Riku Vanha-Rauvola

KUITU KARTALLE -OHJELMA

Tietotekniikan koulutusohjelma

2011

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

KUITU KARTALLE -OHJELMA

Vanha-Rauvola, Riku
Satakunnan ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Huhtikuu 2011
Ohjaaja: Javanainen, Mikko
Sivumäärä: 43

Asiasanat: ohjelmointi, GPS, GPX, Kartta, Navigointi

Eurajoen Puhelin Osuuskunta tarvitsee avukseen räätälöidyn ohjelman, jolla voidaan piirtää ja tallentaa valokuitujen sijaintitietoja kartoille. Valokuitujen reittejä halutaan piirtää pelkästään tietokoneohjelmalla tai käyttäen apuna automaattisesti siirrettyä GPS-navigaattorin reittitietoa.

Opinnäytetyön aiheena oli suunnitella ja ohjelmoida kaikki Eurajoen Puhelimen asettamat vaatimukset täyttävä ohjelma. Kuitu kartalle -projekti on ohjelmointityö, jossa ohjelmointikielenä oli Visual Basic NET. Ohjelmointialustana toimi Microsoftin Visual Studio 2010. Ohjelma suunniteltiin Windows 7 -käyttöjärjestelmälle, mutta se on täysin yhteensopiva myös Windows XP:n kanssa.

Ohjelman nykyisessä versiossa on toteutettu alkuperäisen suunnitelman mukaiset päätoiminnot. Päätoimintoihin kuuluvat uuden kartan kalibrointi, kuitujen ja kohteiden piirto, GPX-datatiedoston tallentaminen tietokantaan ja tietokannasta piirto kartalle. Ohjelmaan lisätään jatkokehityksen myötä uusia toimintoja kuten mm. kuvien lisääminen ja kuitujen tietojen esittäminen.

Työhön kuului myös pienenä lisänä GPS-navigaattorin valinta Eurajoen Puhelimelle. Vaihtoehtoina olivat kalliit, erittäin tarkat GPS-navigaattorit, sekä toisena vaihtoehtona olivat älypuhelimet ja niiden GPS-navigointilaitteisto, jotka mahdollistavat käyttötarkoitukseen nähden riittävän tarkkuuden. Ohjelman testaamista varten valittiin GPS-navigointilaitteeksi Nokian Symbian käyttöjärjestelmällä toimiva älypuhelin. Puhelimen GPS-sovellus Sports Tracker luo yleisen GPS-dataa ja mm. reittitietoja sisältävän GPX-tiedoston. Kuitu kartalle -ohjelma lukee tällaisen GPX-tiedoston ja tallentaa siitä halutut tiedot tietokantaansa.

KUITU KARTALLE -PROGRAM

Vanha-Rauvola, Riku
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in information technology
April 2011
Tutor: Javanainen, Mikko
Number of pages:

Keywords: Programming, GPS, GPX, Map, Navigation

Eurajoen Puhelin Cooperative needs a custom made program, which can draw and save optical fiber location information to maps. Optical fibers routes are wanted to be drawn using only the computer program or using the help of automatically transferred GPS-navigators route information data.

The purpose of this thesis was to design and program a program, which meets all the requirements set by Eurajoen Puhelin Cooperative. Kuitu kartalle-project is a programming job, where the programming language was Visual Basic NET. The development environment was Microsoft Visual Studio 2010. The Program was designed to Windows 7 operating system, but it also works on Windows XP.

The current version of the program has all the main functions mentioned in the original design plan. Main functions includes the calibration of a new map, the drawing of fibers and targets, the saving of GPX-data files to database and the drawing from database to map. Later on the program will have more functions like: the adding of pictures and the showing the information of optical fibers.

The job also included to choose a GPS-navigation device to Eurajoen Puhelin. The first option to choose from was expensive and very accurate GPS-navigators. The second option was smartphones and their GPS-navigation hardware, which were accurate enough to the purpose. A smartphone with Symbian operating system from Nokia was selected to be the GPS-navigation device for the testing the program. The smartphones GPS-application Sports Tracker creates a general GPX-file, which includes among other things GPS-route information. Kuitu kartalle-program reads GPX-data files and saves the wanted information to its database.

SISÄLLYS

SYMBOLIT JA LYHENTEET

GPS	Global Positioning System on satelliittipaikannusjärjestelmä.
VB	Visual Basic on Microsoftin kehittämä BASIC-sukuinen yleiskäyttöinen ohjelmointikieli.
VB.NET	Visual Basic kielestä kehittynyt nykyinen versio.
RGP	RGB-värimalli on väriavaruus, jossa eri värejä muodostetaan sekoittamalla keskenään punaista (R), vihreätä (G) ja sinistä (B) väriä.
GPX	GPS eXchange Format on XML-pohjainen formaatti, jolla voi käsitellä GPS-kordinaatteja määrämuotoisina.
PNG	Portable Network Graphics on on häviötön bittikarttagrafiikan tallennusformaatti kuvatiedostolle.

1 JOHDANTO

Valokuidut ovat lisääntyneet huomattavasti internetin yhteysnopeuksien kasvaessa. Eurajoen Puhelin Osk on alkanut asentaa Eurajoen seudulle isoja valokuituverkostoja. Tällä hetkellä Eurajoen Puhelimella ei ole täysin toimivaa ohjelmistoa, jolla maahan kaivettujen valokuitujen paikat voitaisiin piirtää kartalle. Kuitu kartalle -ohjelma kehitettiin, jotta Eurajoen Puhelimen henkilöstön ei tarvitse muistaa valokuitujen reittejä ulkoa.

Työssäni esittelen tekemääni ohjelmaa yleisesti ja kooditasolla. Opinnäytetyö etenee vaiheittain ohjelmoinnin alkuvaiheista loppuun. Opinnäytetyössäni tulen esittelemään mielestäni tärkeimpien vaiheiden ohjelmointiratkaisuja ja selittämään ohjelmakoodia. En selitä läheskään kaikkea tekemääni ohjelmakoodia, koska työni sivumäärä olisi tuolloin hyvin suuri. Kerron myös yleistä tietoa käyttämistäni tekniikoista sekä ohjelmointikielestä. Koordinaattimuunnoksiin liittyvissä ohjelmointivaiheissa esiintyi myös matemaattisia ongelmia, joita numeroita käsittelevissä ohjelmissa yleensä aina on.

Ohjelmalla tulee pystyä selaamaan käyttäjän valitsemia ja tuomia karttoja. Karttojen selaamisella voidaan yleisesti käsittää kartan tarkentaminen, loitontaminen ja kartan liikuttelu. Ohjelmaan pystyy tallentamaan useita karttoja ja niille valokuitureittejä sekä kohteita. Tallentaminen ja tietojen säilytys onnistuvat tietokannan avulla. Jokainen kartta kalibroidaan GPS-koordinaattien avulla ja kordinaattipisteet tallennetaan tietokantaan. Karttaan voi ohjelmassa piirtää kuituja ja kohteita. Käyttäjän valitsemaan karttaan voidaan tallentaa GPX-tiedostosta saatuja reittejä. Käyttäjän täytyy ainoastaan tietää, missä alueella reitti on tallennettu, jotta reitti voidaan tallentaa oikeaan karttaan. Tietokannassa olevat kuidut ja reitit näkyvät ainoastaan niihin yhdistetyissä kartoissa. Ohjelma piirtää halutut kuidut ja kohteet halutun värisinä, kokoisina ja muotoisina.

2 GPS-JÄRJESTELMÄSTÄ

2.1 GPS

GPS eli Global Positioning System on Yhdysvaltain puolustusministeriön kehittämä ja rahoittama satelliittipaikannusjärjestelmä, viralliselta nimeltään se on Navstar GPS. Se on nykyään yleisimmin käytetty GNSS-järjestelmä (Global Navigation Satellite System). Navstar-GPS:n kehitystyö aloitettiin 1970-luvun puolivälissä ja tarkoituksena oli luoda sekä sotilas- että siviilikäyttöön tarkka, reaaliaikainen ja yksisuuntainen paikannusmenetelmä. (Wikipedia: GPS)

2.1.1 Järjestelmän rakenne

Navstar-GPS-järjestelmä koostuu kolmesta segmentistä, jotka ovat avaruus, kontrolliverkko ja käyttäjäosa. Avaruussegmentin muodostavat satelliitit. Satelliitteja on 24 kappaletta noin 20200 km:n korkeudessa. Jokainen satelliitti kiertää maapallon ympäri kaksi kertaa vuorokaudessa. GPS-vastaanotin mittaa aikaeroja, jossa signaalit tulevat satelliiteista maahan vastaanottimelle. Kun satelliittien sijainnit tunnetaan, eri satelliiteista tulevien signaalien aikaerojen avulla kyetään laskemaan vastaanottimen tarkka sijainti.

Kontrolliverkossa tarkkaillaan satelliittien tilaa, ratoja ja toimintoja. Päävalvontakeskus on Yhdysvalloissa, Colorado Springsissä. Päävalvontakeskuksen lisäksi päiväntasaajan tuntumassa on neljä tarkkailuasemaa. Käyttäjäosan muodostavat miljoonat GPS-vastaanottimet. (Wikipedia: GPS)

2.1.2 GPS-paikannus

GPS-paikannus perustuu siihen, että satelliitit lähettävät atomikellon ajan ja navigaatiosignaalin, jonka GPS-laite vastaanottaa. GPS-laite vastaanottaa signaalia samanaikaisesti useasta satelliitista. Satelliitteja tulee olla vähintään neljä, sillä päätelaitteen kello ei ole tarkka kuten satelliittien atomikellot. Päätelaitteen kellon ja todellisen ajan ero (kellovirhe) täytyy asettaa yhtälöissä tuntemattomaksi, jolloin

tarvitaan vähintään neljän yhtälön ryhmä, jotta ratkaisu on yksikäsitteinen. Tämä vaatii havaintoa vähintään neljästä satelliitista.

Päätelaite laskee vastaanottamistaan radiosignaaleista joko pseudoetäisyyttä – käyttämällä satelliittisignaalin päälle moduloituja pseudosatunnaiskoodeja (PRN) – tai kantoaallon vaihetta. Pseudoetäisyyden ratkaisu perustuu olennaisesti signaalin kulkuajan mittaukseen. Paikkaratkaisu pystytään tekemään C/A-koodista (coarse acquisition) eli salaamattomasta koodista. Toisella radiotaajuudella lähetettävä P-koodi eli salattu koodi tuottaa tarkemman paikkaratkaisun. Molempia — koodi ja kantoaalto — voidaan käyttää paikannukseen, jälkimmäistä tosin vain monimutkaisissa geodeettisissa GPS-tarkkuusvastaanottimissa, sillä kantoaaltoa käytettäessä laskentaa ei voi tehdä reaaliajassa. (Wikipedia: GPS)

2.1.3 GPS-paikannustarkkuus

Vaikka satelliittien minimimäärä on neljä, päästään sitä tarkempaan lopputulokseen mitä useamman satelliitin avulla laskutoimitukset tehdään. Mitä enemmän mittauksessa käytettävät satelliitit ovat erillään toisistaan sitä tarkempi lopputulos saadaan. Paikannus onnistuu myös kolmen satelliitin avulla, kun oletetaan, että vastaanotin on jollakin tietyllä korkeudella vertailuellipsoidista. Tällöin vastaanottimen korkeuskoordinaattia ei ratkaista, jolloin tuntemattomia suureita on vain kolme (kaksi koordinaattia ja kellovirhe). Tämä oletus on hyvä merellä, jossa korkeuseroja ei ole, mutta esimerkiksi vuoristossa oletusta vakiokorkeudesta ei voi tehdä jos vastaanotin liikkuu.

Navstar-GPS:n tarkkuus on siviilikäytössä vaakasuunnassa muutama metri. Korkeussuunnassa tarkkuus on n. 2–3 kertaa heikompi. 1. toukokuuta 2000 asti Yhdysvaltojen puolustusministeriö heikensi tahallisesti satelliittien siviilikäyttäjiin lähettämät rataelementit ja kellon käyntitiedot (*Selective Availability*, SA), jota ennen siviilikäyttöön suunnattujen GPS-laitteiden tarkkuus oli vaakasuunnassa 100 metriä ja korkeussuunnassa 156 metriä. Paikannusvirhettä tuovat satelliittien rata- ja kellovirhe, ilmakehä, monitieheijastuminen, satelliittigeometria, paikantimen virheet, tahallinen häirintä ja käyttäjän virheet. (Wikipedia: GPS)

3 OHJELMOINTIKIELI

3.1 Visual Basic

Visual Basic on Microsoftin kehittämä BASIC-sukuinen yleiskäyttöinen ohjelmointikieli. Visual Basic pohjautuu 1980-luvulla julkaistuun QuickBASIC-kieleen. Visual Basic saavutti 1990-luvulla laajan suosion, joka on jatkunut 2000-luvun aikana. Kieltä käyttävät sekä aloittelijat että ammattilaiset ja se soveltuu sekä pienten että laajojenkin ohjelmien laadintaan. (Halvorson, M 2002)

3.1.1 Historia

Visual Basic -kielen ensimmäinen versio esiteltiin vuonna 1991. Versio 1.0 oli saatavissa sekä Windowsille että MS-DOS-käyttöjärjestelmään. DOS-version kehitys kuitenkin loppui ja seuraavat versiot (2.0-6.0) toimivat enää vain Windowsissa. Vuonna 2002 tapahtui kielessä merkittävä uudistus. Tuolloin julkaistiin Visual Basic .Net (VB.NET), joka on myös kielen nykyinen versio. Se kuuluu Microsoftin .NET-perheeseen.

