Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka
Jyri Leppänen

Opinnäytetyö

Perustuskaivantojen mallintaminen ja siirto koneohjausjärjestelmään

Työn ohjaaja Lehtori Erkki Hietalahti
Työn teettäjä Ramboll Finland Oy, valvojana kehityspäällikkö Teemu Anttila

Tampere 48/2009

Tampereen ammattikorkeakoulu opinnäytetyö
Tietotekniikan koulutusohjelma, ohjelmistotekniikka

Tekijä	Jyri Leppänen
Työn nimi	Perustuskaivantojen mallintaminen ja siirto koneohjausjärjestelmään
Sivumäärä	42 sivua
Valmistumisaika	48/2009
Työn ohjaaja	lehtori Erkki Hietalahti
Työn tilaaja	Ramboll Finland Oy, valvojana kehityspäällikkö Teemu Anttila

TIIVISTELMÄ

Opinnäytetyön tarkoituksena oli suunnitella ja toteuttaa ohjelma tai ohjelmisto, jonka avulla rakennesuunnittelijat voivat Tekla Structures -suunnitteluohjelmaa käyttäessään mallintaa perustuskaivannot perustusten ympärille automatisoidusti. Mallinnuksen jälkeen perustuskaivannot tulee pystyä siirtämään Novatron Vision 3D -koneohjausjärjestelmään, jonka avulla kaivannot ja kaivuupinnat voidaan näyttää tietokoneen ruudulla työkoneen ajajalle.

Opinnäytetyön tuloksena valmistui kaksi ohjelmaa. Modeller suorittaa kaivantojen mallintamisen perustusten ympärille ja Exporter siirtää mallinnetut kaivannot LandXML-muotoiseen tiedostoon, josta ne voidaan lukea koneohjausjärjestelmään. Molemmat ohjelmat on toteutettu C#-ohjelmointikielellä käyttäen Microsoftin .NET-ohjelmistokomponenttikirjaston versiota 2.0. Opinnäytetyön viimeisessä vaiheessa piti muodostaa kolmioverkko maanpintaa kuvaavista muodoista. Tässä yhteydessä on käytetty Bojan Nicenon luomaa EasyMesh-kolmiointisovellusta ja Manchesterin yliopiston Polygon Clipper -sovellusta, jonka avulla suoritetaan monikulmioille yhdistämisoperaatioita. Molemmat ohjelmat on lisensoitu käyttöä varten. Opinnäytetyössä luotujen ohjelmien lähdekoodit on määritetty salaisiksi.

Koneohjausjärjestelmiä käytetään pääasiassa suurilla työmailla, sillä pienemmillä tai monimutkaisia rakennelmia sisältävillä työmailla kaivantojen mallintamisen kustannukset ovat korkeammat suhteessa saavutettuun hyötyyn. Opinnäytetyössä luotujen ohjelmien kehittämisen päätarkoitus oli tutkia Tekla Structuresin ja koneohjausjärjestelmien yhteiskäyttömahdollisuuksia ja löytää tapa luoda kaivannot automatisoidusti. Tämä mahdollistaisi koneohjausjärjestelmien käytön myös pienemmissä rakennushankkeissa.

Avainsanat perustukset, mallintaminen, maanrakennus, koneohjaus

TAMK University of Applied Sciences, Engineering Thesis

Information Technology

Software Engineering

Writer	Jyri Leppänen
Thesis	Modelling Foundation Excavations and Transferring into Machine Control System
Number of Pages	42 pages
Graduation time	50/2009
Thesis Supervisor	Erkki Hietalahti
Co-operating Company	Ramboll Finland Oy, Development Manager Teemu Anttila

---

ABSTRACT

The purpose of this engineering work was to design and implement a program or set of programs that could be used for modelling excavations around foundations in Tekla Structures models. These excavations should be parametric in terms of distances from foundations' sides and bottom to excavation.

Modelled excavations are written into XML-file in LandXML format, which then can be read using Novatron's Vision 3D Machine Control System. Vision 3D is used in earthworks machinery by touch screen computer and it shows bucket and machinery positions compared to modelled excavations in real time.

As a result of the project, two programs were developed. Modeller is used for modelling the excavations in Tekla Structures and Exporter is used to export the excavations into XML-file. Modeller uses external software called EasyMesh, which is created by Bojan Niceno. The function for EasyMesh is to triangulate soil plate in Tekla Structures model, which will make the view of excavations more understandable. Modeller also uses Polygon Clipper for uniting polygon and making it possible to use EasyMesh effectively. Polygon Clipper is created by Manchester University. Licenses for both programs have been acquired.

Main purpose for developing these programs was to test if it could be possible to use Tekla Structures and Vision 3D Machine Control System together. Vision 3D is currently used in large projects like motorways, but uniting TS and Vision 3D would make it possible to use Machine Control Systems also in smaller scale.

Keywords Foundations, Modelling, Earthworks, Machine Control Systems

ALKUSANAT

Tämä opinnäytetyö on tehty vuoden 2009 aikana Ramboll Finland Oy:lle. Haluan kiittää Ramboll Finlandin kehityspäällikkö Teemu Anttilaa työn ohjaamisesta ja Novatron Oy:n tuotekehityspäällikkö Visa Hokkasta opinnäytetyön toimeksiannosta.

Tampereella 27.11.2009

Jyri Leppänen

Sisällysluettelo

Termit ja lyhenteet

C#	C# on Microsoftin kehittämä ohjelmointikieli. Pääosa lopputyöstä on ohjelmoitu tätä käyttäen.
EasyMesh	EasyMesh on Bojan Nicenon kehittämä kolmiointisovellus, jonka avulla voidaan luoda monikulmio kolmioista.
Exporter	Exporter on toinen opinnäytetyön yhteydessä luoduista ohjelmista. Sen tehtävä on siirtää mallinnetut perustuskaivannot XML-tiedostoon. Pidempi nimitys ohjelmalle on Foundation Excavation Exporter.
Input Object	Plugin-tyyppisissä ohjelmissa käyttäjän valitsemia osia sanotaan input objekteiksi.
KKJ	KKJ on lyhennys sanasta kartastokoordinaatistojärjestelmä. KKJ on eräs Suomessa käytössä olevista kartastojärjestelmistä.
Komponentti	Komponentti on Tekla Structuresissa osa tai kooste, joka koostuu useista objekteista. Esimerkiksi liitos, joka koostuu levyistä, pulteista ja hitseistä.
LandXML	LandXML on LandXML-organisaation julkaisema tiedonsiirtostandardi eri teollisuusaloja varten.
Liitos	Liitos tarkoittaa kahden tai useamman kappaleen välille luotavaa yhdistettä. Esimerkiksi betonipilarin ja -palkin välille luotava liitos katkaisee pilarin ja palkin oikeasta kohdasta niin, että ne eivät mene sisäkkäin. Tämän jälkeen liitos luo rakenteisiin pulttien reiät, pultit ja vahvikkeet.
Modeller	Modeller on toinen opinnäytetyön yhteydessä luoduista ohjelmista. Sen tehtävä on mallintaa perustuskaivannot. Pidempi nimitys ohjelmalle on Foundation Excavation Modeller.
Plugin	Tekla Structuresin yhteydessä plugin tarkoittaa ohjelmatyyppiä, jonka käynnistyksen yhteydessä valitaan muokattavat kohteet.
Polygon Clipper	Polygon Clipper on Manchesterin yliopiston kehittämä ohjelma, jonka avulla voidaan tehdä erilaisia operaatioita

	monikulmioiden välillä, esimerkiksi yhdistää toisiaan leikkaavat monikulmiot.
Real Time Kinematic	Real Time Kinematic (lyhyesti RTK) /7/ on pääasiassa rakentamisessa ja maanmittauksessa käytettävä GPS-satelliittipaikannuksen mittausmenetelmä. Tavallisessa GPS-mittauksessa mitataan GPS-satelliittien lähettämän signaalin kulkuaikaa ja paikka määritetään tämän tiedon perusteella, jolloin tarkkuus on muutamien metrien luokkaa. RTK mittaa GPS-satelliittien signaalien vaihetta, jolloin paikannustarkkuus on muutama senttimetri. RTK-mittaus tarvitsee erillisen maa-aseman, joka mittaa GPS-satelliittien signaaleja ja välittää paikannustiedon sitä tarvitseville.
Tahko	Tahko on geometriassa monikulmio, jollaisista monitahokas koostuu. Esimerkiksi kuutiolla on kuusi tahkoa, jotka kaikki ovat neliön muotoisia.
Tekla Oyj	Tekla Oyj on yhdyskuntarakentamiseen liittyviä ohjelmistoja valmistava yritys.
Tekla Open API	Tekla Open API on Teklan kehittämä ohjelmointirajapinta, jonka avulla voi kehittää lisäsovelluksia tai lisätoimintoja Teklan valmistamiin ohjelmistoihin.
TS	TS on lyhennys sanoista Tekla Structures. Tekla Oyj:n kehittämä rakennuksen tietomallinnusohjelmisto. Lisää kohdassa 2.
UDA	UDA on lyhennys sanoista User Defined Attributes. Kaikille TS:n objekteille voidaan määrittää lisäasetuksia tämän avulla.
Vision 3D	Vision 3D on Novatron Oy:n kehittämä koneohjausjärjestelmä. Järjestelmän avulla työkoneen kuljettaja näkee näytön kautta työkoneen ja kauhan aseman työmaalla, jolloin esimerkiksi kaivuusyvyys voidaan todeta suoraan näytöltä. /6/

1 Johdanto

Ramboll Finland Oy (myöhemmin Ramboll) on monella alalla toimiva insinööri- ja konsultointiyritys, joka kuuluu Ramboll Group -konserniin. Talotoimialalla Ramboll tekee pääasiassa rakennesuunnittelua, mutta myös rakennussuunnittelua (= arkkitehtisuunnittelua) ja kiinteistönhallintaa.

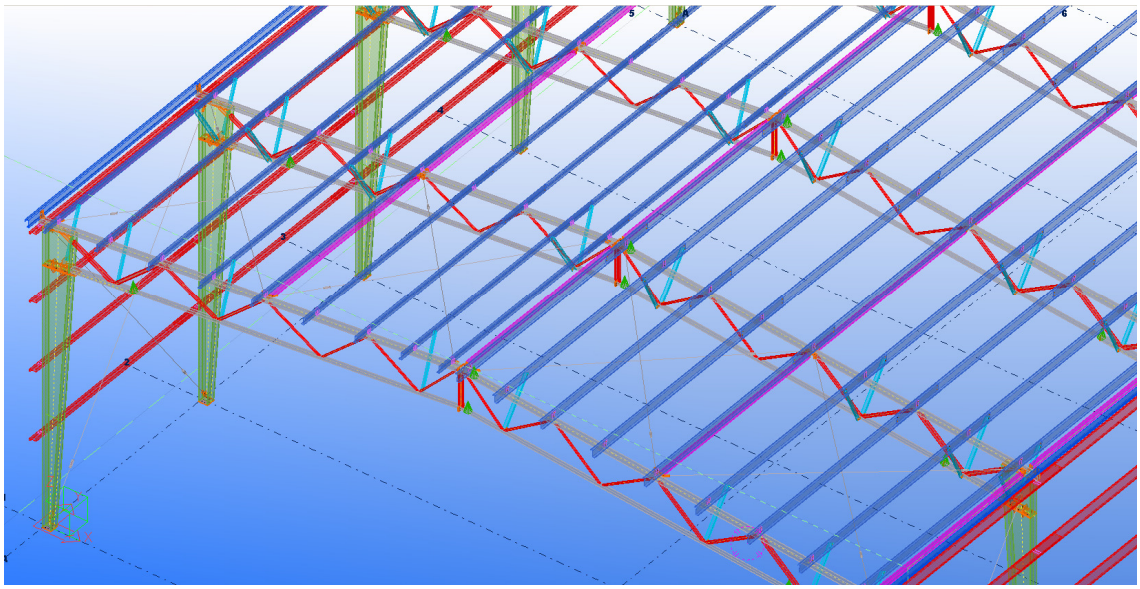
Työn tavoitteena oli tutustua Tekla Structures -rakennesuunnitteluohjelman tarjoamiin erilaisiin mahdollisuuksiin mallintaa perustuskaivannot mallissa olevien kaivanto-osien ympärille. Käyttäjän tulisi pystyä syöttämään kaivannoille parametrit, joiden perusteella kaivanto sijoitettaisiin tietylle etäisyydelle perustuksesta vaaka- ja pystysuunnassa. Kaivannon kaltevuus tulisi myös pystyä asettamaan. Ohjelman tulisi myös mallintaa maanpinta perustuskaivantojen ympärillä, jotta kaivantomallin lukeminen tietokoneen ruudulta helpottuisi.

Mallinnuksen jälkeen kaivannot tulisi pystyä siirtämään Vision 3D-koneohjausjärjestelmään. Siirto tapahtuu kirjoittamalla kaivantojen tiedot XML-tiedostoon, joka luetaan Vision 3D:llä.

Luvuissa 2 ja 3 käsitellään opinnäytetyöhön liittyvät ohjelmistot ja kerrotaan työn taustoista. Luku 4 käsittelee ohjelmat, joiden avustuksella mallinnetaan maanpinta kolmioista. Luvuissa 5 ja 6 käydään läpi pääperiaatteet opinnäytetyön tuloksena syntyneistä mallinnus- ja siirto-ohjelmista.

