STADIA
HELSINKI POLYTECHNIC

TEKNIIKAN JA LIIKENTEEN TOIMIALA

Tietotekniikka
Information Technology and Communications

INSINÖÖRITYÖ

BETONIRAUDOITUKSEN PIIRUSTUKSEN OHJELMALLINEN LUOMINEN

Työn tekijä: Mikko Aspiala
Työn valvoja: Matti Luukkainen, yliopettaja
Työn ohjaaja: Marie-Louise Litonius, tuotantopäällikkö

Työ hyväksytty: 23.4.2008

Matti Luukkainen
yliopettaja

ALKULAUSE

Tämän työ tehtiin Digia Oyj:n toimeksiantona betoniraudoituksia myyvälle Tammet Oy:lle, jonka toimintaan olen projektin puitteissa päässyt erityisellä tarkkuudella tutustumaan. Olen myös päässyt näkemään tekemäni työn paikan Tammetin toiminnassa ja siten saanut itseni tuntemaan tehneeni jotain merkityksellistä.

Tahdon kiittää muita projektiin osallistuneita, projektipäällikkö Anu Pitkästä, joka puhuu pälkäneläisten kanssa samaa kieltä, sekä ohjelmistosuunnittelija Esa Pappilaa, jota ilman olisin voinut joutua ohjelmoimaan ne muutkin projektin osuudet itse.

Työ sai myös projektin ulkopuolista opastusta, jota löytyy aina huutoetäisyydeltä järjestelmäsuunnittelija Asko Rajamäen suunnalta.

Kiitän myös tyttöystävääni Unnaa, jonka roolia tämän lopputyön valmistumisen valvonnassa ei voi vähätellä.

Helsingissä 6.4.2008
Mikko Aspiala

INSINÖÖRITYÖN TIIVISTELMÄ

Tekijä: Mikko Aspiala	
Työn nimi: Betoniraudoituksen piirustuksen ohjelmallinen luominen	
Päivämäärä: 4. huhtikuuta 2006	Sivumäärä: 38 + 6 liitettä
Koulutusohjelma: Tietotekniikka	Suuntautumisvaihtoehto: Information Technology and Communications
Työn valvoja: yliopettaja Matti Luukkainen Työn ohjaaja: tuotantopäällikkö Marie-Louise Litonius	

Tämän insinöörityön tavoitteena oli uuden toiminnallisuuden lisääminen suureen ERP-järjestelmään. Asiakasyritykselle suunniteltiin ratkaisu, jonka avulla voidaan tulostaa loppukäyttäjälle tekninen piirustus betoniin valettavista raudoituksista.

Sovellus luo piirustuksen dynaamisesti käyttäjän antamien mittojen ja parametrien perusteella. Tehtävänanto edellytti myös, että kehikot voivat olla taivutettuina kolmiulotteisesti käyttäjän syötteen määrittämiin muotoihin. Tämä edellytti ratkaisuja lukuisiin trigonometrisiin ja loogisiin ongelmiin. Sovellus suunniteltiin jatkokehityksen kannalta joustavaksi.

Ohjelmointi toteutettiin Progress-kielellä, jolla itse ERP-järjestelmä on myös toteutettu. Lopputuloksena on käyttökelpoinen sovellus, joka täyttää asiakasyrityksen tarpeet.

Avainsanat: ERP, konfiguraattori, raudoitus, Progress

STADIA
HELSINKI POLYTECHNIC

ABSTRACT

Name: Mikko Aspiala

Title: Programmatic creation of rebar beam cage schematic

Date: 4th April 2008 Number of pages: 38 + 6 appendices

Degree Programme: Information Technology

Specialisation: Information Technology and Communications

Instructor: Matti Luukkainen, Principal lecturer

Supervisor: Marie-Louise Litonius, Production manager

The goal of this study was to develop a new feature for a vast ERP-system. The feature includes the dynamic creation of a rebar beam cage schematic for the end-user.

The form of the cage can be freely designed by the user allowing such features as bending the cage in all three dimensions. Achieving this saw the solutions for numerous trigonomical and logical problems. Also the software design was done further development in mind.

The programming was done in Progress-language, native to the ERP-system. The result is a practical application that meets the demands made by the customer enterprise.

Key words: ERP, concrete reinforcement, configurator, Progress

SISÄLLYS

LIITTEET

1 JOHDANTO

Asiakasyrityksen pääasialliseen toimialaan kuuluu betoniraudoitusten valmistus ja niiden myynti teollisuusasiakkaille. Eri raudoitusten muoto ja muut vaatimukset ovat tilauskohtaisia, mikä tekee niiden sulavasta myymisestä sekä hallinnasta haasteellista. Asiakas onkin nähnyt tarpeelliseksi lisätä tätä prosessia helpottavan työkalun ERP- eli toiminnanohjausjärjestelmäänsä. Tämän työkalun suunnittelu ja toteutus ovat tämän lopputyön aihe.

Asiakas tarvitsee sovelluksen, joka mahdollistaa raudoitusverkon valmiin piirustuksen liittämisen ERP-järjestelmänsä myyntitilauksen yhteyteen. Piirustus on myös kyettävä tuottamaan riittävän nopeasti, jotta siitä olisi hyötyä jo myyntitapahtuman aikana, esim. virheellisten mittojen havaitsemisen muodossa. Sovellus pyrkii täten parantamaan asiakasyrityksen toiminnallista tehokkuutta helpottamalla tuotantoketjussa ihmiskäyttäjän aikaa vievää suunnitteluprosessia.

Toteutuksen vaihtoehtona on erinäisen CAD-sovelluksen käyttö. Tästä toimintatavasta halutaan päästä eroon seuraavista syistä:

- Omaksuttavuus – henkilökunnan perehdyttäminen täyden CAD-sovelluksen käyttöön on hidasta ja kallista. ERP-järjestelmään integroitu ratkaisu tarjoaa ainoastaan tarvitut toiminnot tehden perehdytyksestä helpompaa ja käyttöönotosta nopeampaa.

- Jäykkyys – CAD-ohjelma on erillinen sovelluksensa joka jättää käyttäjän vastuulle kaikki hallinnolliset toimenpiteet. Tehty sovellus on puolestaan osa ERP-järjestelmää, joten se hyötyy kaikista sen tuomista eduista ja automaatioista. Esim. piirustus voidaan automaattisesti tallentaa myyntitilauksen yhteyteen, jolloin se kulkeutuu automaattisesti sitä tarvitseville tahoille.

- Huolimattomuusvirheet – räätälöity sovellus tarvitsee toimiakseen ainoastaan vähimmäismäärän käyttäjän syöttämiä parametreja ja sisältää lukuisia erikoistuneita virheenhavaitsemistoimintoja. Tämä vähentää huolimattomuusvirheiden esiintymistä.

- Käyttönopeus – CAD-suunnittelu on usein liian hidasta hoidettavaksi esim. puhelinkeskustelun aikana. Räätälöity sovellus toteuttaa piirustuksen nopeasti pelkkien mittojen perusteella.

- Käyttäjälisenssien hinta – CAD-lisenssien hinta on kohtuuton verrattuna räätälöityyn ratkaisuun.

Tarve ohjelmalle on siis perusteltu ja selvitettäväksi jäi, kuinka se tulisi tehokkaasti toteuttaa.

Projekti toteutettiin monivaiheisesti ilman tietoa tulevista vaiheista. Tällä oli vaikutuksensa suunnittelutyöhön ja siinä sovellettuun tutkimustyyppiin. Koska tulevien vaiheiden määrä ja luonne olivat tuntemattomia, tuli ohjelmasta suunnitella mahdollisimman joustava ja muokattava[1]. Nähtyään tietyn vaiheen toimivan tilasi asiakas toteutukseen uusia ominaisuuksia. Tätä ennakoitiin tekemällä koodista korostetun joustavaa. Tiettyjä ominaisuuksia, kuten kulmien pyöristystä, osattiin jo odottaa aikaisemmassa vaiheessa, joten pohjustus tehtiin ne huomioon ottaen.

Suuri rooli oli myös ennakoivalla tutkimustyöllä, jonka kautta pystyttiin mittaamaan mahdollisten ominaisuuksien toteutettavuutta. Tämä synnytti nimenomaan pyöreiden kulmien tapauksessa kokonaisen työvaiheen, jonka tarkoitus oli tutkia pyöreitä taivutuksia sekä niiden käytännöllisyyttä.

Kuten jo annettiin ymmärtää, tehty työ on jaoteltu työvaiheisiin, jotka muodostavat tämän tutkimuksen toteutuksellisen rakenteen:

- Työvaihe 1: Kaksiulotteinen piirustus – toteutus aloitettiin kaksiulotteisesta piirustuksesta, jossa teräsverkko kuvataan mittoineen litteänä.

- Työvaihe 2: Kolmiulotteinen piirustus ja taivutukset – uutena ominaisuutena piirustukseen lisättiin raudoitusten taivutukset ja niiden myötä poikkileikkauskuvat sekä isometriset lintuperspektiivikuvat.

- Työvaihe 3: Pyöreät kulmat – todelliset taivutukset eivät ole teräviä vaan pyöreitä. Tämä aiheuttaa mm. lyhenemän taivutetuissa teräksissä sekä sitä kautta ei-toivottua siirtymää poikkiteräksissä.

Jokainen työvaihe on jaoteltu edelleen pienemmiksi kokonaisuuksiksi joita jatkossa kutsutaan nimellä ongelma. On yleistä, että käsiteltävät ongelmat sisältävät sisäkkäisiä aliongelmia. Kokonaisuuden helpomman hallinnan, helpomman virheenpaikannuksen ja joustavan suunnittelun vuoksi kaikki mikä pystytään näkemään erillisinä ongelminaan, sellaisina myös kuvataan.

[1] Tämä on kirjoittajan mielestä usein hyvä ohjelmointikäytäntö.

Sisäkkäisten ongelmien ilmentyessä ne ratkaistaan niiden tullessa vastaan, ja omassa luvussaan. Esimerkiksi jos ongelman ratkaisu edellyttää taivutuksen aloituspisteen sekä taivutuksen suunnan määrittävää algoritmia, nämä ratkaistaan ongelman sisällä erillisinä aliongelminaan.

Kuten sanottu, jokainen työvaihe sisältää vaihtelevan määrän ratkaistavia ongelmia. Nämä ongelmat pyritään käsittelemään toisiaan vastaavalla tavalla. Koska monen työhön liittyvän ongelman ratkaisu edellyttää jonkinlaisen algoritmin tuottamista, tarkastellaan tarvittavaa algoritmia ensin loogiselta, sitten matemaattiselta ja lopulta ohjelmalliselta kannalta.

Looginen vaihe määrittelee ongelman ja siihen ratkaisun. Matemaattinen vaihe tuottaa siihen yleispätevän mallin, joka viimeisessä vaiheessa lisätään sovellukseen ohjelmalliselta kannalta toimivana. Tähän vaiheeseen on sisällytetty myös viite ratkaisun kannalta oleelliseen ohjelmalohkoon (liitteessä 6), joka usein tarkoittaa yhden proseduurin tai funktion koodirivejä. Lopuksi tehdään päätelmiä mahdollisista virhe- sekä erikoistilanteista. On huomattava, että matemaattisen vaiheen läsnäolo ei ole kaikissa tilanteissa tarkoituksenmukainen, sillä ongelmat saattavat olla puhtaasti loogisia tai muuten yksinkertaisia. Näiden vaiheiden jälkeen algoritmi sisällytetään ohjelmakoodiin, joka on luettavissa liitteestä 6.

Lukijalle on haluttu tarjota kaksi tapaa tutustua ohjelmalliseen ratkaisuun: toimintaperiaatteet ovat ensin kuvattu sanallisesti ongelmaan liittyvässä luvussa ja itse lähdekoodin muuttujien- sekä muiden elementtien nimet ovat korostetun kuvaavasti nimetty, jotta lukijat jotka perehtyvät ratkaisuun mieluiten suoraan koodiin syventymällä voivat sen helpommin sisäistää. Työssä käytetyn Progress-kielen luonnollinen syntaksi yhdistettynä ratkaisuja kuvaaviin proseduurin- ja funktionnimiin ovat riittävän lähellä pseudokoodia, joten tätä havainnollistamismenetelmää ei erikseen sovelleta.

Toteutetun ohjelman lähdekoodi on sivumääräisesti laaja[2], joten sen lukemista on pyritty helpottamaan eri tavoin: keskeiset sanat on väritetty omalla värillään[3], ja sivut ovat vaakatasossa pitkien rivien rivityksen ehkäisemiseksi. Lisäksi ohjelman korostetun proseduraalinen luonne mahdollistaa yhden yksilöllisen ongelman ratkaisevan ohjelmalohkon käsittelemisen kerrallaan.

[2] 16 sivua, vaakatasossa 24 sivua. Lähdekoodin rivien määrä vastaa suurpiirteisesti tämän dokumentin sivunumeroitujen sivujen (38 kpl) rivien määrää. Vaakatason käytöstä johtuva sivujen määrän lisäys katsottiin tarpeelliseksi helppolukuisuuden parantuessa.

[3] Toiselta nimeltään Syntax highlight.

Ongelmaan liittyvä ohjelmalohko kuvaa usein yhden ohjelmaproseduurin tai funktion toiminnan. Näin ollen yhtä edellä mainitulla tavalla käsiteltävää ongelmaa vastaa aina yksi yhdessä paikassa oleva ohjelmalohko, jossa epäoleellisen koodin määrä on vähäinen.

Kytkökset oleellisen ohjelmalohkoon tehdään selväksi aina luvun lopussa, otsikon ”Oleelliset lähdekoodin proseduurit” alla. Lisäksi liitteestä 1 löytyy lyhyt listaus proseduureista viittauksine lähdekoodin (liite 6) proseduuria käsitteleviin kohtiin. Vastaava listaus käytetyistä funktioista löytyy liitteestä 2.

Ohjelman toteutukseen kuuluu suuri joukko ominaisuuksia, joiden jokaisen tarkka esittely ei tarkoituksenmukaista, joten täysin epäoleellisten tai muuten triviaalien ongelmien käsittely on sivuutettu. Myös työnantajan yrityssalaisuuksiksi luettavat sekä tietoturvariskin mahdollisesti aiheuttavat kohdat jätetään käsittelemättä.

Tutkielma on kirjoitettu käyttäen työnantajan vakiintunutta sanastoa, joten esimerkiksi puskurista käytetään nimeä ”bufferi”. Alaviitteitä käytetään lauseessa ajatuksen sivuraiteille johtavien huomautuksien esille tuomiseen.

2 TAUSTATIEDOT JA KÄSITTEET

Tässä kappaleessa esitellään tutkimukseen liittyvät taustatiedot sekä käsitteet. Edellämainitut ovat pyritty esittämään johdonmukaisessa järjestyksessä siten, että niiden perättäinen lukeminen on mielekästä.

2.1 Digia Oyj

Digia Oyj (ent. SYSOPENDIGIA) on suomalainen ohjelmistoyhtiö, joka on noteerattu Helsingin pörssissä. Yhtiössä työskentelee 1200 IT-alan ammattilaista, ja sen liikevaihto vuonna 2007 oli 105,8 miljoonaa euroa.

Digia on tämän lopputyön toimeksiantaja, ja tutkija on kirjoitushetkellä vakituisessa työsuhteessa sen kanssa työnimikkeellä ohjelmistosuunnittelija.

2.2 Digia Enterprise

Digia Enterprise on teollisuuden ja tukkukaupan toiminnan- ja talousohjausjärjestelmä, jonka asiakasräätälöidyksi lisäominaisuudeksi tämän tutkimuksen lopputulos tehtiin.

Järjestelmä sisältää työkalut yrityksen sisäisten prosessien hallintaan ja hoitaa yrityksen operatiiviset toiminnat hankinnasta laskutukseen saakka. Enterprisen avulla karsitaan manuaalisia työvaiheita, vähennetään virhemahdollisuuksia sekä parannetaan ja standardoidaan toimintamalleja.

Enterprise perustuu käyttöliittymäriippumattomaan objektiarkkitehtuuriin, missä järjestelmän liiketoimintalogiikka on erotettu käyttöliittymistä. Ohjelmistoa voidaan käyttää Windows-käyttöliittymällä client/server -ympäristössä tai internetin kautta. Ohjelman käyttöä voidaan tarvittaessa laajentaa myös erilaisiin mobiilipäätelaitteisiin, kuten kämmentietokoneisiin ja matkapuhelimiin [1].

2.3 Enterprisen proseduurit

Ohjelmatekniseltä kannalta Enterprise sisältää lukuisia ohjelmaproseduureja, joilla yhdenmukaistetaan usein toistuvien operaatioiden suoritusta. Tähän tutkimukseen sisältyvissä ohjelmalohkoissa Enterprisen omat proseduurit erottuvat tehdyn ohjelman sisäisistä proseduureista tiedostopäätteestä ”.p” sekä proseduuriin osoittavasta polusta. Esim.

```
RUN pgsubr/error.p(”Virhe”).
```

on Enterprisen sisäinen proseduuri ja

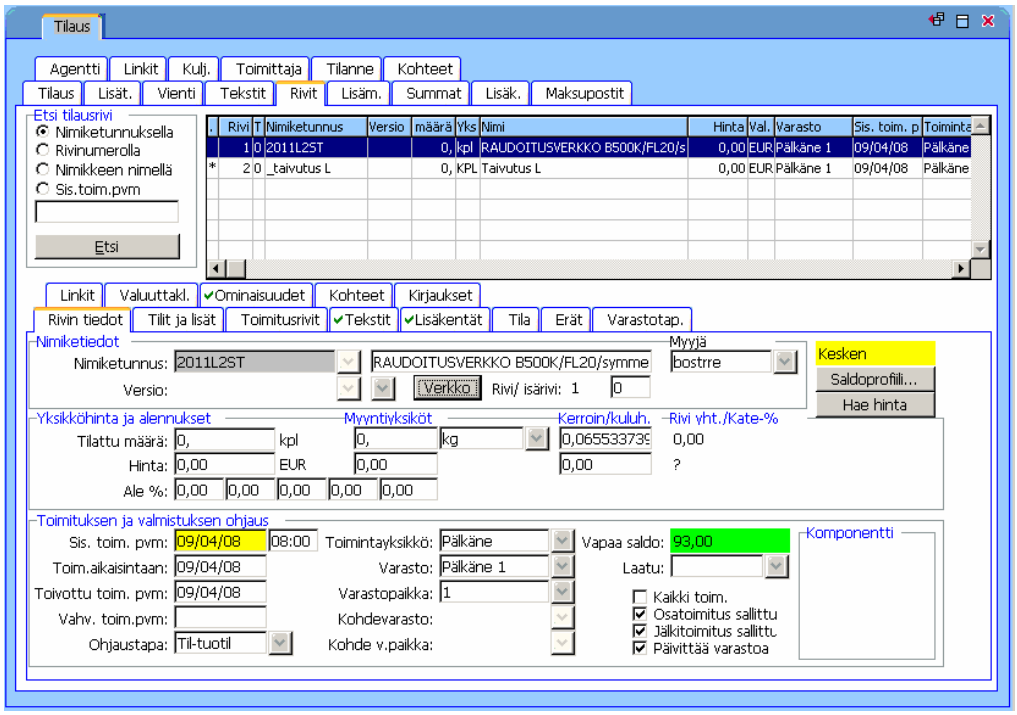
```
RUN OuterInit(OUTPUT xOk).
```

on ohjelman sisäinen proseduuri. Tutkimusta luettaessa proseduurien jaotteleminen on hyödyllistä koska ohjelman sisäisten proseduurien toiminta selitetään tarkasti, kun taas Enterprisen sisäisten proseduurien tarkka kuvaus on epäolennaista. Ohjelmasta löytyy myös PDFincluden (ks. luvussa 2.11) sisäisiä proseduureja, joiden merkitys on samoin vähäisempi.

2.4 Myyntitilaus ja myyntitilausrivi

Enterprisen myyntitilaus on Enterprisen käyttäjän asiakasta varten tekemä tilaus, joka pitää sisällään vaihtelevan määrän rivejä. Riveillä ovat myytävät palvelut ja tuotteet, kuten tämän lopputyön tapauksessa betoniraudoitukset.

Piirustuksen tulostussovellus käynnistetään myyntitilausriviltä.



Kuva 1. Enterprisen myyntitilauksen myyntitilausrivin ulkoasu

Kuvassa 1 näkyy Enterprisen myyntitilausriviohjelma. Huomioitavaa on ”Verkko”-painike, josta käynnistetään verkon määritysohjelma. Itse piirustuksen tulostus käynnistetään ikkunan työkaluriviltä.

Raudoituksen mahdollinen taivutus näkyy itse raudoituksen lapsirivinä. Tämä mahdollistaa mm. taivutusprosessin erillisen hinnoittelun sen monimutkaisuuden mukaan.

2.5 Tammet Oy

Tammet Oy on vuonna 1946 perustettu verkko- ja raudoitustuotteiden valmistaja, jolla on tehtaat Tammisaaressa ja Pälkäneellä. Yritys käyttää Digia Enterprisea toiminnanohjausjärjestelmänään, jonka räätälöidyksi lisäominaisuudeksi tämän lopputyön ohjelmisto-osuus on tehty. Tammet Oy:hyn tullaan viittaamaan tässä lopputyössä nimillä asiakas tai asiakasyritys.

2.6 Betoniraudoitus

Raudoitusta (ks. kuva 2) käytetään vahvistamaan betonista valmistettavien rakenteiden tai rakennelmien kuten siltojen lujuutta vaativia osia.

Kuva 2. Tyypillinen taivutettu betoninraudoitusverkko

Yksinomaan betonista valetun rakenteen vetomurtolujuus on hyvin pieni verrattuna sen puristuslujuuteen. Ilman raudoitusta ei olisi mahdollista valmistaa tarvittavan lujuisia etenkään vaakasuoria kantavia rakenteita, kuten palkit ja holvielementit. Teräksen vetomurtolujuus on huomattavasti suurempi kuin betonin ja terästä sekä betonia yhdistämällä on mahdollista saavuttaa riittävä lujuus [2].

Tässä insinöörityössä betoniraudoituksesta voidaan käyttää myös nimitystä verkko.

2.7 Progress 4GL

Progress 4GL tai toiselta nimeltään OpenEdge Advanced Business Language (OpenEdge ABL) on erityisesti liiketoimintaohjelmistojen kehitykseen erikoistunut ohjelmointikieli, jota kehittää ja ylläpitää Progress Software Corporation [3]. Tämän lopputyön ohjelmointiosuus on toteutettu Progress 4GL -kielellä.

2.8 Temp-table

Temp-table on ohjelmistotekninen nimitys väliaikaiselle, useimmiten client-päähän luodulle tietokantataululle, johon talletetaan haluttua tietoa helpompaa käsittelyä varten. Tässä työssä temp-tablet on valittu tavaksi, jolla suu-

rempia esim. laskennasta aiheutuvia tietomääriä hallitaan. Progress-kielessä temp-tableja käsitellään samalla tavalla kuin muitakin tietokantatauluja. Temp-tablet erottaa koodiriveiltä niiden nimestä, jotka alkavat w-kirjaimella.