VB.NET-uudistuksesta huolimatta vuonna 1998 julkaistu Visual Basic 6.0 on edelleen laajalti käytössä. Yhtenä syynä vanhan version käyttöön on se, että VB.NET poikkeaa huomattavasti aiemmista versioista. Vanhojen ohjelmien muunnos VB.NET-ympäristöön on suhteellisen työlästä. 6.0-versiolla laaditut ohjelmat myös toimivat paremmin vanhoissa käyttöjärjestelmissä. (Halvorson, M 2002)

3.1.2 Käyttöympäristöt

Visual Basic -kieltä kirjoitettiin alun perin käyttäen ohjelmointiympäristöä, jonka nimi oli myös Visual Basic kuten itse kielenkin. Se oli graafinen ohjelmointiympäristö. Nykyisellä Visual Basic .Net -kielellä ohjelmoidaan yleensä käyttäen Microsoftin Visual Studiota, joka sekin on graafinen ohjelmointiympäristö, mutta joka nykyisin on osa Microsoft .NET-perhettä. Visual Basic for Applications

-ohjelmat laaditaan yleisesti isäntäohjelman sisään rakennetussa VBA-ohjelmointiympäristössä.

Visual Basic .Net -kielellä ohjelmoidut ohjelmat toimivat .Net-ympäristössä. Ohjelmien ajamista varten tulee asentaa .NET framework, jonka saa ladattua Microsoftin sivuilta. Vanhemmilla Visual Basic -versioilla (1.0-6.0) laaditut ohjelmat toimivat ainoastaan Windows-käyttöjärjestelmissä. Poikkeuksena on vanha Visual Basic 1.0 DOS-versio, jolla tehdyt ohjelmat toimivat MS-DOS-järjestelmässä. Nykyään Visual Basicilla tehtyjä ohjelmia voi ajaa myös esimerkiksi Linuxissa käyttäen jotakin Windows-emulaattoria. (Halvorson, M 2002)

3.1.3 Syntaksi ja ominaisuudet

Basicin kielioppi periytyy suoraan Microsoftin aiemmista tuotteista (Mbasic, SV-Basic, MSX-Basic, GW-Basic, QuickBasic, AmigaBASIC). Visual Basic on rivipohjainen kieli. C-sukuisista ohjelmointikielistä poiketen lauseet erotetaan rivinvaihdolla tai (tarvittaessa) kaksoispisteellä, ei puolipisteillä. Lohkoja ei varsinaisesti määritellä millään tietyllä standardilla tavalla (vrt. C:n aaltosulut), vaan lohkon alussa ja lopussa on varatut sanat (esimerkiksi While ... Wend, If ... End If, For ... Next), jotka määrittelevät lohkon.

Visual Basic .Net on ohjelmointikielenä jonkin verran lähempänä C-sukuisia kieliä kuin aiemmat Visual Basicin versiot. Tämä johtuu pitkälti .NET-frameworkin käytöstä. Ominaisuuksiltaan VB.NET vastaa hyvin pitkälle C#-kieltä. Ensimmäisissä versioissa mahdollisuudet olio-ohjelmointiin olivat hyvin rajalliset, mutta VB.NET:in myötä kielestä on tullut täysiveroinen oliokieli.

Korkean tason kielenä Visual Basic on ilmaisuvoimaltaan sillä tavoin rajoittunut, että laitteistotasoon on vaikea päästä suoraan käsiksi. Toisaalta samasta syystä useiden abstraktien asioiden käsittely on kielessä selkeätä ja yksinkertaista.

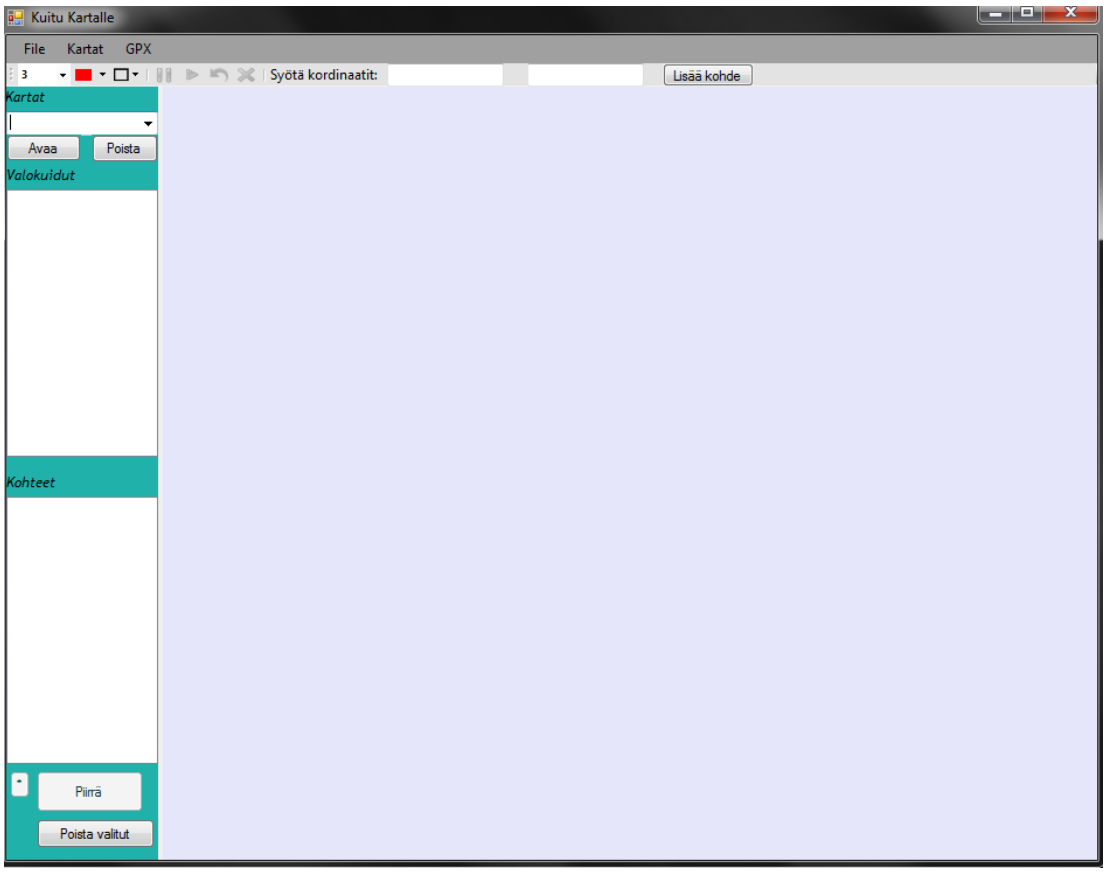
Suuri hyöty Visual Basicissä on myös kääntäjän mukana tuleva laaja työkaluvalikoima. Ohjelmaa on mahdollista tulkata virheiden paikannusta varten.

Tulkkauksen aikana ohjelmoijalla on mahdollisuus keskeyttää ohjelman ajo lähes milloin tahansa, tarkastella muuttujien arvoja ja jopa muuttaa niitä. Visual Basic -ohjelmat on mahdollista myös kääntää. Tällainen työkaluvalikoima ei kuitenkaan ole pelkästään Visual Basicin ominaisuus. Muillekin ohjelmointikielille on kehitetty vastaavia työkaluja, mutta usein ne on hankittava erikseen, joka voi lisätä kustannuksia merkittävästi. Toisaalta Visual Basicille on mahdoton löytää muita kuin Microsoftin kääntäjiä. (Halvorson, M 2002)

4 OHJELMOINTIVAIHEET

4.1 Ulkoasu

Yleisesti voidaan sanoa, että ohjelma tarvitsee mahdollisimman yksinkertaisen ja helppokäyttöisen ulkoasun kuten kuvassa (Kuva 1).



Kuva 1. Kuitu kartalle -ohjelman valmis ulkoasu

Yksinkertaisuus tulee jättämällä pois kaikki ylimääräinen, joka ei välttämättä liity ohjelman peruskäyttötarkoitukseen. Yksinkertaisuutta ei saa silti sekoittaa yksipuolisuuteen. Kaikki, mitä ohjelmalta halutaan, pitäisi olla heti käden ulottuvilla. Ehkä parempi ilmaisu tämän ohjelman tapauksessa olisi hiiren ulottuvilla. Ohjelma on helppokäyttöinen, kun sitä osataan käyttää ilman ohjeiden lukemista. Jos ohjelman ulkoasu myös muistuttaa jotain toisia ohjelmia, ohjelmaa on jo puoliksi opittu käyttämään. Windowsin ohjelmat muistuttavat huomattavasti toisiaan. Tietyt toiminnot sijaitsevat samoissa paikoissa, vaikka ohjelmat on suunniteltu täysin eri käyttötarkoituksiin.

Suunnittelussani halusin luoda ulkoasun, joka ei erotu liikaa muista Windowsin ohjelmista. Suunnittelu käynnistyi, kun valittiin ohjelman avaamisen jälkeinen näkymä. Ylös tulivat Windowsin ohjelmien tyyliin kuuluvasti päävalikko ja työkalurivi. Karttaa täytyy pystyä selaamaan samaan aikaan, kun kaikki ohjelman työkalut sekä muut toiminnallisuudet ovat näkyvissä. Varsinaiseen pääikkunaan tuli kaksi osaa: vasemmalle puolelle tulivat listat kuiduista ja kohteista, ja oikealle puolelle tuli suurempi osa eli varsinainen kartta.

4.2 Tietokanta

Tietokanta on tietotekniikassa käytetty termi tietovarastolle. Se on kokoelma tietoja, joilla on yhteys toisiinsa. Tietokannan ei välttämättä tarvitse olla sähköisessä muodossa, vaan sellaista voidaan pitää esimerkiksi kynällä ja paperilla. Kalenterikin on tietokanta.

Tietokanta saattaa edustaa jotain selvästi rajattua kohdetta reaalimaailmasta. Tällainen kohde voi olla esimerkiksi yrityksen keräämät tiedot asiakkaistaan.

Tietokantojen koot voivat vaihdella suuresti, yhteen tiedostoon tallennetuista taulukoista hyvin suuriin tietokantoihin joissa on useita miljoonia tietueita lukuisista kiintolevyistä koostuvilla levypakoilla. Tietokantaan voidaan tallentaa eri formaateissa olevaa tietoa, esimerkiksi tekstiä, ääntä ja videokuvaa. (Wikipedia: Tietokanta)

Microsoft Access on Microsoft Office -ohjelmistopaketteihin kuuluva tietokantojen hallintaohjelma. Microsoft Access-tietokannat ovat niin sanottuja relaatiotietokantoja. Access -tietokannat soveltuvat käytettäviksi kooltaan keskisuuriin tietokantoihin asti. Raporttien ja lomakkeiden suunnittelu tehdään yleensä Access -ohjelman ohjattujen toimintojen avustamana tai valmiita mallipohjia käyttäen, ja siksi ohjelma on suhteellisen helppokäyttöinen. Ohjelmaa käytetään paljon käyttöliittymän suunnittelun helppouden takia. Sitä voidaan käyttää

esimerkiksi käyttöliittymänä SQL-tietokantoihin. Microsoft Accessin vanhemmat versiot ovat käyttäneet sisäisesti Jet-tietokantamoottoria.

Acces-tietokannat toimivat hyvin yhteistyössä muiden Microsoftin ohjelmien kanssa. Esimerkiksi Microsoft Excel -taulukon tiedot voidaan tuoda Microsoft Accessiin, jolloin niitä voidaan tarkastella lomake- tai raporttimuodossa. Halvemmissa Office-paketeissa, esimerkiksi Officen "peruspaketeissa", kuten "Personal" ja "Student" ei ole Microsoft Accessia mukana. Microsoft Access sisältyy Microsoft Officen "Enterprise"- ja "Professional"- versioihin. (Wikipedia: Access)

Ohjelmaa suunnitellessani oli selvää, että tietokannan käyttö tulisi olemaan välttämätöntä. Tietokanta luotiin Microsoft Access -ohjelmalla. Käyttämäni Access 2000 on vanhempaa tiedostoformaattia, joka on yhteen sopiva käyttämissäni VB.NET-tietokantaan liittyvissä komennoissa. Testeissä nimittäin havaitsin, etteivät uusimmat Access -tietokannan tiedostoformaattiversiot toimineet ongelmitta ja siksi siis tein valintani vanhemmasta formaatista.

Ohjelmani käyttää tietokantaa ja sen sisältämiä taulukoita. Käytetyt taulukot sisältävät tekstiä ja numeroita. Jokaiselle ohjelmaan tallennetulle kartalle luodaan oma taulukko. Taulukot muodostuvat sarakkeista ja riveistä (Kuva 2).

KARTAT

ID	nimi	kuvaus
33	Raumanni	
34	Kuivalahti	Kuivista, pinkjärveee
35	IsoRauma	rauman yltä kuvaa
36	Eurajokki	
37	lapijoki	lappari
38	ejKeskusta	
39	Forssa	forssaa

Raumanni

ID	name	point1	point2	point3	point4	point5	tietoja
1	kalib	61,132101	21,494121	61,149731	21,545451		
2	vreitti	3	2				
3	vTrack201011111353	3	4				
7	kristeys	5	4	3	61,14019	21,50131	
8	vala-asteen kaapeli	2	2				
9	kala-asteen jakamo	3	3	2	61,1404	21,51222	

Sarakkeet

Rivit

Kuva 2. Esimerkki ohjelman käyttämistä tietokantataulukoista.

Kartat-taulukossa on kolme saraketta, jotka ovat ID, nimi ja kuvaus. Rivien määrä muodostuu ohjelman tietokantaan tallennettujen karttojen määrästä. Jokaisen kartan taulukossa on kahdeksan saraketta, jotka ovat ID, name, points 1-5 ja tietoja. Tietoja-saraketta ei nykyisessä ohjelmaversiossa käytetä, mutta jatkokehitystä varten se on tärkeä olla olemassa.

