän opinnäytetyön tarkoituksena oli kehittää ohjelmoinnin perusteiden opetuksen avuksi soveltuva sovellus. Ohjelmointikielenä oli C# ja kehitysympäristönä toimi Microsoftin Visual Studio. Sovellus toteutettiin Windows Presentation Foundation-työpöytäsovelluksena.

Teoriaosuudessa käydään läpi C#:n historiaa, ominaisuuksia ja kehitysympäristö Visual Studiota. Olio-ohjelmointi saa oman osuutensa ja ohjelmoinnin opetuksen historiaa ja opetusmetodeja käydään lyhyesti läpi. Myös avoimesta lähdekoodista kerrotaan kattavasti tärkeimmät seikat. Raportin käytännön osuudessa käydään läpi sovelluksen rakennetta, toimintaa ja katselmoidaan miten ohjelma toimii. Kuvituksena olevat lukuisat kuviot auttavat ymmärtämään ohjelmakoodia paremmin.

Sovelluksen ideana on antaa käyttäjälle ohjattavaksi robotti, jota hän pystyy käyttöliittymän kontrollien avulla käskemään tekemään erilaisia toimintoja. Näitä ovat muun muassa liikkuminen alueella, piirtäminen ja piirron eri ominaisuudet kuten piirtoviivan väri, tyyppi ja paksuus. Käyttäjä pystyy myös ohjelmoimaan sovellukseen uusia toimintoja robotille valmiiksi määriteltyjen ominaisuuksien avulla.

Työn tuloksena syntyi helppokäyttöinen ohjelma, jota käyttämällä ohjelmoinnin perusteita opetteleva käyttäjä pääsee tutkimaan WPF-sovelluksen peruselementtejä ja kontrolleja. Hän oppii yksinkertaisen käyttöliittymän avulla tunnistamaan, miten esimerkiksi kontrollien toiminnallisuuden mahdollistavat tapahtumametodit toimivat. Kevyt käyttöliittymä aktivoi käyttäjää tutkimaan sovelluksen toiminnallisuutta tarkemmin. C# ja WPF todistivat loogisella ja toimivalla käytettävyydellään olevansa todella hyvä vaihtoehto Windows-pohjaisten työpöytäsovellusten kehittämiseen.

Asiasanat: C#, WPF, Visual Studio, avoin lähdekoodi, ohjelmoinnin opetus, olio-ohjelmointi

ABSTRACT

Oulu University of Applied Sciences
Degree programme of Business Information Systems
Web Application Development

Author(s): Ville Viitala
Title of Bachelor´s thesis: Open source application for teaching programming
Supervisor(s): Sinikka Suutari
Term and year of completion: Autumn 2014 Number of pages: 34 + 3

The purpose of this thesis was to develop an application to support teaching of programming. Microsoft Visual Studio IDE was used to program the application with C# programming language. Application was developed as Windows Presentation Foundation desktop application.

The theory section consists history and features of C# and Microsoft Visual Studio. Object oriented programming was given its own section and the history and teaching methods of programming are considered an important part of this thesis. Also the principles of Open Source software development are processed in the theory section. The empirical part of the thesis focuses on the knowledge about the application. The part discusses about the application's structure, functionality and how the program works. Many figures are there to help understanding the program code better.

The idea of the application is to give user a robot to control. The user can control the robot and order it to make different functions with the controls on the user interface. The functions are for example moving on the area, drawing and its different features like drawing line's color, type and thickness. You can also program your own new functionality for the robot with the help of robot's existing attributes.

The result of this work was a user friendly application what with you can learn basics of programming by experimenting basic elements and controls of a WPF application. With the simple UI the user can learn how for example the WPF controls work with help of their event methods. The light UI activates the user to explore the functionality of the application more precisely. C# and WPF proved to be with their logical and working usability a great alternative for developing Windows based desktop applications.

Keywords: C#, WPF, Visual Studio, Open Source, teaching programming, object-oriented programming

SISÄLLYS

1 JOHDANTO

Opinnäytetyön aiheena on Avoimen lähdekoodin sovellus ohjelmoinnin opetuskäyttöön ja kyseessä on kvalitatiivinen tutkimus. Työn toimeksiantajana toimii lehtori Liisa Auer Oulun Ammattikorkeakoulusta. Työn metodina on toiminnallinen tutkimus, johon liittyy kehittämistehtävä, jossa luodaan C#-ohjelmointikielellä avoimen lähdekoodin WPF-työpöytäsovellus, jota pystytään käyttämään ohjelmoinnin perusteita opetettaessa hyödyksi. Tavoitteena on syventää omaa ohjelmointiosaamista ja kehittää apukeino ohjelmoinnin opetusta varten. Kehitysympäristönä toimii Microsoftin Visual Studio.

Ohjelmointitaito vaatii ihmiseltä paljon kognitiivisia taitoja kuten päättelykykyä, suunnittelukykyä ja ongelmanratkaisukykyä. Looginen ajattelu korostuu, kun ohjelmoijan täytyy ymmärtää sovelluksen toimintaprosesseja, jäsentää niitä loogisiksi struktuureiksi ja kääntää ohjelman toiminta toimivaksi ohjelmakoodiksi oikealla ohjelmointikielellä. Ohjelmoinnin opettelu ei siis ole yksinkertainen tehtävä. Ohjelmointia on opetettu Suomessa 1960-luvulta lähtien, mutta varsinaisia opetussovelluksia tehtävää auttamaan ei markkinoilla liiemmin ole. Puutteeseen on varmasti tulossa muutos lähivuosina, koska ohjelmoinnin opetus on tulossa pakolliseksi suomalaisiin peruskouluihin vuodesta 2016 lähtien.

Ohjelmoinnin opettamiseen ei ole samanlaisia vakiintuneita pedagogisia sapluunoita kuten esimerkiksi matematiikan opettamiseen. Tässä opinnäytetyössä tutkitaan erilaisia ohjelmoinnin opetuksen metodeja ja niiden hyötyjä. Tämän opinnäytetyön kehittämistehtävänä luotava sovellus on tarkoitettu opetuksen apukeinoksi ohjelmoinnin perusteiden opettamiseen. Oulun ammattikorkeakoulun tietojenkäsittelyn koulutusohjelmassa ohjelmoinnin perusteita on opetettu vuoden 2013 syksystä lähtien C#-ohjelmointikielellä. Tämä määräsi myös kehittämäni sovelluksen ohjelmointikielen, jotta sitä voitaisiin käyttää maksimaalisella tavalla hyödyksi opetuksen lomassa. Visiona oli, että sovelluksessa käyttäjälle annetaan ohjattavaksi robotti, jota hän pystyy käskemään käyttöliittymän kontrollien avulla tekemään erilaisia toimintoja. Näitä toimintoja olisivat muun muassa liikkuminen alueella, piirtäminen ja piirron eri ominaisuudet kuten väri, tyyppi ja paksuus.

Raportin tietoperustassa käsitellään C#-ohjelmointikielen historiaa, ominaisuuksia ja kehitysympäristönä toimivaa Visual Studiota. Olio-ohjelmointiin ja sen ominaisuuksiin tutustutaan lyhyesti. Avoimen

lähdekoodin eli Open Sourcen periaatteita, lisenssejä ja tunnetuimpia sovelluksia käsitellään muutaman sivun verran. Myös ohjelmoinnin opetuksen historiaa, haasteita ja opetusmetodeja käydään kattavasti läpi. Raportin loppupuolella tutustutaan kehittämistehtävänä luodun opetussovelluksen kehitysvaiheisiin tarkemmin.

2 C# JA VISUAL STUDIO

2.1 C#:n historia

C# on Microsoftin kehittämä olio-orientoitunut ja tyyppiturvallinen ohjelmointikieli, joka käyttää hyväkseen .NET Framework-ohjelmistokomponenttikirjastoa. C# on suunniteltu toimimaan yhdessä .NET-alustan kanssa. Microsoftin koodinimi kielen kehitystyölle oli nimeltään COOL (C like Object Oriented Language). Tämä nimi ei päätynyt käyttöön kieltä lanseerattaessa vuonna 2000, vaan se sai nimekseen C#. Kieli pohjautuu vanhempiin C- ja C++-kieliin, mutta siinä on myös paljon samanlaisia ominaisuuksia kuin Javassa, kuten esimerkiksi virtuaalikoneen myötä automaattinen muistinhallinta sekä kattavat luokkakirjastot. C#:lla on tehty pääasiassa Windows-sovelluksia, mutta nykyään sillä luodaan paljon myös web-sovelluksia (ASP.NET) ja Windows Phone-mobiilisovelluksia. Kieltä ohjelmoidaan yleisesti Microsoftin omalla Visual Studio-kehitysympäristöllä. (Software Engineer Insider 2014, hakupäivä 14.7.2014.)

2.2 C#:n syntaksi ja ominaisuudet

C# on vahvasti tyypitetty ohjelmointikieli. Muuttujille on annettava jokin tietotyyppi kuten kokonaisluku tai merkkijono. Kielen syntaksi on hyvin samantyyppistä kuin esimerkiksi C++:ssa tai Javassa, mutta siitä on pyritty tekemään helpommin ymmärrettävää ja yksinkertaisempaa esimerkiksi C++:aan verrattuna. Kielen kehittäjien tavoitteena oli kumminkin säilyttää C:n ja C++:n hyvät ominaisuudet, joten syntaksia muutettiin ja laajennettiin vain, jos siihen nähtiin syytä parannusmielessä. (MSDN 2014, hakupäivä 14.7.2014.)

C# on oliopohjainen kieli, joka tukee muun muassa toimintoja kuten olioita, luokkia ja periytymistä. Myös esimerkiksi kapselointiin, sisäluokkiin ja monimuotoisuuteen löytyy tuki. C++:aan verrattuna C#-koodia ajetaan virtuaalikoneessa, joka hoitaa roskienkeruun eli helpottaa ohjelmoijan työtä automatisoimalla muistinhallintaa. Myös osoittimien tuki löytyy C#:sta, mikä taas puuttuu Javasta. Muita eroja kielten välillä ovat esimerkiksi C#:n tapahtumat (events) ja ominaisuudet (properties), joita C++:ssa ei

ole tuettu. C#:iin on siis tuotu hyviksi todettuja ominaisuuksia sekä C++:sta että Javasta. (MSDN 2014, hakupäivä 14.7.2014.)

```
// Hello1.cs
public class Hello1
{
   public static void Main()
   {
      System.Console.WriteLine("Hello, World!");
   }
}
```

KUVIO 1. Hello World with C#. (MSDN 2014, viitattu 23.7.2014)

```
<Grid x:Name="ContentPanel" Margin="12,0,12,0">
    <Button Height="72" Width="160" Content="Click Me" />
</Grid>
```