Erityisen merkityksellisiä temp-tableja kuvauksineen:

- wLineList – lista PDF-asiakirjalle tulostettavan kuvan muodostavista janoista
- wMeasures – lista käyttäjän syöttämistä mitoista
- wMeasures3D – lista käyttäjän syöttämistä taivutusmitoista.

Merkityksellisiiin temp-tableihin tullaan viittaamaan tutkimuksessa suoraan nimeltä ja niiden erityinen esittely on siksi tarpeellista. Temp-table on muista ohjelmointikielikohtaisista käsitteistä poiketen esitelty tässä osiossa johtuen lukuisista suorista viittauksista ongelmia ratkaistaessa ja sitä kautta epäselvyyksien välttämiseksi.

2.9 PDF

PDF (Portable Document Format) on Adobe Systems Inc:in kehittämä PostScript-kieleen pohjautuva käyttöjärjestelmäriippumaton siirrettävä tiedostomuoto. Sitä käytetään pääasiallisesti sähköiseen julkaisemiseen, tulostamiseen ja painamiseen. PDF-tiedosto on tulostimen ja näytön tarkkuudesta riippumaton. PDF:n versio 1.7 on hyväksytty ISO 32000 -standardi [5].

Tässä lopputyössä suunniteltu sovellus tulostaa piirustuksen PDF-asiakirjana.

2.10 Include-tiedosto

Progress-kielen tapa kapseloida ohjelmakoodia sekä mahdollistaa erinäisten työkalujen käyttäminen Progress-kääntäjän esiprosessointivaiheessa. Tässä työssä oleellinen käyttötapa on kapselointi, jolla mm. ennalta lasketut sini-funktion arvot sijoitettiin erilliseen tiedostoon. Myös PDFinclude (ks. luku 2.11) liittyy ohjelmaan include-tiedostona.

2.11 PDFinclude

PDFinclude on include-tiedostossa (ks. luvussa 2.10) oleva ohjelmistokirjasto, joka mahdollistaa monimutkaisten PDF-asiakirjojen tuottamisen ilman

tarvetta kolmannen osapuolen tuotteisiin tai työkaluihin. Kyseessä on avoimen lähdekoodin sovellus [5].

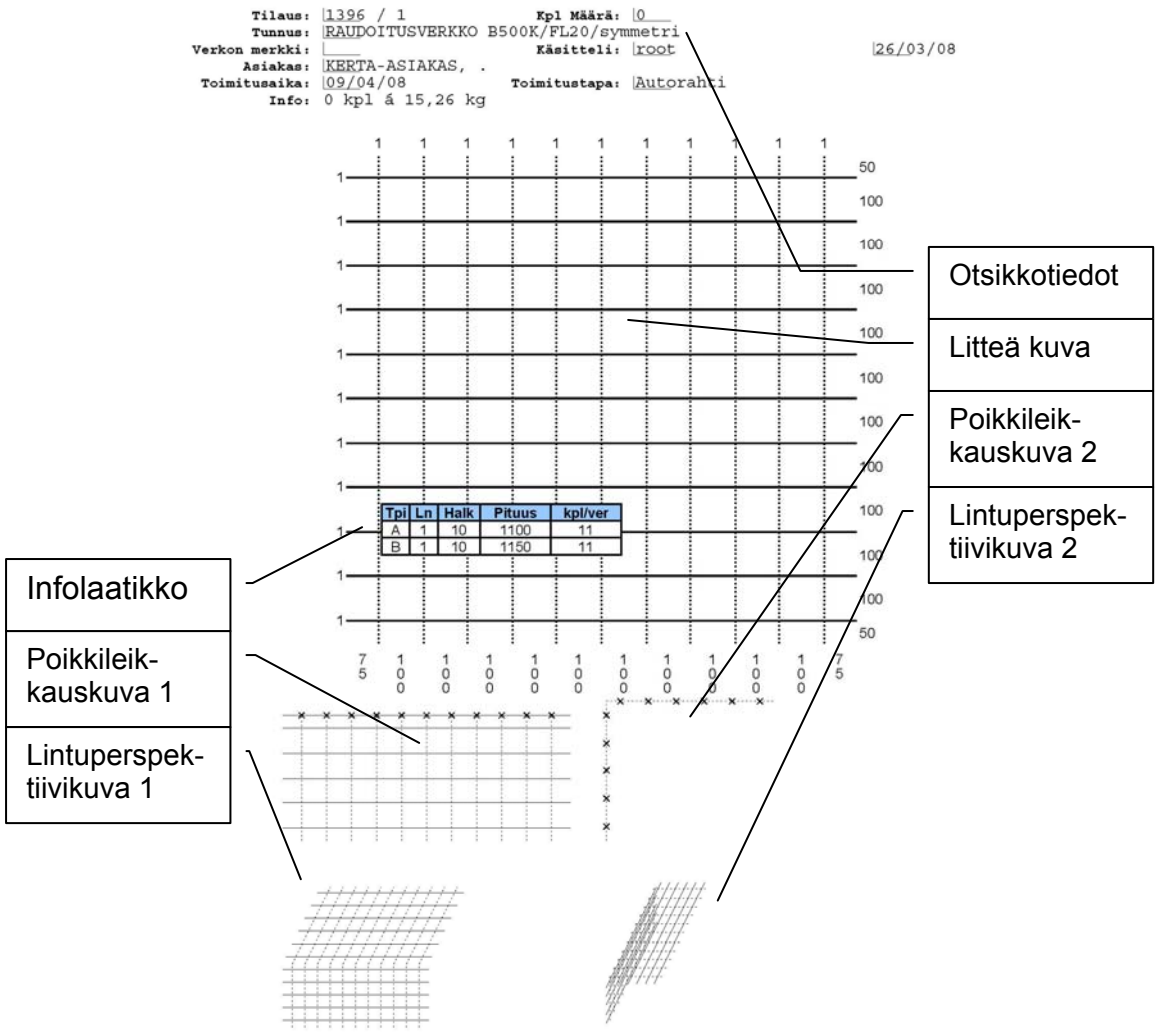
2.12 PDF-stream

PDFinclude käsittelee työn alla olevaa PDF-asiakirjatiedostoa tietovirtana, joka avataan ja käsittelyn jälkeen suljetaan. Vaikka prosessi oleellisesti muistuttaa käyttöjärjestelmille tyypillistä tiedostojen käsittelyä, ei kyseessä sellainen ole, vaan PDFincluden oma tapa hallinnoida limittäisiä tulostusoperaatioita.

2.13 Piirustuksen kuvat ja muut elementit

Lopputyön kannalta valmiiseen versioon kuuluu viisi eri kuvaa. Näistä käytetään nimitystä litteä tai kaksiulotteinen kuva, poikkileikkauskuva 1 ja 2 sekä lintuperspektiivikuvat 1 ja 2. Jokainen kuva on isometrinen. Piirustuksessa on myös otsikkotiedot sekä infolaatikko-elementit. Otsikkotietoihin kuuluu joukko piirustusta ohjaavia tietoja ja infolaatikkoon raudoituksen teknisiä tietoja.

Kuva 3 havainnollistaa elementtien sijoittelua ja nimityksiä valmiissa piirustuksessa:



Kuva 3. Piirustuksen kuvat ja muut elementit

Kuvassa 3 näkyvät piirustuksen eri elementit. Huomattavaa on, että poikkileikkaus- ja lintuperspektiivikuvat otetaan käyttöön vasta työvaiheessa 2. Näistä neljästä pienemmästä kuvasta voidaan käyttää tässä tutkielmassa yhteisnimitystä taivutuskuvat.

2.14 A- ja B-suuntaiset teräkset

A- ja B-suuntaisilla teräksillä tarkoitetaan samassa järjestyksessä pysty- ja vaakasuuntaisia raudoituksen teräksiä. Tätä merkintätapaa harjoitetaan ohjelmakoodien muuttujien nimeämisessä. Perimmäisenä syynä tähän ovat asiakasyrityksen vallitsevat toimintatavat. Ohjelmakoodia on helpompi seurata kun muuttujannimet vastaavat asiakasyrityksen antamaa palautetta.

2.15 Terässarjat

Terässarjalla tarkoitetaan yhtä joukkoa A- tai B-suuntaisia raudoituksen teräksiä, joille on määritetty yhteiset mitat ja muut parametrit. Tutkimuksessa viitataan eri sarjoihin ensin niiden suunnan ja sitten järjestysnumeron mukaisesti. Näin ollen esim. A2-sarja tarkoittaa toista vaakasuuntaista terässarjaa.

2.16 Raudoituskonfiguraattori

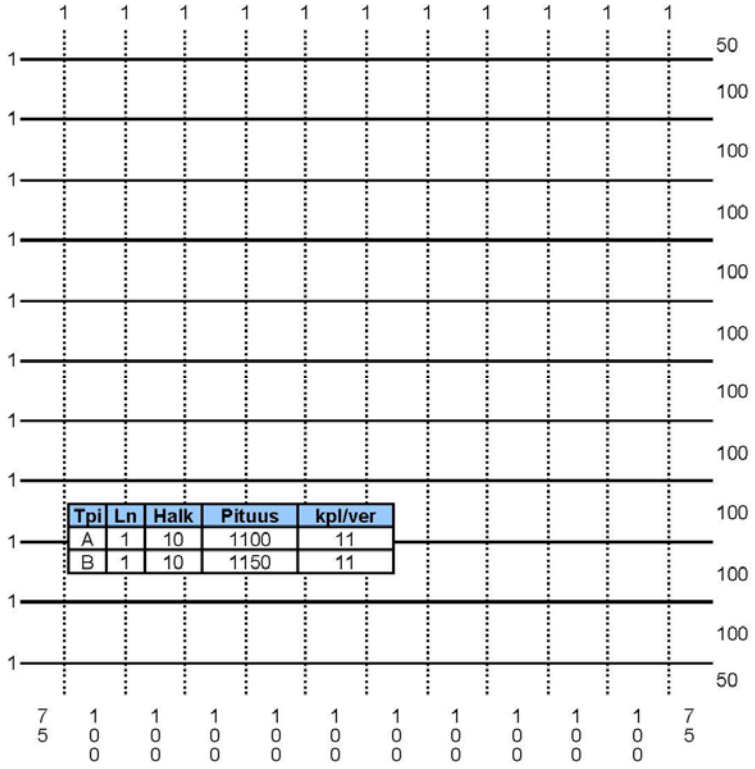
Raudoituskonfiguraattori on yhteisnimike verkon määrittelyohjelmalle, piirustuksen piirto-ohjelmalle sekä muille asiakasyrityksen betoniraudoitusten hallintaa parantaville Enterprisen ominaisuuksille, jotka kuuluvat saman projektin piiriin.

2.17 Esimerkkiverkko

Tässä lopputyössä käytettyjä esimerkkejä sovelletaan aina yksinkertaiseen esimerkkiverkkoon, jonka kaksi terässarjaa ovat määritetty seuraavien mittojen mukaisesti:

- A1 – Lkm = 11, Jako = 100, Posit = 0, Halk = 10, Pituus = 1100, Ylit1 = 50, Ylit2 = 50.
- B1 – Lkm = 11, Jako = 100, Posit = 0, Halk = 10, Pituus = 1150, Ylit1 = 75, Ylit2 = 75.

Selvennykset mittojen nimityksille löytyvät luvusta 4.2. Näillä mitoilla saatava esimerkki on ohessa olevan kuvan kaltainen:



Kuva 4. Esimerkkiverkko

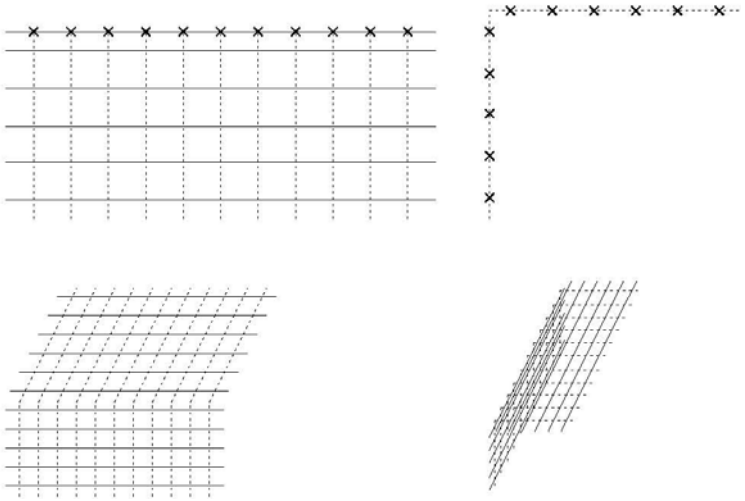
Kuva 4:ssä näkyy, kuinka esimerkkiverkossa on 11 saman paksuista pysty- ja vaakaterästä. Pystyteräkset eli A-teräkset ylittävät verkon molemmista päistä viidenkymmenen millimetrin verran ja B-teräkset puolestaan seitsemänkymmenviiden millimetrin verran. Huomattavaa on, että A-teräkset piirretään aina katkoviivalla piirustuksen helpomman hahmottamisen vuoksi[4].

[4] A- ja B-verkkojen erottaminen toisistaan saattaa olla kolmiulotteisten kuvien tapauksessa haastavaa ilman katkoviivan käyttöä.

Esimerkkiverkko on myös työvaiheesta 2 alkaen taivutettu mitoilla:

- Posit1 = 500, Halk = 90
- Posit2 = 600, Halk = 0

Näillä mitoilla saadaan esimerkkiverkolle seuraavanlainen taivutus:



Kuva 5. Esimerkkiverkon taivutus

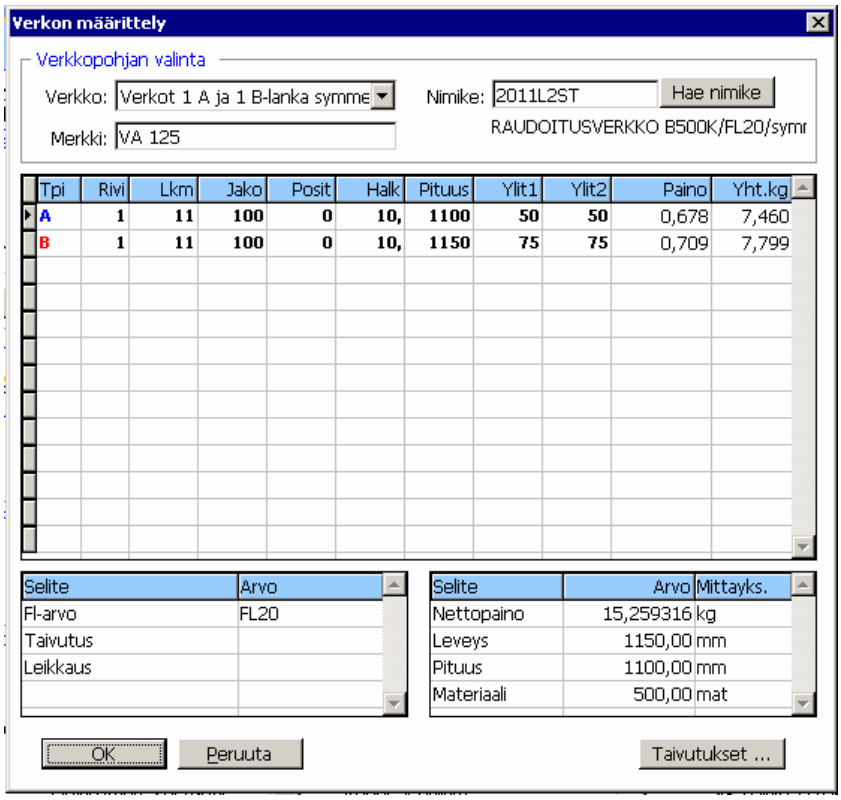
Kuva 5 näyttää, kuinka esimerkkiverkon ensimmäiset A-suuntaiset 500 millimetriä ovat pystysuunnassa, eli yhdeksänkymmenen asteen kulmassa ja loput 600 millimetriä vaakasuorassa, eli nollan asteen kulmassa.

On huomattava, että verkkoja syötettäessä taivutettavien osioiden yhteenlasketun pituuden tulee vastata raudoitusverkon A-sarjojen ääripäiden erotusta. Taivutus tulee siis olla määritelty koko verkon matkalta. Esimerkkiverkossa siis 500 + 600 = 1100, joka on A1-terässarjan pituus.

2.18 Verkon määrittelyohjelma

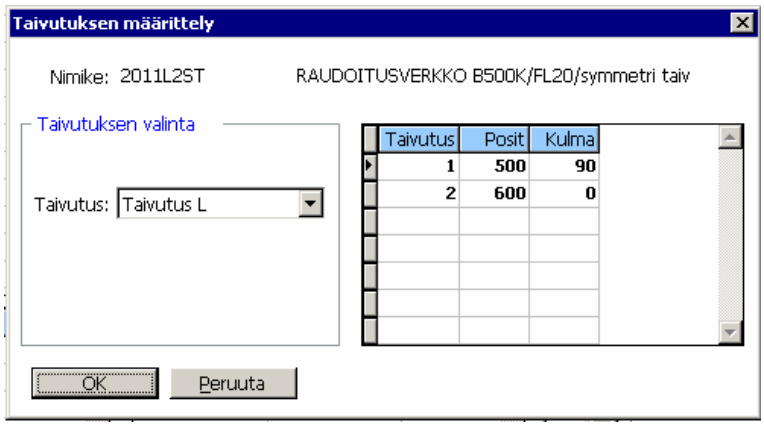
Verkon määrittelyohjelmassa syötetään mitat sekä muut parametrit halutunlaiselle myyntitilausrivin raudoitusverkolle. Ohjelma mahdollistaa erisuuntaisten terässarjojen vaihtelevan määrän syöttämisen sekä puuttuvien syötteiden automaattisen laskennan ylläpidettävien kaavojen mukaisesti.

Taivutusten syöttö on mahdollista vain nimikkeille joiden taivutus on erikseen sallittu, tämä estää väärinkäsitysten syntymistä.



Kuva 6. Verkon määrittelyohjelman ulkoasu

Kuvassa 6 näkyy esimerkkiverkon määrittely.



Kuva 7. Verkon määrittelyohjelman taivutuksenmäärittelyikkuna

Kuvassa 7 näkyy esimerkkiverkon taivutuksen määrittely. Tämä lopputyö keskittyy piirustuksen laskennallisen puolen toteuttamiseen, joten verkon määrittelyohjelman suunnittelu sekä tarkempi esittely ei kuulu tämän tutkimuksen piiriin.

2.19 Bufferi

Bufferi, eli puskuri on Progress-kielen nimitys tietyn tietokantataulun eräälle tietokantariville viittaavalle pointterille. Jokaiselle tietokantataululle on olemassa oletuspuskuri, joka on samanniminen tietokantataulun kanssa. Puskuri voi viitata tiettyyn riviin tai olla viittaamatta mihinkään.

Tarpeen tullen yhden tietokantataulun useampaan riviin pystytään viittaamaan luomalla lisä-buffereita. Tällaiset bufferit ovat ohjelmakoodissa nimetty alkamaan b-kirjaimilla ja muuten vastaamaan kyseessä olevan tietokantataulun nimeä. Lisä-buffereita käytetään tässä työssä mm. eri taivutusosioiden[5] samanaikaiseen viittaamiseen.

2.20 OLE

OLE (Object Linking and Embedding) on Enterprisen tapa liittää ulkoisia asiakirjoja sekä muita tiedostoja eri tilanteen mukaisiin yhteyksiin. Tätä OLE:a ei tule sekoittaa Microsoftin OLE-standardiin, joka periaatteeltaan on asiaa sivuava, mutta toiminnaltaan aivan eri asia.

Kun tässä lopputyössä puhutaan OLE:sta, kyseessä on aina Enterprisen määrittelemä OLE. OLE:a käytetään raudoituksen piirustuksen liittämiseen Enterprisen myyntitilausrivin yhteyteen.

2.21 DLL

DLL (Dynamic Link Library) on Microsoft Windows -käyttöjärjestelmän käyttämä jaettu kirjasto. Jaetut kirjastot ovat tietotekniikassa käyttöjärjestelmien tapa jakaa ohjelmakoodia ja dataa useiden ohjelmien kesken [6].

DLL:iä käytetään tässä työssä kahteen tarkoitukseen: trigonometristen funktioiden arvojen laskemiseen sekä käyttäjän oletus PDF-lukuohjelman automaattiseen käynnistämiseen.

[5] Tiedot taivutusosioista ovat temp-tablessa, joka kuitenkin käyttäytyy buffereiden tapauksessa kuten normaali tietokantataulu.

3 ALUSTAVA SUUNNITTELU JA TOTEUTUKSEN HAHMOTTELU

Tässä luvussa käydään läpi suuremman kokonaisuuden asiat, jotka vaikuttavat koko sovelluksen toteuttamiseen ja sen pitkän tähtäimen suunnitteluun.

3.1 ERP-rajapinta

Ohjelman suunnittelussa noudatettiin yrityksen sisäisiä ohjelmointikäytäntöjä. Näiksi luetaan mm. määrätty liittymä käyttöliittymän ja toiminnallisuuden välillä, virheenhallinta sekä kieliversiointi. Myös epävirallisia, vakiintuneita käytäntöjä, kuten temp-taulujen hyödyntämistä, sovellettiin.

3.2 Asiakkaan käyttäjärajapinta

Ohjelman toteutuksessa otettiin huomioon asiakkaan toivomukset sekä vallitsevat toimintatavat. Nämä määrittivät mm. muodon, jossa mitat syötettiin sekä lukuisan määrän lainalaisuuksia jotka ovat oleellisia kulmien laskennan kannalta. Nämä lainalaisuudet sekä mittojen merkitykset esitellään luvussa 4.2. ja esimerkki niiden soveltamisesta luvussa 2.17.

Asiakas oli myös tottunut käyttämään piirustuksessa esiintyvistä poikki- ja pystytangoista nimitystä A- ja B-tangot. Näitä nimityksiä käytetään myös ohjelman muuttujannimissä.

3.3 Graafisen median valinta

Piirustuksen mediaksi valittiin PDF-tuloste seuraavin perustein:

- Asiakkaan suostumus – PDF:n ominaisuudet täyttivät asiakkaan tarpeet myös hänen omasta mielestään.
- Ennalta löytyvät työkalut ja tutkijan kokemus niiden käyttöön – PDF-tiedostoja kirjoittava ohjelmakirjasto PDFinclude oli jo valmiiksi hankittuna[6] ja tutkija oli tutustunut niihin jo aikaisemmissa toimeksiannoissa.
- Tiedoston koko – PDF-tiedoston koko on pienempi kuin esim. saman tarkoituksen täyttävät ei-vektorigrafiikkaan pohjautuvan kuvatiedoston.

[6] PDFinclude on avoimen lähdekoodin kirjasto.

- Tietoturva – PDF-tiedostot voidaan tarpeen tullen signeerata digitaalisesti. Tätä toimintoa ei vielä nykyisissä työvaiheissa tarvittu.

3.4 Tulosteen ulkoasu

Asiakas toimitti mallin, joka saneli tulostettavan piirustuksen karkean ulkoasun. Tähän malliin kuitenkin tehtiin lisäyksiä uusien työvaiheiden myötä mitä helpotti se, että tulosteen ulkoasuun vaikuttavat proseduurit suunniteltiin alusta asti mahdolliset muutokset huomioon ottaen.