2 Tekla Structures

Tekla Structures on Tekla Oyj:n kehittämä tietomallinnuspohjainen rakennesuunnitteluohjelmisto, jonka avulla voidaan erittäin suurella tarkkuudella mallintaa erilaisia rakennuskohteita. Mallintamisessa perusosat, esimerkiksi pilarit, palkit ja laatat, mallinnetaan pääsääntöisesti osista, jotka nimetään ja numeroidaan tiettyä logiikkaa noudattaen. Kuva 1 esittää kuvaruutukaappauksen teräsrakenteisesta hallista.



Kuva 1. Teräshalli Tekla Structures -mallissa.

Mallista voidaan luoda neljän tyyppisiä piirustuksia:

- Yleispiirustukset (General arrangement drawing), jotka sisältävät näkymiä malliin. Esimerkiksi teräs- ja betonirakenteiden asennuskaaviot, paikallavalurakenteiden piirustukset ja rakennetyyppien kuvaaminen tehdään yleispiirustuksilla. Myös perustuskaivantojen piirustukset tehtäisiin yleispiirustuksina.
- Teräs- ja puurakenteiden kokoonpanopiirustuksia (Assembly drawing), joissa esitetään osien tunnukset ja liittyminen toisiinsa.
- Teräs- ja puurakenteiden osapiirustukset (Single part drawing), joissa esitetään osan valmistamiseksi tarvittava tieto.
- Betonielementtien piirustukset (Cast unit drawing), joissa esitetään elementin valmistamiseksi tarvittava mitta- ja raudoitustietous. Esimerkki (kuva 19, liite 1).

Rakennuksen tietomallia käytetään nykyään pääasiassa rakenteista ja rakenneosista tehtävien yleis- ja valmistuspiirustusten tuottamiseen. Opinnäytetyössä käytössä on Tekla Structuresin versio 15.0 ja sen johdannaiset.

Mallintamisen nopeuttamiseksi ohjelman käyttäjällä on käytettävissään erilaisia komponentteja, joita on kolmea eri perustyyppiä:

1) Systeemikomponentit
- Systeemikomponentit ovat Teklan ohjelmoimia liitoksia, detaljeja tai makroja, jotka on käännetty exe-tiedostoiksi applications-hakemistoon. Liitosta käyttäessä valitaan pääosa ja toissijainen osa, detaljeja käyttäessä pääosa ja sijainti. Makroilla on yleensä toteutettu laajempia kokonaisuuksia tai esimerkiksi tekstitiedostoformaatteihin perustuvia export-työkaluja.

2) Custom componentit
- Custom componentit ovat parametrisointiin perustuvaa mallintamistekniikkaa käyttäen tehtyjä komponentteja (liitokset, detaljit, käyttäjän osat), joita voi tarvittaessa viedä mallista uel-tiedostoon ja tuoda toiseen malliin. TS:n asennuksen yhteydessä tulee custom component -kirjasto ja komponentteja voi ladata lisää Tekla Structuresia käyttävien yritysten internetsivuilta.

3) Ohjelmoidut sovellukset
- Vanhemmat sovellukset on toteutettu C-kielisellä rajapinnalla ja käännetty exe-tiedostoiksi. Uudemmat sovellukset on toteutettu käyttäen Tekla Open APIa, josta lisää kohdassa 2.1.

2.1 Tekla Open API -rajapinta ja ohjelmakehitys

Tekla Open API /3/ on Teklan kehittämä ohjelmointirajapinta Tekla-tuoteperheen ohjelmistoja varten. Open API -rajapinta perustuu Microsoftin .NET-tekniikan versioon 2.0. Rajapinta koostuu dll-kirjastoista, jotka sisällytetään sovelluksen projektiin viitteenä (reference). Rajapinnan käyttö mahdollistaa lisäsovellusten kehittämisen tukemaan oman yrityksen tarpeita. Tekla Open Api -rajapinnan kautta sovellukset voivat pyytää tietoa TS:stä ja auki olevasta mallista sekä muokata mallia. Opinnäytetyösovellukset perustuvat Tekla Open APIn kautta tapahtuvaan mallin tietojen käsittelyyn.

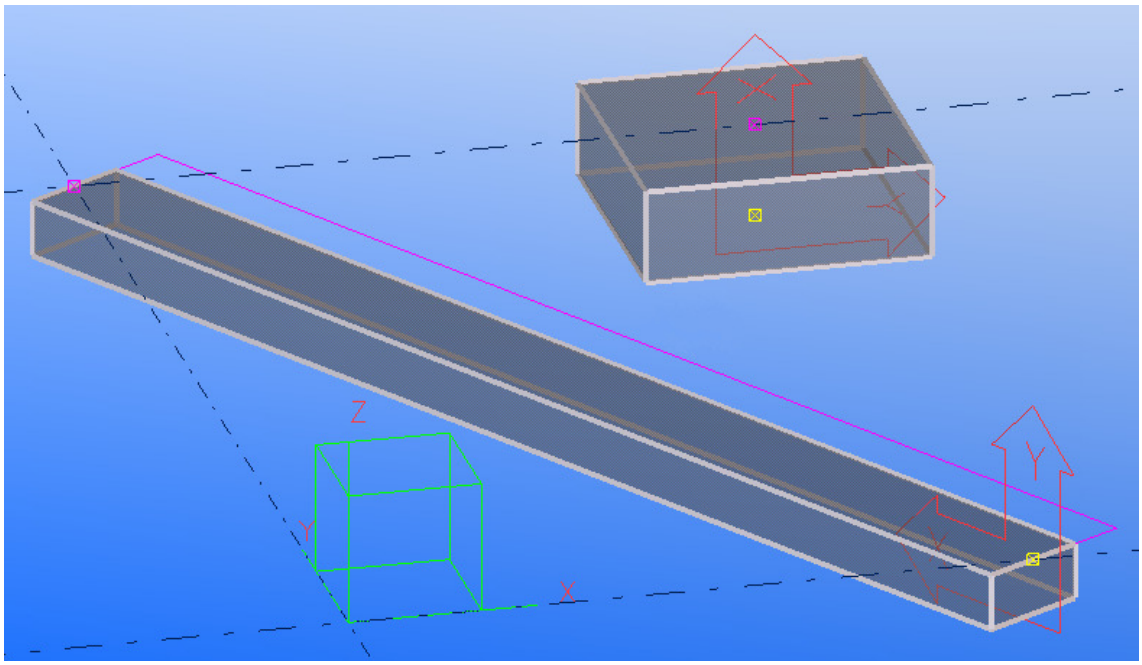
Tekla Open APIlla Luotavat sovellukset periytetään kahdesta perustyypistä, jotka ovat PluginBase ja ConnectionBase. PluginBase-luokkaa käyttäen voidaan luoda erittäin monipuolisia sovelluksia, sillä se toteuttaa DefineInput-metodin, jolla siirretään tietoa TS:n ja sovelluksen välillä. ConnectionBase-luokan avulla voidaan toteuttaa kahden tai useamman rakenneosan välisiä liitoksia, mutta input-osien valinta on rajoitetumpaa,

sillä luokkaan ei ole määritetty DefineInput-metodia. ConnectionBase tarjoaa joitain liitosten perusominaisuuksia valmiina ilman ohjelmointia, kuten esimerkiksi liitoksen sisäisen koordinaattijärjestelmän.

Opinnäytetyössä luodut ohjelmat on kehitetty käyttäen Microsoftin Visual Studio 2008 Express Edition -ohjelmankehitysympäristöä. Molemmat ohjelmat on käännetty dll-kirjastoiksi, jolloin ohjelman asentaminen ja käyttäminen tapahtuu siirtämällä dll-tiedosto oikeaan hakemistoon. Tekla Structures osaa käyttää dll- ja exe-tiedostoiksi käännettyjä ohjelmia.

2.2 Kappaleiden koordinaattijärjestelmät ja muunnokset niiden välillä

Kaikilla TS:n kappaleilla on oma karteesinen koordinaatistonsa, jonka x-akseli on kappaleen alkupisteestä loppupisteeseen suuntautuvan vektorin suuntainen (Kuva 2). Alkupiste on merkitty keltaisella neliöllä ja loppupiste pinkillä neliöllä. Y-akseli on kohtisuorassa x-akseliin nähden. Mallin globaali koordinaatisto on nähtävissä kuvan vasemmassa laidassa.



Kuva 2. Erilaisia koordinaatistoja.

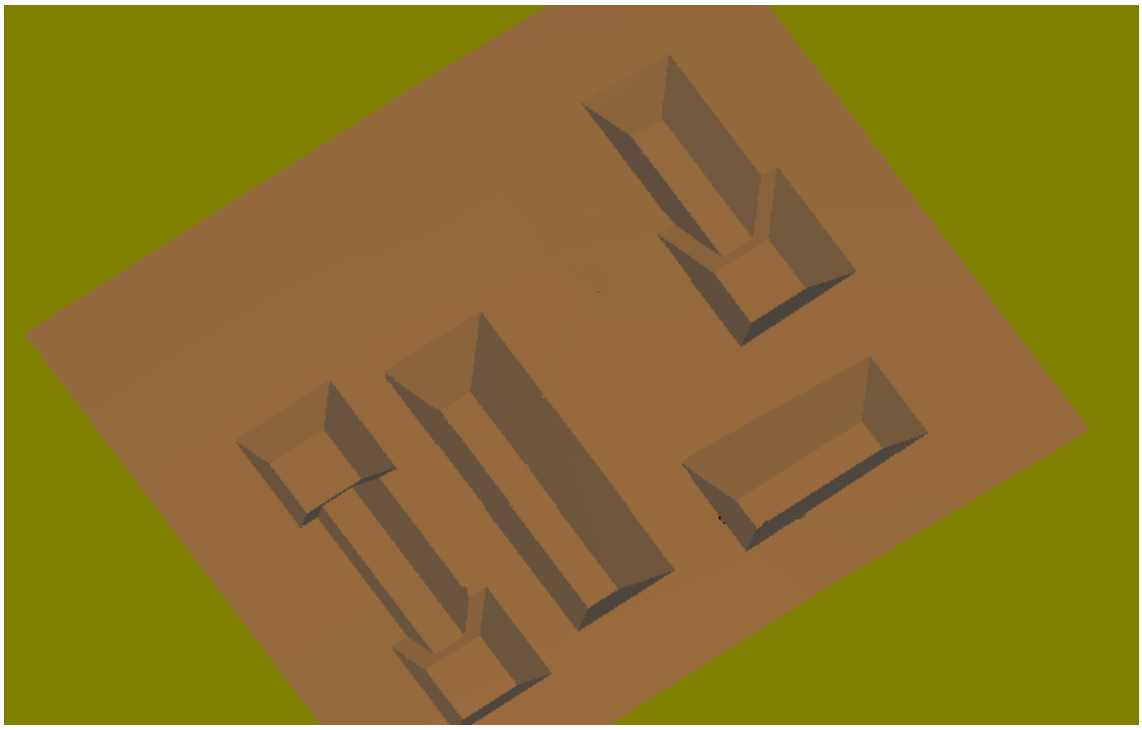
Opinnäytetyön eri vaiheissa kappaleilta pyydetään esimerkiksi niiden kulmapisteitä. Jos halutaan tutkia kahden eri kappaleen pisteitä toisiinsa nähden, tulee se tehdä samassa koordinaatistossa. Tällöin tulee tehdä koordinaatistomuunnos. Alla olevassa esimerkissä (ohjelmalistaus 1) luodaan matriisi, jonka avulla muunnetaan piste perustuksen omasta koordinaattijärjestelmästä globaaliin järjestelmään.

```
// CSys = short for coordinatesystem
Matrix FoundationCSysToGlobalCSys =
      MatrixFactory.ByCoordinateSystems(FoundationCoordinateSystem, GlobalCSys);

Point oStartPointInGlobalCSys = new Point(FoundationCSysToGlobalCSys.Transform(oPoint));
```

Ohjelmalistaus 1. Koodinaatistomuunnos.

3 Novatron Vision 3D -koneohjausjärjestelmä

Vision 3D on Novatron-ohjelmistoyrityksen valmistama koneohjausjärjestelmä, jonka avulla työkoneen kuljettaja näkee halutut kaivuupinnat suoraan työkoneessa tietokoneen ruudulta. Tämä vähentää tarvetta mittamiehille ja nopeuttaa työskentelyä etenkin isoilla työmailla. Kuva 3 on kuvanruutukaappaus Vision3D:n simulaattorista, johon on syötetty kuvitteellisen työmaan perustuskuvat. Käyttötilanteessa Vision3D näyttäisi ruudulla tietoa esimerkiksi kaivinkoneen ja kauhan sijainnista, mutta nämä tiedot on jätetty kuvasta pois havainnollisuuden vuoksi.



Kuva 3. Vision 3D kuvankaappaus

3.1 Toiminta

Vision 3D -koneohjausjärjestelmä koostuu työkoneeseen (esimerkiksi kaivinkone) asennetuista antureista, tietokoneyksiköstä ja kosketusnäytöstä. Paikannukseen järjestelmä käyttää satelliittipaikannusta (GPS, GLONASS) tai robottitakymetriä. Satelliittipaikannuksessa hyödynnetään RTK-korjaus (Real Time Kinematic), jolla saavutetaan tarvittava muutaman senttimetrin tarkkuus. Järjestelmään syötetään halutun työkohteen kaivuupinnat joko langattomasti FTP-palvelimelta tai muistitikulta. Tietokoneyksikkö laskee antureiden ja paikannustietojen perusteella työkoneen sijainnin ja asennon työmaalla ja näyttää sen näytöllä työkoneen ajajalle. Näytön tietojen perusteella maata voidaan työstää muutamien senttien tarkkuuteen asti. Erilaisilla lisävarusteilla

järjestelmä osaa mitata mm. kahden suunnan kallistuksia tai kauhan sivuttaiskallistuksen.