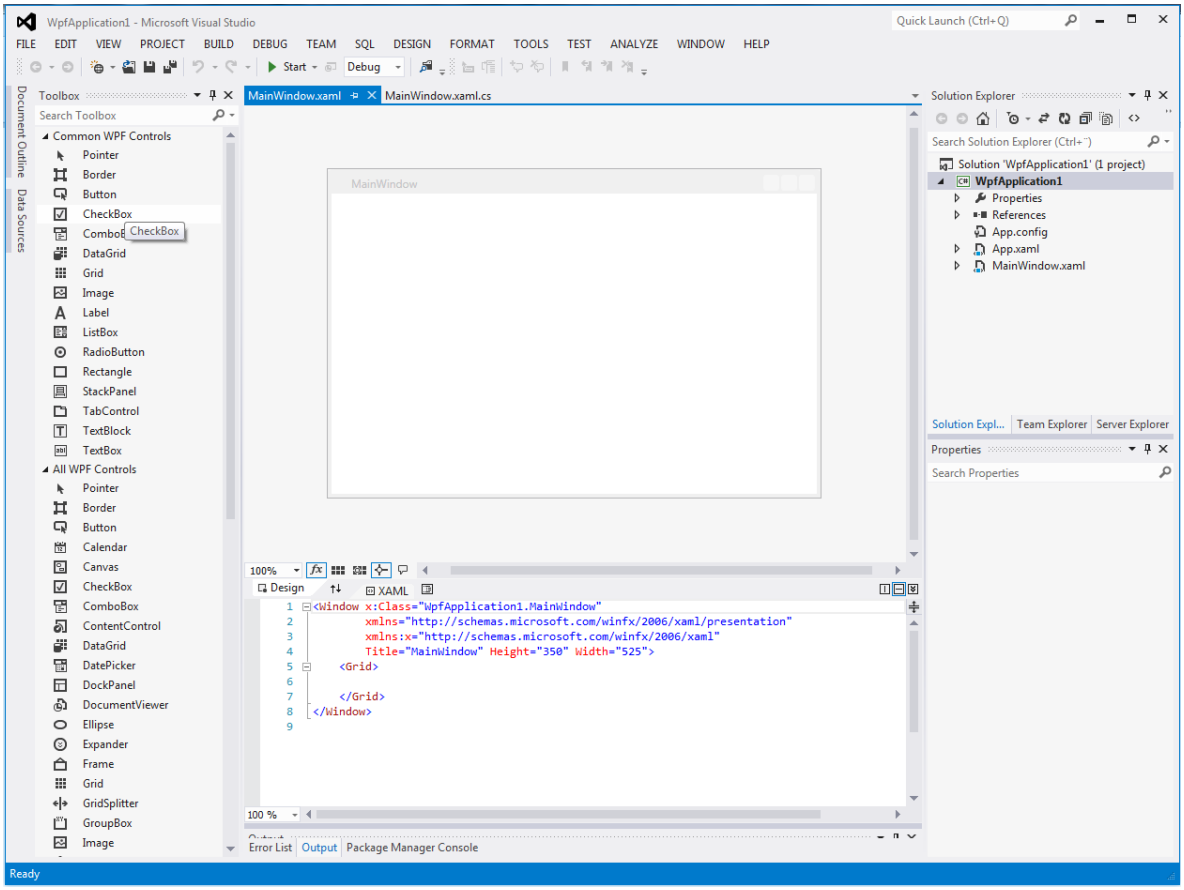
KUVIO 2. Esimerkki XAML:sta. (MSDN 2014, viitattu 23.7.2014)

Microsoft on luonut C#-kehitystä varten myös graafisia ohjelmointirajapintoja helpottamaan graafisten sovellusten tekemistä. Näitä ovat WinForms (Windows Forms) ja WPF (Windows Presentation Foundation), jota käytän opetussovellukseni ohjelmoimiseen. Rajapinnat helpottavat ohjelmoijan työtä huomattavasti tarjoamalla graafisessa käyttöliittymässään esimerkiksi käyttöliittymäelementtejä suoraan raahattavaksi kehitettävään sovellukseen. Näin ohjelmoijan ei tarvitse ohjelmoida kaikkia mahdollisia elementtejä itse. Tämä nopeuttaa huomattavasti kehitystyötä. WPF:ssä käyttöliittymä syntyy ja sitä voidaan muokata XAML-kielellä (Extensible Application Markup Language), joka on Microsoftin kehittämä deklaratiivinen merkintäkieli. (What is WPF? 2014, hakupäivä 22.7.2014.)

2.3 **Microsoft Visual Studio**

Microsoft Visual Studio on Windows-alustoilla toimiva kehitysympäristö, jolla pystyy kehittämään muun muassa Windows-työpöytäsovelluksia, web-sovelluksia ja mobiilisovelluksia. Näitä varten tuki löytyy ohjelmointikielille kuten C#, Visual Basic, C++ ja F#. Kieliä pystyy myös sekoittamaan keske-

nään. Visual Studion ensimmäinen versio Visual Studio 97 lanseerattiin nimensä mukaisesti vuonna 1997. Yhteensä versioita on ilmestynyt yhdeksän kappaletta, joista Visual Studio 2013 on uusin valmis julkaisu. Ohjelmistosta on saatavilla neljä erilaajuista pakettia. Visual Studio Express on ilmainen ja kaikista suppein paketti, kun taas Professional, Premium ja Ultimate ovat maksullisia ja sitä myötä toinen toistaan kattavampia ohjelmistopaketteja. (MSDN 2014, hakupäivä 15.7.2014.)



KUVIO 3. Microsoft Visual Studio 2012 käyttöliittymä – WPF-applikaatio.

Visual Studion työkaluista löytyy muun muassa sisäänrakennettu ohjelmointieditori, joka tukee IntelliSenseä eli ohjelma ehdottaa mahdollisia ominaisuuksia ohjelmoijalle, sekä koodin refaktorointia eli koodin automaattista selkeyttämistä ohjelmiston avulla. Työkaluista löytyy myös debuggeri ohjelmakoodin virheenetsintää varten, käyttöliittymäsuunnittelutyökalu työpöytäsovelluksille sekä suunnittelutyökalut muun muassa web-käyttöliittymille ja tietokannoille. (MSDN 2014, hakupäivä 15.7.2014.)

3 OLIO-OHJELMOINTI

Olio-ohjelmointi (object-oriented programming) on ohjelmointiparadigma, joka yleistyi merkitsevissä määrin 1990-luvulla. Ensimmäinen oliokieli katsotaan syntyneeksi jo 1960-luvulla Simulan muodossa, mutta varsinainen olio-ohjelmointi syntyi 1970-luvun lopulla, kun ohjelmointikieli Smalltalk kehitettiin. Muutama vuosi myöhemmin vuonna 1983 julkaistiin tänäkin päivänä suuressa käytössä oleva C++.

Tällä hetkellä olio-ohjelmointi on vallitseva ohjelmointimalli ohjelmistotuotannossa ympäri maailmaa. Tämän hetken käytetyimmistä ohjelmointikielistä esimerkiksi C++, C#, Objective-C, Java ja Python ovat oliopohjaisia. Olio-ohjelmointi kehitettiin helpottamaan monimutkaistunutta ohjelmistokehitystä mallintamalla reaalimaailmaa havainnollisemmin kuin aikaisemmin vallalla olleessa proseduraalisessa ohjelmoinnissa. Varsinkin suurissa sovellusprojekteissa tämä on tervetullut ominaisuus. (Kompelli, K. 2012, hakupäivä 15.7.2014.)

Olio (object) on olio-ohjelmoinnissa sovelluksen perusyksikkö, joka pystyy käsittelemään ja lähettämään tietoa sekä vastaanottamaan viestejä. Jokaisella oliolla on sovelluksen toiminnan kannalta jonkinlainen rooli tai vastuu. Olio on sovelluksessa yleensä joko abstraktin tai reaalimaailman käsitteen ilmentymä, joka toteuttaa sille määriteltyjä tehtäviä. Koska olioon määritelty koodi on yleensä lyhyt, on se ohjelmoijan näkökulmasta helposti ylläpidettävissä ja ymmärrettävissä.

Olio-ohjelmoinnissa olion yläkäsite on luokka (class), joka määrittelee millaisia ominaisuuksia oliolla on. Tietystä luokasta voidaan tehdä lukuisia olioita, jotka kaikki omaavat samat luokassa määritellyt piirteet. Olio on siis luokan instanssi. Kehittämistehtävänä luomani sovellus pitää sisällään esimerkiksi luokan Robot.cs, johon on ohjelmoitu sovelluksessa toimivan robotin ominaisuudet ja toiminnallisuus. Perintä (inheritance) tekee mahdolliseksi luokan ominaisuuksien kopioimisen suoraan toiselle luokalla. Tämä ominaisuus säästää merkittävissä määrin ohjelmakoodin kirjoittamista. Ohjelmassa voi olla esimerkiksi yksi perusluokka, josta useampi luokka perii ominaisuuksia ja näin ollen näillä kaikilla alemmilla luokilla on samat perusominaisuudet kuin ylemmällä perusluokalla. Joissain ohjelmointikielissä, kuten esimerkiksi C++:ssa, on mahdollista käyttää moniperintää (multiple inheritance),

joka mahdollistaa alemman luokan periä ominaisuuksia useammasta ylemmästä luokasta. C#:ssa ei tätä ominaisuutta ole.

Kapselointi (encapsulation) on myös olio-ohjelmoinnille tyypillinen ominaisuus. Siinä luokan sisäinen toiminnallisuus ja muuttujat eristetään muiden luokkien ulottumattomiin. Kapseloidun luokan tietoja ei pystytä muualta ohjelmasta muuttamaan, mutta sen toiminnallisuuksia pystytään käyttämään hyväksi rajapinnan (interface) kautta. Tällä ominaisuudella pyritään parantamaan ohjelman ylläpidettävyyttä ja vähentämään ohjelmointivirheiden määrää. Muita olio-ohjelmoinnille tyypillisiä ominaisuuksia ovat muun muassa Metodit (method), jotka ovat sovelluksen aliohjelmia ja käsittelevät olion tietoja ja jäsenmuuttujat (variable), joihin tallennetaan tietoja. Olion toiminnan kannalta kriittisen tärkeitä metodeja ovat alustaja (constructor), jota kutsutaan aina olioita luotaessa ja hajottaja (destructor), jota kutsutaan olion tuhoamiseksi. (Seed 2001, 9-23.)

4 OPEN SOURCE – AVOIN LÄHDEKOODI

Termi ”open source” eli avoin lähdekoodi keksittiin vuonna 1998 Kalifornian Palo Altossa. Tähän vaikutti suuresti julkistus, että web-selain Netscapen kehittäjät olivat päättäneet julkaista ohjelman lähdekoodin kaikkien saatavilla. Ohjelmistotuotannossa on aina ollut myös avointa ajatuksenvaihtoa ja yhteistyötä ammattilaisten ja harrastajien kesken, mutta vasta suurten ohjelmistojen lähdekoodien julkaisut (Linux, Netscape) saivat aikaan reaktion, joka synnytti avoimen lähdekoodin käsitteen ja lopulta yleishyödyllisen yhteisön nimeltä OSI (Open Source Initiative), jonka missiona on opettaa ja puolustaa avoimen lähdekoodin hyötyjä. OSI on julkaissut dokumentin nimeltä The Open Source Definition, jossa kerrotaan kriteerit, jotka täyttämällä ohjelmistoa voidaan kutsua avoimen lähdekoodin ohjelmistoksi ja saada open source-sertifioinnin. (OSI 2014, hakupäivä 15.7.2014.)

The Open Source Definitionin suomennos (COSS 2014, hakupäivä 16.7.2014):

1. Ohjelman täytyy olla vapaasti levitettävissä ja välitettävissä.
2. Lähdekoodin täytyy tulla ohjelman mukana tai olla vapaasti saatavissa.
3. Myös johdettujen teosten luominen ja levitys pitää sallia.
4. Lisenssi voi rajoittaa muokatun lähdekoodin levittämistä vain siinä tapauksessa, että lisenssi sallii korjaustiedostojen ja niiden lähdekoodin levittämisen. Voidaan myös vaatia, ettei johdettua teosta levitetä samalla nimellä tai versionumerolla kuin lähtöteosta.
5. Yksilöitä tai ihmisryhmiä ei saa asettaa eriarvoiseen asemaan.
6. Käyttötarkoituksia ei saa rajoittaa.
7. Kaikilla ohjelman käsiinsä saaneilla on samat oikeudet.
8. Lisenssi ei saa olla riippuvainen laajemmasta ohjelmistokokonaisuudesta, jonka osana ohjelmaa levitetään, vaan ohjelmaan liittyvät oikeudet säilyvät, vaikka se irrotettaisiin kokonaisuudesta.
9. Lisenssi ei voi asettaa ehtoja muille ohjelmille. Ohjelmaa saa levittää myös yhdessä sellaisten ohjelmien kanssa, joiden lähdekoodi ei ole avointa.
10. Lisenssin sisällön pitää olla riippumaton teknisestä toteutuksesta. Oikeuksiin ei saa liittää varaumia jakelutavan tai käyttöliittymän varjolla.