3.5 Ohjelmarunko

Tässä luvussa esitellään ohjelmarunko. Ohjelmarungolla tarkoitetaan ohjelmahierarkiassa korkeinta toteutuksen kannalta oleellista tasoa. Tätä korkeampi taso on esim. ERP-järjestelmän kernel-proseduuri, jonka pelkkä olemassaolo on ainoana seikkana tutkimuksen kannalta oleellinen. Myös PDF-tiedoston kirjoituksesta vastaavat työkalut sijoitetaan ohjelmarunkoa korkeammalle tasolle. Syynä tähän ovat PDF-kirjoitukseen erikoistuneen ohjelmakirjaston käyttövaatimukset.

Ohjelmarunko on ensimmäinen taso, joka rikkoo käsiteltävät ongelmat pienempiin, helpommin hahmotettaviin kokonaisuuksiin. Alemmas hierarkiassa mentäessä päästään lopulta tasolle, jolla ongelmat päästään ratkaisemaan esitellyllä, kolmivaiheisella tavalla.

Ohjelmarunko toimeenpanee jokaiseen kolmeen työvaiheeseen kuuluvia operaatioita, joten se kuuluu niihin kaikkiin. Suunnittelu on tehty siinä mielessä kumulatiivisella tavalla, että aikaisempien vaiheiden operaatiot eivät ole ikinä riippuvaisia tulevien vaiheiden operaatiosta. Näin toimien helpotettiin virheenpaikannusta ja lisättiin suunnittelun joustavuutta. Työvaiheet ovat siis rakennettu toinen toistensa päälle, eivät parannellen ja lisäten ominaisuuksia aikaisempiin vaiheisiin. Tämä ohjelmistoarkkitehtuurinen ratkaisu osoittautui arvokkaaksi ja lisäksi sen sivutuotteena eri työvaiheet pystytään tässä tutkielmassa järjestelmällisesti kuvaamaan, kuten seuraavia vaiheita ei vielä olisikaan. Tämä helpottaa kokonaisuuden käsittelyä.

Seuraavasta ohjelmalohkossa on esitetty ohjelmarungon takenne:

```
RUN OuterInit(OUTPUT xOk). /* Ohjelman alustus ERP-järjestelmän puolesta */
IF NOT xOk THEN RETURN.
RUN pgsubr/waiton.p.
RUN GetMeasures("A"). /* Verkon mittojen hakeminen ERP:istä */
RUN GetMeasures("B").
RUN GetInfo. /* Temp-taulun luonti table-työkalua varten */
RUN GetLineList("A"). /* Tee mittojen perusteella lista piirrettävistä viivoista */
RUN GetLineList("B").
RUN GetDimensions. /* Määritä piirrettävän verkon koko */
RUN GetBent("A"). /* Hae taivutuksen mitat */
RUN GetBent("B").

RUN GetBendEffects. /* Pyöreiden kulmien vaikutukset eri mittoihin */

RUN InitPdfFile(xFileName). /* Tiedostoonkirjoituksen alustus */
DO ON ERROR UNDO, LEAVE:
    RUN DrawHeader. /* Otsikkorivien piirto */
    RUN GetScaleDivider(6 / 8, 28 / 64). /* Hae kerroin, joka muuntaa kuvan mitat
piirustukseen sopiviksi */
    RUN DrawLines(0, 7 / 64). /* Piirrä kaksiulotteinen kuva */
    RUN DrawInfoBox2(hHandle). /* Piirrä tietoja piirustuksesta */

    RUN GetLineList3DBetterA. /* Terävien taivutusten vaikutukset, huomattavaa että pyöreät
kulmat huomioidaan tätä ennen joukoksi teräviä kulmia */
    RUN GetLineList3DBetterB.

    RUN ThreeD2TwoDPlane. /* Muunna kolmiulotteiset mitat kaksiulotteiseen tasoon,
poikkileikkaus 1 */
    RUN GetDimensions.
    RUN GetScaleDivider(1 / 3, 1 / 8).
    RUN Draw3DLines(0, 3 / 64). /* Piirrä kuva kaksiulotteiseen tasoon */

    RUN ThreeD2TwoDPlane2. /* …poikkileikkaus 2 */
    RUN GetDimensions.
    RUN GetScaleDivider(1 / 3, 1 / 8).
    RUN Draw3DLines(3 / 8, 3 / 64).

    RUN ThreeD2TwoDPlane3. /* …isometrinen lintuperspektiivi 1 */
    RUN GetDimensions.
    RUN GetScaleDivider(1 / 3, 1 / 8).
    RUN Draw3DLines(0, 1 / 6 + 3 / 64).

    RUN ThreeD2TwoDPlane4. /* …isometrinen lintuperspektiivi 2 */
    RUN GetDimensions.
    RUN GetScaleDivider(1 / 3, 1 / 8).
    RUN Draw3DLines(3 / 8, 1 / 6 + 3 / 64).

    RUN pdf_close(xPdfStream). /* Tiedostoonkirjoituksen sulku */
    RUN OpenIt(xFileName). /* Tiedoston avaaminen käyttäjän oletusohjelmalla */

    RUN SaveOle. /* PDF:n tallennus ERP:iin OLE-objektina */
    RUN pgsubr/waitoff.p.

    RETURN.
END.
RUN pgsubr/waitoff.p.
RUN pdf_close(xPdfStream). /* Varotoimenpide tiedostojen lukkiintumisen varalle */
```

Ohjelmalohko 1. Ohjelmarunko

Ohjelmalohkossa 1 nähdään ohjelmarunko kommentoituine proseduureineen. Huomattavaa on, kuinka eri toiminnot on jäsennelty ja kuinka proseduurit jakavat loogiset kokonaisuudet itseensä. Koko ohjelman karkea toimintalogiikka on luettavissa tältä tasolta.

4 TYÖVAIHE 1: KAKSIULOTTEINEN PIIRUSTUS

Tässä työvaiheessa pyritään vastaamaan kysymyksiin:

- Kuinka toteuttaa litteän kuvan tulostus ja muut piirustuksen elementit siten, että ratkaisun jatkokehittäminen on helppoa?
- Mitä muita asioita pitää ottaa huomioon piirustusta tulostaessa?

4.1 PDF-tulostuksen alustus ja alkutarkistukset

Ennen kuin tulostukseen vaadittava PDF-stream halutaan avata, on tehtävä joukko tarkistuksia. Näiden paikka on proseduurissa OuterInit, jossa tarkistetaan

- Onko luotava tiedosto jo jonkun käytössä?
- Onko kyseessä oleva myyntitilausrivi raudoitusverkko?

PDF-tulostus alustetaan luomalla uusi PDF-stream ja määrittämällä uudelle tiedostolle sen nimi, osoite sekä info-tiedot kuten PDF-author ja PDF-subject. Luotavan tiedoston osoite määräytyy Enterprisen general/olefile.p -proseduurin mukaan, joka varmistaa että Enterprisen OLE-tiedostot tallennetaan Enterprisen OLE-kansioon.

Oleelliset lähdekoodin proseduurit

- OuterInit – lähdekoodin (viite 6) sivu 5

4.2 Mittojen haku ERP-järjestelmästä

Käyttäjä on syöttänyt ERP-järjestelmän myyntitilausrivin taakse verkon määrittelyohjelmalla raudoitusverkon mitat ja muut piirustuksen edellyttämät parametrit. Kun piirustuksen tulostusohjelma ajetaan, tulee tarvittavat tiedot noutaa juuri tästä paikasta.

ERP-järjestelmä on suunniteltu eri käyttötarkoituksiin mukautuvaksi siten, että monet muutokset pystytään järjestelmään sisältyvillä työkaluilla toteuttamaan ohjelmakoodiin koskematta[7]. Yksi tällainen asiakaskohtainen muutos on raudoitusverkon mittatietojen lisääminen myyntitilausriville. Tästä toteu-

[7] Myyntitilausriviltä löytyvä mittojensyöttöohjelma on kuitenkin kovakoodaten tehty.

tuksesta johtuen kaikki asiakaskohtaiset lisäkentät[8] eivät ole omassa tietokantataulussaan, vaan taulussa johon asiakkaan kaikki omat lisäkentät ovat arkistointitietoineen syötetty. Tietojen hakeminen tästä virtuaalisesta taulusta toteutetaan käyttämällä ERP-järjestelmään tehtyä hakuproseduuria[9].

Haettavat muuttujat merkityksineen:

- Lkm – sarjaan kuuluvien terästen lukumäärä.
- Jako – sarjan terästen etäisyys toisistaan.
- Posit – terästen positio.
- Halk – terästen halkaisija.
- Ylit1 – verkon ylimenevän osuuden pituus teräksen alkupäässä.
- Ylit2 – verkon ylimenevän osuuden pituus teräksen loppupäässä.
- Kulma – varalle jätetty muuttuja jota ei käytetä toistaiseksi.

Esimerkki muuttujien käytöstä ja merkityksistä löytyy luvusta 2.17.

Muuttujat haetaan erikseen A- ja B-suuntaisille terässarjoille. Mittojen määrä vaihtelee käytettyjen terässarjojen määrän mukaisesti, ja haku toistetaan, kunnes uutta mittaa ei enää löydy. Mitat tallennetaan ohjelmakoodissa temp-tableen nimeltä wMeasures.

Seuraavaksi mitat muunnetaan kuvastamaan piirustuksen kaavakuvaan piirrettäviä janoja alku- ja loppukoordinaatteineen.

Oleelliset lähdekoodin proseduurit

- GetMeasures – lähdekoodin (viite 6) sivu 6

4.3 Mittojen muuntaminen piirrettäviksi janoiksi

Edellisessä ongelmassa hankitut mitat ovat siinä muodossa, jossa asiakas on ne tottunut yrityksensä sisällä ilmaisemaan. Tämä muoto ei ole käytännöllinen itse janojen mielekkään piirtämisen kannalta, joten sitä pitää jalostaa edelleen.

[8] Kuten tässä tapauksessa raudoitusverkon mitat ja muut parametrit
[9] Vaihtoehtona perinteinen tietokantahaku, joka olisi olosuhteista johtuen monimutkainen. Tämä monimutkaisuus on toteutettu varmasti oikealla tavalla ERP-järjestelmän yhteisessä hakufunktiossa.

Jalostus tehdään luomalla wMeasures-temp-table:n mittojen perusteella rivejä wLineList-temp-table:en, joka on oleellisesti lista piirrettävien janojen alku- ja loppukoordinaateista. Koordinaattien mittayksikköinä käytetään millimetriä. Tähän käytäntöön pyritään jatkossa jokaisessa koordinaatteja koskevassa laskutoimituksessa, syynä tähän on yksikkömuunnosten määrän minimoiminen[10].

Millimetristä luovutaan mittayksikkönä vasta, kun janoista koostuvaa kuvaa skaalataan PDF-asiakirjan sivulle sopivaksi. Tämä on seuraavan ongelman aihe.

Oleelliset lähdekoodin proseduurit

- GetLineList – lähdekoodin (viite 6) sivu 17

4.4 Kuvan koon skaalaus sivulle sopivaksi

Piirrettävän kuvan muodostavien janojen koordinaatit ovat toteutuksen tässä vaiheessa millimetreinä. PDF-asiakirjan käyttämät mittayksiköt ovat puolestaan omassa formaatissaan, joten koordinaatit eivät sellaisenaan sovellu kuvan piirtämiseen. Koordinaatit ovat skaalattava sivulle sopiviksi.

Skaalaus toteutettiin laskemalla jakaja, jota käyttämällä janojen koordinaatit saadaan mahtumaan sille määrättyjen, PDF-muotoisten rajojen sisälle. Koska oletetaan PDF-asiakirjan koon ja asennoitumisen voivan vaihdella asiakkaan tarpeiden mukaisesti, mainitut rajat ilmaistaan aina suhteellisena sivun kokoon nähden.

Jotta mainittu jakaja voitaisiin laskea, tulee tietää kuinka suuresta kuvasta on kysymys. Lasketaan siis kuvan, eli wLineList:in suurimpien ja pienimpien x- ja y-koordinaattien erotus saaden kuvan leveys sekä korkeus. Näiden avulla lasketaan arvo jakajalle käyttämällä kaavaa 1:

$$jakaja = \max\left(\frac{kuvanLeveys}{sivunLeveys * sivuRaja}, \frac{kuvanKorkeus}{sivunKorkeus * pystyRaja}\right) \quad (1)$$

Verkkojen muodot voivat vaihdella, joten kaava 1 ottaa huomioon molemmat sivu- ja pystysuunnassa ensimmäisenä vastaan tulevan rajan.

[10] Tämä helpottaa virheenpaikannusta ja vähentää mahdollisten sekaannusten määrää.

Oleelliset lähdekoodin proseduurit

- GetDimensions – lähdekoodin (viite 6) sivu 9
- GetScaleDivider – lähdekoodin (viite 6) sivu 10

4.5 Kuvan tulostus ja piirustukseen liittyvät tekstit

Kuvan muodostavien janojen koordinaatit sekä ne sivulle sopivaksi skaalaava jakaja ovat tiedossa, joten kuva voidaan piirtää. Käydään läpi wLineList:in jokainen rivi ja piirretään asiakirjalle janat, jotka muodostavat kaksiulotteisen kuvan. Myös piirustukseen liittyvät tekstit lisätään tässä vaiheessa.

Piirustukseen liittyviä tekstejä ovat A- ja B-suuntaisten lankojen ohjausnumerot sekä lankojen väliset etäisyydet. Ohjausnumeroiden paikka on A-suuntaisilla langoilla kasiulotteisen kuvan vasemmassa reunassa ja B-suuntaisilla kuvan yläpuolella. Lankojen väliset etäisyydet kuuluvat vastaavasti oikealla ja alas.

Tekstien kirjoitus toteutetaan sitä varten luoduilla funktioilla, jotka osaavat kahden vierekkäisen janan koordinaattien perusteella määrittää sijaintinsa sekä sisältönsä.

Oleelliset lähdekoodin proseduurit

- DrawLines – lähdekoodin (viite 6) sivu 10

4.6 Otsikkotietojen haku ja tulostus

Piirustuksesta tulee löytyä otsikkotiedot itsestään selvistä syistä.

Otsikkotiedot haetaan tietokantahauilla myyntitilaus- ja myyntitilausrivitauluilta sekä suorista relaatioista niiden takaa.

Tulostusta varten tehtiin TextTopic ja TextValue -proseduurit, jotka helpottavat kirjoitettavien tekstien muotoilua ja sijoittelua. TextTopic-proseduuri huolehtii kirjoitettavan tiedon tyypin kirjoittamisesta ja TextValue puolestaan itse tiedon kirjoittamisesta.

Oleelliset lähdekoodin proseduurit

- DrawHeader – lähdekoodin (viite 6) sivu 13

4.7 PDF-tulostuksen lopetus

Kun tulostus on valmis tulee PDF-stream sulkea. Sulkeminen tapahtuu käyttämällä PDFinclude-ohjelmakirjaston proseduuria pdf_close.

4.8 PDF-tiedoston avaaminen käyttäjän oletusohjelmalla

Kun piirustus on onnistuneesti tallennettu PDF-muotoon, tulee se avata nähtäväksi käyttäjän oletus PDF-lukuohjelmalla. Tästä suoriuduttiin käyttämällä Windowsin DLL:ää shell32.dll ja tarkemmin sen toimintoa ShellExecuteA, joka hakee Windowsin rekisteristä oletusohjelman eri tiedostotyypeille ja sitten käynnistää sen.

Oleelliset lähdekoodin proseduurit

- OpenIt – lähdekoodin (viite 6) sivu 15

4.9 PDF-tiedoston liittäminen ERP-järjestelmän myyntitilausrivin yhteyteen OLE:a hyväksikäyttäen

Piirustus on tallennettu Enterprisen OLE-kansioon, mutta se ei ole vielä linkitetty myyntitilausriville. Tämä ratkaistiin käyttämällä Enteprisen proseduuria prog/sysole/oleadd.p.

Oleelliset lähdekoodin proseduurit

- SaveOle – lähdekoodin (viite 6) sivu 21

4.10 Työvaiheen lopputulos

Ensimmäisen työvaiheen lopussa ohjelma piirtää käyttäjän myyntitilausriville antaman syötteen perusteella kaksiulotteisen, litteän piirustuksen mittoineen. Otsikkoriville kirjoitetaan asiakkaan määrittelyssä toivotut tiedot. Piirustus on sellaisenaan käyttökelpoinen ja riittävä havainnollistamaan taivuttamattomia verkkoja. Tyypillinen ensimmäisen työvaiheen jälkeinen piirustus on nähtävissä liitteessä 3.

Seuraava työvaihe laajentaa piirustuksen kuvaamaan myös taivutettuja verkkoja.

5 TYÖVAIHE 2: KOLMIULOTTEINEN PIIRUSTUS JA TAIVUTUKSET

Tässä työvaiheessa pyritään vastaamaan kysymyksiin:

- Kuinka toteuttaa taivutukset siten että ratkaisun jatkokehittäminen on helppoa?

5.1 Taivutusmittojen haku ERP-järjestelmästä

Tämän ongelman ratkaisu vastaa lukua 4.2 lähes täydellisesti. On huomattava, että ERP:in hallinnoinnin helpottamiseksi haettavien muuttujien nimet eivät havainnollista taivutusmittojen luonnetta, vaan ovat samoja kuin edellä mainitussa luvussa. Selvennös muuttujien merkityksistä:

- Posit – kertoo taivutettavan osion pituuden.
- Halk – kertoo osion taivutuskulman.
- Loput muuttujat vastaavat merkityksiltään luvussa 4.2 kuvattuja. Jotkut taivutusten kannalta epäoleelliset muuttujat jäävät tarpeettomiksi.

Saadut muuttujat eivät ole luvussa 4.2 poiketen terässarjakohtaisia, vaan koskevat kaikkia teräksiä. A- ja B-suuntaiset taivutukset haetaan erikseen[11].

Seuraavassa luvussa sovelletaan näin saatuja mittoja aikaisemmin muodostetun litteän kuvan kanssa saaden aikaiseksi taivutettua verkkoa kuvaava koordinaattijoukko.

Oleelliset lähdekoodin proseduurit

- GetBent – lähdekoodin (viite 6) sivu 7

5.2 Piirrettävien janojen jakaminen taivutettaviin osioihin ja taivutettujen kolmiulotteisten koordinaattien laskeminen

Edellisessä luvussa haettiin tarvittavat mitat, jotta aikaisemmin hankittu litteä raudoitus voitaisiin taivuttaa. Tämän toteuttamisessa olennaiset muuttujat ovat Posit ja Halk, joiden luonteesta ja käytöstä löytyy esimerkki luvussa 2.17.

[11] B-terästen taivutusten syöttäminen on estetty verkonmääritysohjelmassa, mutta niiden nopea käyttöönotto on ennakoivasti mahdollistettu.

5.2.1 Osiointi

Litteän kuvan janat on tallennettu temp-tableen nimeltä wLineList. Nämä janat pilkotaan pienemmäksi janoiksi katkaisten ne Posit-muuttujien määräämiltä pituuksilta. Näin saadut janat tallennetaan wLineList3D-temp-tableen, jonne ne jäävät odottamaan taivutusta.

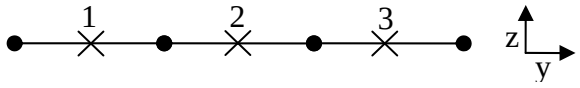
Temp-tableen tallennetaan myös tieto osioiden perättäisestä järjestyksestä. Tätä tullaan tarvitsemaan itse taivutusvaiheessa joka käsitellään seuraavaksi.

5.2.2 Osioiden taivutus

Kun janat ovat edellisen luvun mallin mukaisesti osioitu, ne voidaan taivuttaa. Tämän ongelman ratkaisu käsitellään esimerkkien avulla, ja kuten taivutukset tehtäisiin ainoastaan A-suuntaisille teräksille. B-suuntaiset taivutukset ohjelma suorittaa kääntämällä jo taivutetun verkon 90 astetta xy-tason mukaisesti ja toistamalla saman proseduurin.

Tulevissa tämän luvun esimerkeissä käytetään luvun 2.17 esimerkkiverkon tapausta paremmin tarkoitukseen soveltuvaa mielikuvituksellista verkkoa. Tässä verkossa on 3 yhdenpituista taivutusosiota ja se kuvataan ainoastaan poikkileikkauskuvakulmasta, josta parhaiten näkyvät taivutusten vaikutukset.

Alkuasetelmassa janat ovat taivutusten mukaisesti osioituna temp-tablessa wLineList3D. Otetaan käyttöön aikaisemmin esitelty, taivutuksiltaan kolmiosainen verkko.



Kuva 8. Taivutusta selventävä verkko alkuasetelmassa

Kuvassa 8 näkyy litteästä verkosta kolmeen taivutusosioiden mukaiseen osaan katkotut janat. Numerot janojen päällä merkitsevät niiden järjestysnumeroa alusta loppuun ja rastit katsojasta poispäin kulkevia teräksiä. On huomattava, että tästä kuvakulmasta katsoen y-akselin kanssa yhdensuuntaiset janat piiloutuvat toistensa taakse.

Taivutetaan seuraavaksi osiot seuraavien mittojen mukaisesti:

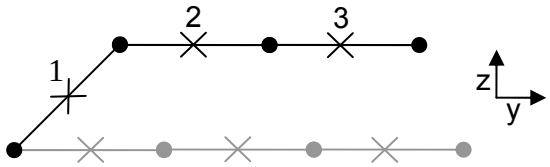
- Osio 1: Posit = 100mm, Halk = 45 astetta.

- Osio 2: Posit = 100mm, Halk = -45 astetta.
- Osio 3: Posit = 100mm, Halk = 0 astetta.

Janat käsitellään osiomukaisessa järjestyksessä, vaikutusten jatkuessa myös kaikkiin järjestyksessä seuraaviin osioihin. Käsittelyyn kuuluu:

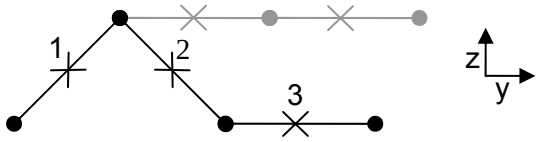
- käsiteltävän osion kaikkien janojen päätepisteiden[12] kääntäminen haluttuun asentoon osion alkupisteen toimiessa käännöskohtana.
- järjestyksessä kaikkien seuraavien osioiden janojen siirtäminen osion päätepisteen edellisellä tavalla kulkema matka.