4.2.1 ADO.NET

ADO.NET on Microsoftin .NET Framework -ohjelmointiympäristöön sisältyvä tietokanta-rajapinta. Se on Microsoftin uusin versio yleiskäyttöisestä tietokantarajapinnasta. ADO.NET tarjoaa ohjelmoijalle suoran yhteyden lukuisiin eri tietokantatyyppeihin. Edellisiä versioita olivat ODBC, OLE DB ja ADO.

ADO.NET kuuluu VB.NET kirjastoihin ja sisältää monia hyödyllisiä komentoja tietokantojen käsittelyyn liittyen. ADO.NET oli siis selvä valinta Kuitu kartalle -ohjelmaan. Ohjelma käyttää monessa eri aliohjelmassa taulukkojen muokkaamista tai lukemista. Kaikissa tietokantaan liittyvissä aliohjelmissa on samat tietokantataulukon avaukseen ja käsittelyyn liittyvät pääpiirteet.

4.2.2 Yhteyden avaaminen ADO.NET:n avulla

Yhteyden avaaminen tietokantaan on ensimmäinen vaihe. Yhteys avataan kertomalla OLEDB-yhteysobjektille, mitä teknologiaa käytetään ja missä tietokanta sijaitsee (Kuva 3). Käytetty SQL-komento kerrotaan data-adapterille, joka kommunikoi tietokannan ja datasetin välillä. Dataset luodaan ja data-adapter asettaa siihen tietokannasta saadut tiedot. Dataset on tässä vaiheessa täysin luettavissa ja muokattavissa. Jos halutaan muokata datasettiä, on komentojen eteen luotava commandbuilder-olio. Muokattu dataset täytyy lopuksi lähettää data-adapterin kautta tietokantaan, jos halutaan tietojen tallentuvan. Varsinaisen yhteyden avaamisen suorittaa con.open ja con.close komennot ohjelman avautuessa ja sulkeutuessa.

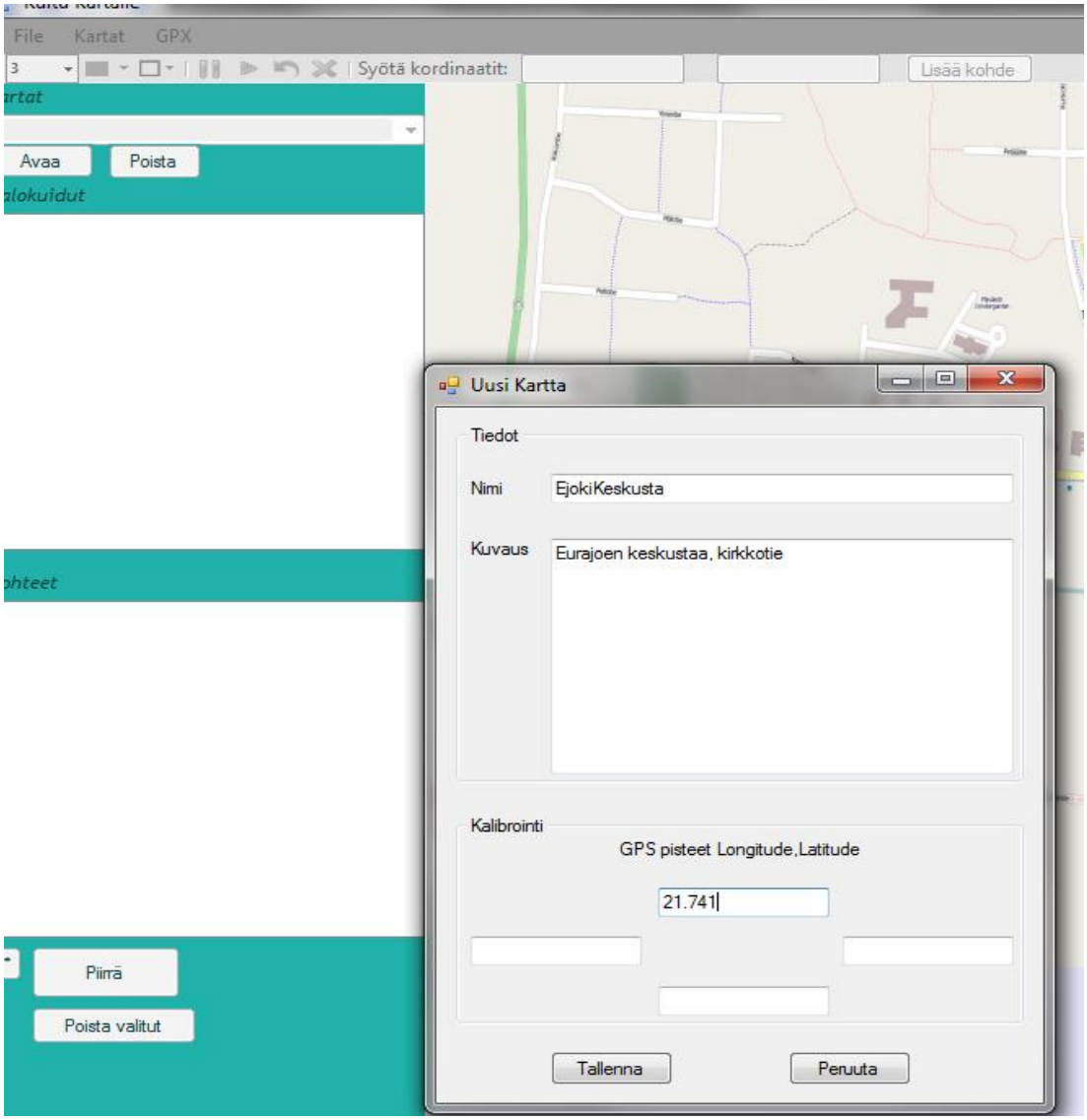
```
Dim tietokannan_polku As String = ohjelman_aloitus_polku + "\Data\Tietokanta\KuituKartalleTietokanta.mdb"
Public con As OleDbConnection = New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + tietokannan_polku)
con.open
-------------------------------------------------------------------------------------------------------------------------

strSQL = "SELECT * from " + kartan_nimi
Dim da As New OleDbDataAdapter(strSQL, con)
Dim ds As New DataSet

da.Fill(ds, kartan_nimi)
Dim cb As New OleDb.OleDbCommandBuilder(da)

****Käskyjä****

da.Update(ds, kartan_nimi)


con.close
```

Kuva 3. Yhteyden avaaminen ja sulkeminen ADO.NET.

4.3 Uusi kartta toiminto

Yksi ohjelman päätoiminnoista on tuo uusi kartta. Kun tuodaan uusi kartta, ohjelma aukaisee tiedostonavausikkunan avulla käyttäjän valitseman tiedoston, jonka täytyy olla .png muotoa oleva kuva. Ohjelma aukaisee uuden ikkunan, jossa kysytään kartan

nimeä ja kuvausta. Kartan kuvaukseen voidaan kirjoittaa tietoja kartasta. Lisäksi ikkuna kysyy kartan reunakordinaatteja, jotta ohjelma pystyisi kalibroimaan kartan. Uusi karttaikkuna aukeaa kartan päälle (Kuva 4).



Kuva 4. Uusi Kartta -ikkuna

4.3.1 Uusi Kartta -ikkunan toiminta

Visual basic -kielessä ikkunoita kutsutaan form-nimellä. Pääikkuna ja muut ikkunat eivät automaattisesti käytä samoja muuttujia tai aliohjelmia. Ikkunat ovat silti samaa ohjelmaa, ja ne saadaan kommunikoimaan keskenään helposti. Uusi Kartta -ikkunan muuttujat on esitelty yleisiksi muuttujiksi, jolloin pääikkuna pystyy hyödyntämään

niitä vapaasti. Uusi Kartta -ikkunan koodi tarkistaa, että kaikki pakolliset kentät on täytetty. Lisäksi kartan sivukordinaatit muutetaan ohjelmallisesti käsiteltävään muotoon. Tämä tarkoittaa pisteiden vaihtamista pilkuiksi. Kun pisteet on vaihdettu pilkkuihin, voidaan string-lause muuttaa double-muuttujaksi. Double-muuttujalla voidaan ohjelmallisesti tehdä matemaattisia laskuja. Kun Uusi Kartta -ikkuna aktivoituu, pääikkuna menee horrokseen, jolloin sitä ei voi tilapäisesti käyttää. Tallenna-näppäintä painettaessa tiedot tallentuvat muuttujiin ja Uusi Kartta -ikkuna sulkeutuu, jolloin pääikkuna aktivoituu jälleen.

4.3.2 Uuden kartan tallennus

Ohjelman tuotua uuden kartan se tallentaa kopion kartasta omaan hakemistoonsa ja avaa kartan kopion selaamiseen tarkoitettuun osioon eli kuvalaatikkoon. Tietokanta sijaitsee ohjelman suorituspolussa, Data-kansiossa. Data-kansiossa ovat kaikki ohjelman käyttämät tiedostot. Ohjelma avaa yhteyden tietokantaan ja tarkistaa, ettei samaa karttaa ole jo ennestään. Jos kyseinen kartta löytyy tietokannasta, se korvataan uudella.

Kartan nimi ja kuvaus tallentuvat tietokannassa sijaitsevaan Kartat-taulukkoon. Kartat-taulukossa on kaikki ohjelmaan tallennetut kartat. Tietokantaan luodaan myös uusi taulukko kartan nimellä, ja siihen tulevat tallentumaan kaikki karttaan tuodut kuidut ja muut kohteet. Kartan kalibrointipisteet tallentuvat taulukon ensimmäiseen riviin. Seuraaviin riveihin tallennetaan karttaan tuotuja kuitujen ja kohteiden nimiä. Kohteiden koordinaatit tallennetaan suoraan tietokantaan, koska kohteista täytyy tietää ainoastaan yksi piste, koko, väri ja muoto. Kuiduissa pisteiden määrä voi kasvaa suuriin määriin, jolloin tietokannan rajoitukset voivat tulla esteeksi. Tietokantojen rajoittuvaisuuden takia kuiduista tallennetaan ainoastaan nimi, koko ja väri uuteen tietokantataulukkoon. Kuituja varten ohjelma luo uuden kansion kartan nimellä Data-kansiossa sijaitsevaan Kuidut-kansioon. Uuteen kansioon tulevat tallentumaan tekstitiedostona kaikki karttaan tuodut kuidut.

Ohjelmaan tallennettu uusi kartta voidaan avata Kartat-alasvetolistasta. Alasvetolistasta valitsemalla voidaan avata ja poistaa ohjelmaan tallennettuja

karttoja. Avaa-nappulaa painamalla ohjelma avaa valitun kartan karttaosioon, ja kaikki siihen tuodut kuidut ja kohteet näkyvät valokuitu- ja kohderastivalintalistoissa. Kartan selaamisosioon avautuva kartta kutistetaan pieneksi, jotta käyttäjä näkisi kartan kokonaisuudessaan, vaikka tarkkoja kohteita ei pystykään erottamaan. Kartan leveys ja korkeus voivat olla erisuuruisia, joten kuvasuhde (leveys/korkeus) pitää säilyttää.

4.4 GPS-kordinaattien esittäminen

Kartan yläpuolella oleviin merkki-komponentteihin eli labeleihin ohjelma tallentaa GPS-kordinaatteja hiiren kartalla liikkumisen mukaan. Ohjelmalla on oma kaksiulotteinen kordinaatisto, jossa leveys- ja pituusasteet on kokonaisluvuilla ilmoitettu. Joillain tietyillä komponenteilla on myös oma kordinaatisto, joka alkaa vasemmalta ylhäältä ja kasvaa oikealle alas mentäessä. Kuvalaatikko eli pixturebox on komponetti, joka näyttää karttan. Kuvalaatikkolla on oma kordinaatisto. Vasen yläpiste on 0,0 ja oikea alapiste on kuvakoosta riippuva eli esim. 999,666. Varsinainen kalibrointi on jo tapahtunut, kun ohjelma tietää kartan sivujen leveys- ja pituusasteet kartan tuonnin mukana tallennetuista GPS-kordinaateista. Tiedossa on siis kaksi kordinaatistoa, jotka ovat käytännössä päällekkäin. Lukuja pystytään ilmoittamaan kumpaakin suuntaan, kun luodaan väliin rajapinta käyttämällä matematiikkaa.

Aliohjelma LocationKordinaateiksi suorittaa ohjelman kordinaattien käännön GPS-kordinaateiksi. Ohjelmakoodi (Kuva 5) näyttää, miten käännös tapahtuu.

```
Private Sub PictureBox1_MouseMove(ByVal sender As Object, ByVal e As System.Windows.Forms.MouseEventArgs) Handles PictureBox1.MouseMove

        x = e.X
        y = e.Y

        LocationKordinaateiksi()

        Label1.Text = y
        Label2.Text = x

End Sub

Public Sub LocationKordinaateiksi()

        x = Math.Round((oikea - vasen) * x / (zoomi - 1) + vasen, 5)
        y = Math.Round(yla - ((yla - ala) * y / (zoomi * zoomiErotus - 1)), 5)

End Sub
```

Kuva 5. Käännös ohjelman kordinaateista GPS-kordinaateiksi.