4.1 COSS ry

Suomessa avoimen lähdekoodin puolesta puhuu muun muassa voittoa tavoittelematon yhdistys nimeltä COSS ry (Centre for Open Systems and Solutions). Sen toiminta pyrkii edistämään avointa lähdekoodia, avointa dataa, avoimia rajapintoja ja avoimia standardeja. COSS pyrkii toiminnallaan lisäämään tietoisuutta avoimesta lähdekoodista varsinkin julkishallinnossa. Yhdistys myös järjestää koulutuksia, tapahtumia ja verkostoitumismahdollisuuksia lisätäkseen valtakunnallista avoimen lähdekoodin osaamista. COSS:n kannatusjäseneksi voi liittyä kuka tahansa yksityishenkilö. Varsinainen jäsenyys on tarkoitettu esimerkiksi yrityksille sekä tutkimus- ja koulutuslaitoksille. Yhdistyksen jäsenenä on vuonna 2014 noin 100 eri yritystä ja organisaatioita muiden muassa Nokia Networks, Suomen Yrittäjäopisto, Tieke eli Tietoyhteiskunnan kehittämiskeskus ry, Oulun yliopiston tietojenkäsittelytieteiden laitos ja Oulun ammattikorkeakoulun liiketalouden yksikkö. (COSS 2014, hakupäivä 16.7.2014)

4.2 Avoin lähdekoodi

Terminä avoin lähdekoodi tarkoittaa tapaa kehittää ja jakaa tietokoneohjelmistoja. Käyttäjän ei tarvitse ostaa tuotetta tai tuotteen käyttöön oikeuttavaa lisenssiä. Avoimen lähdekoodin ohjelmaa voi vapaasti käyttää, kopioida, muunnella ja jaella. Osaava käyttäjä voi näin ollen ohjelmoida ohjelmaan esimerkiksi uuden ominaisuuden, mikä hänen mielestään on sovelluksesta jäänyt puuttumaan. Avoimien ohjelmistojen kehitysmallissa asiaan kuuluu, että kehityksessä on mukana maailmanlaajuinen yksityishenkilöistä ja yrityksistä koostuva yhteisö. Kaikki kehitystyö on julkista ja kaikki voivat halutessaan osallistua siihen. Tämä mahdollistaa esimerkiksi ohjelmistovirheiden paikantamisen ja korjaamisen nopeasti, mikä taas johtaa hyvällä todennäköisyydellä laadukkaisiin ja hyvin toimiviin ohjelmistoihin. (COSS 2014, hakupäivä 16.7.2014.)

4.3 Open Source-lisenssit

Mikään kehitetty ohjelma ei kumminkaan automaattisesti ole niin sanottu open source-sovellus, vaikka kehittäjä niin olisi tarkoittanutkin. Kaikissa maissa, jotka ovat Bernin sopimuksen piirissä, ohjelmakoodilla on tekijänoikeussuoja, eikä suojattua teosta saa levittää tai muokata ilman tekijän suostu-

musta. Tätä varten on kehitetty avoimen lähdekoodin lisenssit, joita on lukematon määrä. Lisenssit saattavat rajoittaa paljonkin ohjelmiston käyttöä, mutta OSI:n hyväksymistä lisensseistä kaikki sallivat avoimen lähdekoodin periaatteiden mukaisesti ohjelmiston vapaan käytön, muokkauksen ja jakamisen. Tunnetuimpia ja eniten käytettyjä lisenssejä ovat esimerkiksi GPL-, BSD-, MIT- ja Apache-lisenssit. Muun muassa Linux-kernelin lisenssinä toimiva GPL (GNU General Public License) on erittäin laajasti käytössä ja esimerkki käyttäjänoikeus- eli copyleft-lisenssistä. Tämä tarkoittaa, että lisensoidusta ohjelmasta johdetut uudet sovellukset ovat automaattisesti kaikki myös saman lisenssin alaisuudessa. GPL-lisensoidun ohjelman pohjalta ei siis esimerkiksi saa kehittää kaupallista sovellusta, josta lähdekoodi on piilotettu tai lisenssiä on muuten muutettu, vaan lähdekoodin tulee olla saatavilla ja lisenssin pysyä samana, jos kyseistä ohjelmaa haluaa muokata ja levittää. GPL:n pääehdot ovat, että lisenssin täytyy pysyä samana, muokkauksista pitää olla maininnat, ohjelmassa täytyy olla tieto lisenssistä ja johdannaiset sovellukset täytyy lisensoida samalla lisenssillä. (Laakkonen, C. 2014, hakupäivä 16.7.2014.)

Muista suosituimmista lisensseistä MIT ja BSD ovat erittäin sallivia. Ne eivät ole copyleft-lisenssejä, joten ne sallivat teoksen hyväksikäytön myös suljetun lähdekoodin ohjelmistoissa. MIT-lisenssi yksinkertaisuudessaan antaa oikeudet ohjelman vapaaseen käyttöön, muokkaamiseen, kopioimiseen ja julkaisemiseen, kunhan lisenssiteksti säilyy ohjelman lähdekoodissa. Kehittämistehtävänä luomani sovellus on lisensoitu MIT-lisenssillä, jotta sitä voidaan jatkossa kehittää vapaasti esimerkiksi muiden opiskelijoiden toimesta. BSD-lisenssin ehdot ovat käytännössä samat: ohjelmiston kaikenlainen vapaa käyttö sallitaan, kunhan lisenssin teksti on säilytetty lähdekoodissa. (Laakkonen, C. 2014, hakupäivä 16.7.2014.)

4.4 **Tunnetuimmat avoimen lähdekoodin sovellukset**

Avoimeen lähdekoodiin perustuvan kehitystyön pohjautuessa avoimuuteen ja yhteisöllisyyteen, tämän hetken suurimpia ja suosituimpia avoimen lähdekoodin ohjelmistoja on ollut tekemässä lukematon määrä kehittäjiä. Pelkästään Mozillan tuotteita on ollut mukana tekemässä tuhansia vapaaehtoisia ihmisiä. Trendiä ei selitä pelkkä kehittäjien harrastaneisuus, vaan usein avoimen lähdekoodin ohjelmisto suunnitellaan korvaamaan jokin kaupallinen, jopa monopoliaseman saavuttanut, ohjelmisto. Paras esimerkki tästä on Linus Torvaldsin alun perin kehittämä Linux (GPL-lisenssi), jonka pohjal-

ta on kehitetty monia käyttöjärjestelmiä vallitsevan Microsoftin Windows-käyttöjärjestelmän korvaajaksi tai vaihtoehdoksi. Muita tämän hetken tunnetuimpia open source-ohjelmistoja ovat muun muassa Mozilla Firefox-selain (MPL-lisenssi), toimisto-ohjelmistopaketti Open Office ja sen jälkeläinen Apache OpenOffice (Apache-lisenssi), relaatiotietokantaohjelmisto MySQL (GPL-lisenssi), kuvankäsittelyohjelma GIMP (GPL- ja LGPL-lisenssit) ja http-palvelinohjelma Apache http Server (Apache-lisenssi). (Laakkonen, C. 2014, hakupäivä 16.7.2014.)

5 OHJELMOINNIN OPETUS

5.1 Tietotekniikan ja ohjelmoinnin opetuksen historia Suomessa

1960-luvulla atk eli automaattinen tietojenkäsittely alkoi yleistyä Suomessa. Haluttiin turvata teollisuudelle tarvittavan työvoiman saatavuus, joten yliopistoihin täytyi saada alan oppiaine. Yliopistoissa oli jonkin aikaa tarjottu atk-kursseja matematiikan ja fysiikan opiskelijoille, mutta varsinaisen tietojenkäsittelyopin opetus aloitettiin ensimmäisenä Suomessa Tampereen yliopistossa vuonna 1965. Helsingin yliopistossa opetus aloitettiin vuonna 1967. Toisen asteen koulutuksiin tietotekniikkakoulutus jalkautui vuonna 1969, kun atk-merkonomien koulutus aloitettiin.

1960- ja 1970-lukujen tietokoneet olivat valtavia ja kalliita yksiköitä, joita hankittiin muutamia kappaleita yliopistoa kohden. Yliopistossa opetettavat ohjelmointikielet määräytyivät käytössä olleen keskustietokoneen mukaan. Helsingin yliopiston opetettavia ohjelmointikieliä noihin aikoihin olivat FORTRAN, Algol ja COBOL. 1970-luvulla opetus oli jaettu kolmeen päälinjaan, jotka olivat ohjelmoinnin teoria ja systeemiohjelmointi, hallinnollinen tietojenkäsittely ja systeemianalyysi. 1980-luvun alussa Helsingin yliopiston tietojenkäsittelytieteen laitos sai omia pienempikokoisia mikrotietokoneita eri käyttöjärjestelmillä, joiden myötä aloitettiin esimerkiksi ohjelmointikieli C:n opetus. (Tienari, M. 2014, hakupäivä 17.7.2014.)

Tietotekniikan opetus tuli peruskouluihin ja lukio-opetukseen 1980-luvulla. Vuosikymmenen alkupuolella opetus tapahtui lähinnä atk-kerhoissa ja mahdollisina koulukohtaisina valinnaisaineina. Varsinaisen peruskoulun valinnaisaineen aseman tietotekniikka sai vuonna 1984. Opetussisältö koostui neljästä osa-alueesta, jotka olivat tietotekniikan perusteet, työvälineohjelmat, musiikki ja kuvankäsittely ja ohjelmointi. Ohjelmointikieli saattoi olla esimerkiksi Pascal, C tai Logo. Lukioissa atk-opetus alkoi vuonna 1982 ja 1984 se korvattiin tietotekniikalla, joka oli valinnaisaine. 1990-luvulle tultaessa oppiaineiden integroinnista tuli uusi ilmiö. Tämä tarkoitti sitä, että esimerkiksi tietoteknisiä taitoja voitaisiin oppia muiden oppiaineiden yhteydessä. Vuonna 1994 opetushallitus vahvisti uuden opetussuunnitelman perusteet, joissa tietotekniikka määrättiin paremmin integroitavaksi muun opetuksen sekaan.

Mahdollisuus tietotekniikan opettamiseen valinnaisaineena kumminkin jätettiin. (Jyväskylän yliopisto 2014, hakupäivä 17.7.2014.)