Näin toimien päästään kohdatuissa erikoistilanteissakin toimivaan ratkaisuun[13]. Seuraava kuvasarja havainnollistaa edellä mainittua käsittelyä:



Kuva 9. Näkymä ensimmäisen osion käsittelyn jälkeen

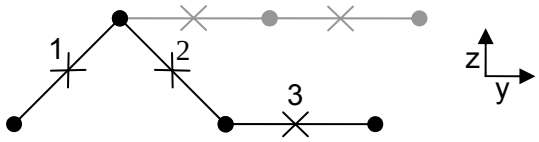
Kuvassa 9 näkyy, kuinka ensimmäisen osion käsittelyn jälkeen janat ovat liikkuneet. Harmaalla väritetty osa näyttää missä janat olivat aikaisemmin. Ensimmäiseen osioon kuuluvat janat ovat kääntyneet 45 astetta osion aloituspisteen mukaisesti. Ensimmäistä osiota seuraaviin osioihin kuuluvat janat ovat siirtyneet saman matkan ensimmäisen osion loppupään kanssa.



Kuva 10. Näkymä toisen osion käsittelyn jälkeen

Kuvassa 10 nähdään, kuinka ensimmäisen osion janat ovat pysyneet paikallaan. Vastaavasti toinen osio on kääntynyt ja kolmas siirtynyt. Kolmannen osion taivutuksen asteluku on 0, joten se ei aiheuta taivutukseen muutosta, kuten Kuva 11 näyttää.

[12] wLineList3D:hen ja muihin wLinelisteihin tallentuu myös tieto onko kyseessä janan alku- vai loppupää.
[13] Tätä ratkaisua pohdittiin ”yritys ja erehdys” -menetelmällä. Kyseinen yritys on järjestyksessä kolmas. Toinen hylätty ratkaisu perustui rekursiiviseen malliin, joka todettiin liian virhealttiiksi. Ensimmäinen yritys luettakoon vedosteluksi.



Kuva 11. Näkymä viimeisen osion käsittelyn jälkeen

Kuva 11:sta nähdään, kuinka kolmannen vaiheen asteluku 0 ei aiheuta muutosta aikaisemman osion käsittelyn lopputulokseen (kuva 10) nähden, koska taivuttamattoman osion alkuasteluku on valmiiksi 0. Esimerkkitaivutus on siis valmis.

Oleelliset lähdekoodin proseduurit

- GetLineList3DBetterA – lähdekoodin (viite 6) sivu 17
- GetLineList3DBetterB – lähdekoodin (viite 6) sivu 17

Janoilla on tämän luvun toteutuksen jälkeen x-, y- ja z-koordinaatit. Sellaisenaan ne eivät sovi PDF-tulosteen 2-ulotteiseen tasoon. Tämä muutos tehdään seuraavassa luvussa.

5.3 Kolmiulotteisen kuvan muuntaminen kaksiulotteisiin tasoihin isometrisinä näkyminä

Taivutuksen jälkeen janat ovat kolmiulotteisessa avaruudessa. Seuraavaksi ne muunnetaan PDF-tulosteen kaksiulotteiseen tasoon sopivaksi. Tämä toteutetaan isometrisessä perspektiivissä, jolloin mittasuhteet säilyvät helposti hahmotettavana palvellen piirustuksen etua. Ratkaisu on myös aitoon perspektiiviin nähden helpompi[14] toteuttaa.

Seuraavaksi esiteltäviä ratkaisuja sovelletaan pienin eroavaisuuksin jokaiseen neljään tehtyä taivutusta esittävään kuvaan, joiden lasketut uudet x- ja y-koordinaatit tallennetaan wLineList–tauluun aikaisempaa piirtämisvaihetta vastaavaa operaatiota varten.

5.3.1 Poikkileikkauskuvat

Poikkileikkauskuvan isometriseen perspektiiviin päästään karkeasti jättämällä yksi janojen koordinaateista pois ja käyttämällä jäljelle jäävää kahta koordinaattia x- ja y-koordinaatteina. Tällöin pois jätetty koordinaatin akseli määrittää kuvan katselukulman suunnan.

[14] Tutkija perustaa tämän väitteen aikaisempaan kokemukseensa aiheesta.

5.3.2 Lintuperspektiivikuvat

Lintuperspektiivikuvan vaatimaan muotoon päästään jatkamalla edellä esiteltyä poikkileikkauskuvan ajatusta hyödyntämällä pois jätettyä, katselukulman määrittävää koordinaattiarvoa.

Selityksen yksinkertaistamiseksi toteutus kerrotaan esimerkkinä käyttäen edellisestä eroten hieman viistoitettua z-akselia näkökulman suuntana. Esimerkistä poikkeavissa tapauksissa toteutus on sama, mutta koordinaattien nimet vaihtelevat. Esimerkki vastaa ensimmäisen lintuperspektiivikuvan toteutusta.

Janojen päätepisteiden kaksiulotteiset x- ja y-koordinaatit lasketaan hyödyntämällä kaavaa 2:

$$f(x, y, z) = (x + 0.5 * y, y + z) \quad (2)$$

joka palauttaa kaksiulotteisen koordinaatin vektorin kolmiulotteisten koordinaattien perusteella.

Oleelliset lähdekoodin proseduurit

- ThreeD2TwoDPlane – lähdekoodin (viite 6) sivu 19
- ThreeD2TwoDPlane2 – lähdekoodin (viite 6) sivu 19
- ThreeD2TwoDPlane3 – lähdekoodin (viite 6) sivu 19
- ThreeD2TwoDPlane4 – lähdekoodin (viite 6) sivu 19

5.4 Kuvan koon skaalaus sivulle sopivaksi

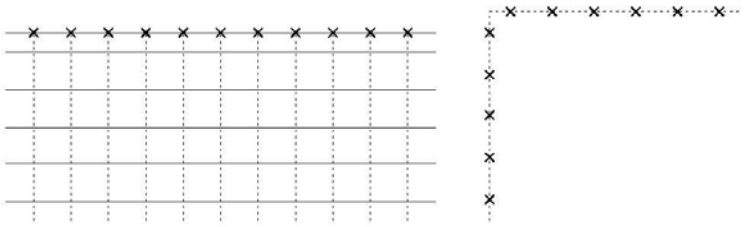
Uudet neljä kuvaa skaalataan vuorollaan sivulle sopivaksi kuten luvussa 4.4. Merkittävää on, että aikaisemmassa vaiheessa suunnitellut työkalut soveltuvat tähän tarkoitukseen muutoksitta.

Oleelliset lähdekoodin proseduurit

- GetDimensions – lähdekoodin (viite 6) sivu 9
- GetScaleDivider – lähdekoodin (viite 6) sivu 10

5.5 Kuvan tulostus

Tulostus toteutetaan lukua 4.5 vastaavalla tavalla sillä poikkeuksella, että piirustuksissa usein esiintyvät, katsojasta suoraan poispäin osoittavat janat kuvataan rastilla. Tällaiset janat tunnistetaan helposti siitä, että niiden molempien päiden koordinaatit ovat samat.



Kuva 12. Rastit taivutuskuvissa

Kuvasta 12 nähdään, kuinka muuten näkymättömät, katsojan näkökentän suuntaiset teräkset kuvataan rasteilla.

Oleelliset lähdekoodin proseduurit

- Draw3DLines – lähdekoodin (viite 6) sivu 20

5.6 Työvaiheen lopputulos

Toisen työvaiheen lopussa sovellus osaa taivuttaa verkon kolmiulotteisiin muotoihin sekä esittää ne graafisesti. Toteutusta voidaan käyttää taivutusmittojen graafiseen tulkitsemiseen sekä esim. tehdasvaraston tehokkaampaan hallintaan. Viimeisessä työvaiheessa tutkitaan mahdollisuuksia luonnottomista, terävistä taivutuskulmista eroon pääsemiseksi.

Tyypillinen toisen työvaiheen jälkeinen piirustus on nähtävissä liitteessä 4, jossa huomioitavaa ovat uudet neljä pientä kuvaa, joista näkyvät taivutukset.

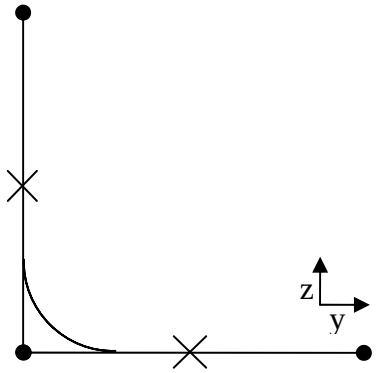
6 TYÖVAIHE 3: PYÖREÄT KULMAT

Tässä työvaiheessa pyritään vastaamaan kysymyksiin:

- Kuinka toteuttaa pyöreät taivutukset siten, että ratkaisun jatkokehittäminen on helppoa?
- Kuinka määrittää poikkiterästen sijainnit pyöreätaivutteisessa raudoituksessa siten, että ne eivät liiku terävätaivutteiseen nähden taivutuksesta johtuvan lyhenemän vuoksi?
- Minkälaisia pyöreisiin taivutuksiin liittyviä piirteitä pitää ottaa huomioon ja miten?

6.1 Lyhenemän huomiointi 2-ulotteisessa kuvassa

Lähtökohtana pyöreiden taivutusten toteutuksessa oli se, että taivutuksesta johtuva lyhenemä ei saa vaikuttaa poikkiterästen sijaintiin.



Kuva 13. Teräväkulmainen sekä pyöreäkulmainen taivutus

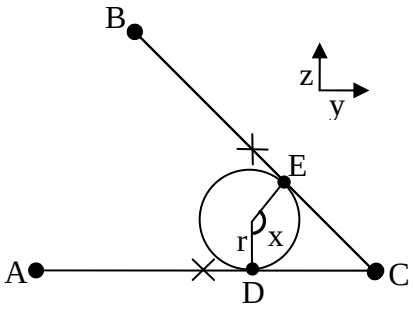
Kuva 13 näyttää tilanteen, jossa päällekkäin olevien taivutusten rasteilla kuvatut poikkiteräkset eivät ole liikkuneet toisiinsa nähden. On otettava huomioon, että pyöreän taivutuksen terästä on lyhennetty, joten molempien taivutusten teräkset myös alkavat ja loppuvat samoissa pisteissä. Tämä on taivutusmalli, johon pyritään.

Jotta esitellyn kaltaiseen taivutukseen päästään, on kyettävä laskemaan kuinka paljon taivutettu teräs on lyhyempi kuin taivuttamaton. Tämä ratkaistaan seuraavassa luvussa.

6.1.1 Pyöreän taivutuksen aiheuttaman lyhenemän laskeminen

Kun teräs taivutetaan käyttäen sovittua[15] taivutussädettä, sen pituuden tulee muuttua lyhyemmäksi, jotta sen alku- ja loppupisteet vastaisivat samoin määrityksin tehtyä teräväkulmaista terästä.

Tämän ongelman ratkaisun apuna käytetään seuraavaa 45 asteen esimerkkitaivutusta:



Kuva 14. Lyhenemää havainnollistava esimerkkitaivutus

Kuva 14 havainnollistaa tilanteen, josta tulee selvittää janojen AC ja CB ja janojen AD, EB sekä kaaren DE pituuksien summien yleispätevä erotus, eli lyhenemä säteen ollessa r ja taivutuskulman ollessa x.

Tämä ratkaistiin kaavalla 3:

$$s = 2 * r * \tan(x / 2) – r * x \quad (3)$$

missä s on lyhenemä, ja x on erisuuri kuin ±π, koska 180 asteen kulmia ei esiinny luonnossa. On huomattava, että taivutuskulman suuruus x kuvataan radiaaneina, mutta sen arvo haetaan asteina sitä varten tehdyllä ohjelmafunktiolla angleDelta, jonka toimintaperiaatetta kuvaa kaava 4:

$$\Delta x = \left| x_1 - x_2 \right| \bmod 180 \quad (4)$$

missä x-muuttujien arvot ovat taivutusosioiden taivutuskulmia. Kuvatussa (Kuva 14) esimerkkitaivutuksessa niiden arvot olisivat 0 ja 135 astetta ja palautettava arvo 45 astetta. Huomattavaa deltakulman kaavassa on jakojäännösoperaattorin käyttö, jota on laajennettu poikkeuksellisesti toimimaan määrätyllä desimaalitarkkuudella pyöristysvirheiden välttämiseksi.

[15] Taivutussäde määräytyy taivutettavan teräksen paksuuden mukaan.

Oleelliset lähdekoodin funktiot

- Shrinkage – lähdekoodin (viite 6) sivu 5
- AngleDelta – lähdekoodin (viite 6) sivu 5

Nyt kun taivutuksen aiheuttaman lyhenemän tarkka suuruus on tiedossa, pitää pystyä valitsemaan mistä kohtaa kaksiulotteista verkkoa se otetaan pois siten, että poikkiteräkset pysyvät teräväkulmaiseen taivutukseen nähden liikkumattomina. Tämä ratkaistaan seuraavassa luvussa.

6.1.2 Lyhenemän soveltaminen kaksiulotteiseen kuvaan sekä taivutusosioihin

Taivutuksista aiheutuvaa lyhenemää ei voida yksinkertaisesti vähentää terästen kokonaispituudesta, sillä tästä seuraisi poikkiterästen ja taivutusosioiden sijainnin arvaamattomuutta.

Ratkaisuna taivutuksen lyhenemää tulee soveltaa sen tapahtumapaikalla, eli taivutusalueella. Tällä tarkoitetaan aluetta, joka alkaa taivutusosioiden välistä, ja jatkuu siitä molempiin suuntiin niin pitkälle kuin pyöreän taivutuksen vaikutus jatkuu.

Taivutusalueen keskipiste sijaitsee luvun 5.2.1 mallista johtuen aina janojen päätepisteiden kohdalla. Lyhenemää sovelletaan tästä alueesta käsin siten, että teräsverkko vetäytyy sitä kohti molemmin puolin yhteensä lyhenemän verran, jättäen taivutusalueen keskipisteessä sijaitsevat janojen päätepisteet ennalleen.

Tätä toimenpidettä helpottaa sääntö, jonka mukaan taivutusalueella ei saa sijaita poikkiteräksiä, koska tehdaslaitteisto ei sitä mahdollista. Jokainen janan päätepiste siis muuttuu joko puolet lyhenemästä tai ei ollenkaan, ja taivutusalueella tapahtuvalta poikkiterästen tilanteenmukaisen siirtymän huomioinnilta vältytään.

Vastaavalla tavalla lyhenemää sovelletaan myös taivutusosioiden sijainteihin, joiden tulee liikkua kaksiulotteisen kuvan muutosten mukana. Näin ollen temp-tablien wLineList ja wMeasuresBend samansuuntaiset kumulatiiviset pituudet pysyvät harmoniassa, kuten esimerkissä luvussa 2.17.

Oleelliset lähdekoodin proseduurit

- GetBendEffects – lähdekoodin (viite 6) sivu 21

Oleelliset lähdekoodin funktiot

- Shrinkage – lähdekoodin (viite 6) sivu 5

Lyhenemän seuraukset ovat nyt riittävällä tarkkuudella huomioitu. Seuraavassa luvussa tutkitaan itse pyöreiden kulmien graafisen esityksen toteutusta.

6.2 Pyöreiden taivutusten graafinen toteutus

Käytettävissä olevat PDFincluden työkalut eivät pidä sellaisenaan sisällään pyöreiden kulmien kuvaamiseen soveltuvaa työkalua. Se pitää siis tätä tarkoitusta varten tehdä. Luvussa 5.2 kerrottiin, kuinka janat pilkotaan terävästi taivutettaviin osioihin. Tätä toimintoa voidaan hyödyntää edelleen lisäämällä yhtä taivutusta kohden määrä osioita, joiden taivutuskulma muuttuu lineaarisesti vanhan osion taivutuskulmasta järjestyksessä seuraavan osion taivutuskulmaksi. Tällaisten osioiden määrää kasvattaessa kulma alkaa näyttää pyöreältä. Lopullisessa toteutuksessa osioiden lukumääräksi valittiin 6. Luku on pieni, ja sen valintaa perusteltiin PDFincluden[16] huonolla vektoreiden erottelutarkkuudella, joka hienojakoisilla viivoilla aiheutti arvaamattomia tuloksia.

Tässä luvussa esiteltyä menetelmää käytetään ainoastaan graafisen ulkoasun toteuttamiseen. Muissa luvuissa esiteltävät pyöreisiin taivutuksiin liittyvät laskelmat toteutetaan tarkkoja arvoja hyödyntämällä.

Esitellyistä lyhyistä osioista käytetään jatkossa nimitystä osataivutus ja niiden tarkempaan toteutukseen perehdytään seuraavaksi.

6.2.1 Osataivutusten toteutus

Osataivutusten pituuden ja asettelun valintaan nähtiin olevan 3 vaihtoehtoa, joissa kussakin on omat hyötynsä ja haittansa:

- osataivutusten yhteispituus = taivutettavan kulman kaarta pitkin kulkevien jänteiden pituuksien summa
- osataivutusten yhteispituus = taivutuksen kaaren tarkka pituus
- optimaalinen ratkaisu, joka on tuntematon.

[16] Huono erottelutarkkuus saattaa myös olla peräisin itse PDF-formaatista. Asiaa ei tutkittu sen tarkemmin, koska ongelmia ei esiintynyt osataivutusten lukumäärällä 6.

Näistä toteutuksista tullaan käyttämään nimityksiä jänne-, kaari-, sekä optimaalinen toteutus. Jännetoteutuksen etuna luvun 5.2 mallia soveltamalla taivutettavan teräksen suorien osuuksien sijainnit pysyvät täsmälleen samalla paikalla kuin terävässä taivutuksessa, mutta piirretyssä kokonaispituudessa esiintyy pientä lyhenemää[17]. Kaaritoteutus käyttäytyy täsmälleen päinvastoin ja mahdollinen optimaalinen toteutus puolestaan suoriutuu molemmista aspekteista virheettömästi. Valinta näiden toteutuksien välillä tehdään niiden aiheuttamien haittojen perusteella.

Jännetoteutuksen haittana piirretyn teräksen pituus ei vastaa tarkalleen todellista terästä. Piirustuksen luettavuuden kannalta tästä ei ole olennaista haittaa, mutta ohjelmallinen toteutus vaatii kohtuuttomia muutoksia tiettyjen virhetilanteiden huomioimisen suhteen. Tämä toteutus hylätään, mutta säilytetään silti lähdekoodissa uusien asiakaslähtöisten määritysten varalta.

Kaaritoteutuksen haittapuoli on sen aiheuttama marginaalinen siirtymä taivutettavan teräksen suorissa osissa. Käytännön kokeiden perusteella virhe ei silminnähtävä ja täten toteutus on käyttökelpoinen siinä määrin, että mahdollisen optimaalisen toteutuksen etsimiseen ei käytetty aikaa.

Oleelliset lähdekoodin proseduurit

- GetBendEffects – lähdekoodin (viite 6) sivu 21

Oleelliset lähdekoodin funktiot

- AngleDelta – lähdekoodin (viite 6) sivu 5
- ChordLen – lähdekoodin (viite 6) sivu 5
- ArcLen – lähdekoodin (viite 6) sivu 5
- IsClockwiceTurn – lähdekoodin (viite 6) sivu 5

Seuraavaksi tutkitaan, kuinka tämä luvun toteutuksien myötä kasvaneesta, trigonometristen funktioiden määrällisestä tarpeesta aiheutuva toiminnan hitaus saadaan estettyä.

[17] Tätä lyhenemää pystyttäisiin pienentämään kasvattamalla osataivutusten määrää, joka ei kuitenkaan ole tarkoituksenmukaista PDF-tulostuksen huonon erottelutarkkuuden vuoksi.

6.3 Trigonometristen funktioiden laskennan optimointi

Pyöreiden kulmien myötä trigonometristen funktioiden laskennan määrä kasvoi keskimäärin muutamasta sadasta moniin tuhansiin. Tästä aiheutui tuntuva, kymmenien sekuntien viive ohjelmaa ajaessa. Käytössä oli lisäksi perustoimistokäyttöön verrattuna nopea tietokone, joten loppukäyttäjän viive olisi oletettavasti tätäkin suurempi. Näin ollen viiveeseen tuli puuttua.

Nykyisessä ratkaisussa yhden trigonometrisen funktion arvon laskeminen edellyttää yhden DLL-kutsun (ks. luku 2.21) tekemistä. Perustuen tutkijan kokemukseen DLL-kutsujen hitaudesta viivettä lähdettiin pienentämään vähentämällä kutsujen määrää. Tämä toteutettiin laskemalla jo ohjelman käynnistyksen yhteydessä 360 arvoa jokaiselle käytetylle trigonometriselle funktiolle, yksi jokaista astetta kohden. Piirustuksiin liittyvää laskentaa suoritettaessa funktioiden arvot haetaan näiden arvojen joukosta valiten toivottua astelukua lähimmäksi osuva arvo.

Näin tehden havaittiin, että ohjelman käynnistäminen oli edelleen epämiellyttävän hidasta ja piirustuksen tarkkuus oli astelukujen pyöristämisen myötä huonontunut. Ratkaisuna näihin ongelmiin DLL-kutsujen määrää tuli vähentää entisestään ja laskentatarkkuutta lisätä.

6.3.1 DLL-kutsujen määrän vähentäminen

DLL-kutsujen määrää vähennettiin vetoamalla sinin ja kosinin tunnettuihin symmetriaominaisuuksiin: kutsumalla ainoastaan 90 ensimmäistä sinin arvoa DLL:n kautta saatiin loput sinin ja kosinin arvot laskettua niiden perusteella. Myös tangentin arvot laskettiin näin saatujen sinien ja kosinien perusteella. DLL-kutsujen määrää vähennettiin tällä tavalla yhteen kahdestoistaosaan[18].

Ajatusta jatkaen 90 sinin arvoa koettiin niin pieneksi määräksi lukuja, että se taulukoitiin osaksi ohjelmakoodia. Näin ollen DLL-kutsuja ei tarvita laisinkaan ja ohjelma käynnistyy nopeammin[19]. Ongelmaksi jäi näin toteutettujen funktioiden epätarkkuus, jota korjataan seuraavassa luvussa.

---

[18] $\frac{90}{360*3} = \frac{1}{12}$

[19] Tutkija tiedostaa, että kaikkien 1080 arvon vastaava taulukointi saattaa olla tehokkaampi toteutus.

6.3.2 Optimoitujen trigonometristen funktioiden laskentatarkkuuden lisääminen

Nykyisessä ratkaisussa trigonometriset funktioiden arvot ovat kvantisoituneita mistä seuraa piirustuksen näkyvää epätarkkuutta. Funktioiden tarkkuutta lisättiin lineaarisella interpoloinnilla, jolla virhemarginaali saatiin käytännön testeillä todennetusti riittävän pieneksi. Lineaarista interpolointia tarkempien interpolointimenetelmien käyttöä ei katsottu tarvittavan toteutuksen raskauden, toiminnan mahdollisen hidastumisen sekä häviävän pienen hyödyn vuoksi.

Oleelliset lähdekoodin proseduurit

- TrigonInit – lähdekoodin (viite 6) sivu 24
- Sin – lähdekoodin (viite 6) sivu 23
- Cos – lähdekoodin (viite 6) sivu 23
- Tan – lähdekoodin (viite 6) sivu 24

Oleelliset lähdekoodin funktiot

- Sini – lähdekoodin (viite 6) sivu 4
- Kosini – lähdekoodin (viite 6) sivu 4
- Tangentti – lähdekoodin (viite 6) sivu 2

6.4 Työvaiheen lopputulos

Kolmannen työvaiheen jälkeen valmiina on asiakkaalle esiteltävä demoversio mahdollisuuksista pyöreiden kulmien toteutuksen suhteen. Laaditut työkalut ovat joustavia ja mahdollistavat asiakkaan eri ehdotuksiin nopeasti reagoimisen.