Yleisimpiä työkohteita koneohjausjärjestelmille ovat isot infrapuolen työmaat, esimerkiksi moottoritiet, joissa yleislinjat ovat selkeitä ja koneohjausjärjestelmä tarjoaa parhaan hyödyn alkukustannuksiin nähden. Pienissä työmaakohteissa maanpinnan mallintamisen kustannusten merkitys saatavaan hyötyyn pienenee oleellisesti.

3.2 LandXML-tiedonsiirtostandardi

Vision 3D -koneohjausjärjestelmä käyttää LandXML:n mukaista määrittelyä lukiessaan kaivantodatan tiedostosta. Opinnäytetyössä on käytetty aluemaisten rakenteiden kuvaushierarkiaa. /1/ Tällöin pinnat muodostetaan kolmioista, jotka kierretään vastapäivään. Perusperiaate kuvauksesta on esitetty seuraavassa:

Tiedoston alussa määritetään käytettävä kuvauksen versionumero ja merkistö, käytettävät yksiköt sekä joitain tunnistetietoja (ohjelmalistaus 2).

```
<?xml version="1.0" encoding="Windows-1252"?>
<LandXML version="1.0" date="28.07.2009" time="12:25:56"
xmlns="http://www.landxml.org/schema/LandXML-1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.landxml.org/schema/LandXML-1.0
http://www.landxml.org/schema/LandXML-1.0/LandXML-1.0.xsd">
  <Units>
    <Metric areaUnit="squareMeter" linearUnit="meter" volumeUnit="cubicMeter"
temperaturUnit="celcius" pressureUnit="milliBars" />
  </Units>
  <Application name="Foundation Excavation Modeller" manufacturer="Ramboll Finland Oy"
manufacturerURL="http://www.ramboll.fi" />
```

Ohjelmalistaus 2. LandXML-tyyppisen tiedoston alkumäärittely

Seuraavaksi suoritetaan pintojen määrittely. Tämä tapahtuu listaamalla kaikki mallissa olevien kaivantojen kulmapisteet Pnts-elementin sisään. Pisteistä kerrotaan tunniste, sijainti leveys- ja pituussuunnassa sekä korkeus merenpinnasta. Ohjelmalistaus 3 ja ohjelmalistaus 4 esittävät, kuinka LandXML-formaatissa määritellään yksinkertainen kaivanto. Kuva 4 esittää määritellyn kaivannon Tekla Structures -mallissa ja Vision 3D-simulaattorissa. TS-mallissa on nähtävissä myös perustus, jonka perusteella kaivanto on mallinnettu.

```
<Surfaces>
    <Surface name="Alin Pinta">
      <Definition surftype="TIN" elevMin="5" elevMax="6">
        <Pnts>
          <P id="1" xmlns="">6900013.25 3500020.35 -0.9</P>
          <P id="2" xmlns="">6900010.75 3500020.35 -0.9</P>
          <P id="3" xmlns="">6900010.75 3500022.85 -0.9</P>
          <P id="4" xmlns="">6900013.25 3500022.85 -0.9</P>
```

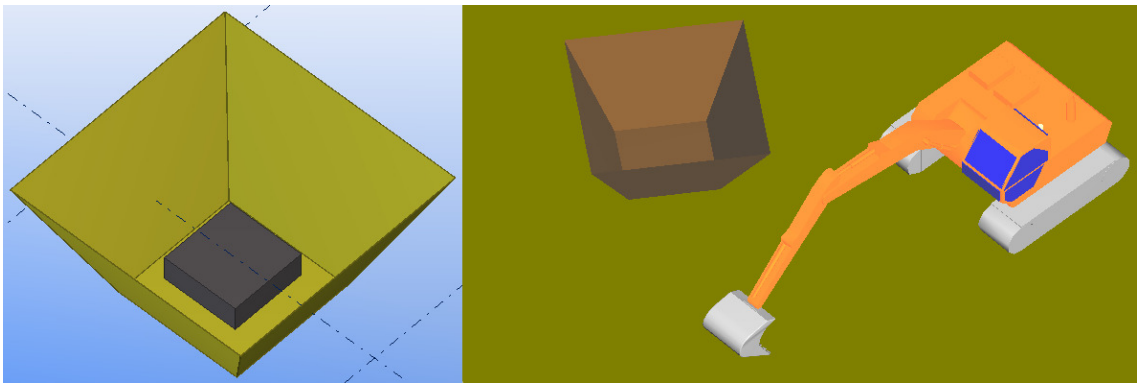
```
          <P id="5" xmlns="">6900009.906 3500019.506 2.25</P>
          <P id="6" xmlns="">6900009.906 3500023.694 2.25</P>
          <P id="7" xmlns="">6900010.75000003 3500022.85 -0.899999983817249</P>
          <P id="8" xmlns="">6900010.75000003 3500020.35 -0.899999983817249</P>
          <P id="9" xmlns="">6900009.906 3500023.694 2.25</P>
          <P id="10" xmlns="">6900014.094 3500023.694 2.25</P>
          <P id="11" xmlns="">6900013.25 3500022.84999997 -0.899999983819131</P>
          <P id="12" xmlns="">6900010.75 3500022.84999997 -0.899999983819131</P>
          <P id="13" xmlns="">6900014.094 3500023.694 2.25</P>
          <P id="14" xmlns="">6900014.094 3500019.506 2.25</P>
          <P id="15" xmlns="">6900013.24999997 3500020.35 -0.899999983817249</P>
          <P id="16" xmlns="">6900013.24999997 3500022.85 -0.899999983817249</P>
          <P id="17" xmlns="">6900014.094 3500019.506 2.25</P>
          <P id="18" xmlns="">6900009.906 3500019.506 2.25</P>
          <P id="19" xmlns="">6900010.75 3500020.35000003 -0.89999998381913</P>
          <P id="20" xmlns="">6900013.25 3500020.35000003 -0.89999998381913</P>
        </Pnts>
```

Ohjelmalistaus 3. Pisteiden määrittely LandXML-formaatissa

Lopuksi määritetään kolmiot pisteiden tunnisteiden perusteella ja suljetaan tiedosto (ohjelmalistaus 4). Esimerkiksi ensimmäinen kolmio muodostuu pisteistä 4, 3 ja 2.

```
<Faces>
          <F xmlns="">4 3 2</F>
          <F xmlns="">4 2 1</F>
          <F xmlns="">8 7 6</F>
          <F xmlns="">8 6 5</F>
          <F xmlns="">12 11 10</F>
          <F xmlns="">12 10 9</F>
          <F xmlns="">16 15 14</F>
          <F xmlns="">16 14 13</F>
          <F xmlns="">20 19 18</F>
          <F xmlns="">20 18 17</F>
        </Faces>
      </Definition>
    </Surface>
  </Surfaces>
</LandXML>
```

Ohjelmalistaus 4. Kolmioiden rakentaminen kulmapisteiden perusteella LandXML-formaatissa



Kuva 4. Yksinkertainen kaivanto TS-mallissa ja koneohjausjärjestelmän simulaattorissa.

4 Opinnäytetyössä käytetyt avustavat ohjelmistot

Kaikki koneohjausjärjestelmälle välitettävä tieto perustuskaivannoista ja maanpinnasta tulee muodostaa kolmioista. Tätä varten opinnäytetyössä on käytetty kahta ulkopuolista ohjelmakomponenttia mahdollistamaan ohjelmiston toteutus ja parantamaan lopputuloksen laatua. Kaivantojen mallintamiseen liittyvä maanpinnan kolmioverkon luominen on toteutettu käyttämällä Bojan Nicenon luomaa EasyMesh-sovellusta /4/. EasyMesh-sovellus osaa kolmioida n-kulmaisen aukkoja sisältävän monikulmion.

Jotta EasyMesh toimisi oikein, eivät monikulmion sisällä olevat aukot saa olla vierekkäin. Käytännön elämässä yksittäiset perustuskaivannot ovat usein vierekkäin, joten näiden kaivantojen maanpintaan luomat aukot tulee yhdistää toisiinsa. Aukoista luodaan polygonit, jotka yhdistetään toisiinsa Manchesterin yliopiston luoman Polygon Clipper -sovelluksen avulla /5/

Seuraavissa aliluvuissa käsitellään tarkemmin molempien sovellusten käyttökohteet ja käyttöperiaatteet.

4.1 Kolmioinnin toteutus EasyMeshin avulla

Kolmioinnilla tarkoitetaan pinnan muodostamista erisuuruisista kolmioista. Opinnäytetyössä Tekla Structures -mallissa olevat maanpintalevyt kolmioidaan niin, että kaivantojen kohdalle jätetään aukot. Kolmiointia varten selvitetään maanpintalevyjen kulmapisteet ja maanpintalevyjä leikkaavien kaivantojen kulmapisteet. Jos havaitaan, että kaivannot ovat vierekkäin, näiden kaivantojen kulmapisteistä luodaan polygonit, jotka yhdistetään toisiinsa. Näin saadaan selvillä useamman vierekkäisen kaivannon muodostaman reiän kulmapisteet.

Maanpintalevyn kulmapisteet muodostavat kolmioinnille ulkoreunat, niin sanotun boundaryn, joka määritetään EasyMeshin input-tiedostoon ensimmäiseksi (Ohjelmalistaus 5). Boundaryn pisteet määritetään vastapäivään kiertäen. Tämän jälkeen input-tiedostoon määritellään mahdolliset reiät boundaryn sisällä vastapäivään kiertäen. Pisteiden määrittelysyntaksi on <*juokseva järjestysnumero*> <*pisteen X-arvo*> <*pisteen Y-arvo*> <*kolmion koko*> <*marker*>. Kolmion koko kertoo EasyMeshille, kuinka suuria kolmioita käyttäjä haluaa lopputulokseksi. Markerin avulla

pisteitä ja segmenttejä voidaan liittää tiettyyn haluttuun boundaryyn, mutta opinnäytetyössä tälle ei ollut tarvetta, joten jokainen boundary ja reikä saa oman marker-arvonsa.

```
#=========
| POINTS |
=========#
12 # number of points #

# Nodes which define the boundary #
0:   0   0   11   1
1:   20   0   11   1
2:   20   20   11   1
3:   0   20   11   1

# Nodes which define the hole #
4:   4   4   11    2
5:   4   8   11    2
6:   8   8   11    2
7:   8   4   11    2
8:   10   10   11    3
9:   10   15   11    3
10:  15   15   11    3
11:  15   10   11    3
```

Ohjelmalistaus 5. Pisteiden määrittely EasyMeshiä varten

Pisteiden määrittämisen jälkeen tiedostoon määritetään boundaryn ja reikien segmentit. Segmentti määritetään kertomalla, mitkä pisteistä yhdistetään toisiinsa. Myös segmentit eritellään juoksevalla numerolla, jonka jälkeen määritetään, minkä pisteiden välillä segmentti on. Lopuksi annetaan marker-arvo, joka täsmää pistemäärittelyn kanssa.

```
#===========
| SEGMENTS |
===========#
12 # Number of segments #

# Boundary segments #
0:   0   1   1
1:   1   2   1
2:   2   3   1
3:   3   0   1

# Hole segments #
4:   4   5   2
5:   5   6   2
6:   6   7   2
7:   7   4   2
8:   8   9   3
9:   9   10   3
10:   10   11   3
11:   11   8   3
```

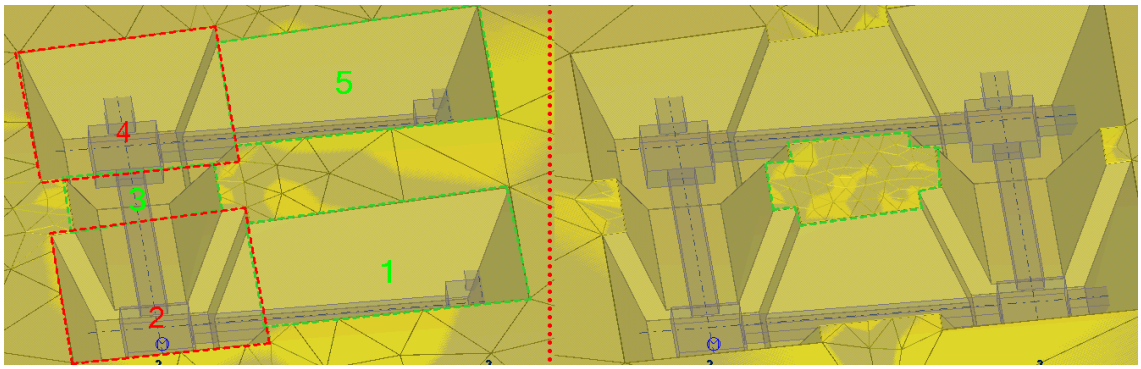
Ohjelmalistaus 6. Segmenttien määrittely EasyMeshiä varten.

4.2 Polygonien väliset operaatiot Polygon Clipperillä

Polygon Clipperin avulla voidaan tehdä erilaisia operaatioita kahden monikulmion välillä. Operaatioiden avulla voidaan selvittää polygonien erotus, yhdistelmä, leikkausala ja Xor (eksklusiivinen tai). Opinnäytetyössä käytetään yhdistys-operaatiota, jonka avulla luodaan vierekkäisistä kaivanto-polygoneista yksi polygoni. Kuvassa Kuva 5 vasemmalla on esitetty, miten kaivanto koostuu viidestä kaivanto-osasta, ja millä tavalla kaivannot muodostavat reikiä maanpintaan. Näistä rei'istä muodostetut polygonit

yhdistetään toisiinsa Polygon Clipperillä , jonka jälkeen EasyMesh voi suorittaa kolmioinnin.