2000-luvulla opetus- ja kulttuuriministeriö on ottanut missiokseen kehittää ja pitää Suomi tietoyhteiskuntien kärkiosastossa. Se on ideoinut useamman kehitysohjelman, kuten Koulutuksen ja tutkimuksen tietostrategia 2000–2004, Koulutuksen ja tutkimuksen tietoyhteiskuntaohjelma 2004–2006 ja Tietoyhteiskuntastrategia 2007–2015, joilla on pyritty nostamaan ja pitämään suomalaisten osaaminen tieto- ja viestintätekniikan saralla maailman huipulla. Nykytilanne tietotekniikan opetuksessa on hyvällä mallilla. Korkeakouluissa löytyy paljon opetustarjontaa ja uusia asiantuntijoita syntyy hyvää vauhtia. Peruskouluissa tietotekniikka on tällä hetkellä yksi suosituimpia valinnaisaineita. Vuonna 2013 opetushallitus on linjannut, että tieto- ja viestintäteknisen osaamisen painotusta lisätään kouluissa huomattavasti. Ohjelmoinnista ei itsestään ole tulossa ainakaan vielä omaa oppiainetta, mutta peruskoulujen uuteen opetussuunnitelmaan (OPS 2016) on sisällytetty ohjelmoinnin opetusta ensimmäisestä luokasta aina yhdeksänteen luokkaan asti. (Opetus- ja kulttuuriministeriö 2014, hakupäivä 17.7.2014.)

5.2 **Ohjelmoinnin opetusmetodeja**

Ohjelmointitaito vaatii ihmiseltä paljon kognitiivisia taitoja kuten päättelykykyä, suunnittelukykyä ja ongelmanratkaisukykyä. Looginen ajattelu korostuu, kun ohjelmoijan täytyy ymmärtää sovelluksen toimintaprosesseja, jäsentää niitä loogisiksi struktuureiksi ja kääntää ohjelman toiminta toimivaksi ohjelmointikoodiksi oikealla ohjelmointikielellä. Tämän takia ohjelmoinnin opetuksen kolme tärkeintä tavoitetta tulisi olla ohjelmointikielen syntaksin opettaminen, ohjelman suunnittelutaidon kehittäminen ja luovan ajattelun kehittäminen.

Ihmisen oppimistyylit ovat hyvin yksilöllisiä. Joku on auditiivinen oppija eli oppii parhaiten kuuloaistin avulla. Toinen on visuaalinen oppija, jolla oppi menee parhaiten perille näkemisen kautta. Kolmas on kinesteettinen oppija, joka oppii parhaiten tekemällä opetettavaa asiaa itse. Oppimisen kannalta tärkeitä asioita ovat myös esimerkiksi motivaatio, itseluottamus ja mielihyvän tunne. Jos näistä kolmesta joku ei ole oppijalla kunnossa, tai hän ei esimerkiksi saa opiskelustaan minkäänlaisia onnistumisen tunteita, oppiminen vaikeutuu huomattavasti. Ohjelmoinnin opetuksessa tulisi ottaa huomioon nämä

seikat ja miettiä miten opiskelija saadaan stimuloitua pitämään hyvä motivaatio yllä esimerkiksi mielenkiintoisilla tehtävillä. Ohjelmointitehtävät ovat muutenkin ratkaisevassa roolissa opiskelijan oppimisen kannalta. Opettajan tehtävä on pedagogisesti todella haastava, koska hänen pitäisi pystyä löytämään tehtäville sopivat vaikeustasot, jotta opiskelija joutuu todella miettimään tehtävän ratkaisua ja täten kehittämään ongelmanratkaisukykyään. Samalla tehtävät täytyisi pitää haastavuudeltaan sellaisella tasolla, että opiskelija saisi itseluottamuksensa kannalta tärkeitä onnistumisen tunteita. Juuri se kantaa opiskelijan paneutumaan aina seuraavaan haasteeseen täydellä innolla ja näin oppimiskyky pysyy tehokkaana.

Opetusmetodilla tarkoitetaan menettelytapaa, jolla opettaja vie opetettavan asian oppilailleen. Tietojenkäsittelytieteissä ei ole vielä yhtä vakiintuneita opetusmetodeja kuin esimerkiksi matematiikan saralla, mutta jonkin verran tieteellisesti testattuja metodeja on tarjolla. Yksi tapa määritellä opetusmetodia on miettiä mihin opetus liittyy ja mitkä sen tavoitteet ovat. Opetusta voidaan lajitella esimerkiksi näkökulman mukaan, kuten ohjelmistoteknologian, ohjelmointikielen, ohjelmointi työkaluna tai tehtäväorientoitumisen mukaan. Esimerkiksi tehtäväorientoituneissa metodeissa voidaan esitellä käytännön ongelmia, joita varten opetetaan ohjelmointikielen osa-alueita siinä järjestyksessä missä niitä tarvitaan, jotta ongelma saadaan ratkaistua loogisesti. Ohjelmointi työkaluna-metodit taas esittelevät ohjelmoinnista tai ohjelmointikielestä vain jonkin tietyn osa-alueen, jonka osaamisella pystytään hoitamaan jotain muita tehtäviä jonkin muun opetettavan alueen sisällä. Opetusta voidaan jaotella myös opetettavan asian järjestyksellä, kuten jaottelu tyyliin alhaalta-ylös ja ylhäältä-alas. Alhaalta-ylös-metodissa keskitytään opettelemaan ohjelmointikielen syntaksia ja sen yksityiskohtia, ennen kuin siirrytään haastavampiin ja suurempiin kokonaisuuksiin. Ylhäältä-alas-metodissa toimitaan päinvastoin. Ensin hahmotetaan kokonaisuuksien toimintaa abstraktilla tasolla, ennen kuin siirrytään opettelemaan tarkemmin syntaksin yksityiskohtia.

Kaikki ohjelmoinnin opetusmetodit tähtäävät opetuksen helpottamiseen ja laadun parantamiseen, jotta ohjelmoinnin konseptit olisivat helpompia oppia. Yhteinen nimittäjä kaikille metodeille on opiskelijoiden motivaation ylläpito ja rohkaiseminen, jotta turhautuminen ja motivaation lasku eivät pääsisi yllättämään opiskelijan kohdatessa vaikeuksia yrittäessään ymmärtää ohjelmoinnin haastavia konsepteja. Metodeilla rohkaistaan opiskelijoita kohtaamaan ongelmat ja sitoutumaan paremmin annettuihin tehtäviin. Opettajan vastuulla on valita sopivat opetusmetodit tai tehdä niistä omat yhdistelmän-

sä, joilla hän saa motivoitua opiskelijoita parhaiten sitoutumaan suunniteltuihin tehtäviin ja toimeksiantoihin. (Mohorovicic, S. & Strcic, V. 2011, hakupäivä 18.7.2014.)

5.2.1 Ongelmaperustainen oppiminen

Ongelmaperustaisessa oppimisessa (Problem-based learning) keskitytään opiskelijan omaan haluun ratkaista ongelmia. Metodissa on tarkoituksena tarjota opetetun tiedon perusteella ratkaistavaksi ongelmia, joita vastaavia todennäköisesti tulee vastaan ohjelmoijan tehtävissä työelämässä. Opiskelijalle annetaan rooli aktiivisena ongelmanratkaisijana, mikä kehittää loogista ja luovaa ajattelukykyä, käytännön taitoja ja luonnollisesti ongelmanratkaisukykyä. Monet ongelmat pyritään ratkaisemaan ryhmissä. Ryhmät tunnistavat ongelmat, tekevät listan mitä he tehtävästä tietävät ja mitä heidän täytyy vielä tietää, minkä jälkeen he tunnistavat oppimiskohteet ja opiskelevat itsenäisesti näiden kohteiden oppimiseksi. Lopuksi opiskelijat kerääntyvät jälleen yhteen keskustellakseen oppimastaan ja tehdäkseen yhteenvedon lopputuloksistaan ongelman ratkaisemiseksi. Ongelmaperusteisen oppimisen metodia on myös sovellettu esimerkiksi niin, että ongelmat jaetaan vasta oppituntien päätteeksi ja oppilaat saavat miettiä ratkaisujaan itsenäisesti. (Mohorovicic, S. & Strcic, V. 2011, hakupäivä 18.7.2014.)

5.2.2 Palapelioppiminen

Palapelioppiminen (Puzzle-based learning) tähtää opiskelijan kokonaisuuden hahmottamisen ja ongelmanratkaisukykyjen kehittämiseen. Metodissa opiskelijalle annetaan esimerkiksi toimivan ohjelman lähdekoodista koodirivejä, jotka hänen täytyy koota oikeaan järjestykseen, jotta ohjelma toimisi opettajan esittämällä tavalla. On tutkittu, että ohjelmoinnin perusteita opetettaessa metodi aktivoi oppilaita osallistumaan enemmän opetukseen. (Mohorovicic, S. & Strcic, V. 2011, hakupäivä 18.7.2014.)

5.2.3 Pariohjelmointi

Pariohjelmoinnissa (Pair programming) kaksi ohjelmoijaa työskentelee yhdessä saman tietokoneen ääressä. He kummatkin osallistuvat sovelluksen kehittämiseen, mutta roolit vaihtuvat tasaisin väliajoin. Toinen ohjelmoija (driver, controller) toimii kuskina kirjoittaen ohjelmakoodia ja toinen (navigator, co-pilot) tarkkailee vierestä koodissa olevia mahdollisia virheitä antaen myös ehdotuksia mahdollisiksi parannuksiksi. On tutkittu, että pariohjelmoinnin seurauksena syntyvä ohjelma valmistuu nopeammin ja sisältää vähemmän virheitä. Hyödyistä huolimatta pariohjelmointi ei sovi kaikille. Jotkut kokevat sen puuduttavaksi ja ärsyttäväksi. Paljon riippuu työparien luonteesta, miten he työskentelevät yhteen. Toinen pariohjelmoinnin toimivuuden kannalta merkittävä seikka on, että työparin tasoero ei saa olla liian suuri. Heidän täytyy olla suurin piirtein samantasoisia ohjelmoijia. Tämä helpottaa suuresti yhdessä työskentelyä. Virtuaalinen pariohjelmointi on myös mahdollista. Tällöin työparin ei tarvitse olla saman tietokoneen ääressä vierekkäin, jos kehitysympäristö sen sallii, vaan he voivat kommunikoida reaaliaikaisesti ja työskennellä ohjelmakoodin parissa omilta päätteiltään. (Mohorovicic, S. & Strcic, V. 2011, hakupäivä 18.7.2014.)

5.2.4 Nauhoitetut luennot

Nauhoitetut luennot (Prerecorded lectures) on opetusmetodi, joka on suunniteltu täydentämään muuta tavallista opetusta. Opettaja tekee esimerkiksi online-opetusympäristöön saatavaksi multimedianauhoituksia, jotka ovat huolellisesti suunniteltuja ja rajattu tiettyyn opetusalueeseen. Koska korkeakouluissa kurssien lähiopetuksen tuntimäärät ovat rajalliset, nauhoitetut luennot antavat opettajalle mahdollisuuden syventää tai kerrata opetusta. Tämä auttaa opiskelijoita kertaamaan oppimaansa, varsinkin jos opiskelija ei ole omaksunut opetettua asiaa vielä luennolla tai on ollut poissa kokonaan. Kertauksen kannalta metodi on hyvä, koska nauhoitettuun luentoon voidaan pelkistää opetettava asia ja rajata vähempi tärkeä aines pois. Opettajalle kertyy vuosien saatossa kokemusta, missä aihealueissa oppilaat monesti omaavat ongelmia ja hän voi varautua tähän tekemällä esimerkiksi näistä aihealueista nauhoitettuja luentoja. (Mohorovicic, S. & Strcic, V. 2011, hakupäivä 18.7.2014.)

