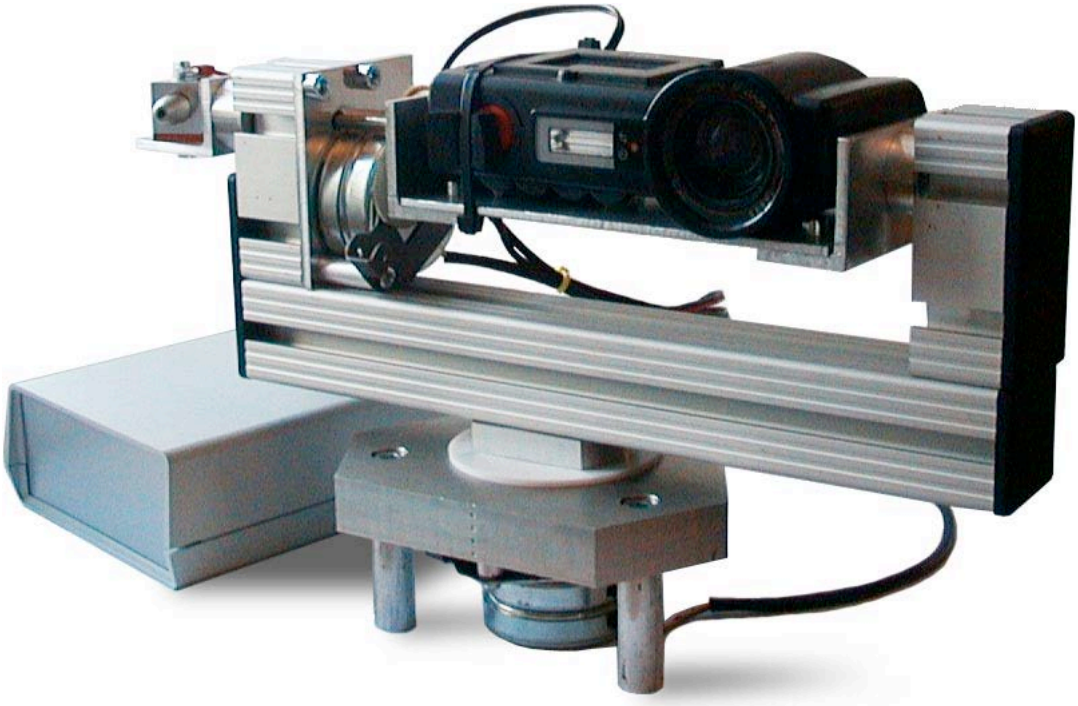
Arno Solin

Päättötyö



Turun Steiner-koulu
2004 – 2005

Sisällysluettelo

Saatteeksi

Päättötyö on vapaavalintainen projekti, joka toteutetaan Steiner-lukion kolmannella luokalla. Tämä ylimääräinen työ on osa Steiner-koulujen opetussuunnitelmaa, ja sen tarkoitus on opastaa oppilasta itsenäiseen työskentelyyn valitun aiheen parissa. Kuulee päättötyötä kutsuttavan jopa kypsyyskokeeksikin.

Omalla kohdallani päättötyön aiheen valinta ei ollut helppo. Kun alkujaan aloitin lukion, odotin päättötyön tekemistä innolla sen suuremmin aihetta miettimättä. Ajattelin, että aihe kyllä olisi valmiina ajan tullessa. Ei ollut. Kiinnostuskohteeni ovat moninaiset ja töitä saa – ainakin tietääkseni – tehdä vain yhden. Halusin löytää mahdollisimman mielenkiintoisen, monipuolisen ja samalla haastavan aiheen. Lisäksi kriteerinä oli, ettei työ saanut olla pelkästään teoreettinen. Aiheen tuli olla alustavasti valittuna keväällä 2004, ja pari viikkoa ennen viimeistä jättöpäivää oli yhä sormi suussa: Aiheesta, kun ei ollut tietoakaan.

Pelastuksen toi äkkiä iskenyt kevätflunssa, joka kaatoi vuoteenomaksi useaksi päiväksi. Kuumeen noustua 39 °C tuntumaan ei ajatus enää juossut jouheasti. Tässä luovassa tilassa sain idean ympärillään olevaa tilaa kartoittavasta laitteesta. Hieman myöhemmin keksin toisen aiheen, nojatuolin tekemisen.

Nämä kaksi aihetta nousivat ylitse muiden. Lopullinen ratkaisu mittalaitteen hyväksi syntyi huonekaluammattilaisen konsultoinnin jälkeen. Hän piti kunnon nojatuolin rakentamista aivan liian vaikeana tehtävänä vasta-alkajalle. Tiedä häntä, mahdoinko sittenkään valita aiheista helpomman.