Tyypillinen kolmannen työvaiheen jälkeinen piirustus on nähtävissä liitteessä 5, jossa huomioitavaa ovat pyöreät taivutukset kolmiulotteisissa kuvissa, sekä kaksiulotteisesta kuvasta ilmenevä lyhenemä.

7 YHTEENVETO JA JOHTOPÄÄTÖKSET

Tässä työssä suunniteltiin ja toteutettiin betoniraudoitusten piirustusten tulostusohjelma, jonka avulla kyseinen toiminto saatiin liitettyä osaksi Digia Enterprise -toiminnanohjausjärjestelmää. Asiakasyritys hyötyy tästä toteutuksesta raudoitusten myyntiprosessin tehostumisen muodossa.

Tavoitteet saavutettiin ja ohjelma on siitä todisteena asiakasyrityksen tuotantokäytössä. Sovellus kohtaa asiakkaan silloisella hetkellä sanelemat määritykset ja piirtää raudoitusten piirustukset taivutetuistakin verkoista CAD-ohjelmaa sopivammin. Myös jatkokehityksen kannalta tehty tutkimustyö on osoittautunut käyttökelpoiseksi ja lisäominaisuuksien käyttöönotto on sen ansiosta mielekästä.

Työssä suunnitellut työkalut soveltuvat räätälöiden muidenkin asioiden kuin raudoitusverkkojen kuvaamiseen. Tulevaisuudessa lisäominaisuuksien, kuten kiinteiden tasojen lisääminen mahdollistaa entistä monimutkaisempien suunnittelutehtävien toteuttamisen osana toiminnanohjausjärjestelmää. Tutkimuksen käyttö- ja sovellusmahdollisuudet ovat pienellä räätälöinnillä metalliteollisuudessa ja miksei muussakin teollisuudessa lupaavat.

Sovellus on lisännyt toiminnanohjausjärjestelmään yksilöllisen ominaisuuden, joka tarjoaa potentiaalisen kilpailuedun tarkkaan määritetyn metalliteollisuuden harjoittajille. Kilpailuedun se tarjoaa myös toiminnanohjausjärjestelmän kehittäjälle, eli työnantajalleni, joka nyt pystyy uuden räätälöitävän ominaisuuden avulla markkinoimaan tuotettansa eräille teollisuusasiakkaille entistä tehokkaammin. Tätä väitettä tukee seuraava ote:

> Toiminnanohjausjärjestelmät eivät ole perinteisesti pitäneet sisällään ominaisuuspohjaista konfiguraattoria ja sen valintojen mukaista lopputuloksen visualisointia.
>
> Työssä visualisointi on onnistuttu saumattomasti ja helppokäyttöisesti liittämään osaksi myyjän tavanomaista työvälineistöä.
>
> Toiminnanohjaukseen tiukasti paketoidulla visualisoinnilla on huomattava kaupallinen merkitys tarjottaessa toiminnanohjausjärjestelmiä toimialoille, joissa tuote kuvataan tapauskohtaisesti syöttämällä mittoja ja materiaalitietoja [7].

Tutkimuksen merkityksellisyyttä kuvaa seuraava sitaatti:

> Tällaiset toiminnallisuudet ovat se, mikä saa asiakkaan pysymään ikuisesti [tietyn] toiminnanohjausjärjestelmän käyttäjänä [8].

VIITELUETTELO

[1] Digia, Digia Enterprise [verkkodokumentti, viitattu 6.4.2008]. Saatavissa: http://www.digia.com/C2256FEF0043E9C1/0/405001395.

[2] Wikipedia, Raudoitus [verkkodokumentti, viitattu 6.4.2008]. Saatavissa: http://fi.wikipedia.org/wiki/Raudoitus.

[3] Wikipedia, OpenEdge [verkkodokumentti, viitattu 6.4.2008]. Saatavissa: http://en.wikipedia.org/wiki/OpenEdge_Advanced_Business_Language.

[4] Wikipedia, PDF [verkkodokumentti, viitattu 6.4.2008]. Saatavissa: http://fi.wikipedia.org/wiki/Pdf.

[5] Pro-sys consultants, PDFinclude [verkkodokumentti, viitattu 6.4.2008]. Saatavissa: http://www.epro-sys.com/pdfinclude.htm.

[6] Wikipedia, DLL [verkkodokumentti, viitattu 6.4.2008]. Saatavissa: http://fi.wikipedia.org/wiki/DLL.

[7] Vanhemman konsultin Mikko Sorjosen haastattelu. 10.4.2008. Digia Oyj.

[8] Projektipäällikkö Anu Pitkäsen haastattelu. 21.11.2007. SysOpenDigia Oyj, nyk. Digia Oyj.

AAKKOSELLINEN LUETTELO OHJELMAN SISÄISISTÄ PROSEDUUREISTA

Tämä liite pitää sisällään luettelon ohjelman sisäisistä proseduureista sekä viitteistä lähdekoodin (liite 6) asiaa käsittelevälle sivulle. Proseduurien kuvauksissa on poikettu asiatekstimäisestä kieliasusta, jotta luettelo pysyisi kompaktina.

PROCEDURE cos EXTERNAL "MSVCRT40.DLL" CDECL: (sivu 23)

Kutsuu Windowsin DLL:ää saaden tarkan kosinifunktion arvon.

PROCEDURE Draw3DLines: (sivu 20)

Piirtää 2-ulotteiseen tasoon muutetun 3-ulotteisen kuvan. Ero normaaliin DrawLinesiin on suoraan katsojasta poispäin menevien janojen erityisessä käsittelyssä.

PROCEDURE DrawHeader: (sivu 13)

Piirustuksen otsikkotietojen piirtäminen.

PROCEDURE DrawInfoBox2: (sivu 12)

Piirtää infolaatikon.

PROCEDURE DrawLines: (sivu 10)

2-ulotteisen piirustuksen piirtäminen.

PROCEDURE GetBendEffects: (sivu 21)

Proseduuri, jossa huomioidaan pyöreiden taivutusten vaikutukset työvaiheen 3 kuvaamissa paikoissa.

PROCEDURE GetBent: (sivu 7)

Taivutusmittojen haku ERP-järjestelmästä.

PROCEDURE GetDimensions: (sivu 9)

Työn alla olevan, eli wLineListissä olevan kuvan koon määritys.

PROCEDURE GetInfo: (sivu 13)

Hae infolaatikon tarvitsemat tiedot temp-table:en.

PROCEDURE GetLineList: (sivu 17)

Jalostaa terässarjojen mitat janalistaksi.

PROCEDURE GetLineList3DBetterA: (sivu 17)

Pilkkoo A-suuntaiset janat taivutuksia vastaaviin osiin ja taivuttaa ne.

PROCEDURE GetLineList3DBetterB: (sivu 17)

Pilkkoo B-suuntaiset janat taivutuksia vastaaviin osiin ja taivuttaa ne.

PROCEDURE GetMeasures: (sivu 6)

Hakee ERP-järjestelmästä litteän raudoituksen mitat.

PROCEDURE GetScaleDivider: (sivu 10)

Laskee jakajan jota käyttämällä työn alla oleva kuva skaalataan sivulle sopivaksi.

PROCEDURE HoriDeltaText: (sivu 11)

Kirjoittaa mittatiedot terästen vaakasuuntaisille etäisyyksille toisistaan.

PROCEDURE InitPdfFile: (sivu 9)

Alustaa PDF-tiedoston kirjoituksten.

PROCEDURE OpenIt: (sivu 15)

Avaa PDF-tiedoston käyttäjän oletusohjelmalla.

PROCEDURE OuterInit: (sivu 5)

Huolehtii tiedoston kirjoittamisen aloitusta edeltävistä tarkistuksista.

PROCEDURE SaveOle: (sivu 21)

Littää tehdyn PDF-tiedoston myyntitilausrivin OLE-tiedostoksi.

PROCEDURE ShellExecuteA EXTERNAL "shell32": (sivu 15)

Kutsuu Windowsin DLL:ää PDF-tiedoston avaamista varten.

PROCEDURE sin EXTERNAL "MSVCRT40.DLL" CDECL: (sivu 23)

Palauttaa sinifunktion tarkan arvon.

PROCEDURE tan EXTERNAL "MSVCRT40.DLL" CDECL: (sivu 24)

Palauttaa tangenttifunktion tarkan arvon.

PROCEDURE TextTopic: (sivu 14)

Vastaa otsikkotietojen aiheen kirjoittamisesta muotoiluineen.

PROCEDURE TextValue: (sivu 14)

Vastaa otsikkotietojen arvon kirjoittamisesta muotoiluineen.

PROCEDURE ThreeD2TwoDPlane: (sivu 19)

Muuttaa kolmiulotteiset koordinaatit sivulle sopivaan kaksiulotteiseen tasoon. Lintuperspektiivikuva 1.

PROCEDURE ThreeD2TwoDPlane2: (sivu 19)

Muuttaa kolmiulotteiset koordinaatit sivulle sopivaan kaksiulotteiseen tasoon. Lintuperspektiivikuva 2.

PROCEDURE ThreeD2TwoDPlane3: (sivu 19)

Muuttaa kolmiulotteiset koordinaatit sivulle sopivaan kaksiulotteiseen tasoon. Poikkileikkauskuva 1.

PROCEDURE ThreeD2TwoDPlane4: (sivu 19)

Muuttaa kolmiulotteiset koordinaatit sivulle sopivaan kaksiulotteiseen tasoon. Poikkileikkauskuva 2.

PROCEDURE trigonInit: (sivu 24)

Laske ennalta muistiin sini-, kosini- ja tangenttifunktioden arvoja.

PROCEDURE VertDeltaText: (sivu 11)

Kirjoittaa mittatiedot terästen pystysuuntaisille etäisyyksille toisistaan.

AAKKOSELLINEN LUETTELO OHJELMAN SISÄISISTÄ FUNKTIOISTA

Tämä liite pitää sisällään luettelon ohjelman sisäisistä funktioista sekä viitteistä lähdekoodin (liite 6) asiaa käsittelevälle sivulle. Funktioiden kuvauksissa on poikettu asiatekstimäisestä kieliasusta, jotta luettelo pysyisi kompaktina.

FUNCTION angleDelta RETURNS DECIMAL (INPUT iAngle1 AS DECIMAL, iAngle2 AS DECIMAL): (sivu 5)

Palauttaa kahden kulman väliin jäävän asteluvun.

FUNCTION arcLen RETURNS DECIMAL (INPUT iAngle AS DECIMAL, iRadius AS DECIMAL): (sivu 5)

Palauttaa sektorin kaaren pituuden.

FUNCTION betterMod RETURNS DECIMAL (INPUT iModdee AS DECIMAL, iModder AS DECIMAL): (sivu 4)

Palauttaa jakojäännöksen, joka ei pyöristy kokonaislukuihin.

FUNCTION chordLen RETURNS DECIMAL (INPUT iAngle AS DECIMAL, iRadius AS DECIMAL): (sivu 5)

Palauttaa ympyrän jänteen pituuden.

FUNCTION isClockwiceTurn RETURNS LOGICAL (INPUT iAngle1 AS DECIMAL, iAngle2 AS DECIMAL): (sivu 5)

Kertoo onko kahden asteluvun välinen kulma lyhyempi myötäpäivään mentäessä.

FUNCTION kosini RETURNS DECIMAL (INPUT iangle AS DECIMAL): (sivu 4)

Palauttaa kosinifunktion likiarvon.

FUNCTION kosini2 RETURNS DECIMAL (INPUT angle AS DECIMAL): (sivu 4)

Palauttaa kosinifunktion tarkan arvon. Tätä käytettiin ainoastaan virhemarginaalien mittaamiseen.

FUNCTION shrinkage RETURNS DECIMAL (INPUT iAngle AS DECIMAL, iRadius AS DECIMAL): (sivu 5)

Palauttaa iAngle-asteisen, iRadius-taivutussäteisen kulman lyhenemän terävään kulmaan verrattuna.

FUNCTION sini RETURNS DECIMAL (INPUT iangle AS DECIMAL): (sivu 4)

Palauttaa sinifunktion likiarvon.

FUNCTION tangentti RETURNS DECIMAL (INPUT angle AS DECIMAL): (sivu 2)

Palauttaa tangenttifunktion likiarvon.

FUNCTION toDegree RETURNS DECIMAL (INPUT iPut AS DECIMAL): (sivu 4)

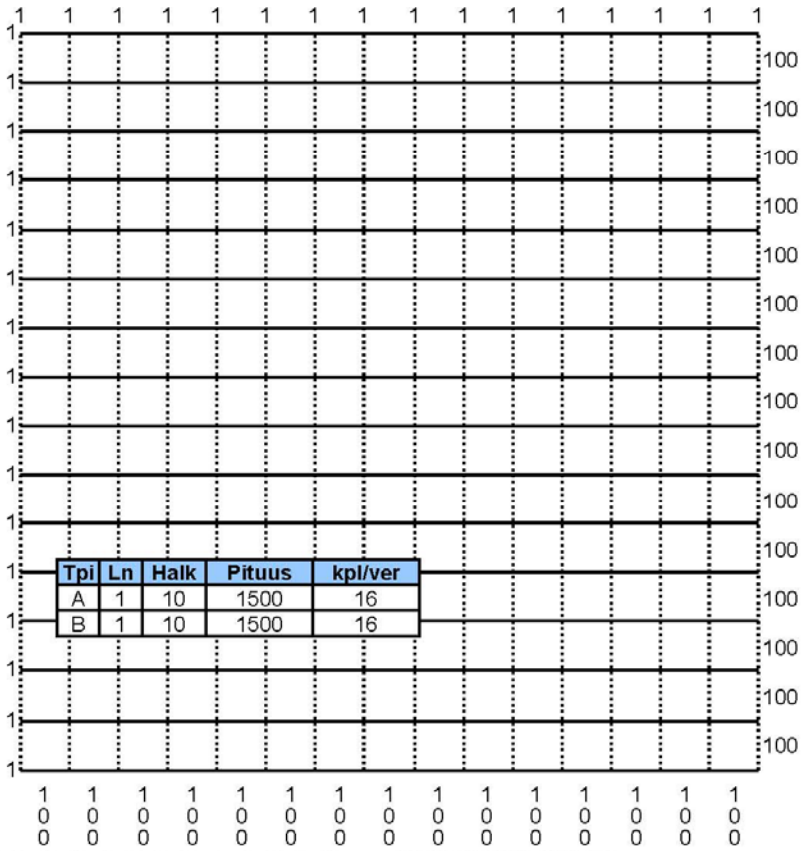
Muuttaa radiaanit asteiksi.

FUNCTION toRadian RETURNS DECIMAL (INPUT iPut AS DECIMAL): (sivu 4)

Muuttaa asteet radiaaneiksi.

PIIRUSTUKSEN ULKOASU TYÖVAIHEEN 1 JÄLKEEN

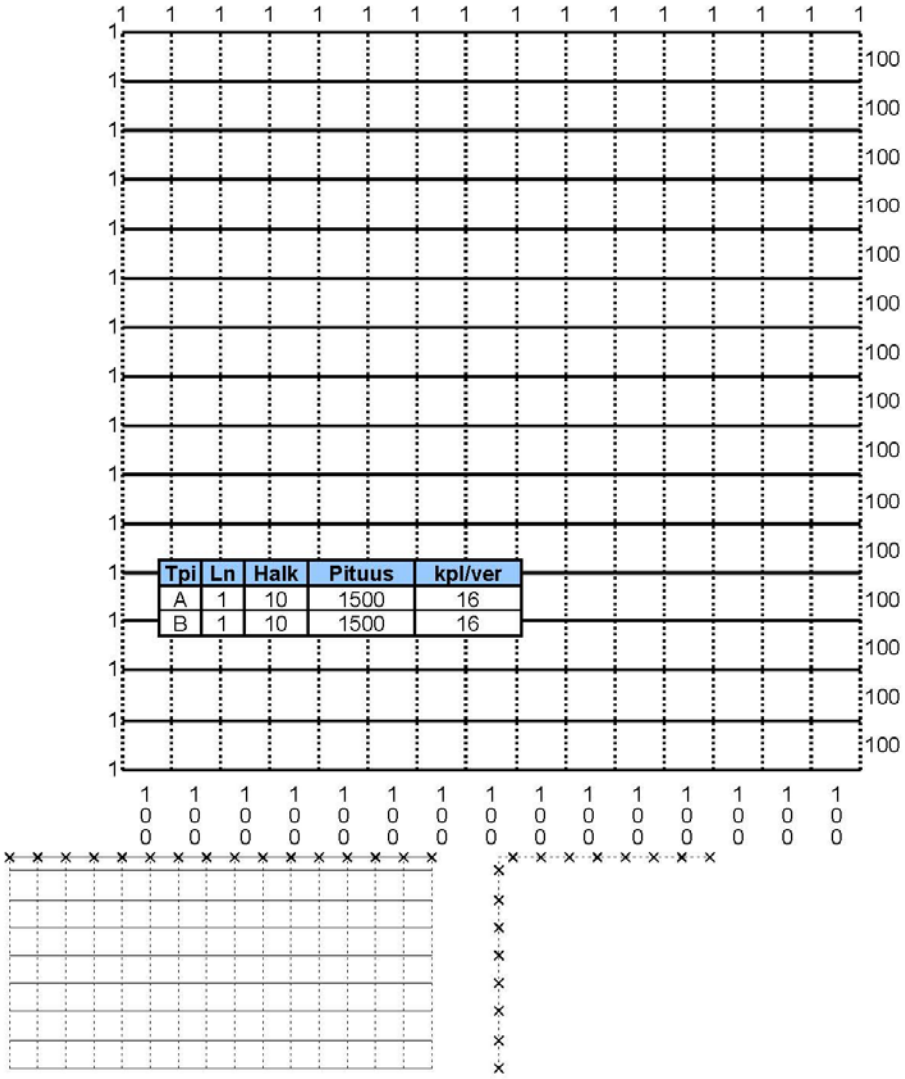
Tilaus: 1394 / 1 **Kpl Määrä:** 1
Tunnus: Armeringsnät 500;OikaisuA1;FL20
Verkon merkki: VA 125 **Käsitteli:** root 18/10/07
Asiakas: YIT-RAKENNUS OY, 33101 TAMPERE
Toimitusaika: 11/02/08 **Toimitustapa:** Autorahti
Info: 1 st á 29,59 kg



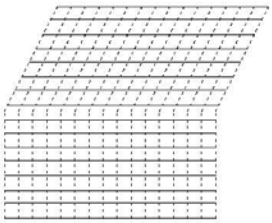
Tpi	Ln	Halk	Pituus	kpl/ver
A	1	10	1500	16
B	1	10	1500	16

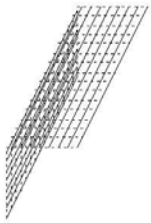
PIIRUSTUKSEN ULKOASU TYÖVAIHEEN 2 JÄLKEEN

Tilaus: 1394 / 1 **Kpl Määrä:** 1
Tunnus: Armeringsnät 500;OikaisuA1;FL20
Verkon merkki: VA 125 **Käsitteli:** root 18/10/07
Asiakas: YIT-RAKENNUS OY, 33101 TAMPERE
Toimitusaika: 11/02/08 **Toimitustapa:** Autorahti
Info: 1 st á 29,59 kg



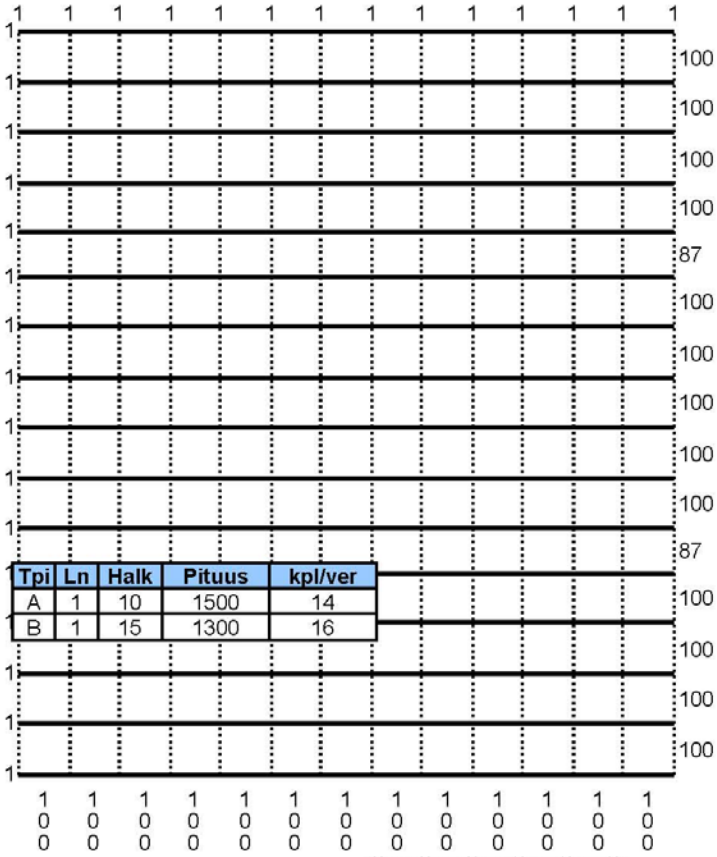
Tpi	Ln	Halk	Pituus	kpl/ver
A	1	10	1500	16
B	1	10	1500	16



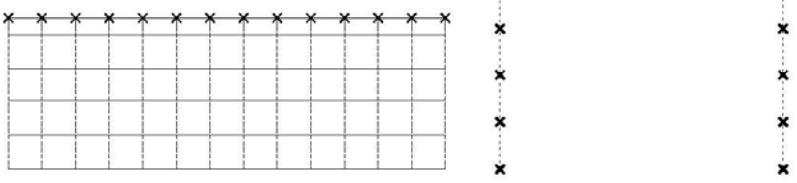


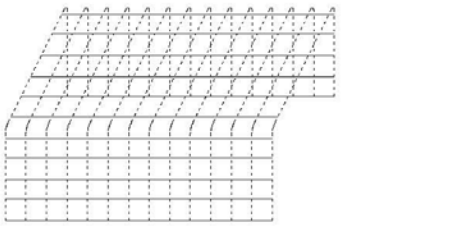
PIIRUSTUKSEN ULKOASU TYÖVAIHEEN 3 JÄLKEEN

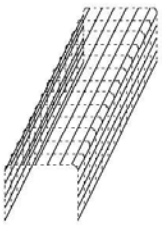
Tilaus: 1394 / 3 **Kpl Määrä:** 2
Tunnus: Armeringsnät 500;OikaisuA1;FL20
Verkon merkki: ____ **Käsitteli:** root 18/10/07
Asiakas: YIT-RAKENNUS OY, 33101 TAMPERE
Toimitusaika: 11/02/08 **Toimitustapa:** Autorahti
Info: 1 st á 29,59 kg