5.2.5 Peliteemainen ohjelmointi

Peliteemainen ohjelmointi (Game-themed programming) on opetusmenetelmä, jossa esimerkiksi ohjelmoinnin perusteita opetettaessa opetusmateriaalina on interaktiivisia pelejä. Tarkoituksena on, että opiskelijat oppivat abstrakteja ohjelmointikonsepteja ohjelmoimalla pieniä pelisovelluksia. Tarkoitus ei ole opettaa opiskelijoille miten tietokonepelejä tehdään, vaan opettaa heille ohjelmoinnin käsitteitä ja logiikkaa pelin tekemisen kautta. Opetusmetodissa siis korvataan opetuksessa tehtävinä käytettävät niin sanotut normaalit sovellukset pelisovelluksilla. Pelien tekeminen motivoi opiskelijoita paremmin miettimään ja ratkaisemaan ohjelmointiongelmia ja näin ollen aktivoi heitä oppimaan innokkaammin ohjelmoinnin perusteita. (Mohorovicic, S. & Strcic, V. 2011, hakupäivä 18.7.2014.)

5.3 Ohjelmoinnin opetuksen haasteet

Ohjelmoinnin opettelu on haastava tehtävä. Korkeakoulujen opiskelijat kohtaavat monia ongelmia aloittaessaan ohjelmoinnin opiskelun ohjelmoinnin perusteiden kursseilla. Benedict du Boulay kirjoitti vuonna 1986 artikkelissaan Some Difficulties of Learning to Program, että aloittelevalla ohjelmoijalla on vastassaan viisi limittäistä pääongelma-aluetta aloittaessaan ohjelmoinnin opintojaan. Ne ovat sopeutumisen ongelma (problem of orientation), kuvitteellinen kone (notional machine), ohjelmointikielten notaatio (notation of formal languages), rakenteiden oppiminen (acquiring structures) ja ohjelmoinnin käytännöt (pragmatics of programming).

Sopeutumisen ongelmalla tarkoitetaan esimerkiksi, miksi ylipäänsä on hyödyllistä opiskella ohjelmointia. Opiskelijan täytyy ymmärtää ohjelmien pääidea ja miksi niitä tarvitaan. Notional machine eli kuvitteellinen kone tarkoittaa idealisoitua abstraktiota tietokoneen laitteistosta ja ajoympäristöstä. Sen idea on tietää miten ajatella mitä tietokone ohjelmaa suorittaessa tekee. Opiskelijan näkökulmasta ongelmallista on oppia ymmärtämään tietokoneen fyysisten laitteiden ja ohjelmien suhdetta ja keskustelua keskenään. Esimerkiksi muistinhallinta ja –käsittely sitä vaativissa ohjelmointikielissä on yksi haaste. Ohjelmointikielten notaatiolla tarkoitetaan kielen syntaksin opiskelua. Rakenteiden oppimisella tarkoitetaan ohjelmointikielen eri ominaisuuksien ja toiminnallisuuksien opiskelua. Esimerkiksi toistorakenteet, perintä, metodit tai dynaaminen muistinkäyttö voivat olla haastavia oppia. Ohjelmoinnin

käytännöillä tarkoitetaan esimerkiksi tietokoneohjelman suunnittelua, määrittelyä, kehitystä, testausta ja debuggausta.

Myös opetuksen haastavuutta opettajien näkökulmasta on tutkittu. Nell B. Dalen tutkimuksessa vuodelta 2006 opettajilta kysyttiin opetusaiheita mitkä he kokivat haastaviksi opettaa. Tutkimustulos osoitti neljä kategoriaa: ongelmanratkaisu ja suunnittelu, yleiset ohjelmointiaiheet, oliokäsitteet ja opiskelijan kypsyys. Ongelmanratkaisu ja suunnittelu sisälsi kolme alakategoriaa: yleinen ongelmanratkaiseminen, algoritmien suunnittelu/kehitys ja olioihin liittyvä ongelmanratkaiseminen ja suunnittelu. Yleisten ohjelmointiaiheiden alakategorioita olivat muun muassa parametrit, arrayt eli listat, toistorakenteet, pointterit eli osoittimet, tiedostot ja ehtolauseet. Oliokäsitteet sisälsi muun muassa aiheita kuten monimuotoisuus, perintä, instanssimetodit, instanssimuuttujat ja staattiset muuttujat. Neljäs kategoria, opiskelijan kypsyys, piti sisällään yleisiä opiskelutaitoja kuten ajanhallinta, opiskelemaan oppiminen, kovan työn tärkeys, suunnittelu ennen ohjelmointia, itsekuri, luennoille osallistuminen ja hyvät työskentelytavat. Myös esimerkiksi ohjelman testaus ja debuggaus koettiin vaikeaksi opettaa.

Opiskelijat ja opettajat kokevat ohjelmoinnin opetuksen saralla monet osa-alueet haastaviksi. Opettajilla on kumminkin olemassa työkaluja auttaa ja rohkaista opiskelijoita tekemään parhaansa oppiakseen vaikealtakin tuntuvat aiheet. Panostus opiskelijoiden motivoimiseen auttaa heitä pitämään tarvittavaa kiinnostusta yllä vaikeankin aiheen parissa. Valitut opetusmetodit ja pedagogiset aktiviteetit voivat auttaa opiskelijoita parantamaan ajanhallintaansa, itsearviointi- ja tiedonetsintäkykyä. Valitsemalla selkeitä opetuskokonaisuuksia, käyttämällä sopivia opetusmetodeja ja antamalla aikaa selittää ja näyttää esimerkkejä voi auttaa opiskelijoita teorian ja vaikeiden konseptien ymmärtämisessä, kuin myös niiden tietojen jalkauttamisessa käytäntöön ohjelmoinnin parissa. Kaikkeen opettaja ei kumminkaan pysty. Jos opiskelija ei itse ole aktiivinen ja halukas oppimaan, ei opettajalla jää paljoa tehtäväksi tällaisen opiskelijan avuksi. (Kinnunen, P. 2009, hakupäivä 21.7.2014.)

Tämän opinnäytetyön kehittämistehtävänä luotu sovellus on suunniteltu helpottamaan ohjelmoinnin opettamista C#-ohjelmointikielellä. Se on työkalu, jolla opettaja voi esimerkiksi näyttää ja opettaa käyttämään WPF-sovelluksen eri komponentteja ja kontrolleja. Kun opiskelijalla on perusasiat kunnossa, opettaja voi luoda sovellukseen perustavia harjoitustehtäviä opiskelijoille ratkaistavaksi ja käyttää opetuksen apukeinoina ja tukena esimerkiksi aiemmin lueteltuja opetusmetodeja.

6 OPETUSSOVELLUKSEN TOTEUTUS

6.1 Lähtökohta ja tavoite

Idea ja toimeksianto opetuskäyttöön soveltuvasta sovelluksesta tulivat Oulun ammattikorkeakoulun Lehtori Liisa Auerilta. Sovellusta on tarkoitus käyttää esimerkiksi Oamkin liiketalouden yksikön tietojenkäsittelyn koulutusohjelmassa ohjelmoinnin perusteita opetettaessa. Kyseisessä yksikössä ohjelmoinnin perusteiden opetuskielenä on syksystä 2013 käytetty C#-ohjelmointikieltä. Tämän vuoksi kehittämäni WPF-työpöytäsovelluksen tuli olla myös ohjelmoitu C#:lla, jotta sitä voitaisiin käyttää maksimaalisella tavalla hyödyksi opetuksen lomassa. Kehitysympäristöksi valikoitui Microsoftin Visual Studio, koska se on käytössä myös Oamkin liiketalouden yksikössä. Opetussovellukseni on avoimen lähdekoodin sovellus (MIT-lisenssi), joten sen jatkokehittäminen Oamkin tai jonkin muun tahon toimesta onnistuu helposti.

6.2 Sovelluksen suunnittelu

Opetussovelluksen suunnittelu aloitettiin kahdella palaverilla Liisa Auerin kanssa. Palavereiden pohjalta syntyneenä visiona oli tuottaa sovellus, jossa käyttäjä voi antaa käskyjä kentällä toimivalle objektille, joka visioitiin robotiksi. Sovelluksen ikkuna jaetaan kahteen osaan, jossa suuremmassa osassa on robotin liikkumisalue ja pienemmästä osasta löytyy kontrollit, millä käyttäjä antaa käskyjä robotille. Kontrolleja ovat muun muassa liikkumisen suunta ja pituus, piirto-ominaisuus ja piirron paksuus, väri ja tyyppi. Piirsin tämän suunnitelman avulla sovelluksen ulkoasusta yksinkertaisen rautalankamallin, jonka pohjalta itse kehitystyö pystyttiin laittamaan alulle.

6.3 Sovelluksen toteutus

Ensimmäiseksi lähdin toteuttamaan sovelluksen ikkunoita ja elementtien asettelua. Rautalankamallin mukaisen kaksiosaisen asetelman tein sisäkkäisillä stackpanel-elementeillä. Robotin kentän pohjana toimii canvas-elementti, jonka päällä robotti pystyy toteuttamaan esimerkiksi piirto-ominaisuuksiaan.

```
<Window x:Class="LearnToCSharp.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="800" Width="1080" ResizeMode="NoResize">
    <StackPanel Orientation="Horizontal" Background="#FFBDBCBC">

        <StackPanel Name="stackArea"  Margin="5" Background="White" Width="800" Height="800" >
            <Canvas Name="canvasArea">
                <Image x:Name="imageRobot"></Image>
            </Canvas>
        </StackPanel>
```

KUVIO 4. Sovelluksen asettelun alkupää XAML-kielellä.

Sovelluksen perusasettelun oltua kohdallaan, lähdin tuomaan toiminnallisia elementtejä kontrollit-osioon sovelluksen ikkunan oikealle laidalle. Otsikointi ja kaikki irrallinen teksti tehtiin labeleilla. Kontrolleiksi tulivat liikkumisen (Move) suunta ja askeleet (Direction and Steps), piirron valinta (Draw Line), piirtoviivan väri (Line Color), piirtoviivan tyyppi (Line Type) ja piirtoviivan paksuus (Line Thickness). Kontrollityypeiksi valikoituivat pudotusvalikko (combo box), valintanappi (radio button), liukusäädin (slider) ja painike (button).