Päättötyön aiheeksi tuli lopulta ”Etäisyysmittalaite, joka kartoittaa ympärillään olevaa tilaa lasermittauksen avulla ja luo tietokoneelle kolmiulotteisen mallin”. Lisäksi aiheen yhteyteen kuului mittalaitteen ja -ohjelman dokumentaatio. Valittu aihe täytti asettamani kriteerit. Se oli haastava ja monipuolinen, muttei pelkästään teoreettinen. Aihe jakautui selkeästi osiin. Laitteen rakentamisen, mekaniikan, lisäksi aiheeseen liittyi ohjauselektroniikan tekeminen. Laitteen toiminta perustui sen liittämiseen tietokoneeseen, joten myös ohjelmointi oli tarpeen. Lisäksi työn kirjallinen osuus oli osa aihetta.

Kirjallista osuutta voi pitää jopa työn tärkeimpänä osuutena. ”Jos sitä ei ole dokumentoitu, sitä ei ole tehty.” Tätä periaatetta löyhästi noudattaen olen koonnut päättötyön kirjallisen osuuden, joka tyylillisesti sivuaa niin työselostusta, rakennusohjeita, tutkielmaakin ja osin jopa esseetä. Kirjallinen osuus pyrkii selittämään ja pohjustamaan päättötyön tekemisessä tehtyjä ratkaisuja sekä seuraamaan työn edistymistä vaihe kerrallaan.

Kirjallisen osuuden tekemistä hankaloitti se tosiasia, etteivät käsiteltävät asiat ja käsitteet ole jokaiselle potentiaaliselle lukijalle ennestään tuttuja. Tämän takia osa työvaiheista on vaatinut perinpohjaista tarkastelua. Olen kuitenkin vetänyt tiettyjä rajoja käsittelyn laajuudelle. Hankalat asiat on selitetty mahdollisesti yksinkertaistaen tai jättämällä aiheeseen liittyneitä ylimääräisiä ominaisuuksia pois. Kirjallisen työn lukemiseen vaaditut perustaidot ovat kutakuinkin sillä tasolla kuin omat tietoni olivat työtä aloitellessa. Kirjalliseen osuuteen kuuluu tekstin lisäksi kuvitus, joka on olennaisessa osassa.

Mikä päättötyöni oikeastaan on? Vaikka aiheena on mittalaitteen rakentaminen ja siihen liittyvä elektroniikka ja ohjelmointi, on päättötyö itsessään paljon muutakin. En lähestynyt päättötyön tekemistä halulla saada mitään konkreettista aikaan. Tavoitteena ei ollut paksu nivaska papereita, ei kiva mittahärveli, vaan tavoitteen oli uuden oppiminen. Lähestyin

aihetta kokemuksena – opettavana sellaisena. Itse asiassa sillä ei ollut loppujen lopuksi mitään väliä, että pystyisikö mittalaite tuottamaan realistisia malleja ympäristöstään. Se ei koskaan ollut tavoitteena. Tavoitteena oli oppia perusteita mekaniikasta, elektroniikasta ja ohjelmoinnista, oppia sellaisia asioita, joiden pariin muuten saattaisi olla suuri kynnys paneutua muun tekemisen ohessa.

Aikaa on kulunut päättötyön parissa niin paljon, etten uskalla varovaistakaan arviota heittää käytettyjen työtuntien määrästä. Oman työpanokseni lisäksi projekti on verottanut myös muiden ihmisten aikaa. Erityisesti ohjaajani Stefan Johansson, joka toimii Åbo Akademin hiukkaskiihdytinlaboratorion pääinsinöörinä, on auttanut ja opastanut minua todella paljon erityisesti mekaniikan ja elektroniikan osa-alueilla. Yksin hän lienee viettänyt projektini parissa enemmän aikaa kuin olisin edes uskaltanut häneltä pyytää.

Haluan kiittää myös toista ohjaajaani opettaja Marjo Lahtea, joka on ollut ohjaajani koulun puolelta. Lisäksi kiitokset ovat ansainneet koulun teknisen työn opettaja Martti Palin, Åbo Akademin Fysiikan laitoksen verstaan esimies Tom Wikström sekä perheenjäseneni auttavista käsistä, neuvoista ja kärsivällisyydestä. Kiitän myös niitä ystäviäni, jotka ovat osoittaneet kiinnostusta projektia kohtaan sekä niitä, jotka ovat teeskennelleet kiinnostunutta.

Noin kymmenen kuukauden työrupeaman jättää mieluusti taakseen sen valmistuttua. Eniten työn valmistumista iloinnee kuitenkin äitini: Vihdoinkin poika raahaa romunsa pois hänen työpöydältään. Odota vain äiti, uusia projekteja siintää jo horisontissa.