Jos perustuksista luodut kaivannot muodostavat yhtenevän ympyrän (kuva 5, oikea puoli), luodaan kaivantojen keskelle jäävästä maanpinnan osasta oma maanpintalevyä vastaava osa, joka kolmioidaan sellaisenaan.



Kuva 5. Perustuskaivantojen maanpintalevyyn muodostamat aukot.

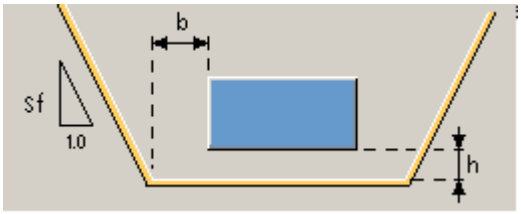
5 Modeller – kaivantojen mallinnus

Modeller on toinen opinnäytetyön aikana luoduista ohjelmista. Sen tehtävä on mallintaa perustuskaivannot käyttäjän valitsemien perustusten ympärille. Kaivantojen alapinta määräytyy perustuksen alapinnan mukaan. Kaivannot katkaistaan yläosastaan suunnittelijan TS-malliin asettamilla maanpintalevyillä. Levyllä tai levyillä on tarkoitus luoda yksinkertainen malli maanpinnan muodoista, jolloin työkoneen kuljettajan on helpompi hahmottaa kaivannot työmaalla tietokoneen ruudulta. Maanpintalevyt kolmioidaan EasyMesh-sovelluksella, jolloin myös ne voidaan siirtää Exporterin avulla koneohjausjärjestelmään. Mallintajan toiminta on monivaiheinen prosessi, ja se käydään yksityiskohtaisesti läpi luvussa 5.6. Merkittävä osa opinnäytetyöhön käytetystä ajasta on kulunut Modellerin suunnitteluun ja toteutukseen.

5.1 Määrittely

Projektin alussa määritellyt ja sen edetessä tarkentuneet vaatimukset mallintajalle olivat seuraavat:

- Ohjelma kykenee mallintamaan Tekla Structures -mallissa olevien valittujen perustuskomponenttien ympärille kaivupohjamallin.
- Kaivuvara määritetään käyttäjältä pyydettävien parametrien perusteella.
- Vaadittavat parametrit on esitetty alla (Kuva 6). h-parametrin avulla säädetään perustuksen alle tulevan murskepinnan korkeus, b-parametri on työstövara sivusuunnassa ja Sf (Slope Factor) määrittää kaivannon seinämän jyrkkyyden asteina.



Kuva 6. Mallintajalta vaaditut parametrit.

- Kaivantojen korkeus määrätään mallintamalla maanpinta graafisesti esimerkiksi levyistä tai asettamalla käyttöliittymässä maksimikorkeus kaivannoille.
- Levyistä muodostettu maanpinta kolmioidaan niin, että kaivantojen kohtiin jää aukot.

5.2 Yleistä

Modeller on Plugin-tyyppinen ohjelma, joka periytyy Tekla Open APIn tarjoamasta abstraktista PluginBase-luokasta. Pluginit tarjoavat mahdollisuuden luoda ohjelman, jonka avulla käyttäjä voi valita avoimesta mallista tietyn määrän tietyntyyppisiä osia, joille valinnan jälkeen suoritetaan halutut toimenpiteet. PluginBase-luokasta on toteutettava metodit DefineInput()- ja Run() -metodit.

DefineInput-metodi käynnistyy, kun käyttäjä käynnistää pluginin suorituksen klikkaamalla sitä hiiren osoittimella. Metodi tallentaa List-tyyppiseen tietorakenteeseen jokaisen käyttäjän valitseman objektin, ns. input-objektin, Picker-olion avulla. DefienInput-metodin toteutus on esitetty ohjelmalistauksessa 7. Käyttäjä lopettaa valitsemisen hiiren keskinapilla, jonka jälkeen suorituksessa siirrytään Run-metodiin ja ohjelman ajo alkaa. Tästä kerrotaan lisää kohdassa 5.6.

```
public override List<InputDefinition> DefineInput()
{
    try
    {
        Picker oFoundationPicker = new Picker();
        List<InputDefinition> inputList = new List<InputDefinition>();
        ModelObjectEnumerator object1 =
            oFoundationPicker.PickObjects(Picker.PickObjectsEnum.PICK_N_OBJECTS);

        foreach (ModelObject o in object1)
        {
            InputDefinition input1 = new InputDefinition(object1.Current.Identifier);
            inputList.Add(input1);
        }
        return inputList;
    }
    catch (Exception e)
    {
        if (!e.Message.Equals("User interrupt"))
        {
            MessageBox.Show(e.StackTrace + "\n" + e.ToString());
        }
        List<InputDefinition> inputList = new List<InputDefinition>();
        return inputList;
    }
}
```

Ohjelmalistaus 7

Run-metodi on Plugin-tyyppisen ohjelman niin sanottu "main-loop". Kaikki aliohjelmat suoritetaan kyseisessä metodissa. Metodin parametrina on lista käyttäjän valitsemista objekteista. Modeller käy for-silmukassa läpi kaikki käyttäjän valitsemat objektit.

Kun plugin on suoritettu läpi onnistuneesti, tuloksena on valittujen perustusten ympärille muodostuneet kaivannot. Kaivannoista muodostuu aina yksi komponentti, jonka parametreja voidaan muuttaa vapaasti jälkikäteen, mutta komponentista ei voida poistaa tai siihen lisätä uusia perustuksia. Jos suunnittelija haluaa jälkikäteen lisätä perustuksia, tulee vanha komponentti poistaa ja luoda uusi.

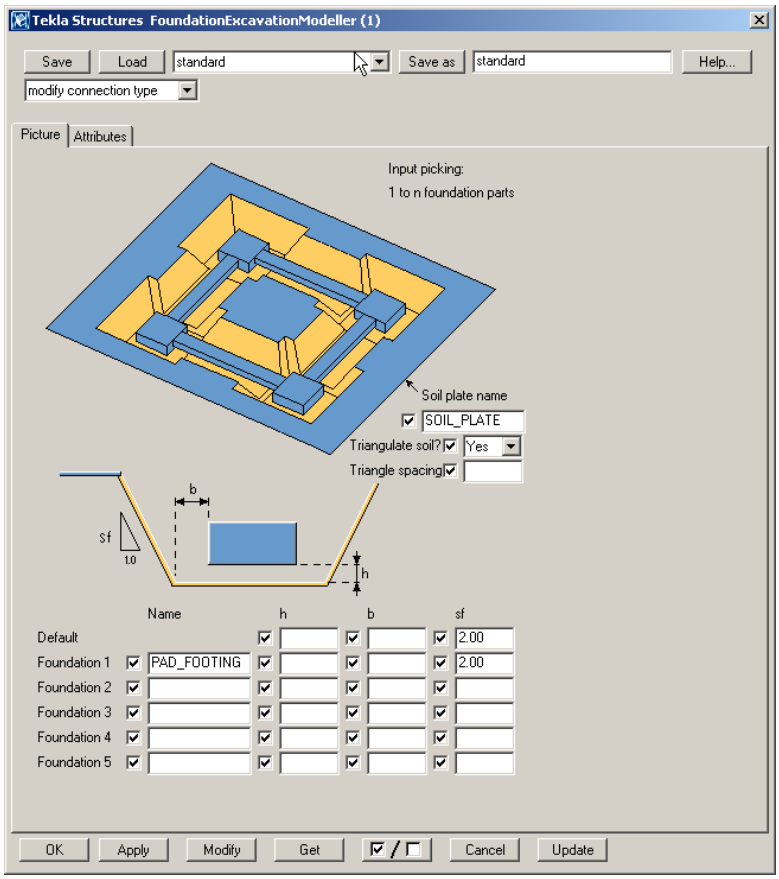
5.3 Käyttöliittymä

Plugin-ohjelmien käyttöliittymä rakennetaan Tekla Structuresin oman inp-tiedostotyypin avulla. Inp-tiedosto on tekstitiedosto, johon määritetään käyttöliittymän osat ja niiden sijainnit tietyn rakenteen mukaan. Modellerin käyttöliittymään on määritetty välilehtiä, tietueita, ja tietokenttiä erilaisia käyttäjältä pyydettäviä arvoja varten. Alla (ohjelmalistaus 8) on yksinkertaistettu versio modellerin käyttöliittymästä. Kommenteissa on kerrottu yksityiskohtaisemmin jokaisen erityyppisen käyttöliittymänosan syntaksi. Kuva 7 esittää modellerin valmiin käyttöliittymän.

```
/* Koska inp-tiedosto on määritetty merkkijonona, tulee kommentoidessa käyttää /* */ -
tyyppisiä kommentteja */
/// this is a nested class containing the UI definition for plug-in component
/// the format for UI definition is same as for custom components
public class UserInterfaceDefinitions
{
      // Määritetään käyttöliittymä vakiomerkkijonona
      public const string dialog =
      @"page(""TeklaStructures"",""""")
      {
      /* Kerrotaan ohjelman tyyppi ja nimi */
      plugin(1, ""FoundationExcavationModeller"")
            {
      /* Määritetään välilehti ja sen järjestysnumero */
            tab_page("""", ""Välilehden_nimi"", 1)
                  {
                        /* Kuva, sen asema x- ja y-akseleilla ja varattava tila */
                        picture (""Näytettävä_kuva"",500, 300, 20, 0)

                        /* Attribuuttien formaatti on:
                        Nimi, Teksti, tyyppi, formaatti (c-kielen tyyppisesti),
                        mahdolliset erikoisliput,tarkistetaanko mini- tai maksimiarvo,
                        minimi- ja maksimiarvot, sijainti ja pituus
                        */

                        /* Nimiötyyppinen attribuutti, joka näyttää tekstin
                        määritetyssä kohdassa merkkijonona */
                        attribute(""label"", ""Input picking:"", label,""%s"", no, none,
                        ""0.0"", ""0.0"",425,0,100)
                        attribute(""label"", ""Soil plate name"", label,""%s"", no,
                        none, ""0.0"", ""0.0"",463,230,100)

                        /* Parametrin avulla luodaan string-tyyppinen tietokenttä, joka
                        käsittelee tekstiä. Tietokentän Sijainti ja pituus määritetään
                        kutem attribuutissa. Parametrin avulla tallennetaan käyttäjän
                        syöttämä tieto muuttujaan. */
                        parameter("""", ""Muuttujan_nimi"", string, text, 463, 255, 120

                        /* Kyllä/ei tyyppinen valinta, ohjelma käsittelee valintaa
                        numerona. Kysymys esitetään attribuutilla ja vastaus
                        tallennetaan parametrin avulla */
                        parameter("""", ""Kolmiointi_muuttuja"", YesNo, number, 510,
                        280, 70)
                        attribute(""CreatePlatesLabel"", ""Kolmioidaanko?"",
                        label,""%s"", none, none, ""0.0"", ""0.0"",380,280,100)
```

```
                }

            /* Määritetään toinen välilehti */
            tab_page("""", ""Toisen_välilehden_nimi"", 2)
                {
                    /* Pyydetään käyttäjältä kolme eri arvoa ja tallennetaan
                    vastaaviin muuttujiin */
                    attribute(""label"", ""Material"", label,""%s"", no, none,
                     ""0.0"", ""0.0"",490,0,60)
                    attribute(""label"", ""Name"", label,""%s"", no, none, ""0.0"",
                     ""0.0"",610,0,60)
                    attribute(""label"", ""Class"", label,""%s"", no, none, ""0.0"",
                     ""0.0"",725,0,60)

                    parameter("""", ""ExcavationMaterial"", material, text, 480, 30,
                    70)
                    parameter("""", ""ExcavationName"", string, text, 600, 30, 70)
                    parameter("""", ""PartClass"", integer, number, 715, 30, 70)

                    parameter("""", ""CreatePlates"", YesNo, number, 300, 130, 150)
                    attribute(""CreatePlatesLabel"", ""Create excavation of
                    plates"", label,""%s"", none, none, ""0.0"", ""0.0"",15,130)
                }

            depend(2) /* Määrittää, suoritetaanko pluginin toimenpiteet uudestaan, jos
                         jokin input-objekti muuttuu */
            modify(1) /* Asetetaan käyttöliittymään modify-nappi, jonka avulla luotua
                         komponenttia/pluginia/vastaavaa) voidaan muokata myöhemmin */
            draw(1, 100.0, 100.0, 0.0, 0.0) /* Määrittää, millä tavalla käyttöliittymä
            esitetään */
           }
      }
      ";
}
```

Ohjelmalistaus 8. Modellerin käyttöliittymä yksinkertaistettuna.



Kuva 7. Modellerin käyttöliittymän ensimmäinen välilehti.

Opinnäytetyössä käyttöliittymä on määritetty ohjelmakoodiin yhtenä string-muuttujana UserInterfaceDefinitions-luokassa. Pluginin alussa määritetään, mistä sen tulee etsiä mahdollista käyttöliittymämäärittelyä:

```
[Plugin("FoundationExcavationModeller")]
[PluginUserInterface(FoundationExcavationModeller.UserInterfaceDefinitions.dialog)]
```

Ohjelmalistaus 9. Käyttöliittymämerkkijonon määritys pluginissa.