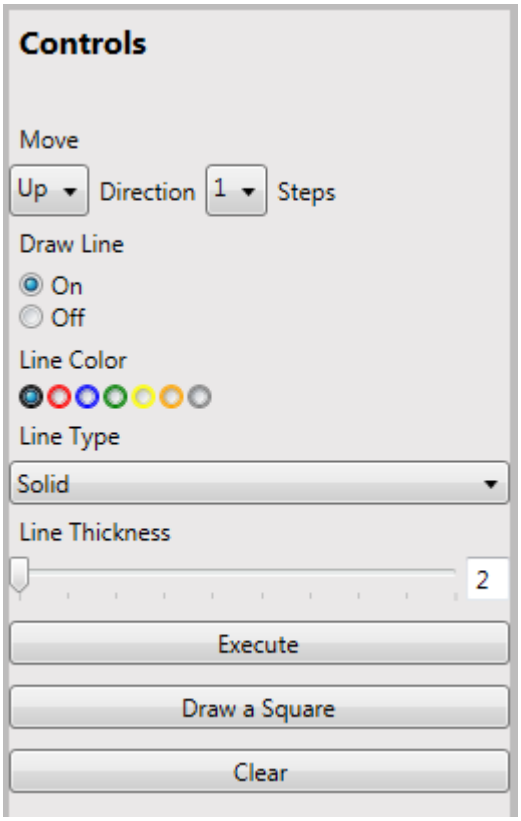
Liikkumisen pudotusvalikoista löytyy muutamia valittavia vaihtoehtoja. Liikkumissuunnan comboboxista löytyy vaihtoehdot ylös, alas, vasemmalla ja oikealle (Up, Down, Left, Right). Toisesta löytyy liikkumismäärän vaihtoehdot 1-6. Käyttäjä valitsee näistä pudotusvalikoista siis robotin liikkumissuunnan ja kuinka monta askelta robotti valittuun suuntaan liikkuu. Piirtovalinta tehtiin kahdella radiobuttonilla yksinkertaiseen tyyliin: päälle/pois (On/Off). Robotin piirto-ominaisuus piirtää päällä ollessaan valitunlaisen viivan canvas-elementtiin robotin liikkeiden mukaan. Robotti siis jättää jälkeensä mahdollisia erityyppisiä viivoja liikkuessaan ympäri aluetta. Piirtoviivan värivalinta toteutettiin myös radiobuttoneilla. Värivaihtoehtoja tuli yhteensä seitsemän: musta, punainen, sininen, vihreä, keltainen, oranssi ja harmaa. Värinvaihto muuttaa koko robotin piirtämän viivan väriä. Piirtoviivan tyypin valinta tehtiin pudotusvalikolla. Viivatyyppejä ovat kiinteä (Solid), katkonainen (Dashed) ja pisteviiva (Dotted). Piirtoviivan paksuuden säätäminen toteutettiin liukusäätimellä (Slider). Säädintä rullaamalla viereisessä tekstikentässä (Text box) näkyy ja vaihtuu arvo, joka kertoo viivan paksuuden. Kun kaikki halutut asetukset ovat valittuina, voi robotin laskea liikkeelle. Tätä varten on painike (Button) nimeltä Execute eli suoritus, joka vie annetut käskyt robotille. Nappia painettaessa voi katsella robotin liikkuvan alueellaan annettujen määräyksien mukaisesti. Suorituspainikkeen lisäksi on kaksi muuta paini-

ketta. Draw a Square-painikkeella käyttäjä voi antaa käskyn robotille piirtää punaisen neliön. Tämä toiminto on ohjelmakoodissa valmiiksi määriteltynä ja vastaavia toimintoja käyttäjä voi halutessaan ohjelmoida robotille lisää. Viimeisenä on Clear-painike, jolla käyttäjä voi puhdistaa koko alueen ja aloittaa puhtaalta pöydältä.

Microsoftin Visual Studio-kehitysympäristö mahdollistaa elementtien ja kontrollien lisäämisen sovellukseen drag and drop-menetelmällä raahaamalla niitä graafisessa ympäristössä. Kontrollien ominaisuuksia ja tietoja pystyy myös muokkaamaan kattavasti graafisella työkalulla. Tämän vuoksi minun ei tarvinnut itse kirjoittaa kovin paljoa sovelluksen käyttöliittymän rakentamiseen tarkoitettua XAML-koodia. Kehitysympäristön graafiset työkalut nopeuttavat ja automatisoivat käyttöliittymän rakennustyötä todella paljon.

```
    <Label Content="Draw Line"></Label>
    <RadioButton Name="radioButtonDrawOn" GroupName="rbDraw" Margin="5,0,0,0" IsChecked="True" Checked="radioButtonDrawOn_Checked">On</RadioButton>
    <RadioButton Name="radioButtonDrawOff" GroupName="rbDraw" Margin="5,0,0,0" Checked="radioButtonDrawOff_Checked">Off</RadioButton>

    <Label Content="Line Color"></Label>
    <StackPanel Orientation="Horizontal">
        <RadioButton Background="Black" BorderBrush="Black" Foreground="Black" Name="radioButtonBlack" GroupName="rbColor"
                     IsChecked="True" Margin="5,0,2,0" Checked="radioButton_Checked"></RadioButton>
        <RadioButton Background="Red" BorderBrush="Red" Foreground="Red" Name="radioButtonRed" GroupName="rbColor"
                     Margin="0,0,2,0" Checked="radioButton_Checked"></RadioButton>
        <RadioButton Background="Blue" BorderBrush="Blue" Foreground="Blue" Name="radioButtonBlue" GroupName="rbColor"
                     Margin="0,0,2,0" Checked="radioButton_Checked"></RadioButton>
        <RadioButton Background="Green" BorderBrush="Green" Foreground="Green" Name="radioButtonGreen" GroupName="rbColor"
                     Margin="0,0,2,0" Checked="radioButton_Checked"></RadioButton>
        <RadioButton Background="Yellow" BorderBrush="Yellow" Foreground="Yellow" Name="radioButtonYellow" GroupName="rbColor"
                     Margin="0,0,2,0" Checked="radioButton_Checked"></RadioButton>
        <RadioButton Background="Orange" BorderBrush="Orange" Foreground="Orange" Name="radioButtonOrange" GroupName="rbColor"
                     Margin="0,0,2,0" Checked="radioButton_Checked"></RadioButton>
        <RadioButton Background="Gray" BorderBrush="Gray" Foreground="Gray" Name="radioButtonGray" GroupName="rbColor"
                     Margin="0,0,2,0" Checked="radioButton_Checked"></RadioButton>
    </StackPanel>

    <Label Content="Line Type"></Label>
    <ComboBox Name="comboBoxLineType">
        <ComboBoxItem Content="Solid" IsSelected="True"/>
        <ComboBoxItem Content="Dashed"/>
        <ComboBoxItem Content="Dotted"/>
    </ComboBox>

    <Label Content="Line Thickness"></Label>
    <StackPanel Orientation="Horizontal">
        <Slider Name="sliderThickness" SmallChange="2" TickFrequency="2" Minimum="2" Maximum="20" LargeChange="2"
                Value="2" IsSnapToTickEnabled="True" TickPlacement="BottomRight" Width="228" ValueChanged="sliderThickness_ValueChanged"/>
        <TextBox Name="textBoxSliderThicknessValue" IsReadOnly="true" Width="22" Text="2"></TextBox>
    </StackPanel>

    <!--Button for executing UI commands-->
    <Button x:Name="buttonExecute" Content="Execute" Click="buttonExecute_Click" Margin="0,10,0,0"/>
    <!--Buttons for executing your own coded commands-->
    <!--Use these for experimenting or create totally new ones-->
    <Button x:Name="buttonSquare" Content="Draw a Square" Margin="0,10,0,0" Click="buttonSquare_Click"/>
    <Button x:Name="buttonClear" Content="Clear" Click="buttonClear_Click"  Margin="0,10,0,0"/>
</StackPanel>
```

KUVIO 5. Kontrollien muokkausnäkymää XAML-kielellä.



KUVIO 6. Sovelluksen kontrollit.

WPF-sovelluksen ydin on jaettu kahteen osaan, joista ensimmäinen on XAML-tiedostosta löytyvä graafinen käyttöliittymä ja toinen XAML.CS-tiedostoon perinteisellä C#:lla ohjelmoitava toiminnallisuus. Vaikka WPF-sovelluksen käyttöliittymän kokoaminen käy hyvin nopeasti graafisten työkalujen avulla, sovelluksen toiminnallisuus täytyy ohjelmoida käsin. Tosin Visual Studio osaa generoida esimerkiksi sovellukseen istutetuista kontrolleista niihin itseensä liittyviä metodirunkoja. Tämäkin ominaisuus nopeuttaa ohjelmointityötä. Suurin hyöty näiden kahden tiedoston symbioosista on kummankin se, että esimerkiksi jotain kontrollia kuten painiketta ei tarvitse luoda kuin toiseen tiedostoon ja se on kummankin tiedoston käytössä. Kun olen luonut käyttöliittymätiedostoon eri elementtejä, niitä pystyy käyttämään suoraan myös toisessa tiedostossa. Kahden päätiedoston lisäksi tein sovellukseeni luokan nimeltä Robot.cs. Tähän tiedostoon tuli kaikki robotin ominaisuudet ja toiminnallisuus.

Seuraava työvaihe sovellukseni parissa oli miettiä, minkälaisia muuttujia tarvitsen tallentamaan käyttöliittymästä saatavia tietoja. Tarvitsin esimerkiksi double-tietotyyppisiä muuttujia hallitsemaan robotin liikkumista ja koordinaatteja canvas-alueella. Vakioin liikkumismatkapituudet kovakoodauksen vält-

tämiseksi. Muita muuttujia tarvittiin muun muassa merkkijonojen tallentamiseen ja totuusarvojen tarkastamiseen. Piirtämistä varten käytin WPF:n objektien piirtämiseen tarkoitettuja luokkia, joista loin olioita hallinnoimaan robotin piirtämisominaisuuksia.

Sovelluksen varsinainen toiminnallisuus alkaa käyttäjän painaessa Execute-painiketta valittuaan haluamansa asetukset robotin liikkumista varten. C#:ssa kontrolleilla on omat tapahtumametodinsa (Event), joiden alla toiminnallisuus tapahtuu. Kun käyttäjää painaa Execute-painiketta, käynnistyy sovelluksessa painikkeen painallustapahtuma eli tässä tapauksessa buttonExecute_Click-metodi. Jos koko toiminnallisuus olisi ohjelmoitu suoraan tämän metodin sisään, metodista olisi tullut erittäin pitkä ja sekava, joten päädyin pilkkomaan toiminnallisuuden eri osa-alueisiin ja luomaan näistä omat metodinsa. Täten painikkeen painallusmetodin sisus pysyi yksinkertaisena sisältäen vain vähän toiminnallisuutta ja pari muuta metodia, jotka pitävät sisällään taas uusia metodeja ja näin paloittelevat sovelluksen toimintalogiikkaa. Suurin osa robotin ominaisuuksista tulee Robot-luokan sisältä.

```
105         /// <summary>
106         /// Method for using the UI and executing user's commands with the Execute-button.
107         /// The method is coded as an async method. This enables the waiting that robot does between the steps it takes.
108         /// The waiting is done by command "await Task.Delay(1000);" that pauses the method for one second.
109         /// </summary>
110         /// <param name="sender"></param>
111         /// <param name="e"></param>
112         private async void buttonExecute_Click(object sender, RoutedEventArgs e)
113         {
114             //These check the user's choices from the UI
115             robot.SetRobotDirection(comboBoxMoveDirection);
116             robot.SetLineStyle(comboBoxLineType);
117             bool result = true;
118             int steps = int.Parse(comboBoxMoveSteps.Text.ToString());
119 
120             //Mandatory for-loop: moves the robot and gives the error message if borders are crossed.
121             for (int i = 0; i < steps; i++)
122             {
123                 result = await robot.MoveTheRobot();
124                 if (result == false)
125                 {
126                     MessageBox.Show("Don't cross the borders!", "Warning", MessageBoxButton.OK, MessageBoxImage.Warning);
127                     return;
128                 }
129             }
130         }
```

KUVIO 7. Suorituspainikkeen metodi nimeltä buttonExecute_Click.