Kuvalaatikossa MouseMove-tapahtuma kertoo, kun hiiren kursori liikkuu kartan päällä. Aina kun hiiri liikkuu kartalla, ohjelma siis käskee aliohjelmaa muuttamaan hiiren sijainnin pisteet GPS-pisteitä vastaaviksi. X-kordinaatti eli leveysaste ja Y-kordinaatti eli pituussaste voidaan laskea matemaattisesta suhdekaavasta. Ohjelma tietää kartan oikean ja vasemman sivun GPS-leveysasteen. Erittäin tärkeässä yleisessä muuttujassa zoomi on tieto kartan nykyisestä kuvakoosta eli suurennuksesta. Zoomi antaa tiedon kuvan leveydestä. Laskutoimitus näkyy kuvan (Kuva 5) ohjelmakoodissa, mutta kirjoitettuna se on kaikessa monimutkaisuudessan seuraava:

1. Kartan GPS-kordinaattileveys kerrotaan hiiren osoittamalla x-akselin leveydellä.
2. Tulos jaetaan koko kartan kuvapisteleveydellä.
3. Tulos lisätään vasempaan GPS-kordinaattiin.

Y-kordinaatti lasketaan samalla periaatteella kuin x. Erona on, että alas mentäessä GPS-kordinaatistossa kordinaatit eivät kasva, vaan vähenevät. Kaavaa täytyi siis muuttaa, ja huomioitava oli myös kuvakoon suhde. Kartat eivät aina ole yhtä leveitä kuin korkeita. Yksinkertaistettuna laskutoimitus on seuraava:

1. Kartan GPS-kordinaattipituus kerrotaan hiiren osoittamalla y-akselin leveydellä.
2. Tulos jaetaan kartan kuvapistekorkeudella.
3. Tulos vähennetään ylä GPS-kordinaatista.

4.5 Kartan suurentaminen ja loitontaminen

Kun käyttäjä klikkaa hiiren oikealla nappulalla avatusta kartasta, näkyviin tulee valintapalkki. Valintapalkissa ovat kohdat Lähennä+ ja Loitonna-. Näillä toiminnoilla voidaan suurentaa ja loitontaa karttaa halutun kokoiseksi. Aliohjelmat zoomaaPlus ja zoomaaMiinus hoitavat suurennusoperaatiot. Kuvalaatikon asetuksiin on määritetty asetus automaattisesti venyttää kartta kuvalaatikon kokoon. Aliohjelmille pitää ainoastaan kertoa, kuinka paljon kuvaa suurennetaan/pienennetään, minkä jälkeen ne muuttavat kartan kokoa halutun

määrän mukaan. Kartta säilyttää aina kuvasuhteensa, koska muuttujaan ”zoomiErotus” ohjelma on tallentanut kartan korkeuden suhteen leveyteen. Korkeus eli pituus saadaan kertomalla zoomiErotus zoomin kanssa. If-lauseet tarkistavat, onko kartan suurennus sellaisessa kohdassa, että suurentamista/loitontamista ei voida enää tehdä. Suurentamisen määräksi eli muuttuja ”zoomin_maara”, on nykyisellä ohjelman versiolla arvo 1000. Ohjelma pitää minimiarvona zoomille 600. Maksimiarvoa ei ole määritelty, koska kartat ovat erikokoisia ja jotkut kartat vaativat paljon suurennusta. Ohjelmankoodissa (Kuva 6) näkyy suurentamisen ja loitontamisen toteutus.

```
zoomiErotus = PictureBox1.Image.Height / PictureBox1.Image.Width
-------------------------------------------------------------------------------

Public Sub zoomaaPlus(ByVal zoomin_maara As Integer)

        If zoomi = 600 Then
            zoomi = 1000
            LoitonnaToolStripMenuItem.Enabled = True
            UusiKuituToolStripMenuItem.Enabled = True
            UusiKohdeToolStripMenuItem.Enabled = True
        Else
            zoomi = zoomi + zoomin_maara
        End If
        PictureBox1.Size = New Size(zoomi, zoomi * zoomiErotus)

        SplitContainer1.Panel2.AutoScrollPosition = New Drawing.Point(aloitusPiste.X, aloitusPiste.Y)

End Sub

Public Sub zoomaaMiinus(ByVal zoomin_maara As Integer)

        If zoomi = 1000 Then
            zoomi = 600
            LoitonnaToolStripMenuItem.Enabled = False
            UusiKuituToolStripMenuItem.Enabled = False
            UusiKohdeToolStripMenuItem.Enabled = False
        Else
            zoomi = zoomi - zoomin_maara
        End If
        PictureBox1.Size = New Size(zoomi, zoomi * zoomiErotus)

End Sub
```

Kuva 6. Suurentamis- ja loitontamisaliohjelmat.

4.6 Kartan selaus ja vierityspalkit

Suurentamalla kuvalaatikon kokoa pystään tutkimaan haluttua kohtaa kartasta tarkemmin. Kartan selausta varten jouduin tekemään alkuperäiseen ulkoasuun vaikuttavia muutoksia. Kuvakoon muuttuessa suuremmaksi osa kuvasta menee käytännössä piiloon pääikkunan alle. Käyttäjä ei siis automaatisesti pääsisi katselemaan alle mennyttä kuvaa tai muita komponentteja. Ainoastaan, jos automaattinenvieritys on laitettu ikkunan asetuksista päälle, on mahdollista vierittää palkkeja haluttuun kohtaan. Ongelmana oli, että tehtäväpalkit ja muut valikot

menivät piiloon aina, kun vierityspalkkeja veti alas ja oikealle. Halusin saada automaattisenvierityksen pelkästään karttapohjalle eli kuvalaatikkoon, jolloin jouduin luomaan jaetunalustan eli splitcontainerin. Splitcontainer luo kaksi paneelia, jonka oikean puoleisessa osassa sijaitsee kuvalaatikko. Kun vaihtaa oikean puolen paneelin ominaisuuksista automaattisen vierityspalkin asetuksen totuusarvoon True eli tosi, paneeli luo automaattisesti vierityspalkit sen komponettien koon kasvaessa arvojensa yli.

Palkkeja vierittämällä pystytään liikuttamaan kartan haluttu kohta näkyviin, mutta se on kömpelöä ja hidasta. Käyttäjät ovat tottuneet tiettyihin standardeihin selatessaan karttoja ja muita kuvia. Hiirestä löytyy rulla, jota on yleisesti käytetty suurentamisessa. Kartan selaamisessa rulla ei olisi ollut hyvä, koska sillä ei pääse kuin yhteen ulottuvuuteen. Toinen vaihtoehto kartan selaamiseen oli ”mouse drag and drop” -toiminto. Suomennettuna se tarkoittaa hiirellä kiinni ja siirrä. Hiirellä raahaus oli selvä valinta ohjelmaani, koska melkein kaikki testaamani kartanselausohjelmat käyttävät samaa ratkaisua.

Toteutus raahaus-selaamiseen oli yllättävän yksinkertainen, vaikka sain kulumaan aikaa yrittäen itse rakentaa tyhjästä omaa ratkaisua. Tiesin, että ratkaisuun tarvittaisiin VB.NET:n MouseDown- ja MouseMove-tapahtumia. Apua löytyi onneksi internetin foorumeilta. Ohjelmakoodissa (Kuva 7) näkyy yllättävän yksinkertainen ratkaisu kartan selaamiseen.

```
Private Sub PictureBox1_MouseDown(ByVal sender As Object, ByVal e As System.Windows.Forms.MouseEventArgs) Handles PictureBox1.MouseDown

        aloitusPiste = New Point(e.X, e.Y)

End Sub

Private Sub PictureBox1_MouseMove(ByVal sender As Object, ByVal e As System.Windows.Forms.MouseEventArgs) Handles PictureBox1.MouseMove

        If piirto_kaynnissa = False And kohteen_piirto_kaynnissa = False Then
            If e.Button = Windows.Forms.MouseButtons.Left Then
                PictureBox1.Cursor = Cursors.SizeAll
                Dim DeltaX As Integer = (aloitusPiste.X - e.X)
                Dim DeltaY As Integer = (aloitusPiste.Y - e.Y)
                SplitContainer1.Panel2.AutoScrollPosition = New Drawing.Point((DeltaX - SplitContainer1.Panel2.AutoScrollPosition.X), _
                                                                              (DeltaY - SplitContainer1.Panel2.AutoScrollPosition.Y))
                PiirraKuidut(pTaulu)
            End If
        End If

End Sub
```

Kuva 7. Kartan selaamisessa käytetyt tapahtumat

Kuvalaatikon MouseDown-tapahtuma kertoo aloituspisteen, josta karttaa aletaan hiirellä vetää. Kuvalaatikon MouseMove-tapahtumaan lisäsin koodin, joka liikuttaa

vierityspalkkeja aina, kun hiiri liikkuu kartalla vasemman hiirennappulan olessa pohjassa. If-lauseet tarkistavat, ettei kuitujen tai kohteiden piirto ole käynnissä, jolloin ei ole mahdollista liikuttaa karttaa. Hiiren osoitin vaihtaa muotoaan Microsoftin järjestelmistä tuttuun sizeAll -muotoon, kun hiirtä liikutetaan kuvalaatikossa vasennappi pohjassa. Muuttujiin ”DeltaX” ja ”DeltaY” sijoitetaan aloituspisteiden ja nykyisten pisteiden erotus. Deltoista vähennetään nykyinen vierityspalkkien sijainti. Tulokseksi saadaan uusi vierityspalkkien sijainti. Kuitujen piirtoon liittyvä aliohjelma PiirräKuidut päivittää mahdollisesti tehtyä piirtoa jatkuvasti kartan siirtyessä.

4.7 Uusi kuitu -toiminto

Manuaalinen piirto kartalle on mielestäni Kuitu kartalle -ohjelman yksi tärkeimpiä ominaisuuksia. Jos käyttäjä tietää, missä kuitu maassa menee, niin käsin tietokoneella piirtäminen on varmasti tarkin mahdollinen keino merkitä kuitu. Suurin osa tulevasta ohjelman käytöstä tulee todennäköisesti liittymään käsin piirrettyihin kuitujen reitteihin. Ohjelmoinnin kannalta kaikista pisin ja vaikein urakka oli Uusi kuitu -ominaisuuden lisääminen ohjelmaan. Työtä syntyi suuri määrä, koska vaiheet olivat pitkiä. Ensimmäiseen vaiheeseen tuli pelkkä graaffinen piirto. Toiseen vaiheeseen kuului piirron tallennus tietokantaan. Molempien vaiheiden suunnittelussa piti huomioida ohjelman tulevien toimintojen yhteensopivuus. Kordinaattien käännökset, piirto ja tietokantaan tallennukset tuottivat yhteensä monta aliohjelmaa. Suunnittelussa piti myöskin huomioida myöhemmin ohjelmoitavaksi tuleva kohteen piirto toiminto.

4.7.1 Kuidun ja kohteen ominaisuudet

Ohjelman ulkoasuun tuli muutoksia. Työkalupalkkiin tuli valintalaatikko, josta voi valita kuidun tai kohteen paksuuden. Koko voi olla ykkösestä äärettömään, vaikka listassa on vaihtoehtoja vain kymmeneen asti.

Työkalupalkkiin tuli nappulalistavalikko, josta valitaan väri. Värin valinta nappulalistavalikolla oli yllävän työläs. Silti se oli helmpompi toteuttaa kuin muut mahdolliset vaihtoehdot. Jokaiselle listan nappulalle piti tehdä oma kuva, joka kuvaa kyseisen nappulan väriä. Kuvat tein Microsoft Paint -ohjelmalla, oikeilla värien RGP-arvoilla. Kuvat olivat 30 x 27 pixeliä kooltaan ja .png-muodossa. Kun käyttäjä painaa valitsemaansa värinappulaa listasta, väri-indeksin arvo vaihtuu ohjelmalle kerrottuun kokonaislukuun. Käytännössä väri-indeksi ilmoittaa tuleville piirtotoiminnoille nykyisen värin lukuna. Kohteen piirtoon liittyen työkalupalkkiin tuli kolmas valintanappula. Alasvetovalikkosta voidaan valita kohteen muoto neljästä vaihtoehdosta, jotka ovat nykyisessä versiossa neliö, kolmio, ympyrä ja tähti.

Muotojen valinnan ohjelmoinnissa oli sama työ kuin värin valinnassa. Työkalupalkkiin lisättiin Pause-, Jatka-, Undo- ja Cancel-painikkeet.

4.7.2 Piirto

Uuden kuidun piirto aloitetaan valitsemalla uudelle kuidulle koko ja väri. Käyttäjän painaessa kuvalaatikkoa eli karttaa oikella hiirennapilla esiin ponnahtaa valintapalkki. Palkista painetaan Uusi kuitu -toimintoa. Toiminnon aktivoituessa ohjelma siirtää kaikki tarpeettomat toiminnot horrokseen, eikä niitä siis pystytä käyttämään kuidun piirron ollessa käynnissä (Kuva 8). Ohjelma luo kartan päälle grafiikat, jotta siihen voitaisiin piirtää. Yleinen kokonaislukutaulukko ”pTaulu” alustetaan, jotta siihen voitaisiin tallentaa uuden kuidun pisteet. Kuvalaatikko voidaan virkistää eli Refresh()-lause pyyhkii vanhat piirrot pois. Totuusarvo ”piirto_käynnissä” muuttuu todeksi, jolloin ohjelma tunnistaa piirron olevan tapahtumassa. Kuvalaatikossa hiirialas-tapahtuma aktivoituu aina, kun hiiren nappulalla painetaan karttaa. Tapahtuma tallentaa nykyisen hiiren sijainnin ”aloituspiste” -pisteeseen. If-lauseet tarkistavat, onko piirto käynnissä. Piirron ollessa käynnissä kutsutaan aliohjelmaa ”PiirraOmiKuidut()”.

```
Private Sub UusiKuituToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles UusiKuituToolStripMenuItem.Click

            UusiKuituToolStripMenuItem.Text = "Lopeta Piirto"
            LähennäToolStripMenuItem.Enabled = False
            LoitonnaToolStripMenuItem.Enabled = False
            UusiKohdeToolStripMenuItem.Enabled = False
            ToolStripComboBoxKoko.Enabled = False
            ToolStripSplitButtonVari.Enabled = False
            ToolStripButtonPause.Enabled = True
            ToolStripButtonUndo.Enabled = True
            ToolStripButtonCancel.Enabled = True
            TextBox1.Enabled = False
            TextBox2.Enabled = False
            manuaalinenLisäysButton.Enabled = False
            ComboBox1.Enabled = False
            AvaaButton.Enabled = False
            DeleteButton.Enabled = False
            FileMenuItem.Enabled = False
            KartatMenuItem.Enabled = False


            grafiikat = PictureBox1.CreateGraphics
            ReDim pTaulu(1)
            PictureBox1.Refresh()
            piirto_kaynnissa = True

End Sub


Private Sub PictureBox1_MouseDown(ByVal sender As Object, ByVal e As System.Windows.Forms.MouseEventArgs) Handles PictureBox1.MouseDown

        aloitusPiste = New Point(e.X, e.Y)

        If piirto_kaynnissa And e.Button = Windows.Forms.MouseButtons.Left Then
            PiirraOmiKuidut()
        End If
        If kohteen_piirto_kaynnissa And e.Button = Windows.Forms.MouseButtons.Left Then
            PiirraKohde()
        End If

End Sub
```

Kuva 8. Kuidun ja kohteen piirto käynnistyy.