Tämän käytännön etuna on se, että ohjelman käyttöliittymä ei vaadi erillisen tiedoston kopioimista toimiakseen. Tekla Structures käsittelee käynnistyksen yhteydessä pluginien ohjelmakoodin ja muodostaa string-muuttujan määrittelyjen mukaisen inp-tiedoston, jonka se tallentaa plugins-hakemistoon. Jos pluginin nimen mukainen inp-tiedosto on jo olemassa, TS ei ylikirjoita sitä.

Käyttöliittymää voi tarpeen mukaan muokata myös suoraan inp-tiedostosta. Jos tiedostoon tehtyjen muutoksien jälkeen haluaa palata ohjelmakoodin tekemään käyttöliittymään, tulee inp-tiedosto poistaa ja käynnistää TS uudelleen.

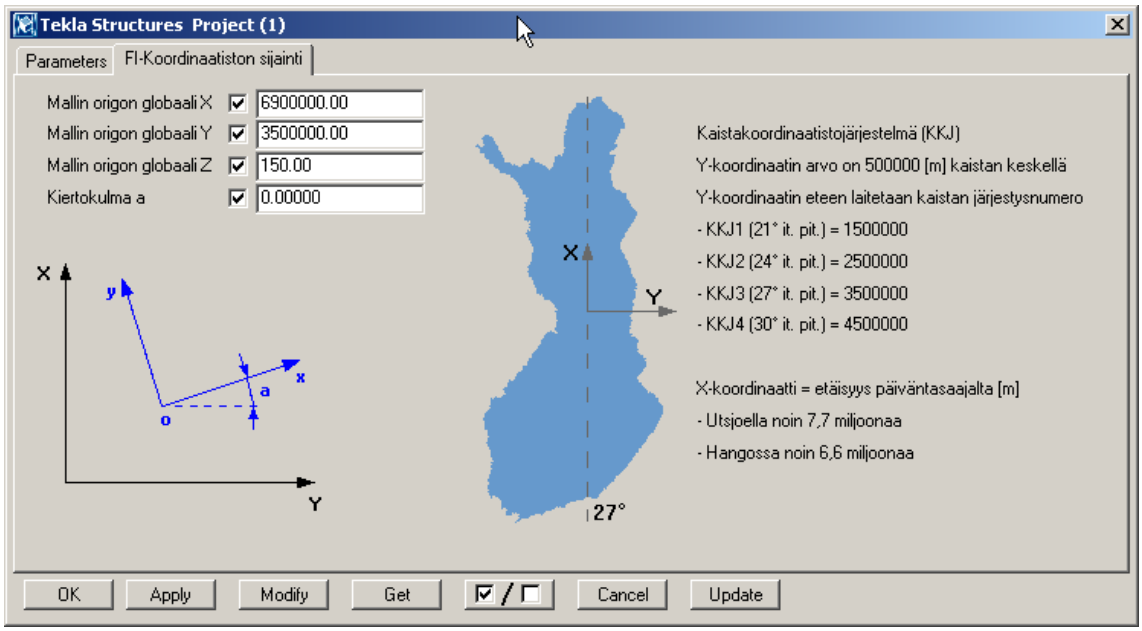
Käyttöliittymien kuvaaminen inp-tiedostoilla on kankea tapa. Jos määrittelyyn eksyy virhe, esimerkiksi puuttuva pilkku, käyttöliittymä ei suostu aukeamaan tai voi kaataa TS:n. Virheen etsiminen tällaisessa tilanteessa tapahtuu esimerkiksi kommentoimalla osia UI-määrittelystä ja etsimällä kohta, joka kaatumisen aiheuttaa. Erilaisten kenttien ja kuvien sijoittaminen paikalleen on myös hankalaa, sillä sijoitus tehdään ilmoittamalla vaaka- ja pystysuunnassa etäisyys ikkunan vasempaan yläreunaan. Jos haluaa siirtää taulukollisen tietokenttiä viisi pikseliä ylemmäs, pitää käydä käsin läpi kaikki määrittelyrivit.

Tekla tarjoaa ohjelmakehitystä varten tavan päivittää IU-määrittely TS:n ollessa käynnissä, mikä nopeuttaa kenttien sijoittamista, sillä muutokset nähdään heti inp-tiedoston muokkaamisen jälkeen. TS tukee versiosta 15.0 eteenpäin käyttöliittymien ohjelmointia .NETin avulla. Opinnäytetyössä päätettiin kuitenkin käyttää vanhaa tapaa, sillä käyttöliittymät olivat melko yksinkertaiset, eikä käyttöliittymien ohjelmoinnista ollut vielä kokemusta.

5.4 Projektikohtaiset attribuutit

Tekla Structures tarjoaa mahdollisuuden luoda käyttäjän määrittämiä ominaisuuksia (user defined attributes) kaikille mallin objekteille (osat, kokoonpanot, hitsit, pultit erityyppiset piirustukset jne.). Opinnäytetyössä projekti-objektille luodaan UDA-määrittely (Kuva 8). Tällainen määrittely on mallikohtainen, ja se tallennetaan mallin hakemiston objects.inp-tiedostoon. Määrittely tapahtuu samoja periaatteita noudattaen kuin kohdassa 5.3 käyttöliittymää määrittäessä.

Opinnäytetyössä projektikohtaisiin attribuutteihin tallennetaan mallin origon sijainti Suomen kartalla kartastokoordinaattijärjestelmän /2/ mukaisesti ja kiertokulma. Tällä tiedolla saavutetaan se etu, että rakennus voidaan Tekla Structuresissa mallintaa sen paikalliseen suorakulmaiseen koordinaatistoon, jolloin origo voidaan asettaa esimerkiksi johonkin rakennuksen nurkista ja määrittää x-akseli pitkän seinän suuntaiseksi. Projektille tallennettuja attribuutteja käyttäen tehdään muunnos paikallisesta koordinaatistosta KKJ-koordinaatistoon, kun kaivantojen tiedot viedään tiedostoon.

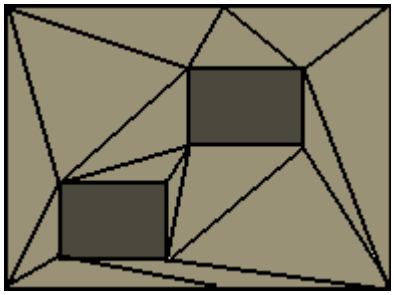


Kuva 8. Projektikohtaiset attribuutit.

5.5 Kolmiointi

Yksi Modellerille määrittelyssä listatuista toiminnoista on perustuskaivantojen yläpintojen yhdistäminen toisiinsa, jolloin tuloksena on maanpinta. Ilman maanpinnan luomista koneohjausjärjestelmässä näkyisi vain perustuskaivannot. Vaikka kaivaminen pelkästään niiden perusteella onnistuukin, helpottaa maanpinta työmaan hahmottamista

huomattavasti. Kolmioiminen tarkoittaa kappaleen muodostamista erikokoisista kolmioista. Maanpinta pitää kolmioida, sillä Vision3D-koneohjausjärjetelmä käsittelee kaikki kappaleet kolmioina. Kuva 9 esittää mahdollisen lopputuloksen, kun kolmioidaan suorakulmainen kappale. Tummemmat suorakulmiot edustavat reikiä kappaleen sisällä.



Kuva 9. Suorakulmainen reikiä sisältävä kappale kolmioituna.

Maanpinnan kolmiointi suoritetaan Bojan Nicenon luomalla EasyMesh-sovelluksella, sillä kolmiointi itsessään on varsin haastava toiminto, eikä sitä voinut millään sisällyttää opinnäytetyöhön. EasyMeshin käyttöä varten Modellerin täytyy selvittää TS-mallissa maanpintaa esittävien maanpintalevyjen kulmapisteet, sekä jokaisen perustuskaivannon ylänurkkapisteet.

5.6 Toiminta

Kaavio 1(Liite 2) kuvaa Modellerin toiminnan vuokaaviomuodossa. Aluksi ohjelma etsii käyttäjän valitsemista perustusosista pohjatahkot. Pohjatahkojen sijainnin ja käyttäjän antamien parametrien perusteella Modeller asettaa kaivannot jokaisen perustuksen ympärille. Seuraavaksi ohjelma tutkii, millä tavalla kaivannot ja perustukset ovat sijoittuneet toisiinsa nähden ja muotoilee kaivanto-osia eri tavoin. Muotoilun tavoitteena on yhdistää kaivannot toisiinsa niin, että kaivantojen geometria vastaisi haluttuja kaivantoja työmaalla. Tämän jälkeen ohjelma tutkii, millä tavalla kaivannot leikkaavat mallin kuvitteellista maanpintaa ja kolmioi maanpinnan tietojen perusteella.

Ohjelman ajon aikana perustuskaivannot luodaan palkeista, joista kerrotaan lisää kohdassa 5.6.1. Kaivantojen muotoilu tehdään muokkaamalla kyseisiä palkki-objekteja. Ohjelman suorituksen lopuksi käydään läpi kaikki palkkien pinnat, eli tahkot, ja luodaan jokaisesta pinnasta levy. Osa levyistä, esimerkiksi kaivantojen katot, ovat ylimääräisiä, joten ne poistetaan mallista. Jos malliin jää joitain ylimääräisiä levyjä, voi suunnittelija poistaa ne käsin. Seuraavissa aliluvuissa on selitetty toiminta yksityiskohtaisesti vaihe vaiheelta ja selitetty alla olevassa kaaviossa esiintyvät uudet termit. Havainnollisuuden

vuoksi useimmissa kuvissa olevat kaivannot on jo muodostettu levyistä, vaikka todellisuudessa ajon aikana niitä kyseisessä vaiheessa käsiteltäisiinkin vielä palkkeina.

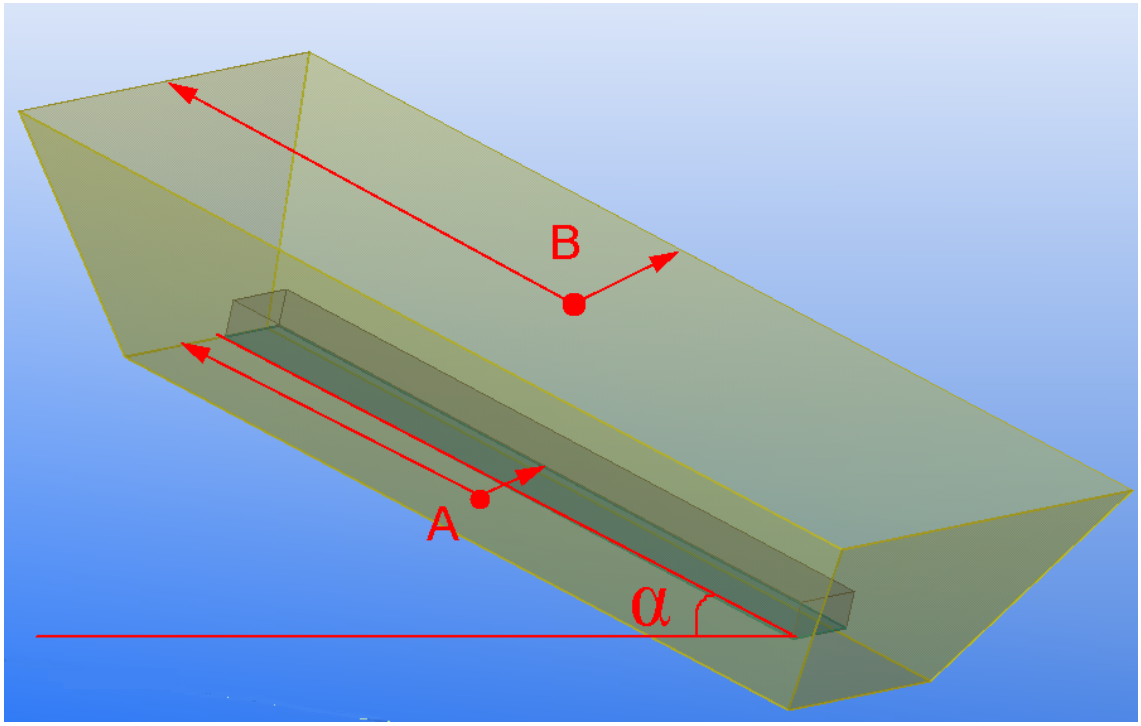
5.6.1 Vaihe 1 - Pohjatahkon etsintä ja kaivannon luominen

Kun Modeller-ohjelma käynnistetään, se käy läpi kaikki käyttäjän syöttämät input-objektit, eli perustukset. Perustusosista etsitään pohjatahkot, eli pinnat, jotka ovat osan alimpana ja kohti maanpintaa. Yksittäisillä perustuksilla pohjatahkoja on vain yksi kappale, mutta suunnittelija on voinut mallintaa myös jatkuvia mutkittelevia perustusosia, joilla on useampia pohjatahkoja. Ohjelma käsittää pohjatahkona sellaisen tahkon, jonka kulma maanpinnan tasoon nähden on alle 45 astetta. Tämän takia Modeller ei toimi oikein, jos perustus on yli 45 asteen kulmassa. Yleinen maksimikaltevuus perustukselle on 1 : 3 (metrin nousu kolmen metrin matkalla, n. 18 astetta), joten puute ei ole merkittävä.