SetRobotDirection()-metodi tarkistaa käyttäjän antamat käskyt yksinkertaisilla valintarakenteilla. Robottia liikuttava metodi on taas hieman kompleksisempi. Sen ohjelmoimisessa on täytynyt käyttää C#:n mahdollistamaa asynkronista ohjelmointia. C#:ssa metodista pystyy tekemään asynkronisen lisäämällä metodin määrittelyyn termin ”async” (esim. private async void moveTheRobot()). Tämä mahdollistaa esimerkiksi sovelluksen toiminnan nopeuden säätämisen. Tätä ominaisuutta olen käyt-

tänyt robotin liikkumisen säätelyssä. Robotin liikkuminen ajetaan for-toistorakenteessa, jossa tarkastetaan myös haluaako käyttäjä robotin piirtävän. Jos käyttäjä on valinnut piirto-ominaisuuden päälle, toistorakenteessa kutsutaan metodia, joka piirtämisen toteuttaa (drawRobotsLine()). Toistorakenteesta löytyy myös virheentarkistus siltä varalta, että robottia yritetään ohjata yli alueen reunojen. Asynkronisuutta hyödynnetään toistorakenteen viimeisellä koodirivillä. Sieltä löytyy käsky, joka pakottaa ohjelman odottamaan tietyn ajan, ennen kuin se jatkaa toimintaansa. Käsky on ”await Task.Delay(1000)”, jossa odotusaika on 1000 millisekuntia eli yksi sekunti.

```
350         /// <summary>
351         /// Method that moves the robot and checks that the robot doesn't cross canvas borders.
352         /// The method is coded as an async method. This enables the waiting that robot does between the steps it takes.
353         /// The waiting is done by command "await Task.Delay(1000);" that pauses the method for one second.
354         /// </summary>
355         /// <returns></returns>
356         public async Task<Boolean> MoveTheRobot()
357         {
358
359             if (moveDirection == "Left" || moveDirection == "Right")
360             {
361                 robotCoordinateX = Canvas.GetLeft(imageRobot);
362
363                 if (moveDirection == "Left")
364                 {
365                     if (robotCoordinateX < 50)
366                     {
367                         return false;  // reached border
368                     }
369                     else
370                     {
371                         Canvas.SetLeft(imageRobot, robotCoordinateX - MOVE_MEASURE);
372                         if (draw == true)
373                         {
374                             DrawRobotsLine();
375                         }
376                     }
377                 }
378                 else if (moveDirection == "Right")
379                 {
380                     if (robotCoordinateX > 700)
381                     {
382                         return false;
383                     }
384                     else
385                     {
386                         Canvas.SetLeft(imageRobot, robotCoordinateX + MOVE_MEASURE);
387                         if (draw == true)
388                         {
389                             DrawRobotsLine();
390                         }
391                     }
392                 }
393             }
394             else
```

KUVIO 8. Robottia liikuttavan metodin alkupää.

Robotin piirtämisen toteuttavassa metodissa piirtäminen on toteutettu WPF:n luokkakirjaston avulla. Esimerkiksi piirto-ominaisuuden aikaansaama viiva on Line-luokasta luotu olio. Luokalla on ominaisuuksia kuten Stroke, jolla määritellään viivan väri. Sovelluksessani väri haetaan viivalle sovelluksen kontrollien värivalintojen tapahtumametodien avulla. Viivan paksuus on myös Line-luokan ominaisuus

nimeltä StrokeThickness. Sovellukseni hakee viivan paksuuden kontrollien viivan paksuus-liukusäätimen tapahtumametodin avulla. Myös esimerkiksi viivan kärkien muotoa pystyy määrittelemään Line-luokan ominaisuuksien avulla. Piirtoviivan tyyppi tehdään DoubleCollection-luokan avulla. Sillä määritellään välien pituus, mikä esimerkiksi pisteviivan pisteiden välissä on.

```
434         /// <summary>
435         /// Method that draws Robot's movement line if drawing is enabled.
436         /// Also creates the line types according the user's choice.
437         /// </summary>
438         public void DrawRobotsLine()
439         {
440             Line robotDrawLine = new Line();
441             SolidColorBrush colorToUse = (SolidColorBrush)new BrushConverter().ConvertFrom(lineColor);
442             robotDrawLine.Stroke = colorToUse;
443             robotDrawLine.StrokeThickness = drawThickness;
444             DoubleCollection dashes = new DoubleCollection();
445 
446             //Line type
447             if (lineType == (int) LineStyle.Solid)
448             {
449                 dashes.Add(0);
450                 dashes.Add(0);
451                 robotDrawLine.StrokeDashArray = dashes;
452                 robotDrawLine.StrokeDashCap = PenLineCap.Square;
453             }
454             else if (lineType == (int) LineStyle.Dashed)
455             {
456                 dashes.Add(2);
457                 dashes.Add(4);
458                 robotDrawLine.StrokeDashArray = dashes;
459                 robotDrawLine.StrokeDashCap = PenLineCap.Round;
460                 robotDrawLine.StrokeStartLineCap = PenLineCap.Round;
461             }
462             else if (lineType == (int) LineStyle.Dotted)
463             {
464                 dashes.Add(1);
465                 dashes.Add(0.5);
466                 robotDrawLine.StrokeDashArray = dashes;
467                 robotDrawLine.StrokeDashCap = PenLineCap.Flat;
468             }
469 
470             //Movement
471             if (moveDirection == "Left")
472             {
473                 robotDrawLine.X1 = Canvas.GetLeft(imageRobot) + MOVE_MEASURE_HALF;
474                 robotDrawLine.Y1 = Canvas.GetTop(imageRobot) + MOVE_MEASURE_HALF;
475                 robotDrawLine.X2 = Canvas.GetLeft(imageRobot) + MOVE_MEASURE + MOVE_MEASURE_HALF;
476                 robotDrawLine.Y2 = Canvas.GetTop(imageRobot) + MOVE_MEASURE_HALF;
477             }
478             else if (moveDirection == "Right")
479             {
480                 robotDrawLine.X1 = Canvas.GetLeft(imageRobot) + MOVE_MEASURE_HALF;
481                 robotDrawLine.Y1 = Canvas.GetTop(imageRobot) + MOVE_MEASURE_HALF;
482                 robotDrawLine.X2 = Canvas.GetLeft(imageRobot) - MOVE_MEASURE_HALF;
483                 robotDrawLine.Y2 = Canvas.GetTop(imageRobot) + MOVE_MEASURE_HALF;
484             }
```

KUVIO 9. Robotin piirtämismetodin koodia.

```
91          /// <summary>
92          /// Draw thickness-slider method
93          /// </summary>
94          /// <param name="sender">In this method the sender is Slider</param>
95          /// <param name="e"></param>
96          private void sliderThickness_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double> e)
97          {
98              var slider = sender as Slider;
99              if (robot != null)
100             {
101                 robot.SetDrawingThickness(slider, textBoxSliderThicknessValue);
102             }
103         }
104
```

KUVIO 10. Piirtopaksuuden liukusäätimen toteutus.

Piirtoviiva maalataan ja sen paikka canvas-elementillä määritetään robotin koordinaattien avulla. Sovellus tunnistaa aktiivisesti missä robotti milloinkin on ja määrittää robotin liikkumissuunnan mukaan viivan horisontaalisen ja vertikaalisen tason. Näin ollen viiva seuraa koko ajan aktiivisesti robottia ja luo illuusion, että robotti piirtää alueelle jättämällä jälkeensä piirtoviivaa.

6.4 **Tulokset**

Ohjelmointityön lopputuloksena syntyi varsin yksinkertainen mutta moniulotteinen sovellus, jossa on käytetty useita WPF-sovelluksen tyypillisimpiä ominaisuuksia hyväksi. Näin ollen ohjelmoinnin perusteita opetettaessa kyseisellä tekniikalla, opettaja pystyy esittelemään tämän sovelluksen avulla esimerkiksi eri kontrolleja ja miten niille ohjelmoidaan toiminnallisuutta tapahtumametodien avulla. Myös käyttöliittymän rakentamiseen suunnittelutyökalun ja XAML-koodin kera pystytään tutustumaan sovelluksen selkeän käyttöliittymän avulla. Käyttöliittymä ja koodi ovat myös rakennettu tarkoituksella sellaisiksi, että niihin pystyy helposti tekemään muutoksia. Opettaja voi kehittää erilaisia ohjelmointitehtäviä, missä opiskelija esimerkiksi lisää käyttöliittymään painikkeen ja ohjelmoi siihen uutta toiminnallisuutta robotille. Sovellus on saatavilla ja vapaasti ladattavissa GitHub-palvelussa osoitteessa https://github.com/Wilbourw/LearnToCSharp.git.

7 POHDINTA

Opinnäytetyön tavoitteena oli syventää omaa C#-ohjelmointikielen osaamista ja luoda hyödyllinen opetuskäyttöön suunnattu sovellus. Microsoftilla on todella kattavat dokumentaatiot ja ohjeet ohjelmointitekniikoidensa opiskeluun ja tutkimiseen. Tämän vuoksi oli helppoa löytää tarvittavaa tietoa C#:sta ja kehitysympäristö Visual Studiosta. Hieman haastavampaa oli etsiä tietoa ohjelmoinnin opettamisesta. Englanninkielistä kirjallisuutta aiheesta löytyi kumminkin paljon ja sainkin mielestäni tähän työhön rakennettua kattavan selostuksen ohjelmoinnin opetuksen historiasta ja opetusmetodeista. Oli mielenkiintoista tutustua, millä eri pedagogisin tavoin ohjelmoinninkin opettamista voi lähestyä.

Vastaanottaessani tehtävänantoa WPF-sovelluksen ohjelmoimisesta C#:lla olin kyllä ohjelmoinut C#:lla ennenkin, mutta vain Windows Forms-sovelluksia. WPF oli minulle uusi tuttavuus ja tämän takia tahdoinkin lähteä haastamaan itseäni ja opettelemaan uutta tekniikkaa. Windows Presentation Foundation osoittautuikin todella mielenkiintoiseksi ja joustavaksi alustaksi, jonka kanssa työskennellessäni opin todella paljon uutta ohjelmoimisesta yleiselläkin tasolla. Joskin WPF ja C# ovat niin valtava osaamisen alue, että niistä ei uuden oppiminen lopu kesken yhden ihmiselämän aikana. Koko opinnäytetyön, sekä ohjelmointisovelluksen että raportin, työstö sujui minulla ilman suurempia vastoinkäymisiä. WPF:n kanssa jouduin opettelemaan paljon uutta, mutta onneksi materiaalia ongelmanratkaisuun on internet täynnä. Koko opinnäytetyöprosessia varten olin suunnitellut alustavan aikataulun tavoitteineen, mitä milloinkin pitäisi olla valmiina, mutta kaunis kesä sekoitti sen varsin pahasti. Tämä ei kumminkaan haitannut, sillä sain työt silti valmiiksi ennen asettamiani aikarajoja.

Avoimen lähdekoodin periaate on mielenkiintoinen nykysuuntaus ohjelmistokehitysalalla. Ensinnäkin ne säästävät kaupallisesta näkökulmasta katsottuna ihmiseltä suuren määrän rahaa tarjoamalla ilmaisia vaihtoehtoja hintaville kaupallisille ohjelmistoille. Toiseksi se yhdistää ohjelmistoalan ihmisiä ympäri maailmaa työskentelemään yhdessä kehittääkseen jotain uutta ja innovatiivista, tai vastaavasti parantaakseen olemassa olevia open source-tuotteita yhdessä mielin. Ohjelman lisenssistä riippuen mikään ei estä myöskään ei-copyleft-lisenssin omaavan avoimen lähdekoodin hyödyntämistä omassa kaupallisessa tuotteessa. Kaiken kaikkiaan avoimen lähdekoodin idea on todella hyvä ja sitä pitäisi mielestäni hyödyntää ohjelmistoalalla entistä enemmän.

LÄHTEET

Check out C# programming and where to learn it 2014. Hakupäivä 14.7.2014, http://www.softwareengineerinsider.com/programming-languages/csharp.html#.U8OBaZR_vy0.

COSS ry 2014. Hakupäivä 16.7.2014, http://coss.fi/.

C# Station 2014. Hakupäivä 14.7.2014, http://csharp-station.com/.

Getting Started with Visual Studio 2014. Hakupäivä 15.7.2014, http://msdn.microsoft.com/library/vstudio/ms165079(v=vs.120).aspx.

Introduction to the C# Language and the .NET Framework 2014. Hakupäivä 14.7.2014, http://msdn.microsoft.com/en-us/library/z1zx9t92.aspx.