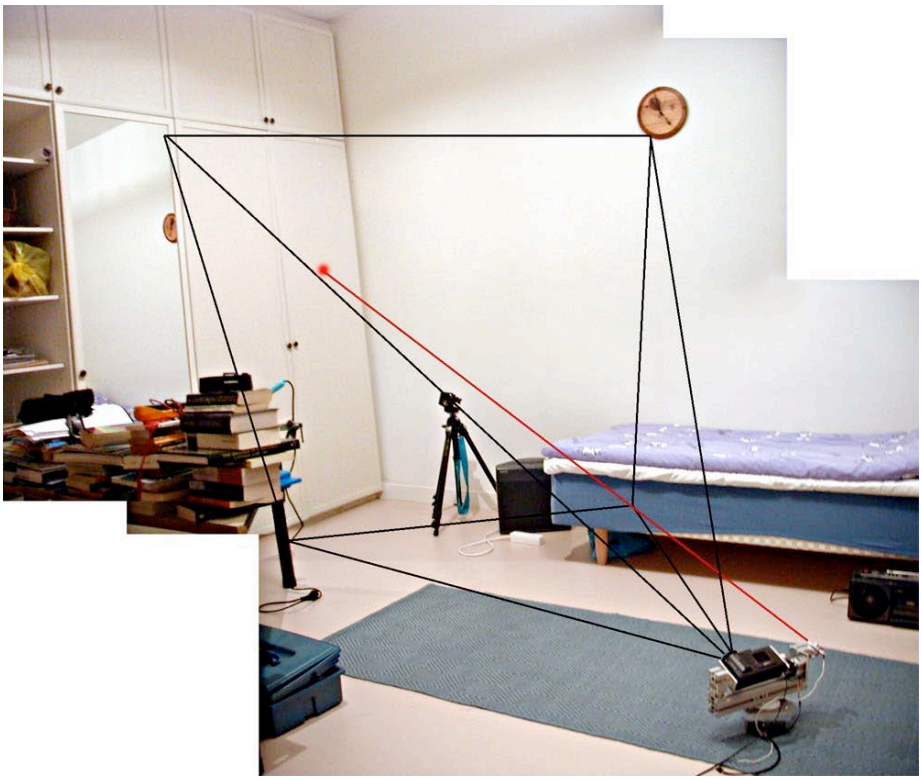
Arno Solin
Turussa 6.3.2005

Toimintaperiaate

Päättötyön mittalaite kartoittaa ympäristöään ja luo saadusta informaatiosta summittaisen kolmiulotteisen mallin tietokoneelle. Laitteen toiminta perustuu etäisyysmittaukseen, jonka avulla on mahdollisuus saada mitattua etäisyyksiä laitteen ja seinän tai muun vastaavan esteen välillä. Etäisyysmittauksessa saadut arvot eivät yksin riitä kolmiulotteisen mallin luomiseen. Tästä syystä on tarpeen tietää mihin suuntaan laite milloinkin osoittaa.

Laite pystyy kääntymään pysty- ja vaakasuunnassa. Pystysuunnassa riittää vajaa 90°, mutta vaakasuunnassa on tarpeen päästä koko kierros ympäri. Kun tiedetään mihin suuntaan etäisyysmittalaite osoittaa ja saadaan mitattua etäisyys esteeseen, voidaan näiden kolmen arvon avulla laskea mittapisteen koordinaatti trigonometristen funktioiden avulla. Kun pisteitä mitataan suuri määrä, voidaan pisteiden kautta hahmottaa laitetta ympäröivää tilaa.

Etäisyysmittaus on toteutettu kameran ja laserin avulla. Aihetta käsitellään perusteellisesti ohjelmoinnin yhteydessä. Kyseessä on mielenkiintoinen – jopa luova – mittausmenetelmä, joskin epätarkka. Etäisyysmittalaitteisto on saatava liikkumaan. Liikkuminen on toteutettu rakentamalla pääosin alumiinista koostuvan laitteen, johon sekä kamera että laser on kiinnitetty. Rakennelman hallittu ohjaaminen on toteutettu askelmoottoreiden avulla. Mittalaitteen tekemistä, rakentamista, käsitellään mekaanisessa osuudessa. Moottoreiden, laserin ja kameran virransyöttöä käsitellään sen sijaan elektroniikka-otsikon alla.



Laitteen toimintaperiaate esitettynä kuvassa.