Yksittäinen perustuskaivanto luodaan palkki-objektista. Palkin paikka määrätään antamalla sille alku- ja loppupiste. Palkin muoto määräytyy profiilin perusteella. Modellerissa käytetyn palkin profiiliin määritellään palkin leveys ja pituus sekä alku-, että loppupisteen kohdalla. Kuva 10 havainnollistaa tilannetta. A on kaivantopalkin alkupiste ja B loppupiste. Nuolet kuvaavat profiilin määrittelyä. Kyseisen kaivantopalkin määrittely on nähtävillä alla olevassa ohjelmalistauksessa. Kuvassa oleva kulma α on vihreällä korostetun perustuksen pohjatahkon ja maanpinnan välinen kulma.

```
Start point [mm] :      x =   25362.15 y =    6000.53 z =    5222.33
End point [mm]   :      x =   24878.71 y =    6000.44 z =    6944.60
Profile          :      PRMD6920*1600-8588*3268
```

Ohjelmalistaus 10. Perustuspalkin alku- ja loppupisteet sekä profiili.



Kuva 10 Perustuksen pohjatahko ja kaivannon määrittely.

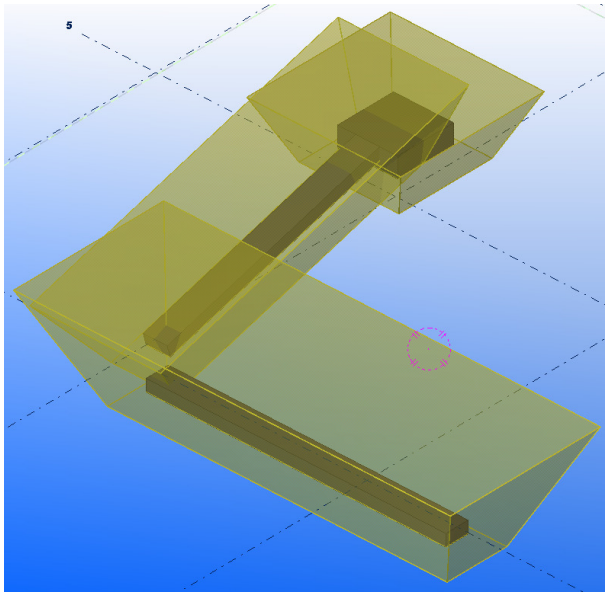
Kaivannon alkupisteeksi asetetaan löydetyn pohjatahkon keskipiste, jota siirretään korkeussuunnassa, eli mallin z-akselilla, käyttäjän asettaman parametrin h verran alaspäin. Näin perustukselle on muodostettu murskevara. Kaivannon loppupisteeksi asetetaan käyttöliittymässä määritetty kaivantojen maksimikorkeus, joka on kaikille sama. Profiili alkupisteen tasolla muodostetaan laskemalla perustuksen pituus ja leveys kulmapisteiden perusteella ja lisäämällä arvoihin käyttöliittymässä asetettu kaivannon b-parametri (työstövara sivusuunnassa, kuva 6). Profiili loppupisteen kohdalla on muuten sama, mutta siihen pitää lisätä kaivannon kaltevuuskertoimesta (slope factor) johtuva leviäminen. Muodostettu kaivanto asetetaan malliin ja se tallennetaan muistiin.

5.6.2 Vaihe 2 - Perustusten vertailu toisiinsa ja muotoilu

Vaiheen 1 jälkeen Tekla Structures -malli on täynnä toisiaan leikkaavia kaivantoja (Kuva 11), jotka tulee muotoilla leikkauskohdistaan. Vaiheessa 2 verrataan kaikkia perustuksia toisiinsa. Jos havaitaan perustuksia, joiden pohjatahkojen jokin kulmapiste on sama, on kyseessä mutkitteleva yksittäisestä kappaleesta luotu perustus. Tässä tapauksessa siirrytään vaiheeseen 3.

Jos yhteisiä kulmapisteitä ei havaita, tutkitaan törmäävätkö perustuskaivannot toisiinsa. Jos törmäystä ei tapahdu, mitään ei tarvitse tehdä ja jatketaan vaihetta 2. Jos kaivannot

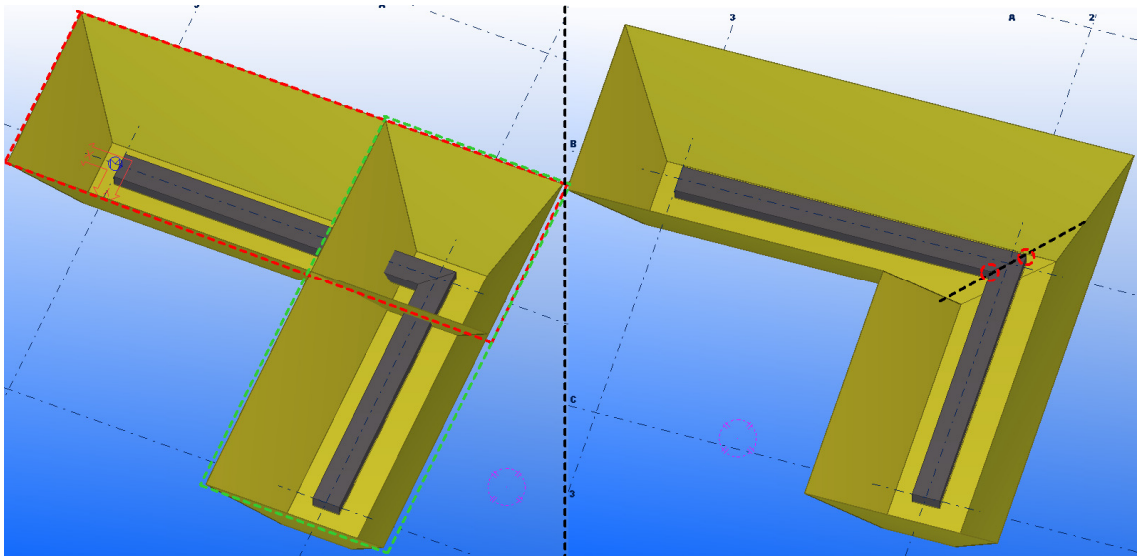
törmäävät, tutkitaan törmäävätkö myös perustukset. Jos perustukset eivät törmää, siirrytään vaiheeseen 4. Jos perustukset törmäävät, vuorossa on vaihe 5.



Kuva 11. Sekalaisia kaivantoja ennen muotoilua.

5.6.3 Vaihe 3 - Yhtenevän perustuksen nurkan muotoilu.

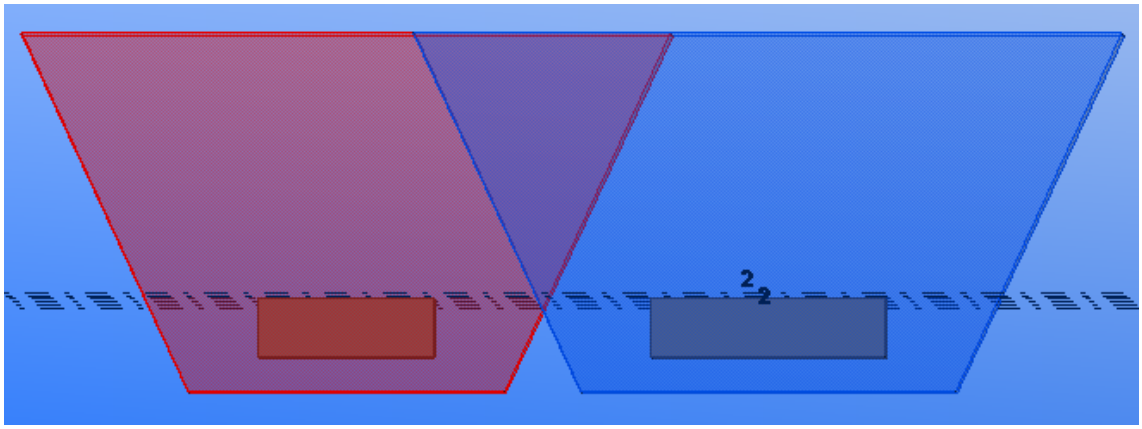
Vaiheessa 3 tilanne ennen kaivantojen muotoilua on nähtävissä alla olevan kuvan vasemmassa reunassa (kuva 12). Muotoilun jälkeinen tilanne on kuvassa oikealla. Muotoileminen tehdään luomalla line cut -operaatio molemmille kaivannoille. Operaatiossa määritetään viiva kahden pisteen perusteella, jolloin viivan kohdalle muodostetaan pystysuuntainen leikkaustaso. Viivan suunnasta riippuen tason vasemmalle tai oikealle puolelle jäävä osa objektista häviää. Nurkan muotoilussa line cut -viivan pisteiksi määritetään perustusten yhteisen nurkkapisteet (kuvassa punaiset ympyrät), jolloin kaivannot katkeavat oikeasta kohdasta. Tällöin malliin jää vielä yksi pystysuuntainen levy keskelle nurkkaa. Tämä poistetaan myöhemmässä vaiheessa sen perusteella, että tiedetään sen olevan täysin pystysuuntainen.



Kuva 12. Yhtenäisen perustuksen nurkan muotoilu.

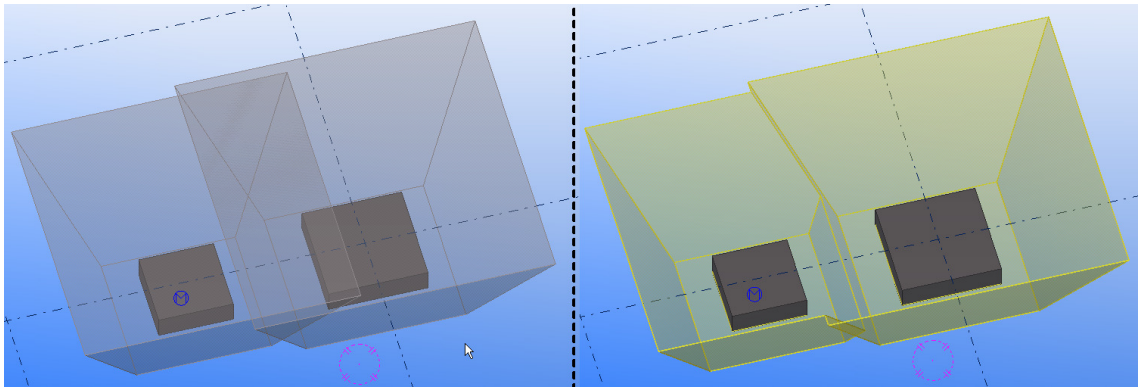
5.6.4 Vaihe 4 - Erillään olevien kaivantojen muotoilu

Tähän vaiheeseen on päädytty, jos on havaittu kahden kaivannon leikkaavan toisiaan, mutta niiden perustukset eivät osu toisiinsa. Tällöin muotoilu tehdään käyttämällä Tekla Structuresin part cut -työkalua. Part cut -operaatiossa valitaan kaksi objektia. Ensin valitusta osasta poistetaan ne osat, jotka ovat jälkimmäisenä valitun osan sisällä. Alla olevassa kuvassa oleva purppuran värinen alue on se osa kaivannosta, joka poistuu leikkauksen yhteydessä.



Kuva 13. Part cut -leikkauksen toimintaperiaate.

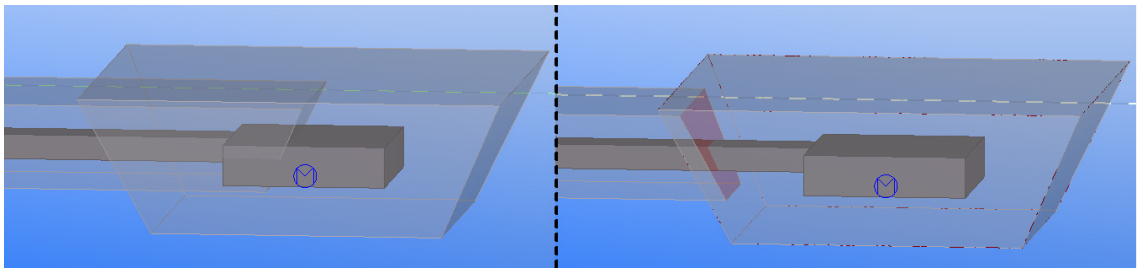
Tämänkaltaisessa tilanteessa yksi part cut -leikkaus ei riitä, sillä molempia kaivantoja pitää leikata toisella kaivannolla. Tämän takia ennen leikkausta molemmista kaivannoista tehdään kopio. Molempia alkuperäisiä kaivantoja leikataan toisen kaivannon kopiolla, jolloin tuloksena on alla olevan kuvan kaltainen tilanne. Myös tämän leikkauksen tuloksena on ylimääräisiä levyjä kaivannoissa. Näiden levyjen poistaminen on ohjelmallisesti ongelmallista, joten suunnittelijan tulee itse poistaa ylimääräiset levyt.



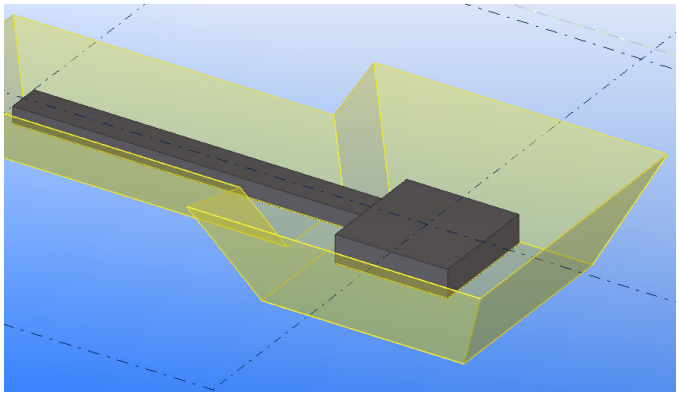
Kuva 14. Erilliset kaivannot ennen ja jälkeen muotoilun.