Tpi	Ln	Halk	Pituus	kpl/ver
A	1	10	1500	14
B	1	15	1300	16







LÄHDEKOODI

```
{pgkernel/DEFINE.i}
{general/qtyfunctions.i}
{general/featval.i}
{pgrepo/pdf_inc.i}
{pgsid/client.i}

DEFINE INPUT PARAMETER iTableName AS CHAR   NO-UNDO.
DEFINE INPUT PARAMETER iROWID     AS ROWID  NO-UNDO.
DEFINE INPUT PARAMETER iParam     AS CHAR   NO-UNDO.

DEFINE TEMP-TABLE wMeasures NO-UNDO
    FIELD AorB AS CHARACTER
    FIELD Order AS INTEGER
    FIELD Lkm   AS INTEGER
    FIELD Jako  AS INTEGER
    FIELD Posit AS DECIMAL DECIMALS 10
    FIELD Halk   AS DECIMAL
    FIELD OldHalk AS DECIMAL
    FIELD Pituus AS DECIMAL
    FIELD Ylit1 AS DECIMAL
    FIELD Ylit2 AS DECIMAL
    FIELD Kulma AS DECIMAL
    FIELD CumulPituus1 AS DECIMAL
    FIELD CumulPituus2 AS DECIMAL
    .
DEFINE TEMP-TABLE wMeasuresBend LIKE wMeasures.
DEFINE BUFFER bwMeasuresBend FOR wMeasuresBend.
DEFINE BUFFER bbwMeasuresBend FOR wMeasuresBend.
DEFINE TEMP-TABLE wMeasuresBendTemp LIKE wMeasures.
DEFINE TEMP-TABLE wMeasuresBendRound LIKE wMeasures.

DEFINE TEMP-TABLE wInfo NO-UNDO
    FIELD AorB AS CHARACTER
    FIELD Order AS CHARACTER
    FIELD Halk   AS CHARACTER
    FIELD Pituus AS CHARACTER
    FIELD Lkm   AS CHARACTER.

DEFINE TEMP-TABLE wLineList NO-UNDO
    FIELD AorB          AS CHARACTER
    FIELD FatherOrder   AS INTEGER
    FIELD FatherHalk    AS DECIMAL
    FIELD Halk          AS DECIMAL
    FIELD Order         AS INTEGER
```

```
    FIELD xCoord1       AS DECIMAL
    FIELD yCoord1       AS DECIMAL
    FIELD xCoord2       AS DECIMAL
    FIELD yCoord2       AS DECIMAL.
DEFINE BUFFER bwLineList FOR wLineList.

DEFINE TEMP-TABLE wLineList3D NO-UNDO
    FIELD AorB          AS CHARACTER
    FIELD FatherOrder   AS INTEGER
    FIELD FatherHalk    AS DECIMAL
    FIELD Halk          AS DECIMAL
    FIELD Order         AS INTEGER
    FIELD xCoord1       AS DECIMAL
    FIELD yCoord1       AS DECIMAL
    FIELD zCoord1       AS DECIMAL
    FIELD xCoord2       AS DECIMAL
    FIELD yCoord2       AS DECIMAL
    FIELD zCoord2       AS DECIMAL
    FIELD bendIndex     AS INTEGER
    FIELD bendMagni1    AS DECIMAL
    FIELD bendMagni2    AS DECIMAL.
DEFINE BUFFER bwLineList3D FOR wLineList3D.

DEFINE TEMP-TABLE wLineList3Db LIKE wLineList3D.

DEFINE BUFFER bSlsLine FOR SlsLine.

DEFINE VARIABLE xInfoRows     AS INTEGER     NO-UNDO.
DEFINE VARIABLE xWidth        AS DECIMAL     NO-UNDO.
DEFINE VARIABLE xHeight       AS DECIMAL     NO-UNDO.
DEFINE VARIABLE xScaleDivider AS DECIMAL     NO-UNDO.
DEFINE VARIABLE hHandle AS HANDLE      NO-UNDO.
DEFINE VARIABLE xFileName AS CHARACTER   NO-UNDO.
DEFINE VARIABLE xOk AS LOGICAL     NO-UNDO.
DEFINE VARIABLE xPdfStream AS CHARACTER   NO-UNDO INIT "sysStream".
DEFINE VARIABLE pi AS DECIMAL NO-UNDO INIT 3.14159265358979323846264338327 9.

DEFINE VARIABLE xSin AS DECIMAL    NO-UNDO EXTENT {&AngleResolution}.
DEFINE VARIABLE xCos AS DECIMAL    NO-UNDO EXTENT {&AngleResolution}.
DEFINE VARIABLE xTan AS DECIMAL    NO-UNDO EXTENT {&AngleResolution}.

hHandle = TEMP-TABLE wInfo:HANDLE.

RUN OuterInit(OUTPUT xOk).
IF NOT xOk THEN RETURN.
RUN pgsubr/waiton.p.
RUN GetMeasures("A").
```

```
RUN GetMeasures("B").
RUN GetInfo.
RUN GetLineList("A").
RUN GetLineList("B").
RUN GetDimensions.
RUN GetBent("A").
RUN GetBent("B").

RUN InitPdfFile(xFileName).
DO ON ERROR UNDO, LEAVE:
    RUN TrigonInit.
    RUN GetBendEffects.
    RUN GetDimensions. /* Added because of chances made in getBendEffects */
    RUN DrawHeader.
    RUN GetScaleDivider(6 / 8, 28 / 64).
    RUN DrawLines(0, 7 / 64).
    RUN DrawInfoBox2(hHandle).

    RUN GetLineList3DBetterA.
    RUN GetLineList3DBetterB.

    RUN ThreeD2TwoDPlane.
    RUN GetDimensions.
    RUN GetScaleDivider(1 / 3, 1 / 8).
    RUN Draw3DLines(0, 3 / 64).

    RUN ThreeD2TwoDPlane2.
    RUN GetDimensions.
    RUN GetScaleDivider(1 / 3, 1 / 8).
    RUN Draw3DLines(3 / 8, 3 / 64).

    RUN ThreeD2TwoDPlane3.
    RUN GetDimensions.
    RUN GetScaleDivider(1 / 3, 1 / 8).
    RUN Draw3DLines(0, 1 / 6 + 3 / 64).

    RUN ThreeD2TwoDPlane4.
    RUN GetDimensions.
    RUN GetScaleDivider(1 / 3, 1 / 8).
    RUN Draw3DLines(3 / 8, 1 / 6 + 3 / 64).

    RUN pdf_close(xPdfStream).
    RUN OpenIt(xFileName).

    RUN SaveOle.
    RUN pgsubr/waitoff.p.