Aliohjelma ”PiirraOmiKuidut()” on pieni välivaihe varsinaiseen piirtoon liittyen (Kuva 9). Pisteet täytyy lisätä yksitellen taulukkoon ja piirtää, jotta käyttäjä näkee jatkuvasti tekemänsä reitin kartalla. Pisteet tallennetaan pTauluun peräkkäin x-kordinaatti ennen y-kordinaattia. ReDim Preserve -lause on uudelleenalustus, joka säilyttää vanhat taulun tiedot määrättyyn arvoon asti. Uudelleen alustamalla voidaan lisätä kätevästi taulukon kokoa uusien pisteiden lisäämisen yhteydessä. Kun taulussa on kaksi tai enemmän pistettä, kutsutaan varsinaisen piirron suorittavaa aliohjelmaa ”PiirraKuidut(pTaulu)”, joka esitellään ohjelmakoodissa (Kuva 8). Aliohjelma vastaanottaa taulukon, jonka se piirtää taulun sisältämien ohjelman koordinaattien avulla. Aliohjelmassa asetetaan piirtokynän arvot valitun koon ja värin perusteella. Kokonaislukumuuttujaan ”Kerrat”-rutiini asettaa luvun, josta se tietää, kuinka monta viivaa tullaan piirtämään. Käydään kaikki taulukon pisteet läpi ja piirretään viivat niiden väliin DrawLine -lauseella.

```
Public Sub PiirraOmiKuidut()
        Dim piste1 As Point
        piste1 = aloitusPiste

        If pTaulu(0) = 0 Then
            pTaulu(0) = piste1.X
            pTaulu(1) = piste1.Y
        Else
            ReDim Preserve pTaulu(pTaulu.Length + 1)
            pTaulu(pTaulu.Length - 2) = piste1.X
            pTaulu(pTaulu.Length - 1) = piste1.Y

            PiirraKuidut(pTaulu)
        End If

End Sub

Public Sub PiirraKuidut(ByVal taulu() As Integer)
        Dim i As Integer = 0
        Dim kerrat As Integer
        Dim piste1 As Point
        Dim piste2 As Point
        pKoko = ToolStripComboBoxKoko.Text
        Dim kyna As New Pen(pVari, pKoko)

        If taulu.Length = 4 Then
            kerrat = 2
        Else
            kerrat = taulu.Length - 2
        End If

        Do Until i = kerrat
            piste1 = New Point(taulu(i), taulu(i + 1))
            piste2 = New Point(taulu(i + 2), taulu(i + 3))

            grafiikat.DrawLine(kyna, piste1, piste2)
            i = i + 2
        Loop

End Sub
```

Kuva 9. Aliohjelmat ”PiirräOmiKuidut()” ja ”PiirräKuidut()”

ToolStripButtonPause-nappula on yksi kolmesta painikkeesta työkalupalkissa, joita voidaan käyttää piirron ollessa käynnissä. Karttaa pystyy lähentämään tai loitontamaan, jos tarve on. Pause-nappula asettaa piirron pois päältä ja mahdollistaa kartan loitontamisen tai lähentämisen. Loitonna- ja Lähennä-nappulat kutsuvat ”taulukonZoomi()”-aliohjelmaa, joka muuttaa pTaulun arvot nykyisen tarkennuksen mukaan (Kuva 10.). Pause-nappula aktivoi Jatka-nappulan, jotta piirtoa voitaisiin jatkaa haluttaessa.

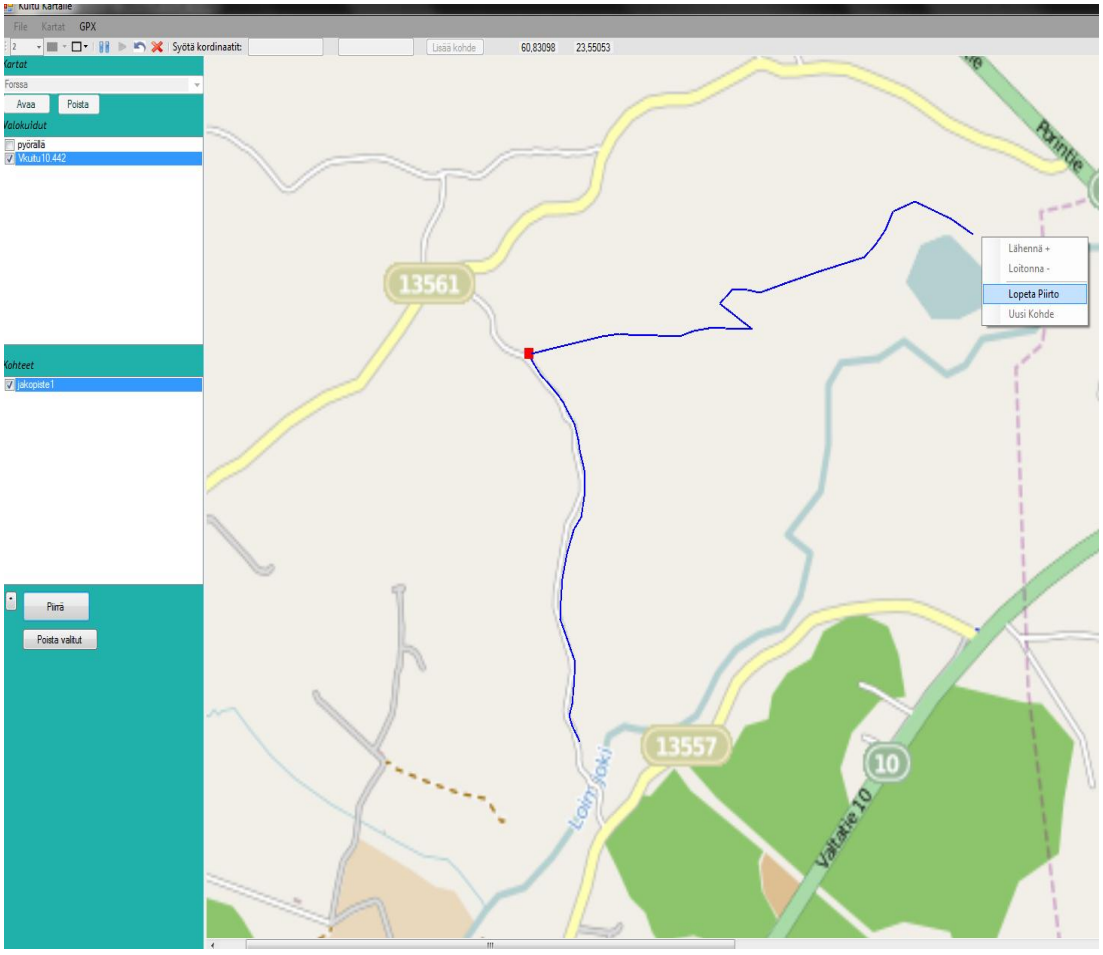
```
Private Sub LähennäToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles LähennäToolStripMenuItem.Click
        zoomaaPlus(1000)
        If pTaulu.Length > 2 Then
            taulukonZoomi(1)
            grafiikat = PictureBox1.CreateGraphics
            PiirraKuidut(pTaulu)
        End If
End Sub

Private Sub LoitonnaToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles LoitonnaToolStripMenuItem.Click
        zoomaaMiinus(1000)
        If pTaulu.Length > 2 Then
            taulukonZoomi(-1)
            grafiikat = PictureBox1.CreateGraphics
            PiirraKuidut(pTaulu)
        End If
End Sub

Public Sub taulukonZoomi(ByVal etuMerkki As Integer)
        Dim i As Integer = 0

        Dim vanhazoomi As Integer = zoomi - 1000 * etuMerkki

        Do Until pTaulu.Length = i
            pTaulu(i) = zoomi - (((vanhazoomi - pTaulu(i)) * zoomi) / vanhazoomi)
            i = i + 1
        Loop
End Sub
```

Kuva 10. Loitonna- ja Lahennä-nappulat sekä taulukonZoomi aliohjelma.

Jatka-nappula piirtää ennen Pausea tallennetun reitin ja asettaa Loitonna- ja Lähennä-nappulat horrokseen. Undo nimensä mukaan mahdollistaa askel taakse -ominaisuuden. Yksinkertaisesti toteutettu nappula käyttää hyväksi ReDim Preserve -lausetta, jolla voidaan pTaulusta ottaa viimeiseksi tallennettu piste pois. Undo-nappula tyhjentää grafiikat ja piirtää pTaulun uudestaan viimeisen pisteen ollessa poissa. Cancel-nappulan toiminto on myös hyvin yksinkertainen, vaikka sillä on ohjelmallisesti enemmän kokoa Undo-nappulan toimintoon verrattuna. Koodin määrä on suuri, koska Cancel-nappula palauttaa kaikki tärkeät komponentit horroksesta. Cancel-nappula peruuttaa piirron. Se yksinkertaisesti alustaa pTaulun oletusarvoon ja tyhjentää kartan piirroksista.

4.7.3 Tallennus tietokantaan

Kun käyttäjä haluaa lopettaa piirron ja tallentaa Uusi kuitu-tietokantaan, pitää hiiren oikealla painikkeella painaa karttaa. Kuvassa (Kuva 11) näkyy tilanne, jossa käyttäjä haluaa lopettaa uuden kuidun piirron.



Kuva 11. Uuden kuidun piirron lopetus.

Uusi kuitu -nappulan tilalle on ilmestynyt eri nimisenä Lopeta piirto -teksti. Kyseessä on sama nappula, mutta eri nimellä. Uusi kuitu -nappula ohjaa ohjelman suorittamaan ”lopetaPiirto()”-aliohjelma, kun halutaan lopettaa piirto. Aliohjelma aktivoi kuidun piirron aloitusvaiheessa horrokseen menneet komponentit. Kuitu tarvitsee nimen, jonka käyttäjä voi antaa teksti-ikkunan ilmestyessä näytölle. Aliohjelma tarkistaa, onko käyttäjän antama kuidun nimi jo käytössä. Jos nimi ei ole käytössä tai se on tyhjä, ”tallennaPiirto()”-aliohjelma käynnistetään. Piirron tallennus on esitelty ohjelmakoodissa (Kuva 12). Aliohjelma on tehty kuidun ja kohteen tallentamiseen. Kun halutaan tallentaa kuitu, pitää aliohjelmaa kutsua tyyliin: ”TallennaPiirto(True)”, jossa sulkujen sisällä oleva totuusarvo on tosi. Toisin ilmaistuna aliohjelma kysyy, onko tallennettava kuitu eli tosi vai onko se kohde eli epätosi. Piirron tallentava aliohjelma suorittaa valitun kartan tietokantataulun käsittelyihin liittyvät toiminnot. Ohjelma alustaa uuden taulukon rivin, johon lisätään kuidun tai kohteen tiedot. Kuidun tallennuksessa uudelle riville tulee tietoa kolmeen kenttään. Taulukossa ensimmäinen kenttä eli kenttä (0) on pääavain, johon tietokanta

luo automaattisesti numeron. Toiseen kenttään eli kenttään 1 aliohjelma kirjoittaa uuden kuidun nimen. Nimen eteen tulee kirjain ”v”, jotta ohjelman tietokantoja käyttävät aliohjelmat erottaisivat valokuidut ja kohteet toisistaan. Kohteiden nimen edessä on vastaavasti kirjain ”k”. Uuden rivin kolmanteen kenttään tallennetaan kuidun koko. Neljänteen kenttään kirjoitetaan värin indeksinumero, joka on väliltä 1—8. Kuidun reitin tallennuksessa luodaan tekstitiedosto (.txt) kuidun nimellä. Tiedostoon kirjoittaminen tapahtuu kätevästi streamwriter -luokalla. Ennen kuin tiedostoon voidaan kirjoittaa, pitää ohjelman kordinaattipisteet kääntää GPS-koordinaattipisteiksi. Käännösfunktiot ”FunktioLocationKordinaateiksi X ja Y vastaanottavat luvun ja palauttavat valmiit käännökset.

```
Public Sub tallennaPiirto(ByVal kuitu As Boolean)

        Dim tiedoston_polku As String = ohjelman_aloitus_polku + "\Data\Kuidut\" + kartan_nimi + "\" + uuden_kuidun_nimi + ".txt"

        strSQL = "SELECT * FROM " + kartan_nimi
        Dim da As New OleDb.OleDbDataAdapter(strSQL, con)
        Dim cb As New OleDb.OleDbCommandBuilder(da)
        Dim ds As New DataSet
        dsNewRow = ds.Tables(kartan_nimi).NewRow()

        If kuitu Then
            dsNewRow.Item(1) = "v" + uuden_kuidun_nimi
            dsNewRow.Item(2) = pKoko
            dsNewRow.Item(3) = pVariIndx

            If File.Exists(tiedoston_polku) Then
                File.Delete(tiedoston_polku)
            End If
            File.Create(tiedoston_polku)
            Dim writer As New StreamWriter(tiedoston_polku)
            Do Until pTaulu.Length + 3 = i
                If i Mod 2 = 0 Then
                    writer.Write(CStr(FunktioLocationKordinaateiksiX(pTaulu(i - 4))) + ";")
                Else
                    writer.Write(CStr(FunktioLocationKordinaateiksiY(pTaulu(i - 2))) + "|")
                End If
                i = i + 1
            Loop
            writer.Close()
        Else
            dsNewRow.Item(1) = "k" + uuden_kohteen_nimi
            dsNewRow.Item(2) = pKoko
            dsNewRow.Item(3) = pVariIndx
            dsNewRow.Item(4) = pMuotoIndx
            dsNewRow.Item(5) = FunktioLocationKordinaateiksiY(aloitusPiste.Y)
            dsNewRow.Item(6) = FunktioLocationKordinaateiksiX(aloitusPiste.X)
        End If
        ds.Tables(kartan_nimi).Rows.Add(dsNewRow)
        da.Update(ds, kartan_nimi)

        KartanTiedot()

End Sub


Function FunktioLocationKordinaateiksiX(ByVal X As Double)
        X = Math.Round((oikea - vasen) * X / (zoomi - 1) + vasen, 5)
        Return X
End Function

Function FunktioLocationKordinaateiksiY(ByVal Y As Double)
        Y = Math.Round(yla - ((yla - ala) * Y / (zoomi * zoomiErotus - 1)), 5)
        Return Y
End Function
```

Kuva 12. Piirron tallentaminen tietokantaan.