5.6.5 Vaihe 5 - Kulman muotoilu

Vaiheeseen 5 on siirrytty, jos on havaittu kahden kaivannon ja kaivantojen perustusten törmäävän toisiinsa. Tällaisessa tilanteessa tunnistetaan nurkassa oleva laajempi perustus, jonka kaivannolla leikataan toisen perustuksen kaivantoa. Erona edelliseen tapaukseen on siis se, että leikkaus tehdään vain toiseen suuntaan. Kuva 15 havainnollistaa, kuinka pidempi kaivanto katkaistaan nurkan laajemmalla kaivannolla. Kuvassa punainen osuus on kaivantojen leikkauspinta-ala. Kuva 16 esittää lopputuloksen, kun palkkien tahkoista muodostetaan levyt ja kattolevyt on poistettu.



Kuva 15. Yhdensuuntainen part cut -leikkaus.



Kuva 16. Kaivanto lopullisessa muodossaan.

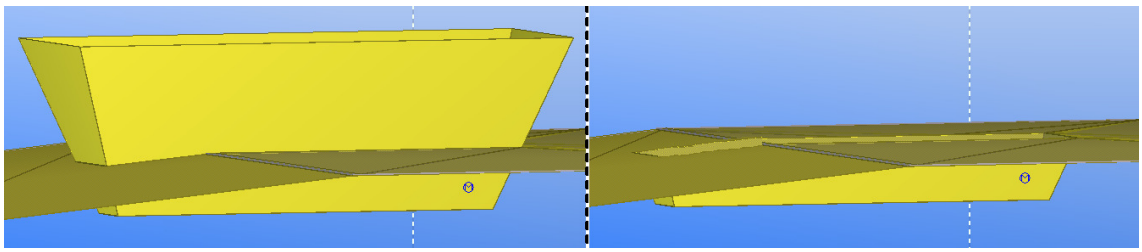
5.6.6 Vaihe 6 - Kolmioinnin valmistelu ja suorittaminen

Kolmioinnin valmistelu tapahtuu käymällä läpi kaikki maanpintalevyt ja tallentamalla niiden kulmapisteet listaan. Tämän jälkeen jokainen maanpintalevy leikataan vuorotellen jokaisella kaivannolla. Aina yhden leikkauksen jälkeen maanpintalevyn kulmapisteet luetaan uudelleen, ja jos kulmapisteisiin on tullut uusia pisteitä, tiedetään missä kohtaa kaivanto leikkaan maanpinnan. Tällä tavalla saadaan selville kaikkien reikien kulmapisteet.

Kolmiointi suoritetaan maanpintalevykohtaisesti. Jos mallissa on useampia maanpintalevyjä, suoritetaan kolmiointi useamman kerran. Kolmiointi tapahtuu luomalla tekstitiedosto, johon aluksi määritetään maanpintalevyn kulmapisteet ja niitä yhdistävät segmentit. Tämän jälkeen tiedostoon listataan kaikki reiät kulmapisteineen ja näitä yhdistävine segmentteineen. Valmis tiedosto luetaan kolmiointiohjelmaan ja kolmioinnin tuloksena valmistuu uusi tiedosto, josta löytyy listattuna kolmioinnissa syntyneet kolmiot kulmapisteineen. Tämän tiedoston tietojen perusteella malliin luodaan kolmiot, jotka muodostavat maanpinnan.

5.6.7 Vaihe 7 - Kaivantojen katkaisu maanpinnan mukaan

Kolmioinnin tuloksena mallissa on näkyvissä kolmioista muodostettu maanpinta, mutta kaivantojen yläreunat ovat edelleen maanpinnan yläpuolella. Vaiheen 7 aikana käydään läpi kaikki kaivanto-osat ja luodaan part cut -leikkaus jokaisen maanpintalevyn kanssa. Lopputuloksena kaikki kaivannot katkeavat yläreunoistaan maanpinnan tasoon (Kuva 17).



Kuva 17. Kaivantojen katkaisu maanpintaa mukaillen.

5.6.8 Vaihe 8 - Kaivantojen luominen levyistä

Tähän mennessä kaikissa vaiheissa kaivantoja on käsitelty palkki-objektista luotuina osina. Tämä sen takia, että objekteja on voitu muokata erilaisilla leikkauksilla ja sopivaa

profiilia käyttäen palkista on saatu lähes automaattisesti kaivannon muotoinen osa. Vision3D-koneohjausjärjestelmä vaatii kuitenkin, että kaikki osat muodostetaan kolmioista. Tämän takia kaivannot luodaan levyistä, jotka Exporter osaa kolmioida ja siirtää koneohjausjärjestelmän vaatimaan muotoon.

Levyjen muodostaminen tapahtuu käymällä läpi kaikki kaivanto-objektien pinnat, eli tahkot, jolloin saadaan selville tahkon kulmapisteet. Pisteiden perusteella Modeller luo malliin levyt ja kun kaikki levyt on luotu, palkki-objektit poistetaan. Käyttöliittymässä on myös olemassa asetus, jolloin palkki-objektit jätetään malliin näkyviin, eikä levyjä luoda.

5.6.9 Vaihe 9 - Ylimääräisten levyjen poisto

Vaiheen 9 jälkeen mallissa on useita ylimääräisiä levyjä, esimerkiksi maanpinnan yläpuolisia levyjä. Nämä poistetaan tutkimalla levyjen painopisteitä verrattuna mallin maanpintalevyihin. Vaiheessa 3 luotiin kaivannot yhtenäisen mutkittelevan perustuksen ympärille. Tällöin kulmaan jää aina yksi täysin pystysuuntainen levy, joka poistetaan tässä vaiheessa.

6 Exporter – kaivantojen siirto tiedostoon

Exporter on toinen opinnäytetyön aikana valmistuneista ohjelmista, jonka tehtävä on siirtää Modellerilla mallinnetut kaivannot LandXML-määrittelyn mukaiseen tiedostoon. Exporterin toteutus on melko yksinkertainen, joten se muodosti vain n. 15 % kokonaistyöajasta.

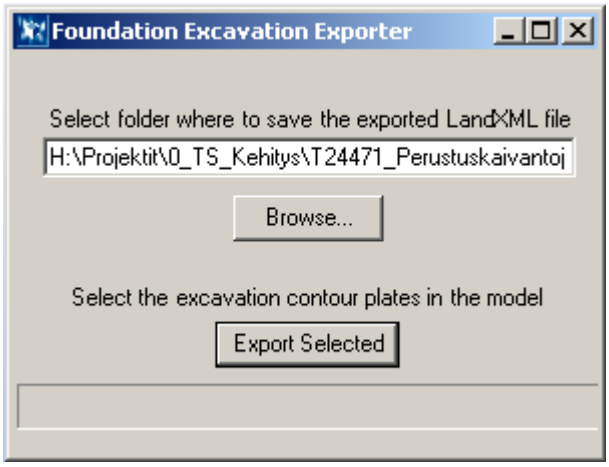
6.1 Määrittely

Projektin alussa määritellyt ja sen edetessä tarkentuneet vaatimukset Exporterille olivat seuraavat:

- Ohjelma kykenee siirtämään mallinnetut perustuskaivannot LandXML InfraModel2-määrittelyn mukaiseen XML-tiedostoon.
- Kaivannot pitää pystyä sijoittamaan kartastokoordinaatistojärjestelmän mukaisesti paikalliselle työmaalle.

6.2 Yleistä

Exporter on toteutettu makrona hyödyntäen Tekla Open API:n tarjoamia ratkaisuja. Makrot ovat yleensä pienehköjä, yhden yksittäisen toiminnon tai sarjan toimintoja suorittavia avustavia ohjelmistoja. Niiden tehtävänä on helpottaa suunnittelijoiden työtä ja automatisoida usein toistuvia työvaiheita. Exporter-makro on periytetty System.Windows.Forms.Form -luokasta, jonka avulla on luotu yksinkertainen käyttöliittymä nappeineen ja erilaisine kontrolleineen (Kuva 18).



Kuva 18. Exporterin käyttöliittymä

Käyttöliittymässä käyttäjä pyydetään kertomaan, mihin tiedostoon mallista valitut kaivannot viedään. Kun käyttäjä on valinnut mallista kaivannot ja valitsee *Export Selected,* makro ryhtyy tutkimaan valittuja osia. Makro käy läpi kaikki valitut osat ja tunnistaa niistä kaivannot. Kaivantopaloista se muodostaa kolmiot, jotka se kirjoittaa valittuun tiedostoon. Samanaikaisesti alareunan edistymispalkki kertoo käyttäjälle, kuinka valmis ohjelman ajo on.

6.3 Toiminta

6.3.1 Valmistelevat toimenpiteet

Makron käynnistymisen jälkeen suoritetaan käyttöliittymään liittyvät alustukset. Oletuksena ohjelma ehdottaa tallennushakemistoksi mallin hakemistoa ja oletustiedostonimeä. Käyttäjän asettamat projektiattribuutit luetaan muistiin, jos ne on asetettu. Muussa tapauksessa käyttäjää pyydetään asettamaan attribuutit, sillä kaivantojen pisteiden siirto ei onnistu ilman niitä.

Exporttauksen käynnistyttyä makro pyytää TS:ltä ModelObjectEnumeratorin-olioon valitut osat ja käy foreach-rakenteessa läpi jokaisen osan yksitellen. Jos valittu osa on komponentti, käydään kaikki komponentin aliosat läpi ja etsitään kaivantolevyjä aliosan nimen perusteella. Muussa tapauksessa tutkitaan suoraan osan nimeä. Jos osan todetaan olevan perustuskaivanto-osa, pyydetään osalta sen nurkkapisteet.

Nurkkapisteet tallennetaan listaan, jotta ne voidaan ajon lopuksi kirjoittaa tiedostoon. Tämän jälkeen n-nurkkainen kappale kolmioidaan niin, että siitä muodostuvien kolmioiden kärkipisteet kiertävät vastapäivään. Kolmiointirutiini on melko yksinkertainen ja toimii vain yksinkertaisilla kappaleilla. Mallintaja luo tietyissä tapauksissa monimutkaisempia kahdeksanpisteisiä levyjä, jotka kolmioidaan exporterissa erillisessä metodissa. Kolmioiden nurkkapisteiden indeksit tallennetaan listaan struktuurissa. Indeksi kertoo, että mistä nurkkapiste-listan pisteistä kolmio koostuu.

6.3.2 Tiedostoonkirjoitus

Kun kaikki valitut kappaleet on käsitelty, ohjelma siirtää tiedot LandXML-tiedostoon. Aluksi Exporter tarkistaa, onko valittu tiedosto jo olemassa. Jos sitä ei ole, luodaan tiedosto käyttäen .NETin *XmlTextWriter*-luokasta luotavaa oliota. Tiedoston alkuun

kirjoitetaan LandXML-määrittelyn mukaiset tiedot, sekä luodaan pisteitä ja tahkoja varten elementit. Alla olevasta ohjelmakoodipätkästä on poistettu ymmärryksen kannalta epäolennaista toistoa. Jos tiedosto on jo olemassa, oletetaan sen olevan oikein rakennettu ja ohitetaan tämä vaihe.

```
// Xml Writer
XmlTextWriter oXMLTextWriter = new XmlTextWriter(m_sXMLPathAndFileName,
                                                 m_encodingInUse);

// Create it and write all necessary stuff in it
oXMLTextWriter.WriteStartDocument();

string sDate = DateTime.Now.ToString("dd/MM/yyyy");
string sTime = DateTime.Now.ToString("HH:mm:ss");
oXMLTextWriter.WriteStartElement("LandXML");

// Write needed attributes
oXMLTextWriter.WriteAttributeString("version", "1.0");
oXMLTextWriter.WriteAttributeString("date", sDate);
oXMLTextWriter.WriteAttributeString("time", sTime);

oXMLTextWriter.WriteStartElement("Application");
oXMLTextWriter.WriteAttributeString("name", "Foundation Excavation Modeller");
oXMLTextWriter.WriteEndElement();   // Close Application

// Points will be written in here, but set later
oXMLTextWriter.WriteStartElement("Pnts");
oXMLTextWriter.WriteEndElement();   // Close Pnts
Faces will be written in here, but set later
oXMLTextWriter.WriteStartElement("Faces");
oXMLTextWriter.WriteEndElement();   // Close Faces

oXMLTextWriter.WriteEndElement(); // End LandXML

oXMLTextWriter.WriteEndDocument();
oXMLTextWriter.Close();
```

Ohjelmalistaus 11. LandXML-tiedoston rungon muodostaminen.