```

```
    RETURN.
END.
RUN pgsubr/waitoff.p.
RUN pdf_close(xPdfStream).

FUNCTION kosini2 RETURNS DECIMAL (INPUT angle AS DECIMAL):
    DEFINE VARIABLE RESULT AS DECIMAL     NO-UNDO.
    RUN cos(INPUT angle, OUTPUT RESULT).
    RETURN RESULT.
END FUNCTION.
FUNCTION kosini RETURNS DECIMAL (INPUT angle AS DECIMAL) FORWARD.

FUNCTION betterMod RETURNS DECIMAL (INPUT iModdee AS DECIMAL, iModder AS DECIMAL):
    DEFINE VARIABLE xFidelity AS DECIMAL    NO-UNDO INIT 1000.
    RETURN ((iModdee * xFidelity) MOD (iModder * xFidelity)) / xFidelity.
END FUNCTION.

FUNCTION toDegree RETURNS DECIMAL (INPUT iPut AS DECIMAL):
    RETURN iPut * 180.0 / pi .
END FUNCTION.

FUNCTION toRadian RETURNS DECIMAL (INPUT iPut AS DECIMAL):
    RETURN iPut / 180.0 * pi .
END FUNCTION.

FUNCTION sini RETURNS DECIMAL (INPUT iangle AS DECIMAL):
    DEFINE VARIABLE xHelper AS DECIMAL    NO-UNDO.

    xHelper = betterMod(toDegree(iangle), 360.0).
    RETURN xSin[INT(1 + TRUNCATE(xHelper, 0))] + (xHelper - TRUNCATE(xHelper, 0)) * (xSin[INT(1 + 1 + TRUNCATE(xHelper, 0))] - xSin[INT(1 +
TRUNCATE(xHelper, 0))]).
END FUNCTION.

FUNCTION kosini RETURNS DECIMAL (INPUT iangle AS DECIMAL):
    DEFINE VARIABLE xHelper AS DECIMAL    NO-UNDO.

    xHelper = betterMod(toDegree(iangle), 360.0).
    RETURN xCos[INT(1 + TRUNCATE(xHelper, 0))] + (xHelper - TRUNCATE(xHelper, 0)) * (xCos[INT(1 + 1 + TRUNCATE(xHelper, 0))] - xCos[INT(1 +
TRUNCATE(xHelper, 0))]).
END FUNCTION.

FUNCTION tangentti RETURNS DECIMAL (INPUT iangle AS DECIMAL):
    DEFINE VARIABLE xHelper AS DECIMAL    NO-UNDO.

    xHelper = betterMod(toDegree(iangle), 360.0).
    RETURN xTan[INT(1 + TRUNCATE(xHelper, 0))] + (xHelper - TRUNCATE(xHelper, 0)) * (xTan[INT(1 + 1 + TRUNCATE(xHelper, 0))] - xTan[INT(1 +
TRUNCATE(xHelper, 0))]).
```

```
END FUNCTION.

FUNCTION angleDelta RETURNS DECIMAL (INPUT iAngle1 AS DECIMAL, iAngle2 AS DECIMAL):
    DEFINE VARIABLE xHelper AS DECIMAL     NO-UNDO.
    xHelper = betterMod(ABS(iAngle1 - iAngle2), 180.0).
    IF iAngle1 <> iAngle2 AND betterMod(ABS(iAngle1 - iAngle2), 180.0) = 0 THEN xHelper = 180.0.
    RETURN xHelper.
END FUNCTION.

FUNCTION shrinkage RETURNS DECIMAL (INPUT iAngle AS DECIMAL, iRadius AS DECIMAL):
    DEFINE VARIABLE xhelper AS DECIMAL     NO-UNDO.
    iAngle = iAngle * pi / 180.0.
    IF iAngle = pi THEN
        MESSAGE "Virhe! 180 asteen kulma."
            VIEW-AS ALERT-BOX INFO BUTTONS OK.
    xHelper = 2 * iRadius * tangentti(iAngle / 2) - 2 * pi * iRadius * (iAngle /(2 * pi)).
    RETURN xHelper.
END FUNCTION.

FUNCTION arcLen RETURNS DECIMAL (INPUT iAngle AS DECIMAL, iRadius AS DECIMAL):
    DEFINE VARIABLE xhelper AS DECIMAL     NO-UNDO.
    iAngle = iAngle * pi / 180.0.
    xHelper = iAngle * iRadius.
    RETURN xHelper.
END FUNCTION.

FUNCTION chordLen RETURNS DECIMAL (INPUT iAngle AS DECIMAL, iRadius AS DECIMAL):
    DEFINE VARIABLE xhelper AS DECIMAL     NO-UNDO.
    iAngle = iAngle * pi / 180.0.
    xHelper = iRadius * 2.0 * sini(iAngle / 2.0).
    RETURN xHelper.
END FUNCTION.

FUNCTION isClockwiceTurn RETURNS LOGICAL (INPUT iAngle1 AS DECIMAL, iAngle2 AS DECIMAL):
    DEFINE VARIABLE xHelper AS DECIMAL     NO-UNDO.
    iAngle1 = betterMod(iAngle1, 360.0).
    iAngle2 = betterMod(iAngle2, 360.0).
    RETURN iAngle2 = (iAngle1 + angleDelta(iAngle1, iAngle2)).
END FUNCTION.


PROCEDURE OuterInit:
    DEFINE VARIABLE xLineNumHelper AS INTEGER     NO-UNDO.
    DEFINE VARIABLE xOrderNumHelper AS INTEGER     NO-UNDO.
    DEFINE OUTPUT PARAMETER xOk AS LOGICAL     NO-UNDO INIT TRUE.
    xOk = FALSE.
    FIND FIRST SlsLine WHERE ROWID(SlsLine) = iRowid NO-LOCK NO-ERROR.
```

```
    xLineNumHelper = SlsLine.FatherLineNum.
    xOrderNumHelper = SlsLine.OrderNum.
    IF AVAILABLE SlsLine THEN IF SlsLine.FatherLineNum <> 0 THEN FIND FIRST SlsLine NO-LOCK WHERE SlsLine.LineNum = xLineNumHelper AND
SlsLine.OrderNum = xOrderNumHelper NO-ERROR.
    IF NOT AVAILABLE SlsLine THEN DO:
        RUN pgsubr/error.p("<<POWERED%prog/slsline/crewireframe.p%nolineavail>>").
        RETURN.
    END.
    RUN general/olefile.p("wf_" + STRING(SlsLine.Ordernum) + "_" + STRING(SlsLine.LineNum) + "_" + SlsLine.ItemCode + ".pdf", TRUE, OUTPUT
xFileName).
    IF SEARCH(xFileName) <> ? THEN DO:
        OS-DELETE VALUE(xFileName).
        IF OS-ERROR <> 0 THEN DO:
            RUN pgsubr/error.p("<<POWERED%prog/slsline/crewireframe.p%fileinuse>>").
            RETURN.
        END.
    END.

    IF(GetHierarcyAttribute("LkmA1", "SlsLine", SlsLine.SlsLineKey) = ?) AND (GetHierarcyAttribute("LkmB1", "SlsLine", SlsLine.SlsLineKey) = ?)
THEN DO:
        RUN pgsubr/error.p("<<POWERED%prog/slsline/crewireframe.p%notawirethingy>>").
        RETURN.
    END.

    xOk = TRUE.
END PROCEDURE.

PROCEDURE GetMeasures:
    DEFINE INPUT  PARAMETER iAorB AS CHARACTER   NO-UNDO.
    DEFINE VARIABLE xi AS INTEGER NO-UNDO INIT 1.
    REPEAT WHILE GetHierarcyAttribute("Lkm" + iAorB + STRING(xi) ,"SlsLine",SlsLine.SlsLineKey) <> ?. /* Lkm määritelty vain mittoja kuvaavilla
paramatreilla. */
        CREATE wMeasures.
        ASSIGN
            wMeasures.AorB = iAorB
            wMeasures.Order = xi
            wMeasures.Lkm =    GetHierarcyAttribute("Lkm"    + iAorB + STRING(xi) ,"SlsLine",SlsLine.SlsLineKey)
            wMeasures.Jako  =  GetHierarcyAttribute("Jako"   + iAorB + STRING(xi) ,"SlsLine",SlsLine.SlsLineKey)
            wMeasures.Posit =  GetHierarcyAttribute("Posit"  + iAorB + STRING(xi) ,"SlsLine",SlsLine.SlsLineKey)
            wMeasures.Halk =   GetHierarcyAttribute("Halk"   + iAorB + STRING(xi) ,"SlsLine",SlsLine.SlsLineKey)
            wMeasures.Pituus = GetHierarcyAttribute("Pituus" + iAorB + STRING(xi) ,"SlsLine",SlsLine.SlsLineKey)
            wMeasures.Ylit1 =  GetHierarcyAttribute("Ylit1"  + iAorB + STRING(xi) ,"SlsLine",SlsLine.SlsLineKey)
            wMeasures.Ylit2 =  GetHierarcyAttribute("Ylit2"  + iAorB + STRING(xi) ,"SlsLine",SlsLine.SlsLineKey)
            wMeasures.Kulma =  GetHierarcyAttribute("Kulma"  + iAorB + STRING(xi) ,"SlsLine",SlsLine.SlsLineKey).

            wMeasures.Lkm = IF wMeasures.Lkm = ? THEN 0 ELSE wMeasures.Lkm.
            wMeasures.Jako = IF wMeasures.Jako = ? THEN 0 ELSE wMeasures.Jako.
```

```
            wMeasures.Posit = IF wMeasures.Posit = ? THEN 0 ELSE wMeasures.Posit.
            wMeasures.Halk = IF wMeasures.Halk = ? THEN 0 ELSE wMeasures.Halk.
            wMeasures.Pituus = IF wMeasures.Pituus = ? THEN 0 ELSE wMeasures.Pituus.

        xi = xi + 1.
    END.
END PROCEDURE.

PROCEDURE GetBent:
    DEFINE INPUT  PARAMETER iAorB AS CHARACTER   NO-UNDO.
    DEFINE VARIABLE Crawler AS DECIMAL     NO-UNDO.
    DEFINE VARIABLE CrawlerMax AS DECIMAL     NO-UNDO INIT 0.
    DEFINE VARIABLE xOldHalk AS DECIMAL     NO-UNDO.

    CREATE wMeasuresBendTemp.
    ASSIGN
        wMeasuresBendTemp.AorB = iAorB
        wMeasuresBendTemp.Lkm = 0
        wMeasuresBendTemp.Jako  = 0
        wMeasuresBendTemp.Posit = IF iAorB = "A" THEN xHeight ELSE xWidth
        wMeasuresBendTemp.CumulPituus1 = 0
        wMeasuresBendTemp.CumulPituus2 = IF iAorB = "A" THEN xHeight ELSE xWidth
        wMeasuresBendTemp.Halk = 0.
    FIND bSlsLine WHERE bSlsLine.FatherLineNum = SlsLine.LineNum AND bSlsLine.OrderNum = SlsLine.OrderNum NO-LOCK NO-ERROR.
    IF NOT AVAILABLE bSlsLine THEN RETURN.

    DEFINE VARIABLE xi AS INTEGER NO-UNDO INIT 1.
    REPEAT WHILE GetHierarcyAttribute("Posit" + iAorB + STRING(xi), "SlsLine", bSlsLine.SlsLineKey) <> ?. /* Posit määritelty vain taivutusta
kuvaavilla paramatreilla. */
        IF xi <> 1 THEN CREATE wMeasuresBendTemp.

        ASSIGN
            wMeasuresBendTemp.AorB = iAorB
            wMeasuresBendTemp.Order = xi
            wMeasuresBendTemp.Lkm =    GetHierarcyAttribute("Lkm"    + iAorB + STRING(xi), "SlsLine", bSlsLine.SlsLineKey)
            wMeasuresBendTemp.Jako  =  GetHierarcyAttribute("Jako"   + iAorB + STRING(xi), "SlsLine", bSlsLine.SlsLineKey)
            wMeasuresBendTemp.Posit =  GetHierarcyAttribute("Posit"  + iAorB + STRING(xi), "SlsLine", bSlsLine.SlsLineKey) /* Kertoo
taivutettavan jakson pituuden. */
            wMeasuresBendTemp.Halk =   GetHierarcyAttribute("Halk"   + iAorB + STRING(xi), "SlsLine", bSlsLine.SlsLineKey) /* Kertoo
taivutuskulman. */
            wMeasuresBendTemp.Pituus = GetHierarcyAttribute("Pituus" + iAorB + STRING(xi), "SlsLine", bSlsLine.SlsLineKey)
            wMeasuresBendTemp.Ylit1 =  GetHierarcyAttribute("Ylit1"  + iAorB + STRING(xi), "SlsLine", bSlsLine.SlsLineKey)
            wMeasuresBendTemp.Ylit2 =  GetHierarcyAttribute("Ylit2"  + iAorB + STRING(xi), "SlsLine", bSlsLine.SlsLineKey)
            wMeasuresBendTemp.Kulma =  GetHierarcyAttribute("Kulma"  + iAorB + STRING(xi), "SlsLine", bSlsLine.SlsLineKey). /* Not all are
needed, but I get them anyway. */

        wMeasuresBendTemp.Lkm = IF wMeasuresBendTemp.Lkm = ? THEN 0 ELSE wMeasuresBendTemp.Lkm.
```

```
        wMeasuresBendTemp.Jako = IF wMeasuresBendTemp.Jako = ? THEN 0 ELSE wMeasuresBendTemp.Jako.
        wMeasuresBendTemp.Posit = IF wMeasuresBendTemp.Posit = ? THEN 0 ELSE wMeasuresBendTemp.Posit.
        wMeasuresBendTemp.Halk = IF wMeasuresBendTemp.Halk = ? THEN 0 ELSE wMeasuresBendTemp.Halk.
        wMeasuresBendTemp.Pituus = IF wMeasuresBendTemp.Pituus = ? THEN 0 ELSE wMeasuresBendTemp.Pituus.

        IF xi <> 1 THEN wMeasuresBendTemp.OldHalk = xOldHalk.
        ELSE wMeasuresBendTemp.OldHalk = wMeasuresBendTemp.Halk.
        xOldHalk = wMeasuresBendTemp.Halk.

        wMeasuresBendTemp.CumulPituus1 = CrawlerMax.
        CrawlerMax = CrawlerMax + wMeasuresBendTemp.Posit.
        IF CrawlerMax <> 0 THEN wMeasuresBendTemp.CumulPituus2 = CrawlerMax. /* CrawlerMax = 0 if this is the only bend in A or B direction. In
this case CumulPituus2 should be left undisturbed */

        xi = xi + 1.
    END.

    FOR EACH wMeasuresBendTemp WHERE wMeasuresBendTemp.AorB = iAorB NO-LOCK:
        CREATE wMeasuresBend.
        BUFFER-COPY wMeasuresBendTemp TO WMeasuresBend.
    END.
END PROCEDURE.

PROCEDURE GetLineList:
    DEFINE INPUT  PARAMETER iAorB AS CHARACTER   NO-UNDO.
    DEFINE VARIABLE xd AS DECIMAL     NO-UNDO.
    DEFINE VARIABLE xi AS INTEGER     NO-UNDO.
    FOR EACH wMeasures NO-LOCK WHERE wMeasures.AorB = iAorB BY Order:
        REPEAT xi = 1 TO wMeasures.Lkm.
            CREATE wLineList.
            ASSIGN
                wLineList.AorB          = iAorB
                wLineList.FatherOrder   = wMeasures.Order
                wLineList.FatherHalk    = wMeasures.Halk
                wLineList.Order         = xi.
            IF iAorB = "A" THEN ASSIGN
                wLineList.xCoord1       = ((xi - 1) * wMeasures.Jako) + wMeasures.Posit
                wLineList.YCoord1       = 0 - wMeasures.Ylit1
                wLineList.xCoord2       = ((xi - 1) * wMeasures.Jako) + wMeasures.Posit
                wLineList.YCoord2       = wMeasures.Pituus - wMeasures.Ylit1.
            IF iAorB = "B" THEN ASSIGN
                wLineList.xCoord1       = 0 - wMeasures.Ylit1
                wLineList.YCoord1       = ((xi - 1) * wMeasures.Jako) + wMeasures.Posit
                wLineList.xCoord2       = wMeasures.Pituus - wMeasures.Ylit1
                wLineList.YCoord2       = ((xi - 1) * wMeasures.Jako) + wMeasures.Posit.
        END.
    END.
```

```
374     xd = 0.6.
375     FOR EACH wLineList NO-LOCK BREAK BY wLineList.FatherHalk:
376         IF FIRST-OF(wLineList.FatherHalk) THEN xd = xd + 0.6.
377         wLineList.Halk = xd.
378     END.
379 END PROCEDURE.
380 PROCEDURE GetDimensions: /* and handle the offset */
381     DEFINE VARIABLE xMinX AS DECIMAL     NO-UNDO INIT 999999999999999999.
382     DEFINE VARIABLE xMaxX AS DECIMAL     NO-UNDO INIT -99999999999999999.
383     DEFINE VARIABLE xMinY AS DECIMAL     NO-UNDO INIT 999999999999999999.
384     DEFINE VARIABLE xMaxY AS DECIMAL     NO-UNDO INIT -99999999999999999.
385     FOR EACH wLineList NO-LOCK:
386         xMaxX = MAXIMUM(xMaxX, wLineList.xCoord1, wLineList.xCoord2).
387         xMinX = MINIMUM(xMinX, wLineList.xCoord1, wLineList.xCoord2).
388         xMaxY = MAXIMUM(xMaxY, wLineList.yCoord1, wLineList.yCoord2).
389         xMinY = MINIMUM(xMinY, wLineList.yCoord1, wLineList.yCoord2).
390     END.
391
392     FOR EACH wLineList NO-LOCK:
393         IF xMinX < 0 THEN DO:
394             wLineList.xCoord1 = wLineList.xCoord1 - xMinX.
395             wLineList.xCoord2 = wLineList.xCoord2 - xMinX.
396         END.
397         IF xMinY < 0 THEN DO:
398             wLineList.yCoord1 = wLineList.yCoord1 - xMinY.
399             wLineList.yCoord2 = wLineList.yCoord2 - xMinY.
400         END.
401     END.
402
403     xWidth  = xMaxX - xMinX.
404     xHeight = xMaxY - xMinY.
405 END PROCEDURE.
406
407 PROCEDURE InitPdfFile:
408     DEFINE INPUT  PARAMETER iFilename AS CHARACTER   NO-UNDO.
409
410     RUN pdf_new(xPdfStream, iFileName).
411
412     RUN pdf_set_info(xPdfStream,"Author","Test").
413     RUN pdf_set_info(xPdfStream,"Subject","Test").
414     RUN pdf_set_info(xPdfStream,"Title","Test").
415     RUN pdf_set_info(xPdfStream,"Keywords","Test").
416     RUN pdf_set_info(xPdfStream,"Creator","SysOpenDigia").
417     RUN pdf_set_info(xPdfStream,"Producer","prog/slsline/crewireframe.p").
418
419     RUN pdf_new_page(xPdfStream).
420
```

```
    RUN pdf_set_LeftMargin(xPdfStream, 80).
    RUN pdf_set_TopMargin(xPdfStream, 80).
END PROCEDURE.

PROCEDURE GetScaleDivider:

    DEFINE INPUT  PARAMETER iWidthRatio AS DECIMAL     NO-UNDO.
    DEFINE INPUT  PARAMETER iHeightRatio AS DECIMAL     NO-UNDO.
    xScaleDivider = MAXIMUM(xWidth / (iWidthRatio * DECIMAL(pdf_PageWidth(xPdfStream))), xHeight / (iHeightRatio *
DECIMAL(pdf_PageHeight(xPdfStream)))).
END PROCEDURE.

PROCEDURE DrawLines:
    DEFINE VARIABLE i AS INTEGER     NO-UNDO.
    DEFINE INPUT  PARAMETER xOffset AS DECIMAL     NO-UNDO.
    DEFINE INPUT  PARAMETER yOffset AS DECIMAL     NO-UNDO.
    xOffset = xOffset * DECIMAL(pdf_PageWidth(xPdfStream)).
    yOffset = yOffset * DECIMAL(pdf_PageHeight(xPdfStream)).
    xOffset = xOffset + (pdf_PageWidth(xPdfStream) - pdf_LeftMargin(xPdfStream) * 2 - (xWidth / xScaleDivider)) / 2.

    RUN pdf_set_font(xPdfStream, "Helvetica", 10.0).
    FOR EACH wLinelist NO-LOCK:
        IF wLineList.AorB = "A" THEN RUN pdf_set_dash (xPdfStream, 1, 2).
        ELSE RUN pdf_set_dash (xPdfStream, 1, 0).
        RUN pdf_line(xPdfStream,
            pdf_LeftMargin(xPdfStream) + wLineList.xCoord1 / xScaleDivider + xOffset,
            (pdf_PageHeight(xPdfStream) - xHeight / xScaleDivider) / 2 + wLineList.yCoord1 / xScaleDivider + yOffset,
            pdf_LeftMargin(xPdfStream) + wLineList.xCoord2 / xScaleDivider + xOffset,
            (pdf_PageHeight(xPdfStream) - xHeight / xScaleDivider) / 2 + wLineList.yCoord2 / xScaleDivider + yOffset,
            wLineList.Halk).
        IF wLineList.AorB = "A" THEN
            RUN pdf_text_xy(xPdfStream,
            STRING(wLineList.FatherOrder),
            pdf_LeftMargin(xPdfStream) + wLineList.xCoord1 / xScaleDivider - 3 + xOffset,
            (pdf_PageHeight(xPdfStream) + xHeight / xScaleDivider) / 2 + 5 + yOffset).
        IF wLineList.AorB = "B" THEN
            RUN pdf_text_xy(xPdfStream,
            STRING(wLineList.FatherOrder),
            pdf_LeftMargin(xPdfStream) - 7 + xOffset,
            (pdf_PageHeight(xPdfStream) - xHeight / xScaleDivider) / 2 + wLineList.yCoord1 / xScaleDivider - 3 + yOffset).
    END.

    FOR EACH wLinelist NO-LOCK WHERE wLineList.AorB = "A" BREAK BY wLineList.xCoord1:
        IF FIRST(wLinelist.xCoord1) THEN FOR EACH bwLineList WHERE bwLinelist.xCoord1 < wLinelist.xCoord1 BREAK BY bwLinelist.xCoord1:
            IF FIRST-OF(bwLineList.xCoord1) THEN RUN HoriDeltaText(bwLineList.xCoord1, wLineList.xCoord1, xOffset, yOffset).
        END.
        FOR EACH bwLineList NO-LOCK WHERE bwLineList.xCoord1 > wLineList.xCoord1 AND bwLineList.AorB = "A" BY bwLineList.xCoord1:
```

```
            RUN HoriDeltaText(wLineList.xCoord1, bwLineList.xCoord1, xOffset, yOffset).
            LEAVE.
        END.
        IF LAST(wLinelist.xCoord1) THEN FOR EACH bwLineList WHERE bwLinelist.xCoord2 > wLinelist.xCoord1 BREAK BY bwLinelist.xCoord2:
            IF FIRST-OF(bwLineList.xCoord2) THEN RUN HoriDeltaText(wLineList.xCoord1, bwLineList.xCoord2, xOffset, yOffset).
        END.

    END.

    FOR EACH wLinelist NO-LOCK WHERE wLineList.AorB = "B" BREAK BY wLinelist.yCoord1:
        IF FIRST(wLinelist.yCoord1) THEN FOR EACH bwLineList WHERE bwLinelist.yCoord1 < wLinelist.yCoord1 BREAK BY bwLinelist.yCoord1:
            IF FIRST-OF(bwLineList.yCoord1) THEN RUN VertDeltaText(bwLineList.yCoord1, wLineList.yCoord1, xOffset, yOffset).
        END.
        FOR EACH bwLineList NO-LOCK WHERE bwLineList.yCoord1 > wLineList.yCoord1 AND wLineList.AorB = "B" BY bwLineList.yCoord1:
            RUN VertDeltaText(wLineList.yCoord1, bwLineList.yCoord1, xOffset, yOffset).
            LEAVE.
        END.
        IF LAST(wLinelist.yCoord1) THEN FOR EACH bwLineList WHERE bwLinelist.yCoord2 > wLinelist.yCoord1 BREAK BY bwLinelist.yCoord2:
            IF FIRST-OF(bwLineList.yCoord2) THEN RUN VertDeltaText(wLineList.yCoord1, bwLineList.yCoord2, xOffset, yOffset).
        END.
    END.

END PROCEDURE.

PROCEDURE HoriDeltaText:
    DEFINE INPUT  PARAMETER iLower AS DECIMAL     NO-UNDO.
    DEFINE INPUT  PARAMETER iUpper AS DECIMAL     NO-UNDO.
    DEFINE INPUT  PARAMETER ixOffset AS DECIMAL     NO-UNDO.
    DEFINE INPUT  PARAMETER iyOffset AS DECIMAL     NO-UNDO.
    DEFINE VARIABLE xText AS CHARACTER   NO-UNDO.
    DEFINE VARIABLE i AS INTEGER     NO-UNDO.

    xText = STRING(ROUND(iUpper - iLower, 0)).
    IF xText <> "0" THEN REPEAT i = 1 TO LENGTH(xText):
        RUN pdf_text_xy(
            xPdfStream, SUBSTRING(xText, i, 1),
            pdf_LeftMargin(xPdfStream) + (iLower + (iUpper - iLower) / 2) / xScaleDivider - pdf_text_width(xPdfStream, SUBSTRING(xText, i, 1)) /
2 + ixOffset,
            10 * - i + (pdf_PageHeight(xPdfStream) - xHeight / xScaleDivider) / 2 - 5 + iyOffset
            ).
    END.
END.

PROCEDURE VertDeltaText:
    DEFINE INPUT  PARAMETER iLeft AS DECIMAL     NO-UNDO.
    DEFINE INPUT  PARAMETER iRight AS DECIMAL     NO-UNDO.
    DEFINE INPUT  PARAMETER ixOffset AS DECIMAL     NO-UNDO.
```

```
    DEFINE INPUT  PARAMETER iyOffset AS DECIMAL     NO-UNDO.
    RUN pdf_text_xy(
                xPdfStream, STRING(ROUND(iRight - iLeft, 0)),
                pdf_LeftMargin(xPdfStream) + xWidth / xScaleDivider + 2 + ixOffset,
                (pdf_PageHeight(xPdfStream) - xHeight / xScaleDivider) / 2 + (iLeft + (iRight - iLeft) / 2) / xScaleDivider - 4 + iyOffset
                ).
END.

PROCEDURE DrawInfoBox2:
    DEFINE INPUT  PARAMETER ittHandle AS HANDLE      NO-UNDO.

    RUN pdf_set_font(xPdfStream,"Helvetica",10.0).

    RUN pdf_tool_add(xPdfStream,"Info","MATRIX", ?).

    RUN pdf_set_tool_parameter(xPdfStream,"Info","X",0,"150").
    RUN pdf_set_tool_parameter(xPdfStream,"Info","Y",0,STRING(pdf_PageHeight(xPdfStream) / 2)).
    RUN pdf_set_tool_parameter(xPdfStream,"Info","Rows",0,STRING(xInfoRows + 1)).
    RUN pdf_set_tool_parameter(xPdfStream,"Info","Columns",0,"5").
    RUN pdf_set_tool_parameter(xPdfStream,"Info","GridWeight",0,"0.1").
    RUN pdf_set_tool_parameter(xPdfStream,"Info","GridColor",0,"0,0,0").
    RUN pdf_set_tool_parameter(xPdfStream,"Info","FontSize",0,"1.0").
    RUN pdf_set_tool_parameter(xPdfStream,"Info","BGcolor",0,"255,255,255").

    /* Setup Row 1 -- the Header */
    RUN pdf_set_tool_parameter(xPdfStream,"Info","BGcolor",1,"151,201,255").
    RUN pdf_set_tool_parameter(xPdfStream,"Info","Font",1,"Helvetica-Bold").

    /* Setup Row 2 -- the Information */
    RUN pdf_set_tool_parameter(xPdfStream,"Info","Font",2,"Helvetica").

    RUN pdf_set_tool_parameter(xPdfStream,"Info","ColumnWidth",1,"20").
    RUN pdf_set_tool_parameter(xPdfStream,"Info","ColumnAlign",1,"CENTER").
    RUN pdf_set_tool_parameter(xPdfStream,"Info","ColumnWidth",2,"20").
    RUN pdf_set_tool_parameter(xPdfStream,"Info","ColumnAlign",2,"CENTER").
    RUN pdf_set_tool_parameter(xPdfStream,"Info","ColumnWidth",3,"30").
    RUN pdf_set_tool_parameter(xPdfStream,"Info","ColumnAlign",3,"CENTER").
    RUN pdf_set_tool_parameter(xPdfStream,"Info","ColumnWidth",4,"50").
    RUN pdf_set_tool_parameter(xPdfStream,"Info","ColumnAlign",4,"CENTER").
    RUN pdf_set_tool_parameter(xPdfStream,"Info","ColumnWidth",5,"50").
    RUN pdf_set_tool_parameter(xPdfStream,"Info","ColumnAlign",5,"CENTER").

    /* Load Row 1 - Header Data */
    RUN pdf_set_tool_parameter(xPdfStream,"Info","CellValue",1,"Tpi").
    RUN pdf_set_tool_parameter(xPdfStream,"Info","CellValue",2,"Ln").
    RUN pdf_set_tool_parameter(xPdfStream,"Info","CellValue",3,"Halk").
    RUN pdf_set_tool_parameter(xPdfStream,"Info","CellValue",4,"Pituus").
```

```
562     RUN pdf_set_tool_parameter(xPdfStream,"Info","CellValue",5,"kpl/ver").
563
564     /* Load Row 2 - Information Data */
565     DEFINE VARIABLE i AS INTEGER NO-UNDO.
566     i = 5.
567     FOR EACH wInfo NO-LOCK:
568         RUN pdf_set_tool_parameter(xPdfStream,"Info","CellValue", i + 1, STRING(wInfo.AorB)).
569         RUN pdf_set_tool_parameter(xPdfStream,"Info","CellValue", i + 2, STRING(wInfo.Order)).
570         RUN pdf_set_tool_parameter(xPdfStream,"Info","CellValue", i + 3, STRING(wInfo.Halk)).
571         RUN pdf_set_tool_parameter(xPdfStream,"Info","CellValue", i + 4, STRING(wInfo.Pituus)).
572         RUN pdf_set_tool_parameter(xPdfStream,"Info","CellValue", i + 5, STRING(wInfo.Lkm)).
573         i = i + 5.
574     END.
575
576     RUN pdf_tool_create(xPdfStream,"Info").
577 END PROCEDURE.
578
579 PROCEDURE GetInfo:
580     FOR EACH wMeasures NO-LOCK:
581         CREATE wInfo.
582         wInfo.AorB   = STRING(wMeasures.AorB).
583         wInfo.Order  = STRING(wMeasures.Order).
584         wInfo.Halk   = STRING(wMeasures.Halk).
585         wInfo.Pituus = STRING(wMeasures.Pituus).
586         wInfo.Lkm    = STRING(wMeasures.Lkm).
587         xInfoRows = xInfoRows + 1.
588     END.
589 END PROCEDURE.
590
591 PROCEDURE DrawHeader:
592     DEFINE VARIABLE xCharHelper AS CHARACTER   NO-UNDO.
593     DEFINE VARIABLE xi AS INTEGER     NO-UNDO.
594
595     FIND SlsOrder NO-LOCK WHERE SlsOrder.OrderNum = SlsLine.OrderNum.
596     FIND Customer NO-LOCK WHERE Customer.CustNum = SlsOrder.DlvCustNum.
597     FIND FIRST Address NO-LOCK WHERE Address.AddressNum = SlsOrder.DlvAddrNum NO-ERROR.
598     FIND SysText NO-LOCK WHERE SysText.TableName = "SlsLine" AND SysTextKey = SlsLineKey AND SysText.NoteType = "OrdConfRowE" NO-ERROR.
599
600     RUN pdf_set_TextY(xPdfStream, pdf_PageHeight(xPdfStream) - 40).
601
602     RUN TextTopic("Tilaus:", 100).
603     RUN TextValue(STRING(SlsOrder.OrderNum) + " / " + STRING(SlsLine.LineNum), 110).
604
605     RUN TextTopic("Kpl Määrä:", 320).
606     RUN TextValue(STRING(SlsLine.Qty), 330).
607
608     RUN pdf_skip(xPdfStream).
```

```
609
610     RUN TextTopic("Tunnus:", 100).
611     RUN TextValue(STRING(SlsLine.ItemName, "x(35)"), 110).
612
613     RUN pdf_skip(xPdfStream).
614
615     RUN TextTopic("Verkon merkki:", 100).
616     RUN TextValue(FeatValueChar("Merkki", "SlsLine", SlsLine.SlsLineKey), 110).
617     RUN TextTopic("Käsitteli:", 320).
618     RUN TextValue(STRING(SlsOrder.RespUser), 330).
619     RUN TextValue(STRING(SlsOrder.OrderDate), 500).
620
621     RUN pdf_skip (xPdfStream).
622
623     RUN TextTopic("Asiakas:", 100).
624     RUN TextValue(Address.CustName1 + ", " + Address.PostOffice, 110).
625
626     RUN pdf_skip(xPdfStream).
627
628     RUN TextTopic("Toimitusaika:", 100).
629     RUN TextValue(STRING(SlsLine.ReqDlvDate), 110).
630     RUN TextTopic("Toimitustapa:", 320).
631     RUN TextValue(STRING(SlsOrder.TraMode), 330).
632
633     RUN pdf_skip(xPdfStream).
634     RUN TextTopic("Info:", 100).
635
636     DEFINE VARIABLE xVoid AS INTEGER     NO-UNDO.
637     RUN pdf_set_font(xPdfStream, "Courier", 12.0).
638     RUN pdf_wrap_Text(xPdfStream,
639          SUBSTRING(SysText.InfoText, 1, 115) + IF LENGTH(SysText.InfoText) > 115 THEN "..." ELSE "",
640          5,
641          65,
642          "left",
643          OUTPUT xVoid).
644 END PROCEDURE.
645
646 PROCEDURE TextTopic:
647     DEFINE INPUT  PARAMETER iText AS CHARACTER   NO-UNDO.
648     DEFINE INPUT  PARAMETER ixCoord AS INTEGER     NO-UNDO.
649     RUN pdf_set_font(xPdfStream, "Courier-Bold", 10.0).
650     RUN pdf_text_align (xPdfStream, iText, "RIGHT", ixCoord, pdf_TextY(xPdfStream)).
651 END PROCEDURE.
652
653 PROCEDURE TextValue:
654     DEFINE INPUT  PARAMETER iText AS CHARACTER   NO-UNDO.
655     DEFINE INPUT  PARAMETER ixCoord AS INTEGER     NO-UNDO.
```

```
    RUN pdf_set_font(xPdfStream, "Courier", 12.0).
    RUN pdf_text_align (xPdfStream, iText, "LEFT", ixCoord, pdf_TextY(xPdfStream)).
    RUN pdf_line(xPdfStream,
        ixCoord - 1, pdf_TextY(xPdfStream) - 1,
        ixCoord + 25, pdf_TextY(xPdfStream) - 1, 0.1).
    RUN pdf_line(xPdfStream,
        ixCoord - 1, pdf_TextY(xPdfStream) + 8,
        ixCoord - 1, pdf_TextY(xPdfStream) - 1, 0.1).
END PROCEDURE.

PROCEDURE OpenIt:
    DEFINE INPUT  PARAMETER iFileName AS CHARACTER   NO-UNDO.
    DEF VAR wH AS INTEGER NO-UNDO.

    RUN ShellExecuteA (0,               /* or current-window:hwnd      */
        "",                             /* "" or "print"           */
        iFileName,
        "",
        "",                             /* current directory           */
        3,                              /* 0x03 = maximized, see below */
        output wH).
END PROCEDURE.

PROCEDURE ShellExecuteA EXTERNAL "shell32" :
    DEFINE INPUT PARAMETER hwnd AS LONG.
    DEFINE INPUT PARAMETER lpOperation AS char.
    DEFINE INPUT PARAMETER lpFile AS char.
    DEFINE INPUT PARAMETER lpParameters AS char.
    DEFINE INPUT PARAMETER lpDirectory AS char.
    DEFINE INPUT PARAMETER nShowCmd AS LONG.
    DEFINE RETURN PARAMETER hInstance AS LONG.
END PROCEDURE.

PROCEDURE GetLineList3DBetterA:
    DEFINE VARIABLE xi AS INTEGER     NO-UNDO.
    DEFINE VARIABLE lastBend AS DECIMAL     NO-UNDO.
    DEFINE VARIABLE CumulMagniHelper  AS DECIMAL     NO-UNDO.
    DEFINE VARIABLE MagniHelper  AS DECIMAL     NO-UNDO.
    DEFINE VARIABLE AngleHelper  AS DECIMAL     NO-UNDO.

    lastBend = 0.
    CumulMagniHelper = 0.
    xi = 1.
    FOR EACH wMeasuresBend NO-LOCK WHERE wMeasuresBend.AorB = "A":
        lastBend = CumulMagniHelper.
        MagniHelper = wMeasuresBend.Posit.
        AngleHelper = wMeasuresBend.Halk * pi / 180.
```

```
        CumulMagniHelper = CumulMagniHelper + MagniHelper.
loop:
        FOR EACH wLineList NO-LOCK:
            CREATE wLineList3Db.
            wLineList3Db.AorB        = wLineList.AorB.
            wLineList3Db.FatherOrder = wLineList.FatherOrder.
            wLineList3Db.FatherHalk  = wLineList.FatherHalk.
            wLineList3Db.Halk        = wLineList.Halk.
            wLineList3Db.Order       = wLineList.Order.
            wLineList3Db.xCoord1     = wLineList.xCoord1.
            wLineList3Db.xCoord2     = wLineList.xCoord2.
            wLineList3Db.bendIndex   = xi.

            IF (wLineList.yCoord1 >= lastBend) AND (wLineList.yCoord2 <= CumulMagniHelper) THEN DO:
                wLineList3Db.yCoord1     = wLineList.yCoord1.
                wLineList3Db.yCoord2     = wLineList.yCoord2.
                NEXT loop.
            END.
            IF (wLineList.yCoord1 <= lastBend) AND (wLineList.yCoord2 >= lastBend)  AND (wLineList.yCoord2 <= CumulMagniHelper) THEN DO:
                wLineList3Db.yCoord1     = lastBend.
                wLineList3Db.yCoord2     = wLineList.yCoord2.
                NEXT loop.
            END.
            IF (wLineList.yCoord1 >= lastBend) AND (wLineList.yCoord1 <= CumulMagniHelper)  AND (wLineList.yCoord2 >= CumulMagniHelper) THEN DO:
                wLineList3Db.yCoord1     = wLineList.yCoord1.
                wLineList3Db.yCoord2     = CumulMagniHelper.
                NEXT loop.
            END.
            IF (wLineList.yCoord1 <= lastBend) AND (wLineList.yCoord2 >= CumulMagniHelper) THEN DO:
                wLineList3Db.yCoord1     = lastBend.
                wLineList3Db.yCoord2     = CumulMagniHelper.
                NEXT loop.
            END.

            DELETE wLineList3Db.
        END.
        xi = xi + 1.
    END.

    xi = 1.
    CumulMagniHelper = 0.
    FOR EACH wMeasuresBend NO-LOCK WHERE wMeasuresBend.AorB = "A":
        MagniHelper = wMeasuresBend.Posit.
        AngleHelper = wMeasuresBend.Halk * pi / 180.

        FOR EACH wLineList3Db NO-LOCK WHERE wLineList3Db.BendIndex = xi:
            wLineList3Db.zCoord1 = wLineList3Db.zCoord1 + (wLineList3Db.yCoord1 - CumulMagniHelper) * sini(AngleHelper).
```

```
            wLineList3Db.zCoord2 = wLineList3Db.zCoord2 + (wLineList3Db.yCoord2 - CumulMagniHelper) * sini(AngleHelper).
            wLineList3Db.yCoord1 = CumulMagniHelper + (wLineList3Db.yCoord1 - CumulMagniHelper) * (kosini(AngleHelper)).
            wLineList3Db.yCoord2 = CumulMagniHelper + (wLineList3Db.yCoord2 - CumulMagniHelper) * (kosini(AngleHelper)).
        END.

        FOR EACH wLineList3Db NO-LOCK WHERE wLineList3Db.BendIndex > xi:
            wLineList3Db.yCoord1 = wLineList3Db.yCoord1 - MagniHelper * (1 - kosini(AngleHelper)).
            wLineList3Db.yCoord2 = wLineList3Db.yCoord2 - MagniHelper * (1 - kosini(AngleHelper)).
            wLineList3Db.zCoord1 = wLineList3Db.zCoord1 + MagniHelper * sini(AngleHelper).
            wLineList3Db.zCoord2 = wLineList3Db.zCoord2 + MagniHelper * sini(AngleHelper).
        END.

        CumulMagniHelper = CumulMagniHelper + MagniHelper * (kosini(AngleHelper)).
        xi = xi + 1.
    END.
END PROCEDURE.

PROCEDURE GetLineList3DBetterB:
    DEFINE VARIABLE xi AS INTEGER     NO-UNDO.
    DEFINE VARIABLE lastBend AS DECIMAL     NO-UNDO.
    DEFINE VARIABLE CumulMagniHelper  AS DECIMAL     NO-UNDO.
    DEFINE VARIABLE MagniHelper  AS DECIMAL     NO-UNDO.
    DEFINE VARIABLE AngleHelper  AS DECIMAL     NO-UNDO.

    lastBend = 0.
    CumulMagniHelper = 0.
    xi = 1.
    FOR EACH wMeasuresBend NO-LOCK WHERE wMeasuresBend.AorB = "B":
        lastBend = CumulMagniHelper.
        MagniHelper = wMeasuresBend.Posit.
        AngleHelper = wMeasuresBend.Halk * pi / 180.
        CumulMagniHelper = CumulMagniHelper + MagniHelper.
loop:
        FOR EACH wLineList3Db NO-LOCK:
            CREATE wLineList3D.
            wLineList3D.AorB        = wLineList3Db.AorB.
            wLineList3D.FatherOrder = wLineList3Db.FatherOrder.
            wLineList3D.FatherHalk  = wLineList3Db.FatherHalk.
            wLineList3D.Halk        = wLineList3Db.Halk.
            wLineList3D.Order       = wLineList3Db.Order.
            wLineList3D.yCoord1     = wLineList3Db.yCoord1.
            wLineList3D.yCoord2     = wLineList3Db.yCoord2.
            wLineList3D.zCoord1     = wLineList3Db.zCoord1.
            wLineList3D.zCoord2     = wLineList3Db.zCoord2.

            wLineList3D.bendIndex   = xi.