Tekstitiedostossa sijaitsevat kaikki kuidun GPS-kordinaattipisteet. Pisteet ovat tiedostossa peräkkäin leveys- ja pituusarvot ”|” -merkillä erotettuina. Kaksi pistettä erotetaan toisistaan puolipisteellä. Tiedostot tallentuvat ohjelman suorituspolkuun Data\Kuidut\(kartan-nimi) kansioon. Uusi kuitu päivitetään näkymään heti tallennuksen jälkeen valokuitujen listaan ”KartanTiedot()”-aliohjelmalla. Kartan kuitujen ja kohteiden listojen päivittämistä (Kuva 13) käytetään myös ohjelman muissa toiminnoissa.

```
Public Sub KartanTiedot()
        Dim strSQL As String, teksti As String
        Dim i As Integer, maxrows As Integer

        CheckedListBox1.Items.Clear()
        CheckedListBox2.Items.Clear()

        kartan_nimi = ComboBox1.Text
        strSQL = "SELECT * FROM " + kartan_nimi

        Dim da2 As New OleDb.OleDbDataAdapter(strSQL, con)
        Dim ds As New DataSet

        da2.Fill(ds, kartan_nimi)

        ala = ds.Tables(kartan_nimi).Rows(0).Item(2)
        vasen = ds.Tables(kartan_nimi).Rows(0).Item(3)
        yla = ds.Tables(kartan_nimi).Rows(0).Item(4)
        oikea = ds.Tables(kartan_nimi).Rows(0).Item(5)

        i = 1
        maxrows = ds.Tables(kartan_nimi).Rows.Count
        Do Until i = maxrows
            teksti = ds.Tables(kartan_nimi).Rows(i).Item(1)
            If teksti.Chars(0) = "v" Then
                CheckedListBox1.Items.Add(teksti.Remove(0, 1))
            Else
                CheckedListBox2.Items.Add(teksti.Remove(0, 1))
            End If
            i = i + 1
        Loop

End Sub
```

Kuva 13. Kartan kuitujen ja kohteiden listojen päivitys.

4.7.4 Kohteen tallennus tietokantaan

Uuden kohteen voi lisätä kartalle kahdella eri tavalla. Ensimmäinen tapa on sama menettely kuin uuden kuidun lisääminen. Työkalupalkista valitaan kohteen koko, väri ja muoto. Klikataan hiiren oikealla karttaa ja valitaan ”Uusi kohde”-nappula. Piirretään kohde manuaalisesti kartalle haluttuun kohtaan.

Toinen tapa on GPS-koordinaattien kirjoittaminen työkalupalkilla sijaitseviin tekstilaatikoihin. Tämä tapa on hyvä keino tarkistaa kartan kalibrointi. Esimerkiksi Googlen karttapalvelusta voidaan hakea tietyn risteyksen koordinaatit ja kirjoittaa nämä ”syötä kordinaatit”-tekstilaatikoihin. Klikkaamalla Lisää kohde -nappulaa käyttäjän kahteen tekstikenttään syöttämät kordinaatit tallentuvat tietokantaan. Jos kartta on kalibroitu onnistuneesti, kohde on suoraan samassa risteyksessä kuin

Googlen kartasta otettu pisteen sijainti. Ohjelmakoodissa (Kuva 14) näkyy tekstikenttien kautta tallentamisen toteutus.

```
Private Sub kordinaattienLisäysButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles kordinaattienLisäysButton.Click

        If tekstiCheck(TextBox1.Text) = "virhe" Or tekstiCheck(TextBox2.Text) = "virhe" Then
            Exit Sub
        End If

        uuden_kohteen_nimi = InputBox("Anna uuden kohteen nimi", "Nimeä kohde", "Kalibrointi1")
        If uuden_kohteen_nimi = vbNullString Then
            Exit Sub
        End If
        Dim kohteiden_maara As Integer = CheckedListBox2.Items.Count
        For i = 0 To (kohteiden_maara - 1)
            If uuden_kohteen_nimi = CheckedListBox2.Items.Item(i) Then
                uuden_kohteen_nimi = InputBox("Nimi oli jo olemassa, anna uusi nimi", "Nimeä kohde")
                i = 0
            End If
            If uuden_kohteen_nimi = vbNullString Then
                Exit Sub
            End If
        Next
        pKoko = ToolStripComboBoxKoko.Text

        aloitusPiste = New Point(kordinaatitLocationiksiX(tekstiCheck(TextBox2.Text)), kordinaatitLocationiksiY(tekstiCheck(TextBox1.Text)))
        tallennaPiirto(False)

End Sub

Function tekstiCheck(ByVal teksti As String)
        If teksti = vbNullString Then
            MsgBox("tekstikenttä on tyhjä")
            teksti = "virhe"
        Else
            teksti = teksti.Replace(".", ",")
            If IsNumeric(teksti) Then
                teksti = teksti
            Else
                MsgBox("Syötteissä oli kirjaimia")
                teksti = "virhe"
            End If
        End If
        Return teksti
End Function
```

Kuva 14. GPS-kordinaattien avulla tapahtuva kohteen piirto.

Funktio tekstiCheck (Kuva 14) tarkistaa ja korjaa käyttäjän antamat koordinaatit. Pisteet vaihdetaan pilkuiksi, jotta ohjelma tunnistaisi kordinaatit luvuiksi. Funktio antaa virheen, jos syötteet eivät kelpaa. Ohjelma kysyy käyttäjältä uuden kohteen nimeä ja tarkistaa sen kelpoisuuden. AloitusPiste-pisteeseen tallennetaan annetut GPS-kordinaatit käännettynä ohjelman kordinaateiksi. Aliohjelma ”tallennaPiirto” hoitaa uuden kohteen tallentamisen tietokantaan, samalla periaatteella kuin kuidun tallentaminen tietokantaan (Kuva 12).

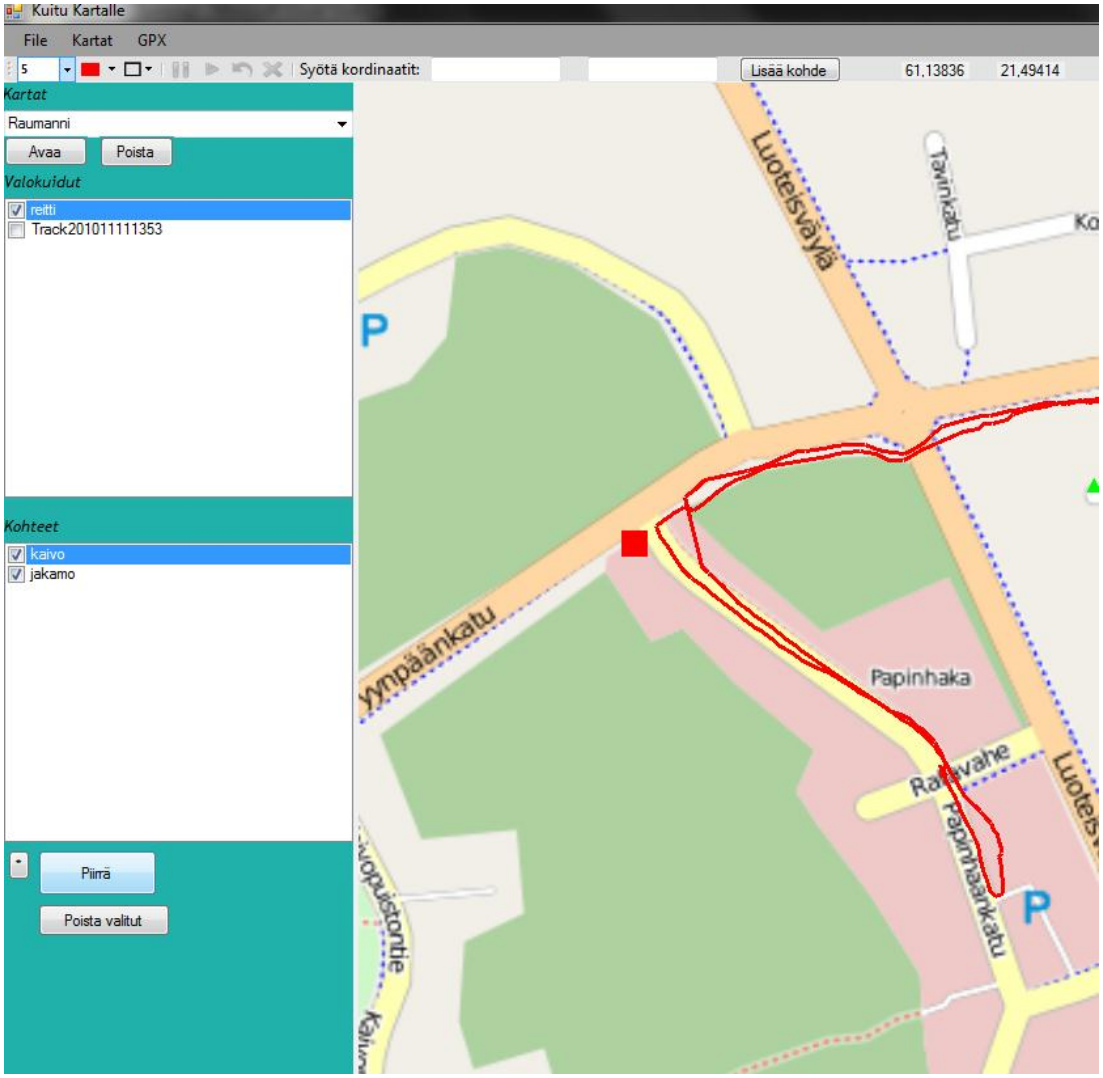
4.8 Tietokantaan tallennetut kuidut ja kohteet

Tietokantaan tallennetut kuidut ja kohteet näkyvät vasemmassa reunassa olevan paneelin valintalistoissa. Kuidut ja kohteet ovat karttaan sidottuja, joten kaikki tietokantaan tallennetut eivät tietenkään näy jokaisessa kartassa. Kuituja ja kohteita

voidaan vapaasti valita klikkaamalla rastia. Piirto- ja Poistonappulat sijaitsevat listojen alla ja ovat kumpikin nimiensä mukaisia toimintoja.

4.8.1 Piirto

Tietokantaan tallennettuja kuituja ja kohteita pystyy valitsemaan ja piirtämään Piirrä-nappulaa painamalla. Kuvassa (Kuva 15) käyttäjä on valinnut haluamansa kuidut ja kohteet ja piirtänyt ne kartalle.



Kuva 15. Valittujen kuitujen ja kohteiden piirto.

Aliohjelma ”piirraValitut” on koodimäärällisesti hyvin suuri. Piirto-nappi käynnistää aliohjelman, jossa on monta vaihetta. Vaiheita on ohjelmassa monta, koska

valintojen lukeminen tietokannasta ei ole niin yksinkertaista, kuin miltä se kuulostaa. Tallennetut kuidut ja kohteet ovat tietokannassa luomisajan mukaisessa järjestyksessä sekaisin keskenään. Ensimmäisessä suunnittelussani halusin tehdä aliohjelman hyvin yksinkertaisesti etsimällä kuituja ja kohteita nimen perusteella. Oletin automaattisesti, että tietokannasta voi etsiä rivejä tietämällä ainoastaan etsityn rivin sarakkeesta tietoa. Päivien tiedonhaun ja testaamiseen jälkeen en löytänyt ratkaisua. Suunnittelin ohjelmaani työläämmän ratkaisun. Esitelläkseni ratkaisua jaoin aliohjelman ”piirraValitut” ohjelmakoodin kolmeen kuvaan vaiheittain, joista ensimmäinen näkyy kuvassa (Kuva 16).

```
Public Sub piirraValitut()

        Dim strSQL As String
        Dim i As Integer = 0
        Dim u As Integer = 0

        Dim vTaulu(0) As Double
        Dim vKTaulu(0) As Double

        Dim iTaulu(0) As Integer
        Dim iKTaulu(0) As Integer
        Dim ciTaulu(0) As Integer
        Dim ciKTaulu(0) As Integer

        Dim iCount As Integer = CheckedListBox1.Items.Count
        Dim icCount As Integer = CheckedListBox1.CheckedItems.Count
        Dim iKCount As Integer = CheckedListBox2.Items.Count
        Dim iKcCount As Integer = CheckedListBox2.CheckedItems.Count
        Dim dsCount As Integer = 0

        Dim count As Integer = 3
        Dim reCount As Integer = 0

        strSQL = "SELECT * FROM " + kartan_nimi
        Dim da As New OleDb.OleDbDataAdapter(strSQL, con)
        Dim ds As New DataSet
        da.Fill(ds, kartan_nimi)

        dsCount = ds.Tables(kartan_nimi).Rows.Count - 1

        Do Until u = dsCount
            If ds.Tables(kartan_nimi).Rows(u + 1).IsNull(6) = True Then
                ReDim Preserve iTaulu(reCount)
                iTaulu(reCount) = u + 1
                reCount = reCount + 1
            End If
            u = u + 1
        Loop
        u = 0
        reCount = 0

        Do Until u = dsCount
            If ds.Tables(kartan_nimi).Rows(u + 1).IsNull(6) = False Then
                ReDim Preserve iKTaulu(reCount)
                iKTaulu(reCount) = u + 1
                reCount = reCount + 1
            End If
            u = u + 1
        Loop
```

Kuva 16. Valittujen piirron ensimmäinen vaihe.