Edellisen vaiheen päätteeksi xml-tiedosto tallennettiin ja suljettiin, joten avataan se *XmlDocument*-luokasta luotavalla oliolla ja käydään *XmlNodeList*-oliolla läpi tiedoston elementit etsien "Pnts"-määrettä. Jos elementti löydetään, pyydetään siltä lapsielementtien lukumäärä. Ensimmäinen tallennettava piste on tunnistenumeroltaan kyseinen arvo+1. Ohjelmakoodi käy läpi aiemmin tallennetut kolmioiden kulmapisteet, muuttaa ne millimetreistä metreiksi, konvertoi TS:n karteesisesta koordinaatistosta KKJ-koordinaatistoon projektiattribuuteissa määriteltyjen arvojen mukaan ja tallentaa ne uusiksi lapsielementeiksi. Tahkot kirjoitetaan tiedostoon saman periaatteen mukaisesti.

```
XmlDocument doc = new XmlDocument();
doc.Load(m_sXMLPathAndFileName);

// Find last id for points in file and start giving id's to new points from that
int iNumberOfPoints = 0;
XmlNodeList parentNodePoints = doc.GetElementsByTagName("Pnts");
if (parentNodePoints != null)
{
      iNumberOfPoints = parentNodePoints[0].ChildNodes.Count + 1;
}
else
{
      bResult = false;
}

// If project attributes can be found, start adding the points to xml-file
```

```
if (InitializeProjectAttributes())
{
      // Write the points to Pnts-node
      for (int i = 0; i < m_lNewVertexPoints.Count; i++)
      {
            XmlElement childElement = doc.CreateElement("P");
            //Create an attribute.
            XmlAttribute attr = doc.CreateAttribute("id");
            attr.Value = (iNumberOfPoints + i).ToString();
            childElement.SetAttributeNode(attr);

            TSG.Point oRotatedPoint = new TSG.Point(0, 0, 0);

            // Rotate and convert point, method will also switch x and y axis
            if (ConvertRotatePointFromTSCSysToKKJ(m_lNewVertexPoints[i], ref
                                                  oRotatedPoint))
           {
                string y = string.Format("{0:0.0000}", (oRotatedPoint.Y / 1000 +
                                                   m_dOriginY).ToString());
                y = y.Replace(",", ".");

                string x = string.Format("{0:0.00}", (oRotatedPoint.X / 1000 +
                                                 m_dOriginX).ToString());
                x = x.Replace(",", ".");

                string z = string.Format("{0:0.00000000}", (oRotatedPoint.Z /
                                          1000 + m_dOriginZ).ToString());
                z = z.Replace(",", ".");

                // Norther, easter, elevation
                childElement.InnerText = x + " " + y + " " + z;
                if (parentNodePoints[0] != null)
                 {
                    parentNodePoints[0].AppendChild(childElement);
                 }
           }
      }
      // Tahkojen tallennus tässä samalla periaatteella kuin yllä pisteiden tallennus
      doc.Save(m_sXMLPathAndFileName);
}
```

Ohjelmalistaus 12. Pisteiden tallentaminen LandXML-tiedostoon.

7 Kehitysehdotukset

Exporter luo yksinkertaisella algoritmilla mallinnetuista perustuskaivannoista kolmiot, jotka siirretään XML-tiedostoon koneohjausjärjestelmää varten. Algoritmi ei osaa käsitellä oikein ulokkeita sisältäviä monikulmioita, joten sen toteutuksen voisi tehdä geneerisemmäksi. Testitapauksissa algoritmin on todettu toimivan oikein, mutta laajempi käyttö voi paljastaa puutteita.

Modeller ei tue useita maanpintalevyjä, joten tällä hetkellä käyttäjä voi vain kuvata maanpinnan tason, mutta ei sen muotoja.

8 Loppupäätelmät

Opinnäytetyön suunnittelu ja toteutus perustuu vahvasti Tekla Structuresin ymmärtämiseen. Suunnittelu on aloitettu noin kolmen kuukauden Tekla Structures -ohjelmointikokemuksella, joten tilanne ei ole ollut optimaalinen. Tekla Open API -rajapinta tarjoaa monia palveluita, mutta kokemattomuuden takia niitä ei ole aina osattu hyödyntää. Opinnäytetyön aikana toteutin itse joitain metodeita, joille myöhemmin löysin valmiin toteutuksen rajapinnasta. Kokemattomuus aiheutti myös sen, että jotkin toiminnallisuudet on projektin edetessä toteutettu kokonaan uudestaan, sillä vanha toteutus on ehditty todeta joko hankalaksi ylläpitää, tai esimerkiksi riittämättömäksi erikoisemmissa tapauksissa. Kokemuksen ja projektin aikana oppimisen myötä Modellerin toteutuksen voisi kirjoittaa joiltain osin uudestaan, jolloin koodi olisi selkeämpää ja tehokkaampaa, sekä toimisi paremmin monimutkaisemmissa tilanteissa.

Ohjelmien määrittely eli projektin edetessä, mutta perusajatus pysyi samana. Alkuperäisessä määrittelyssä ollut C-parametri, jolla oli tarkoitus mallintaa maanpinta, muuttui myöhemmässä vaiheessa maanpintalevyihin ja niiden avulla tehtävään kolmiointiin. Tämä olikin ainoa merkittävä muutos työn edetessä.

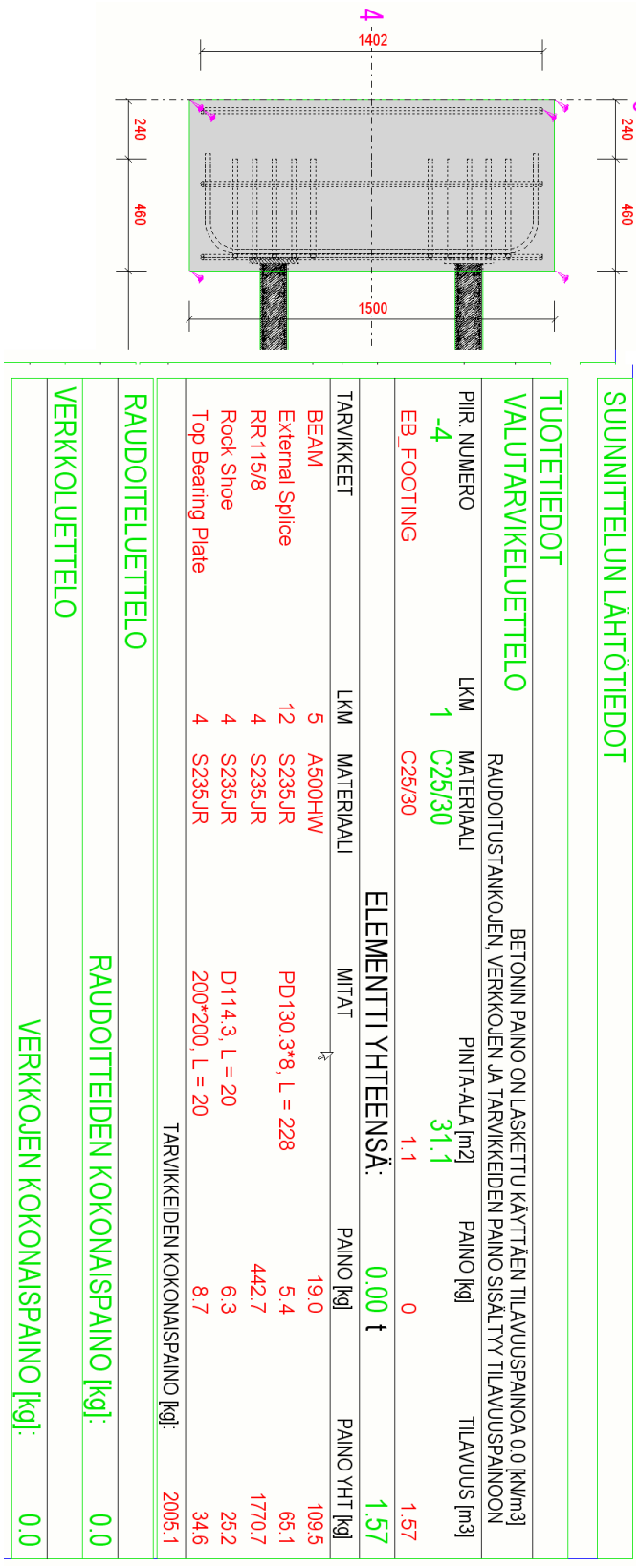
Kokemattomuus Tekla Structuresin kanssa oli ongelma, mutta aikaisempi työkokemus maanrakennuksesta auttoi ymmärtämään työn perusidean hyvin. Kokemus antoi paljon ideoita ja jotkin monimutkaiset tapaukset pystyi suoraan jättämään huomiotta, kun omasta kokemusta tiesi, että sellaiseen ei törmätä.

9 Lähteet

/1/ Infra Model 2 Browser for LandXML 1.0 [online][viitattu 28.7.2009].
Saatavissa: http://cic.vtt.fi/projects/inf/ramodel2/documentation/index_d.html
/2/ Kartastokoordinaattijärjestelmä [online][viitattu 29.7.2009].
Saatavissa: http://fi.wikipedia.org/wiki/KKJ
/3/ Tekla Open Api -ohjelmistokehitysrajapinta [online][viitattu 22.10]
Saatavissa: http://www.tekla.com/fi/solutions/building-construction/Pages/basic-concepts.aspx#tekla-open-api
/4/ EasyMesh -kolmiointisovellus [online][viitattu 1.9.2009].
Saatavissa: http://www-dinma.univ.trieste.it/nirftc/research/easymesh/easymesh.html
/5 / Polygon Clipper [online][viitattu 22.10.2009].
Saatavissa: http://www.cs.manchester.ac.uk/~toby/alan/software/
/6/ Novatron Vision3D -koneohjausjärjestelmä [online][viitattu 22.10].
Saatavissa: http://www.novatron.fi/en/Vision3d.html
/7/ Real Time Kinematic GPS-paikannuksen mittausmenetelmä [online][viitattu 27.10].
Saatavissa: http://fi.wikipedia.org/wiki/Reaaliaikainen_kinemaattinen_mittaus

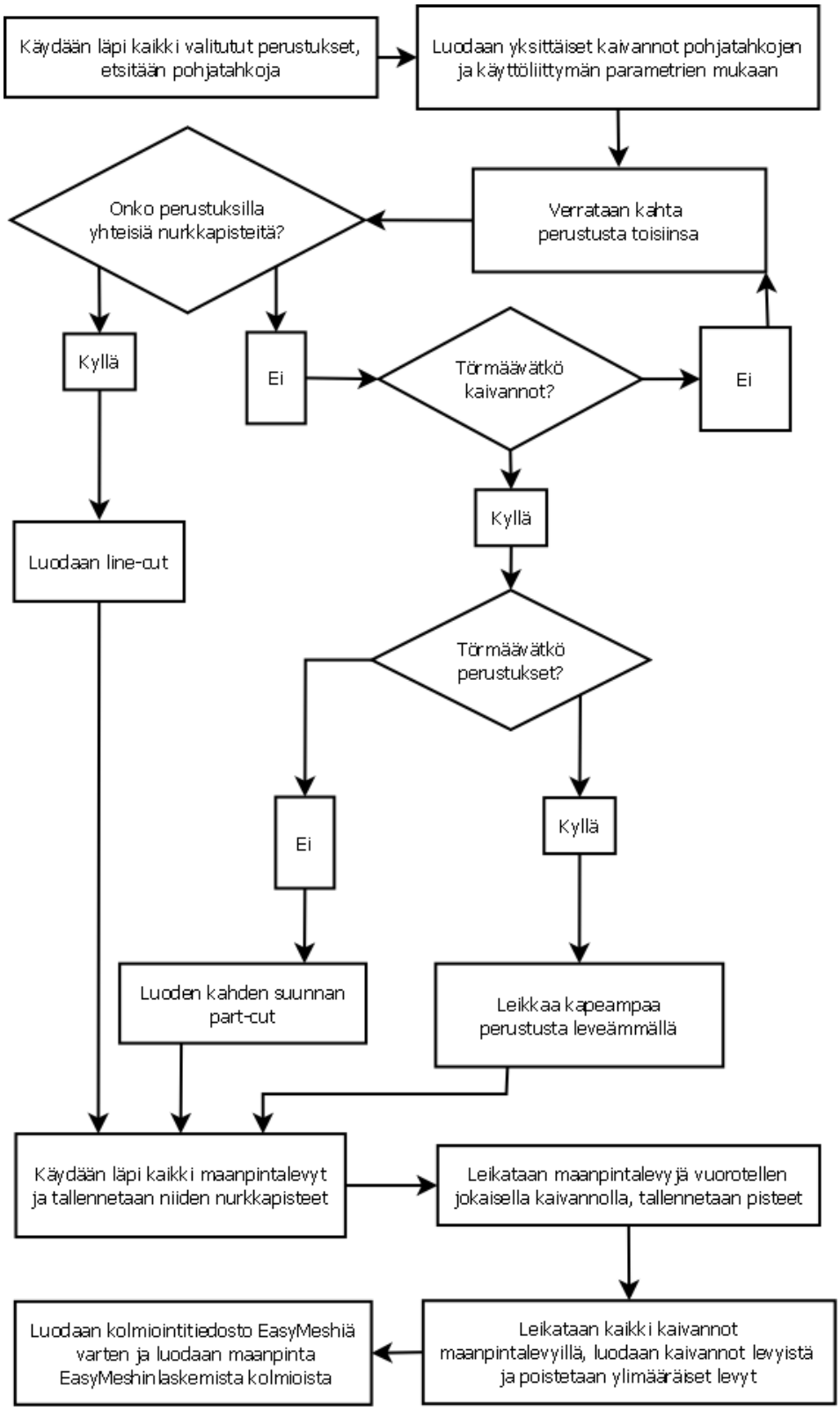
10 Liitteet

Liite 1: Betonielementin piirustus



Kuva 19. Betonielementin piirustus ja tiedot.

Liite 2: Modellerin toiminta vuokaaviona



Kaavio 1. Modellerin toiminta vuokaaviomuodossa.