```

```
            IF (wLineList3Db.xCoord1 >= lastBend) AND (wLineList3Db.xCoord2 <= CumulMagniHelper) THEN DO:
                wLineList3D.xCoord1     = wLineList3Db.xCoord1.
                wLineList3D.xCoord2     = wLineList3Db.xCoord2.
                NEXT loop.
            END.
            IF (wLineList3Db.xCoord1 <= lastBend) AND (wLineList3Db.xCoord2 >= lastBend)  AND (wLineList3Db.xCoord2 <= CumulMagniHelper) THEN DO:
                wLineList3D.xCoord1     = lastBend.
                wLineList3D.xCoord2     = wLineList3Db.xCoord2.
                NEXT loop.
            END.
            IF (wLineList3Db.xCoord1 >= lastBend) AND (wLineList3Db.xCoord1 <= CumulMagniHelper)  AND (wLineList3Db.xCoord2 >= CumulMagniHelper)
THEN DO:
                wLineList3D.xCoord1     = wLineList3Db.xCoord1.
                wLineList3D.xCoord2     = CumulMagniHelper.
                NEXT loop.
            END.
            IF (wLineList3Db.xCoord1 <= lastBend) AND (wLineList3Db.xCoord2 >= CumulMagniHelper) THEN DO:
                wLineList3D.xCoord1     = lastBend.
                wLineList3D.xCoord2     = CumulMagniHelper.
                NEXT loop.
            END.

            DELETE wLineList3D.
        END.
        xi = xi + 1.
    END.
    xi = 1.
    CumulMagniHelper = 0.
    FOR EACH wMeasuresBend NO-LOCK WHERE wMeasuresBend.AorB = "B":
        MagniHelper = wMeasuresBend.Posit.
        AngleHelper = wMeasuresBend.Halk * pi / 180.

        FOR EACH wLineList3D NO-LOCK WHERE wLineList3D.BendIndex = xi:
            wLineList3D.zCoord1 = wLineList3D.zCoord1 + (wLineList3D.xCoord1 - CumulMagniHelper) * sini(AngleHelper).
            wLineList3D.zCoord2 = wLineList3D.zCoord2 + (wLineList3D.xCoord2 - CumulMagniHelper) * sini(AngleHelper).
            wLineList3D.xCoord1 = CumulMagniHelper + (wLineList3D.xCoord1 - CumulMagniHelper) * (kosini(AngleHelper)).
            wLineList3D.xCoord2 = CumulMagniHelper + (wLineList3D.xCoord2 - CumulMagniHelper) * (kosini(AngleHelper)).
        END.

        FOR EACH wLineList3D NO-LOCK WHERE wLineList3D.BendIndex > xi:
            wLineList3D.xCoord1 = wLineList3D.xCoord1 - MagniHelper * (1 - kosini(AngleHelper)).
            wLineList3D.xCoord2 = wLineList3D.xCoord2 - MagniHelper * (1 - kosini(AngleHelper)).
            wLineList3D.zCoord1 = wLineList3D.zCoord1 + MagniHelper * sini(AngleHelper).
            wLineList3D.zCoord2 = wLineList3D.zCoord2 + MagniHelper * sini(AngleHelper).
        END.

        CumulMagniHelper = CumulMagniHelper + MagniHelper * (kosini(AngleHelper)).
```

```
844         xi = xi + 1.
845     END.
846 END PROCEDURE.
847 
848 PROCEDURE ThreeD2TwoDPlane: /* Isometric perspective, populate 3Dviewpoints. */
849     EMPTY TEMP-TABLE wLineList.
850 
851     FOR EACH wLineList3D WHERE NOT (wLineList3D.xCoord1 = wLineList3D.xCoord2 AND wLineList3D.yCoord1 = wLineList3D.yCoord2 AND
852 wLineList3D.zCoord1 = wLineList3D.zCoord2) NO-LOCK:
853         CREATE wLineList.
854         wLineList.AorB        = wLineList3D.AorB.
855         wLineList.FatherOrder = wLineList3D.FatherOrder.
856         wLineList.FatherHalk  = wLineList3D.FatherHalk.
857         wLineList.Halk        = wLineList3D.Halk.
858         wLineList.Order       = wLineList3D.Order.
859         wLineList.xCoord1     = (wLineList3D.xCoord1 + 0.5 * wLineList3D.yCoord1).
860         wLineList.xCoord2     = (wLineList3D.xCoord2 + 0.5 * wLineList3D.yCoord2).
861         wLineList.yCoord1     = (wLineList3D.yCoord1 + wLineList3D.zCoord1).
862         wLineList.yCoord2     = (wLineList3D.yCoord2 + wLineList3D.zCoord2).
863     END.
864 END PROCEDURE.
865 
866 PROCEDURE ThreeD2TwoDPlane2: /* Isometric perspective, populate 3Dviewpoints. */
867     EMPTY TEMP-TABLE wLineList.
868 
869     FOR EACH wLineList3D WHERE NOT (wLineList3D.xCoord1 = wLineList3D.xCoord2 AND wLineList3D.yCoord1 = wLineList3D.yCoord2 AND
870 wLineList3D.zCoord1 = wLineList3D.zCoord2) NO-LOCK:
871         CREATE wLineList.
872         wLineList.AorB        = wLineList3D.AorB.
873         wLineList.FatherOrder = wLineList3D.FatherOrder.
874         wLineList.FatherHalk  = wLineList3D.FatherHalk.
875         wLineList.Halk        = wLineList3D.Halk.
876         wLineList.Order       = wLineList3D.Order.
877         wLineList.xCoord1     = (wLineList3D.yCoord1 + 0.5 * wLineList3D.xCoord1).
878         wLineList.xCoord2     = (wLineList3D.yCoord2 + 0.5 * wLineList3D.xCoord2).
879         wLineList.yCoord1     = (wLineList3D.xCoord1 + wLineList3D.zCoord1).
880         wLineList.yCoord2     = (wLineList3D.xCoord2 + wLineList3D.zCoord2).
881     END.
882 
883 END PROCEDURE.
884 
885 PROCEDURE ThreeD2TwoDPlane3: /* Isometric perspective, populate 2Dviewpoints. */
886     EMPTY TEMP-TABLE wLineList.
887 
888     FOR EACH wLineList3D WHERE NOT (wLineList3D.xCoord1 = wLineList3D.xCoord2 AND wLineList3D.yCoord1 = wLineList3D.yCoord2 AND
889 wLineList3D.zCoord1 = wLineList3D.zCoord2) NO-LOCK:
890         CREATE wLineList.
```

```
        wLineList.AorB        = wLineList3D.AorB.
        wLineList.FatherOrder = wLineList3D.FatherOrder.
        wLineList.FatherHalk  = wLineList3D.FatherHalk.
        wLineList.Halk        = wLineList3D.Halk.
        wLineList.Order       = wLineList3D.Order.
        wLineList.xCoord1     = (wLineList3D.xCoord1).
        wLineList.xCoord2     = (wLineList3D.xCoord2).
        wLineList.yCoord1     = (wLineList3D.zCoord1).
        wLineList.yCoord2     = (wLineList3D.zCoord2).
    END.
END PROCEDURE.

PROCEDURE ThreeD2TwoDPlane4: /* Isometric perspective, populate 2Dviewpoints. */
    EMPTY TEMP-TABLE wLineList.

    FOR EACH wLineList3D WHERE NOT (wLineList3D.xCoord1 = wLineList3D.xCoord2 AND wLineList3D.yCoord1 = wLineList3D.yCoord2 AND
wLineList3D.zCoord1 = wLineList3D.zCoord2) NO-LOCK:
        CREATE wLineList.
        wLineList.AorB        = wLineList3D.AorB.
        wLineList.FatherOrder = wLineList3D.FatherOrder.
        wLineList.FatherHalk  = wLineList3D.FatherHalk.
        wLineList.Halk        = wLineList3D.Halk.
        wLineList.Order       = wLineList3D.Order.
        wLineList.xCoord1     = (wLineList3D.yCoord1).
        wLineList.xCoord2     = (wLineList3D.yCoord2).
        wLineList.yCoord1     = (wLineList3D.zCoord1).
        wLineList.yCoord2     = (wLineList3D.zCoord2).
    END.
END PROCEDURE.

PROCEDURE Draw3DLines:
    DEFINE INPUT  PARAMETER ixOffset AS DECIMAL     NO-UNDO.
    DEFINE INPUT  PARAMETER iyOffset AS DECIMAL     NO-UNDO.
    ixOffset = ixOffset * DECIMAL(pdf_PageWidth(xPdfStream)).
    iyOffset = iyOffset * DECIMAL(pdf_PageHeight(xPdfStream)).

    FOR EACH WLineList NO-LOCK:
        IF wLineList.AorB = "A" THEN RUN pdf_set_dash (xPdfStream, 1, 2).
        ELSE RUN pdf_set_dash (xPdfStream, 1, 0).
        RUN pdf_line_dec(xPdfStream,
            pdf_LeftMargin(xPdfStream) + wLineList.xCoord1 / xScaleDivider + ixOffset, wLineList.yCoord1 / xScaleDivider + iyOffset,
            pdf_LeftMargin(xPdfStream) + wLineList.xCoord2 / xScaleDivider + ixOffset, wLineList.yCoord2 / xScaleDivider + iyOffset,
wLineList.Halk / 4).

        IF wLineList.xCoord1 = wLineList.xCoord2 AND wLineList.yCoord1 = wLineList.yCoord2 THEN DO:
            RUN pdf_set_dash (xPdfStream, 1, 0).
            RUN pdf_line_dec (xPdfStream,
```

```
                pdf_LeftMargin(xPdfStream) + wLineList.xCoord1 / xScaleDivider + ixOffset - 2,
                wLineList.yCoord1 / xScaleDivider + iyOffset - 2,
                pdf_LeftMargin(xPdfStream) + wLineList.xCoord1 / xScaleDivider + ixOffset + 2,
                wLineList.yCoord1 / xScaleDivider + iyOffset + 2,
                wLineList.Halk / 2).
            RUN pdf_line_dec (xPdfStream,
                pdf_LeftMargin(xPdfStream) + wLineList.xCoord1 / xScaleDivider + ixOffset + 2,
                wLineList.yCoord1 / xScaleDivider + iyOffset - 2,
                pdf_LeftMargin(xPdfStream) + wLineList.xCoord1 / xScaleDivider + ixOffset - 2,
                wLineList.yCoord1 / xScaleDivider + iyOffset + 2,
                wLineList.Halk / 2).
        END.
    END.
END PROCEDURE.

PROCEDURE SaveOle:
    RUN prog/sysole/oleadd.p("SlsLine", SlsLine.SlsLineKey, xFileName). /* Tämän ulkopuolisen proseduurin ajaminen on hieman arveluttavaa, SBO:ta
ei käytetä */
END PROCEDURE.

PROCEDURE GetBendEffects:

    DEFINE VARIABLE xRadius AS DECIMAL     NO-UNDO INIT 200.
    DEFINE VARIABLE xi AS INTEGER     NO-UNDO.
    DEFINE VARIABLE xNumSteps AS DECIMAL    NO-UNDO INIT 6.0.
    DEFINE VARIABLE xSnip AS DECIMAL    NO-UNDO.

    FIND FIRST wLineList NO-LOCK WHERE wLineList.AorB = "A".
    xRadius = wLineList.FatherHalk * 3.

    /* Effects on wMeasures */
    /* Effects on info */

    /* Effects on the 2D-linelist */

    FOR EACH wMeasuresBend NO-LOCK WHERE wMeasuresBend.AorB = "B",
        EACH wLineList WHERE
            (wLineList.xCoord1 >= wMeasuresBend.CumulPituus1) BY wLineList.xCoord1:
            wLineList.xCoord1 = wLineList.xCoord1 - shrinkage(angleDelta(wMeasuresBend.OldHalk, wMeasuresBend.Halk), xRadius).
    END.
    FOR EACH wMeasuresBend NO-LOCK WHERE wMeasuresBend.AorB = "B",
        EACH wLineList WHERE
            (wLineList.xCoord2 >= wMeasuresBend.CumulPituus1) BY wLineList.xCoord2:
            wLineList.xCoord2 = wLineList.xCoord2 - shrinkage(angleDelta(wMeasuresBend.OldHalk, wMeasuresBend.Halk), xRadius).
    END.

    FOR EACH wMeasuresBend NO-LOCK WHERE wMeasuresBend.AorB = "A",
```

```
        EACH wLineList WHERE
        (wLineList.yCoord1 >= wMeasuresBend.CumulPituus1) BY wLineList.yCoord1:
            wLineList.yCoord1 = wLineList.yCoord1 - shrinkage(angleDelta(wMeasuresBend.OldHalk, wMeasuresBend.Halk), xRadius).
    END.
    FOR EACH wMeasuresBend NO-LOCK WHERE wMeasuresBend.AorB = "A",
        EACH wLineList WHERE
        (wLineList.yCoord2 >= wMeasuresBend.CumulPituus1) BY wLineList.yCoord2:
            wLineList.yCoord2 = wLineList.yCoord2 - shrinkage(angleDelta(wMeasuresBend.OldHalk, wMeasuresBend.Halk), xRadius).
    END.

    /* Effects on wMeasuresBend */
    FOR EACH wMeasuresBend NO-LOCK:
        xSnip = 2 * xRadius * tangentti((angleDelta(wMeasuresBend.OldHalk, wMeasuresBend.Halk) * pi / 180.0) / 2). /* Etäisyys kulmapisteestä */
        FOR EACH bwMeasuresBend EXCLUSIVE-LOCK WHERE /* further sections */
            bwMeasuresBend.AorB = wMeasuresBend.AorB AND
            bwMeasuresBend.CumulPituus1 >= wMeasuresBend.CumulPituus1 BY bwMeasuresBend.CumulPituus1 DESC:
            IF bwMeasuresBend.CumulPituus1 = wMeasuresBend.CumulPituus1 THEN DO:
                FOR FIRST bbwMeasuresBend EXCLUSIVE-LOCK WHERE /* the previous section */
                    bbwMeasuresBend.AorB = wMeasuresBend.AorB AND
                    bbwMeasuresBend.CumulPituus2 = wMeasuresBend.CumulPituus1:
                    bbwMeasuresBend.CumulPituus2 = bbwMeasuresBend.CumulPituus2 - xSnip * 0.5.
                    bbwMeasuresBend.Posit = bbwMeasuresBend.CumulPituus2 - bbwMeasuresBend.CumulPituus1.
                END.
                bwMeasuresBend.CumulPituus1 = bwMeasuresBend.CumulPituus1 - xSnip * 0.5.
                bwMeasuresBend.CumulPituus2 = bwMeasuresBend.CumulPituus2 - xSnip.
            END.
            ELSE DO:
                bwMeasuresBend.CumulPituus1 = bwMeasuresBend.CumulPituus1 - xSnip.
                bwMeasuresBend.CumulPituus2 = bwMeasuresBend.CumulPituus2 - xSnip.
            END.
            bwMeasuresBend.Posit = bwMeasuresBend.CumulPituus2 - bwMeasuresBend.CumulPituus1.
        END.
    END.

    FOR EACH wMeasuresBendTemp NO-LOCK:
        DELETE wMeasuresBendTemp.
    END.

    DEFINE VARIABLE xii AS INTEGER     NO-UNDO.
    DEFINE VARIABLE xArcLen AS DECIMAL     NO-UNDO.
    DEFINE VARIABLE xStep AS DECIMAL    NO-UNDO.
    DEFINE VARIABLE xChordStep AS DECIMAL    NO-UNDO.

    FOR EACH wMeasuresBend NO-LOCK BREAK BY wMeasuresBend.AorB BY wMeasuresBend.CumulPituus2:
        IF FIRST-OF(wMeasuresBend.AorB) THEN xii = 0.
        xArcLen = arcLen(angleDelta(wMeasuresBend.OldHalk, wMeasuresBend.Halk), xRadius).
```

```
        xStep = angleDelta(wMeasuresBend.OldHalk, wMeasuresBend.Halk) / xNumSteps.
        /* xChordStep = chordLen(angleDelta(wMeasuresBend.OldHalk, wMeasuresBend.Halk) / (xNumSteps - 1), xRadius).  /* Accurate */ */
        xChordStep = arcLen(angleDelta(wMeasuresBend.OldHalk, wMeasuresBend.Halk) / (xNumSteps - 1), xRadius).     /* Inaccurate, but nice. */

        xi = xStep.
        IF angleDelta(wMeasuresBend.OldHalk, wMeasuresBend.Halk) <> 0 THEN DO WHILE xi < angleDelta(wMeasuresBend.OldHalk, wMeasuresBend.Halk):
            CREATE wMeasuresBendTemp.
            ASSIGN
                wMeasuresBendTemp.Posit = xChordStep /* chordLen(xStep, xRadius) */
                wMeasuresBendTemp.Halk =  (IF isClockwiceTurn(wMeasuresBend.OldHalk, wMeasuresBend.Halk) THEN wMeasuresBend.OldHalk + xi ELSE
wMeasuresBend.OldHalk - xi)
                wMeasuresBendTemp.AorB =  wMeasuresBend.AorB
                wMeasuresBendTemp.Order = xii.
            xii = xii + 1.
            xi = xi + xStep.
        END.

        CREATE wMeasuresBendTemp.
        ASSIGN
            wMeasuresBendTemp.Posit = wMeasuresBend.Posit
            wMeasuresBendTemp.Halk  = wMeasuresBend.Halk
            wMeasuresBendTemp.AorB =  wMeasuresBend.AorB
            wMeasuresBendTemp.Order = xii.
        xii = xii + 1.
    END.

    FOR EACH wMeasuresBend NO-LOCK:
        DELETE wMeasuresBend.
    END.
    FOR EACH wMeasuresBendTemp NO-LOCK:
        CREATE wMeasuresBend.
        BUFFER-COPY wMeasuresBendTemp TO WMeasuresBend.
    END.

    /* Effects on Scale */

END.

PROCEDURE sin EXTERNAL "MSVCRT40.DLL" CDECL:
   DEFINE INPUT  PARAMETER dblValue  AS DOUBLE NO-UNDO.
   DEFINE RETURN PARAMETER dblResult AS DOUBLE NO-UNDO.
END PROCEDURE.

PROCEDURE cos EXTERNAL "MSVCRT40.DLL" CDECL:
   DEFINE INPUT  PARAMETER dblValue  AS DOUBLE NO-UNDO.
   DEFINE RETURN PARAMETER dblResult AS DOUBLE NO-UNDO.
END PROCEDURE.
```

```
1079
1080 PROCEDURE tan EXTERNAL "MSVCRT40.DLL" CDECL:
1081    DEFINE INPUT  PARAMETER dblValue  AS DOUBLE NO-UNDO.
1082    DEFINE RETURN PARAMETER dblResult AS DOUBLE NO-UNDO.
1083 END PROCEDURE.
1084
1085 PROCEDURE trigonInit:
1086     DEFINE VARIABLE RESULT AS DECIMAL     NO-UNDO.
1087     DEFINE VARIABLE xInt AS INTEGER    NO-UNDO.
1088     DEFINE VARIABLE xInt2 AS DECIMAL    NO-UNDO.
1089     DEFINE VARIABLE xRound AS INTEGER    NO-UNDO.
1090
1091     {prog/slsline/somesinvalues.i}
1092     REPEAT xInt = 1 TO 91:
1093         RESULT = xSin[xInt].
1094         xSin[180 - xInt + 2] = RESULT.
1095         xSin[xInt + 180] = 0 - RESULT.
1096         xSin[360 - xInt + 2] = 0 - RESULT.
1097     END.
1098     REPEAT xInt = 1 TO 90:
1099         xCos[xInt] = xSin[90 + xInt].
1100         xCos[xInt + 90] = xSin[180 + xInt].
1101         xCos[xInt + 180] = xSin[270 + xInt].
1102         xCos[xInt + 270] = xSin[xInt].
1103         xCos[361] = xCos[1].
1104     END.
1105     REPEAT xInt = 0 TO 360:
1106         xTan[xInt + 1] = xSin[xInt + 1] / xCos[xInt + 1] NO-ERROR.
1107     END.
1108 END.
```