Ensimmäisessä vaiheessa aliohjelman tarvitsemat muuttujat alustetaan. Muuttujiin kuuluvat mm. taulukot, joita käytetään valittujen kuitujen ja kohteiden selvittämisessä. Valitun kartan tietokantatauluun otetaan yhteys, jolloin aliohjelma saa tietoa tallennetuista kuiduista ja kohteista. Aliohjelmassa on varattu kuusi taulukkoa selvittämään, mitkä kuidut ja kohteet piirretään, ja mitkä ei. Ensimmäinen vaihe selvittää kaikkien kuitujen ja kohteiden rivien paikkaindeksit

tietokantataulukosta, ja tallentaa ne järjestyksessä kumpaakin i-tauluun. Kuitujen paikat saadaan selville tietämällä, että taulukon kentän kuusi ollessa tyhjä, kyseessä on kuitu. Kohde erottuu vastaavasti saman kentän ollessa käytössä. Kuitujen paikkaindeksit tietokantataulussa menevät iTauluun ja kohteiden iKTauluun.

Toinen vaihe on lyhyt, mutta tärkeä. Toisen vaiheen ohjelmakoodi näkyy kuvasta (Kuva 17).

```
u = 0
reCount = 0

If icCount = 0 = False Then
    Do Until u = iTaulu.Length
        If CheckedListBox1.GetItemCheckState(u) = 1 Then
            ReDim Preserve ciTaulu(reCount)
            ciTaulu(reCount) = iTaulu(u)
            reCount = reCount + 1
        End If
        u = u + 1
    Loop
End If

u = 0
reCount = 0

If iKcCount = 0 = False Then
    Do Until u = iKTaulu.Length
        If CheckedListBox2.GetItemCheckState(u) = 1 Then
            ReDim Preserve ciKTaulu(reCount)
            ciKTaulu(reCount) = iKTaulu(u)
            reCount = reCount + 1
        End If
        u = u + 1
    Loop
End If
```

Kuva 17 . Valittujen kuitujen ja kohteiden piirron toinen vaihe.

Toisessa vaiheessa ci-tauluihin tallennetaan käyttäjän listoista valitsemien kuitujen ja kohteiden paikkaindeksit tietokantataulukossa. Huomiotava on, että operaatio on eri kuin ensimmäisessä vaiheessa, jossa taulukoihin laitettiin kaikkien kuitujen ja kohteiden rivien paikkaindeksit tietokantataulukosta. Toisessa vaiheessa siis pelkästään hyödynnetään ensimmäisestä vaiheesta saatuja taulukoita, joista jätetään pois kaikki ei-valitut kuidut ja kohteet. Kuitujen paikat tallentuvat ciTauluun ja kohteiden ciKTauluun. If-lauseet tarkistavat, onko kuituja tai kohteita valittuna lainkaan.

Kuitujen ja kohteiden piirto on edennyt kolmanteen vaiheeseen, jossa ne eroavat hieman. Aliohjelman kolmas ja viimeinen vaihe kuitujen suhteen alkaa grafiikan luomisella kuvalaatikkoon (Kuva 18).

```
grafiikat = PictureBox1.CreateGraphics

Dim teksti As String, nimi As String
Dim int As Integer, maara As Integer
Dim cha As Char = "|"

If icCount = 0 = False Then
    Do Until i = ciTaulu.Length
        ReDim vTaulu(1)
        int = 0
        cha = "|"
        nimi = ds.Tables(kartan_nimi).Rows(ciTaulu(i)).Item(1)
        nimi = nimi.Remove(0, 1)
        Dim FILE_NAME As String = ohjelman_aloitus_polku + "\Data\Kuidut\" + kartan_nimi + "\" + nimi + ".txt"
        Dim reader As New System.IO.StreamReader(FILE_NAME)

        teksti = reader.ReadLine
        vTaulu(0) = ds.Tables(kartan_nimi).Rows(ciTaulu(i)).Item(2)
        vTaulu(1) = ds.Tables(kartan_nimi).Rows(ciTaulu(i)).Item(3)
        Do Until int = -1
            ReDim Preserve vTaulu(count)
            If cha = "|" Then
                int = teksti.IndexOf("|")
                maara = teksti.Length
                If int = -1 = False Then
                    vTaulu(count) = kordinaatitLocationiksiY(teksti.Remove(int, maara - int))
                End If
                cha = ";"
            Else
                int = teksti.IndexOf(";")
                maara = teksti.Length
                If int = -1 = False Then
                    vTaulu(count - 1) = kordinaatitLocationiksiX(teksti.Remove(int, maara - int))
                    count = count + 2
                End If
                cha = "|"
            End If
            teksti = teksti.Remove(0, int + 1)
        Loop
        ReDim Preserve vTaulu(count - 2)
        count = 3
        vPiirto(vTaulu)
        i = i + 1
        reader.Close()
    Loop
End If

Public Sub vPiirto(ByVal vTaulu() As Double)

        Dim i As Integer = 2
        Dim kerrat As Integer

        Dim piste1 As Point
        Dim piste2 As Point

        Dim Vari As System.Drawing.Brush = variCheck(vTaulu(1))

        Dim kyna As New Pen(Vari, CSng(vTaulu(0)))

        If vTaulu.Length = 4 Then
            kerrat = 2
        Else
            kerrat = vTaulu.Length - 2
        End If

        Do Until i = kerrat
            piste1 = New Point(vTaulu(i), vTaulu(i + 1))
            piste2 = New Point(vTaulu(i + 2), vTaulu(i + 3))

            grafiikat.DrawLine(kyna, piste1, piste2)
            i = i + 2
        Loop

End Sub
```

Kuva 18. Valittujen kuitujen piirron kolmas vaihe sekä varsinaisen piirron hoitava aliohjelma.

Kohteiden piirron kolmas vaihe on lyhyempi, koska kohteilla ei ole tekstitiedostoja. Kohteiden piirron kolmas vaihe on esillä kuvassa (Kuva 19).

```
    Do Until i = ciKTaulu.Length
        ReDim vKTaulu(5)

        If ciKTaulu(0) = 0 = False Then

            vKTaulu(0) = ds.Tables(kartan_nimi).Rows(ciKTaulu(i)).Item(2)
            vKTaulu(1) = ds.Tables(kartan_nimi).Rows(ciKTaulu(i)).Item(3)
            vKTaulu(2) = ds.Tables(kartan_nimi).Rows(ciKTaulu(i)).Item(4)
            vKTaulu(3) = kordinaatitLocationiksiX(ds.Tables(kartan_nimi).Rows(ciKTaulu(i)).Item(6))
            vKTaulu(4) = kordinaatitLocationiksiY(ds.Tables(kartan_nimi).Rows(ciKTaulu(i)).Item(5))

            vKPiirto(vKTaulu)
        End If
        i = i + 1
    Loop


Public Sub vKPiirto(ByVal vKTaulu() As Double)
    Dim X As Integer = CInt(vKTaulu(3))
    Dim Y As Integer = CInt(vKTaulu(4))
    Dim koko As Integer = vKTaulu(0)

    If vKTaulu(2) = 1 Then
        grafiikat.FillRectangle(variCheck(vKTaulu(1)), X - (koko + koko), Y - (koko + koko), (koko + koko) * 2, (koko + koko) * 2)
    End If
    If vKTaulu(2) = 2 Then
        Dim Pisteet(2) As Point
        Pisteet(0) = New Point(X - (koko + koko), Y + (koko + koko))
        Pisteet(1) = New Point(X + (koko + koko), Y + (koko + koko))
        Pisteet(2) = New Point(X, Y - (koko + koko))
        grafiikat.FillPolygon(variCheck(vKTaulu(1)), Pisteet)
    End If

    If vKTaulu(2) = 3 Then
        grafiikat.FillEllipse(variCheck(vKTaulu(1)), X - (koko + koko), Y - (koko + koko), (koko + koko) * 2, (koko + koko) * 2)
    End If
    If vKTaulu(2) = 4 Then
        Dim Pisteet(5) As Point
        Pisteet(0) = New Point(X - (koko + koko), Y + (koko))
        Pisteet(1) = New Point(X - (koko + koko), Y - (koko))
        Pisteet(2) = New Point(X, Y - (koko + koko))
        Pisteet(3) = New Point(X + (koko + koko), Y - (koko))
        Pisteet(4) = New Point(X + (koko + koko), Y + (koko))
        Pisteet(5) = New Point(X, Y + (koko + koko))
        grafiikat.FillPolygon(variCheck(vKTaulu(1)), Pisteet)
        grafiikat.DrawPolygon(Pens.Black, Pisteet)
    End If
End Sub
```

Kuva 19 . Valittujen kohteiden piirron kolmas vaihe sekä varsinaisen piirron hoitava aliohjelma.

Kuitujen ja kohteiden piirron kolmansissa vaiheissa on sama periaate. Ohjelma tallentaa kuidun ja kohteen piirtoon tarvittavat tiedot vTauluihin. Tiedot haetaan tietokantataulukoista ja kuidun tapauksessa myös tiedostosta. vTauluihin tulevat lopulta kaikki piirtoon tarvittavat tiedot. Nyt ei ohjelman tarvitse tehdä muuta kuin lähettää tehty taulukko varsinaisen piirron suorittavalle aliohjelmalle. Ohjelma toistaa samaa operaatiota niin monta kertaa, kuin käyttäjän valitsemia kuituja ja kohteita on.

4.8.2 Poisto

Poista valitut -toiminto käyttää täysin samanlaisia menetelmiä, kuin Piirrä-toiminto. Käytännössä ohjelmakoodin alkuosa on täysin sama piirrossa ja poistossa. Ensin tarkistetaan, mitkä kuidut ja kohteet ovat valittuna. Kuitujen ja kohteiden paikat on

selvitettävä tietokantataulukossa omiin taulukoihinsa aivan kuin piirrossakin. Poistossa pitää lopuksi molemmista taulukoista tehdä yhteinen, jossa kuitujen ja kohteiden paikkaindeksit tietokantataulukossa ovat samassa taulukossa numerojärjestyksessä. Taulukkojen yhdistäminen oli haankalaa, koska mukana oli kaksi kooltaan tuntematonta taulukkoa, jotka piti järjestää laskevaan numerojärjestykseen. Ratkaisussani käytin 13:aa eri if-lausetta. Jälkikäteen tuli mieleen paljon helpompiakin vaihtoehtoja ratkaista taulukkojen yhdistäminen. Ratkaisuni kahden taulukon yhdistämiseen ja numeroiden järjestämiseen näkyy ohjelmakoodikuvassa (Kuva 20).

```
Dim yTaulu(ciTaulu.Length + ciKTaulu.Length - 1) As Integer
If ciTaulu(0) = 0 Or ciKTaulu(0) = 0 Then
    ReDim yTaulu(yTaulu.Length - 2)
End If
Dim e As Integer = 0
Dim k As Integer = 0
Dim d As Integer = 0
u = 0

If ciTaulu(0) <> 0 Or ciKTaulu(0) <> 0 Then

    Do Until i = ciTaulu.Length + ciKTaulu.Length
        If ciTaulu(0) <> 0 And ciKTaulu(0) <> 0 Then
            If d = 1 Then
                yTaulu(i) = ciTaulu(e)
                If e < ciTaulu.Length - 1 Then
                    e = e + 1
                End If
            End If
            If u = 1 Then
                If k < ciKTaulu.Length Then
                    yTaulu(i) = ciKTaulu(k)
                End If
                If k < ciKTaulu.Length - 1 Then
                    k = k + 1
                Else
                    d = 1
                End If
            End If
            If u = 0 And d = 0 Then
                If ciTaulu(e) < ciKTaulu(k) Then
                    If e < ciTaulu.Length Then
                        yTaulu(i) = ciTaulu(e)
                    End If
                    If e < ciTaulu.Length - 1 Then
                        e = e + 1
                    Else
                        u = 1
                    End If
                Else
                    yTaulu(i) = ciKTaulu(k)
                    If k < ciKTaulu.Length - 1 Then
                        k = k + 1
                    Else
                        d = 1
                    End If
                End If
            End If
        End If
        If ciTaulu(0) = 0 Then
            yTaulu(i) = ciKTaulu(k)
            If k = ciKTaulu.Length - 1 Then
                i = i + 1
            End If
            k = k + 1
        End If
        If ciKTaulu(0) = 0 Then
            yTaulu(i) = ciTaulu(e)
            If e = ciTaulu.Length - 1 Then
                i = i + 1
            End If
            e = e + 1
        End If
        i = i + 1
    Loop
```

Kuva 20. Kahden taulukon yhdistäminen laskevaan numerojärjestykseen.