Kinnunen, P. 2009. Challenges of teaching and studying programming at a university of technology - Viewpoints of students, teachers and the university. Hakupäivä 21.7.2014, https://aaltodoc.aalto.fi/handle/123456789/4710.

Kompelli, K. 2012. OOPS History. Hakupäivä 15.7.2014, https://www.classle.net/content-page/oops-history.

Laakkonen, C. 2013. Avoin lähdekoodi – Tiesitkö mitä rajoituksia se asettaa ohjelmistosi käytölle? Hakupäivä 16.7.2014, http://www.sofokus.com/blogi/avoin-lahdekoodi/.

Mohorovicic, S. & Strcic, V. 2011. An Overview of Computer Programming Teaching Methods. Hakupäivä 18.7.2014, http://www.ceciis.foi.hr/app/index.php/ceciis/2011/paper/viewFile/431/238.

Open Source Initiative 2014. Hakupäivä 15.7.2014, http://opensource.org/.

Opetus ja kulttuuriministeriö 2014. Hakupäivä 17.7.2014, http://www.minedu.fi/OPM/.

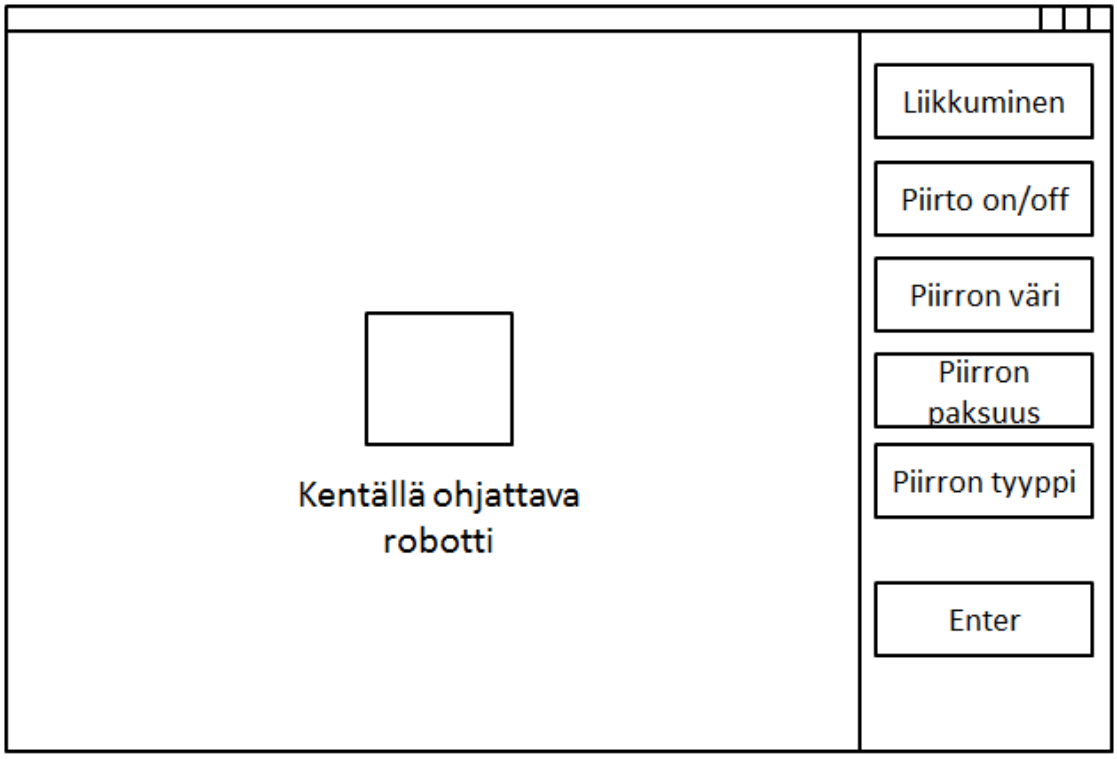
Seed, G. 2003. Object-Oriented Programming in C++ with Applications in Computer Graphics 2nd edition. London: Springer.

Tienari M. 2010. Helsingin yliopiston Tietojenkäsittelyopin laitos 1967-1990. Hakupäivä 17.7.2014, http://www.cs.helsinki.fi/sites/default/files/root/hallinto/tietojenkasittelyopin_laitos_1967_1990.pdf.

Tietotekniikan opetuksen historia 2014. Hakupäivä 17.7.2014, https://koppa.jyu.fi/avoimet/mit/tietotekniikan-opetuksen-perusteet/taustoista-nykyisyyteen/tietotekniikan-opetuksen-historia.

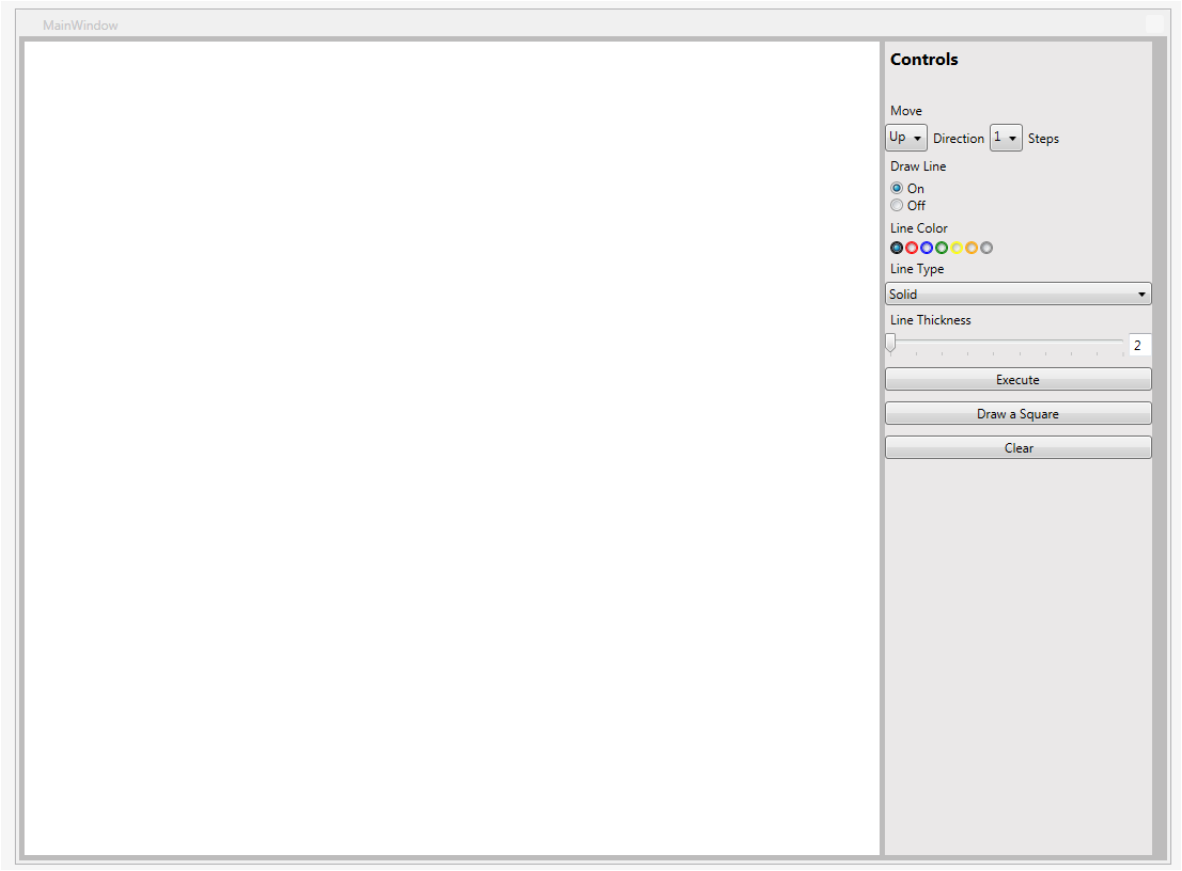
What is WPF? 2014 Hakupäivä 22.7.2014, http://www.wpf-tutorial.com/about-wpf/what-is-wpf/.

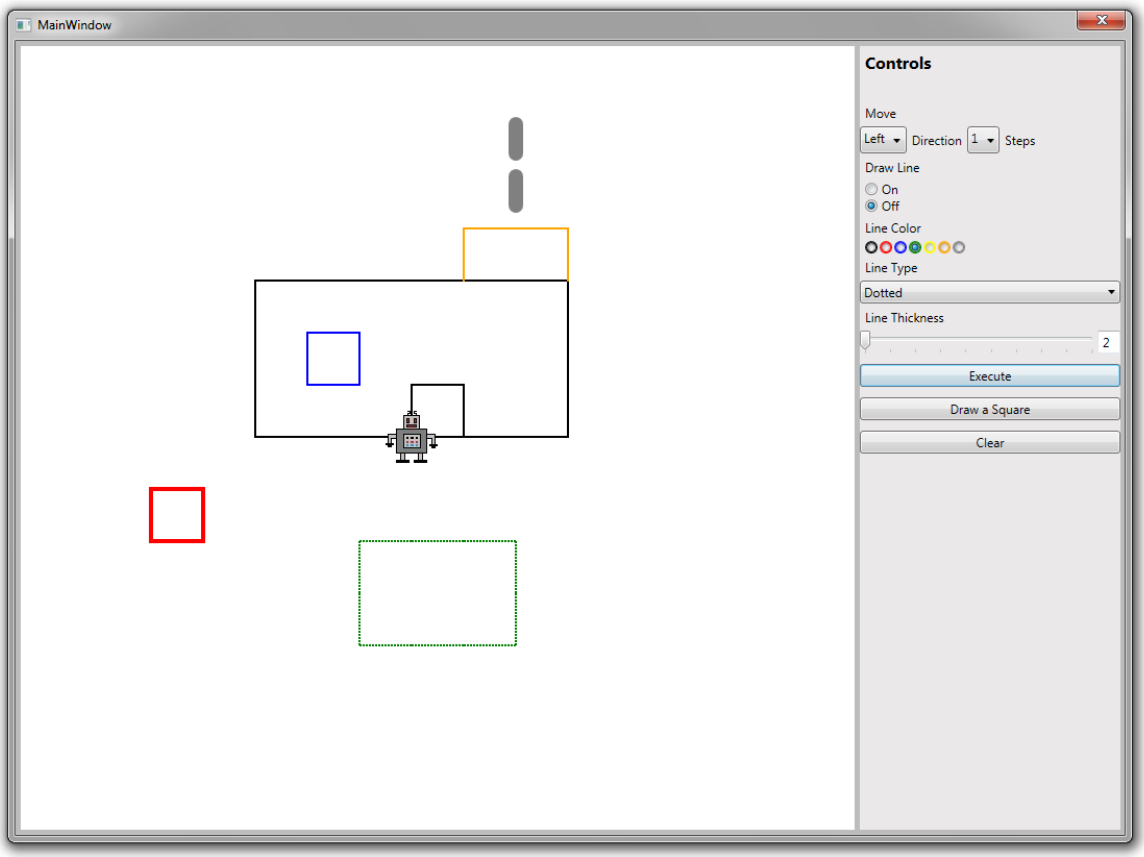
SOVELLUKSEN RAUTALANKAMALLI LIITE 1



VALMIIN SOVELLUKSEN ALOITUSNÄKYMÄ

LIITE 2




MainWindow
Controls
Move
Left
Direction
1
Steps
Draw Line
On
Off
Line Color
Line Type
Dotted
Line Thickness
2
Execute
Draw a Square
Clear