Kun kaikki poistettavat kuidut ja kohteet ovat samassa taulukossa, varsinainen poisto suoritetaan tietokannasta (Kuva 21). Ohjelma tarkistaa, onko poistossa kuitu. Kuidun ollessa kyseessä täytyy myös tekstitiedosto poistaa.

```
Dim nimi As String

Do Until i = yTaulu.Length
    nimi = ds.Tables(kartan_nimi).Rows(yTaulu(i) + u).Item(1)
    If nimi.Chars(0) = "v" Then
        nimi = nimi.Remove(0, 1)
        File.Delete(ohjelman_aloitus_polku + "\Data\Kuidut\" + kartan_nimi + "\" + nimi + ".txt")
    End If
    Dim cb As New OleDb.OleDbCommandBuilder(da)
    ds.Tables(kartan_nimi).Rows(yTaulu(i) + u).Delete()
    da.Update(ds, kartan_nimi)
    i = i + 1
    u = u - 1
Loop
```

Kuva 21. Kuitujen ja kohteiden poisto tietokannasta.

4.9 GPX

GPX eli GPS eXchange Format on XML-pohjainen formaatti, jolla voi siirtää reittipisteitä. (geowiki.fi : GPX)

Ohjelman alkuperäisenä ideana oli tallentaa GPS-navigaattorilla tehty reitti kartalle. Tätä varten ohjelman pitää osata lukea jotain navigaattorin tekemää data-tiedostoa ja tallentaa se tietokantaan. GPX-tiedosto on XML-skeema, joka on suunniteltu sisältämään reitti, reittipiste ja jälki GPS-tietoina. GPX on yleisesti käytetty tiedostoformaatti eri navigointiohjelmien ja laitteiden välillä. Formaatti on avoin, eikä siitä tarvitse maksaa lisenssimaksuja. Useat uusimmat älypuhelimet ja niiden sisältämät navigointi-ohjelmat pystyvät luomaan GPX-tiedostoja. GPX-tiedoston lukeminen ja tallentaminen olivat selvä valinta ohjelman toimintoihin.

Toteutuksen suunnittelu alkoi valitsemalla vaikeasta tai helposta ratkaisusta. Vaikea ratkaisu olisi ollut rakentaa täysin oma GPX-tiedoston luku ohjelmallisesti. Helpompi ratkaisu löytyi internetin keskustelufoorumilta, jossa neuvottiin lataamaan GPX-skeema versio 1.1. Visual studion mukana tulee XSD-työkalu, joka lukee skeemoja ja tekee niistä käytettäviä Visual basic -luokkia. Käytettäessä XSD-työkalua projektiin lisätään tiedoston gpx.vb. Tiedostossa on paljon luokkia ja metodeja, joilla ohjelmani saa kaiken tarvittavan tiedon esiin .gpx -tiedostoista.

GPX-tiedoston tallentaminen kuului lopulta helpoimpiin työvaiheisiin projektissani. Ohjelmalleni piti ainoastaan kertoa, miten käyttää tiedoston gpx.vb -luokkia ja metodeja. Ohjelmakoodikuvassa (Kuva 22) ohjelma hyödyntää vain murto-osaa gpx.vb -tiedoston luokista ja metodeista, koska muita tietoja ohjelman ja käyttäjän ei tarvitse tietää.

```
Public Sub gpxTiedostoTauluun(ByVal polku As String)

        Dim MyFileStream As FileStream = New FileStream(polku, FileMode.Open)
        Dim i As Integer = 0
        Dim TrkPts(0) As Object
        Dim strSQL As String

        kartan_nimi = ComboBox1.Text
        If kartan_nimi = vbNullString Then
            MsgBox("Valitse kartta")
            MyFileStream.Close()
            Exit Sub
        End If

        Dim Myserializer As XmlSerializer = New XmlSerializer(GetType(gpxType))
        Dim mygpx As gpxType = New gpxType
        Try
            mygpx = CType(Myserializer.Deserialize(MyFileStream), gpxType)
        Catch ex As Exception
            MsgBox("gpx tiedostoa ei voitu lukea. Ota yhteyttä koodaajaan. ")
            Exit Sub
        End Try
        Dim Singletrk As trkType
        Dim SingletrkSeg As trksegType
        Dim SingletrkPt As wptType
        For Each Singletrk In mygpx.trk
            uuden_kuidun_nimi = Singletrk.name
            uuden_kuidun_nimi = InputBox("Haluatko varmasti tallentaa tämän kuidun karttaan " + kartan_nimi +
                                         " nykyisillä koko ja väri asetuksilla",
                                         "Tallenna kuitu", uuden_kuidun_nimi)
            If uuden_kuidun_nimi = "" = False Then
                polku = (ohjelman_aloitus_polku + "\Data\Kuidut\" + kartan_nimi + "\" + uuden_kuidun_nimi + ".txt")
                Dim writer As New StreamWriter(polku, False)
                For Each SingletrkSeg In Singletrk.trkseg()
                    For Each SingletrkPt In SingletrkSeg.trkpt()
                        ReDim Preserve TrkPts(i)
                        With TrkPts(i)
                            writer.Write(CStr(SingletrkPt.lat) + "|")
                            writer.Write(CStr(SingletrkPt.lon) + ";")
                        End With
                        i += 1
                    Next
                Next
                writer.Close()
                strSQL = "SELECT * FROM " + kartan_nimi
                Dim da As New OleDb.OleDbDataAdapter(strSQL, con)
                Dim cb As New OleDb.OleDbCommandBuilder(da)
                Dim ds As New DataSet
                Dim dsNewRow As DataRow
                da.Fill(ds, kartan_nimi)
                dsNewRow = ds.Tables(kartan_nimi).NewRow()
                dsNewRow.Item(1) = "v" + uuden_kuidun_nimi
                dsNewRow.Item(2) = pKoko
                dsNewRow.Item(3) = pVariIndx
                ds.Tables(kartan_nimi).Rows.Add(dsNewRow)
                da.Update(ds, kartan_nimi)
            End If
        Next
        KartanTiedot()
End Sub
```

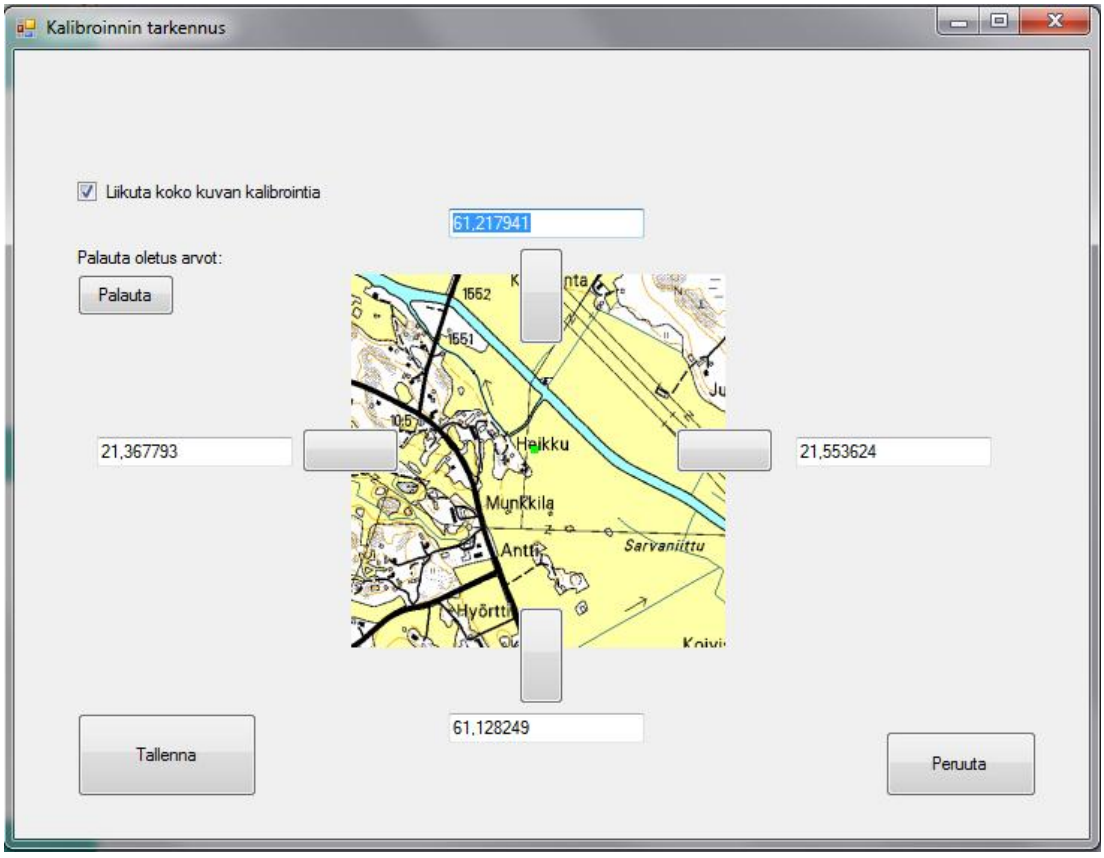
Kuva 22. Aliohjelma, jossa luetaan GPX-tiedosto ja tallennetaan se tietokantaan.

Lukeminen aloitetaan aliohjelmassa ”gpxTiedostoTauluun” avaamalla yhteys valittuun .gpx-tiedostoon. Seuraavaksi käytetään ensimmäistä kertaa hyväksi XSD-työkalun muokkaamaa GPX-skeematiedostoa. Ohjelma selvittää GPX-tyypin, mikä voi olla englanniksi waypoint, track tai route. Ohjelma tarkistaa samalla onko GPX-tiedosto oikeanmuotoinen (validi), jotta ohjelma ei kokonaan kaatuisi sen etsiessä reittipisteitä. Yhdessä GPX-tiedostossa voi olla monta reittiä tallennettuna, joten ohjelma etsii ne kaikki. Reittipisteet tallennetaan suoraan uuteen luotuun tiedostoon ohjelmalle rakennettuun tyyliin. Tietokantaan tallentuu uusi kuitu käyttäjän antaman nimen mukaan.

5 JATKOKEHITYS JA LOPPUSANAT

5.1 Jatkokehitys

Ohjelmaan on nyt lisätty kaikki alkuperäisen suunnitelman mukaiset päätoiminnot. Todellisuudessa Kuitu kartalle -ohjelma ei ole koskaan valmis. Aina löytyy ominaisuuksia, jotka lisäävät ohjelman monipuolisuutta ja käytännöllisyyttä. Käyttäjän ohjelman testaaminen antaa ohjelmoitiin suunnan, johon lähteä. Jatkokehityksen kannalta ohjelmaan tullaan lisäämään toimintoja, joita tilaaja haluaa. Ennen ohjelman luovutusta testikäyttöön lisäsin muutaman tärkeän toiminnon ohjelmaan. Esimerkiksi kalibroinnin tarkennus -toiminto ei kuulu alkuperäisiin päätoiminta vaatimuksiin, mutta on silti tärkeä lisä Kuitu kartalle -ohjelmaan. Kuvassa (Kuva 23) näkyy kalibroinnin tarkennusikkuna.



Kuva . Kalibroinnin tarkennus –ikkuna.

Kartat-valikosta voidaan valita kalibroinnin tarkennus. Kalibroinnin tarkennus mahdollistaa valitun kartan kalibroinnin muokkaamisen, jos se on mahdollisesti epätarkka.

5.2 Loppusanat

Kuitu kartalle-projekti oli ensimmäinen tekemäni isompi ohjelmakokonaisuus. Toisena opiskeluvuotenani suoritin Visual Basic -ohjelmoinnin opintojakson. VB.NET -ohjelmointi muuttuu kuitenkin nopeasti, joten vanhoista opeista ei ollut ratkaisevasti hyötyä. Kurssin viimeisenä työnä tein ohjelmointityöprojektina pelikoneen, jossa oli muutama VB-kielellä luomani peli. Pelikoneprojekti oli täysin eri koko- ja osaamis-luokkaa, kuin Kuitu kartalle -projekti. Pelikoneprojektissa silti tajusin, että VB.NET -ohjelmoinnissa mahdollisuudet ovat käytännössä rajattomat.

Ohjelmointiurakka oli pitkä ja vastoinkäymisiä riitti. Ohjelmoinnissa esiintyi ongelmia melkein jokaisena ohjelmointipäivänä. Jälkeenpäin ajateltuna voidaan todeta, että kaikki ongelmat voidaan ratkaista tai kiertää ohjelmointikielen monia eri vaihtoehtoja ja mahdollisuuksia käyttämällä. Tiedonhaku internetistä oli välttämättömyys, koska ohjelmointia opitaan jatkuvasti lisää samalla, kun sitä tehdään. Google ja internetin keskustelupalstat ovat ohjelmoijan apukeinoja, kun tulee vastaan asioita, joita ei ole koskaan joutunut ennen kohtaamaan tai jotka ovat jo unohtuneet.

LÄHTEET

Wikipedia: GPS Verkkosivut [www-dokumentti]. Viitattu 20.3.2011.
Saatavissa: http://fi.wikipedia.org/wiki/GPS

Halvorson, M. 2002. Microsoft Visual Basic.NET Trainers Kit. Helsinki.

Wikipedia: Access Verkkosivut [www-dokumentti]. Viitattu 6.4.2011.
Saatavissa: http://fi.wikipedia.org/wiki/Access

Geowiki: GPX Verkkosivut [www-dokumentti]. Viitattu 13.4.2011.
Saatavissa: http://geowiki.fi/GPX