Mekaniikka

Pohjustus

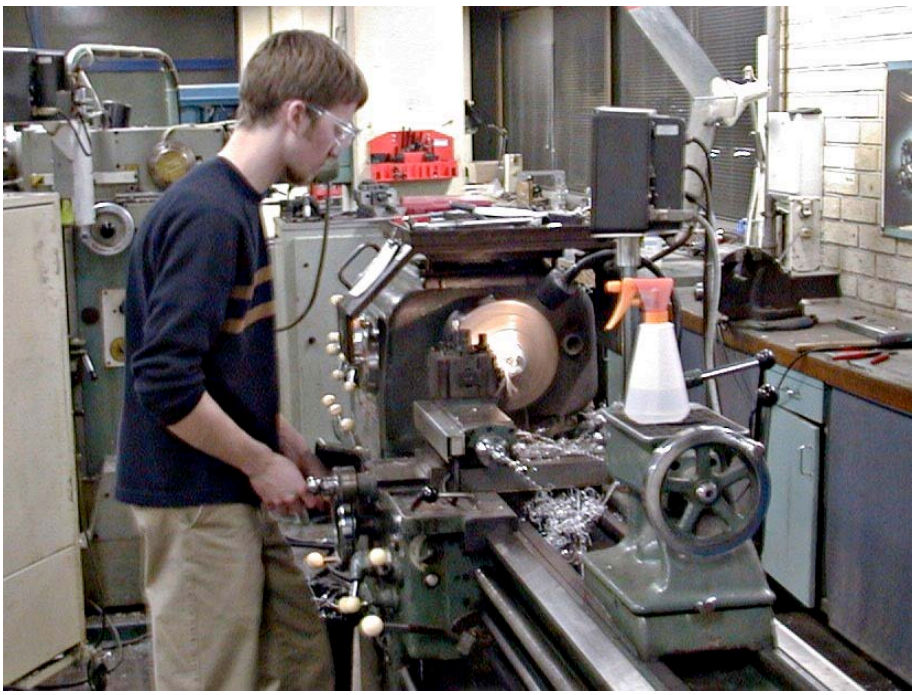
Päättötyöhön liittyvä mekaniikka oli minua alkuun eniten huolestuttanut osa-alue. Vähäinen kokemukseni mekaniikan parista rajoittuu koulun tekniseen työhön ja pienimuotoiseen puuhasteluun kesämökillä. Metallin työstö ja siihen liittyneet työvaiheet ja -kalut olivat minulle miltei tuntemattomia.

Uutuudenviehätys lienee siksi kuvaavin sana luonnehtimaan viettämiäni tunteja verstaassa sorvin ja jyrsimen ääressä. Yksin olisin tuskin päässyt kovinkaan pitkälle. Onneksi minulla oli apunani ohjaajani Stefan Johansson, joka tutustutti minut työn lomassa erilaisiin työstömenetelmiin ja toteutustapoihin.

Opin myös piirustusten tarpeellisuuden. Aina kuulee hoettavan, miten tärkeitä piirustukset ovat. Nyt näki käytännössä etukäteen tehtyjen suunnitelmien tarpeellisuuden ja suunnitelmallisuuden tuoman järjestyksen työntekoon.

Päättötyöni mekaaninen osuus on erittäin tärkeä kokonaisuuden toimimisen kannalta. Mekaaniselta idealtaan laite on yksinkertainen. Perusajatuksena on kiinnittää digitaalikamera alustaan, jota voi kääntää pysty ja vaakatasossa. Tämän lisäksi on saatava pieni osoitinlaser kääntymään samaan tahtiin kameran kanssa. Kääntyminen on lisäksi toteutettava niin, että laitetta hallitseva ohjelma jatkuvasti tietää mihin suuntaan kamera on kääntyneenä.

Olen esittänyt tässä osiossa laitteen rakentamiseen liittyneitä ratkaisuja ja ongelmakohtia, osien tekoa sekä työstöprosessiin liittyneitä seikkoja. Mittalaitteen piirustukset antavat kuvan laitteen toiminnasta ja niiden perusteella lienee jopa mahdollista toistaa työ miltei millilleen. Laitteen mekaniikka toimii pohjana käytetylle elektroniikalle ja ohjelmoiduille ominaisuuksille. Toimivan perusrakenteen suunnittelu ja toteutus vaati keskittymistä ja kärsivällisyyttä. Opettavaisen pitkäjänteisyyden kautta mittalaite, päättötyön fyysinen ilmentymä, valmistui hiljalleen noin neljän kuukauden kuluessa.



Sorvin ääressä.

Suunnittelua määräävät osat

Ajattelen usein mekaanisia ongelmia Lego-palikoiden kautta. Lähes kaikki tuntemani ihmiset ovat jossakin elämänsä vaiheessa olleet tekemisissä niiden kanssa, joten niitä voinee käyttää esimerkkinä mekaanisista osista. Legoilla rakennettaessa palaset itsessään asettavat tiettyjä rajoituksia, ja siksi on aina lähdettävä liikkeelle niistä palasista, jotka ovat elintärkeitä koko rakennelman onnistumisen kannalta. Näiden muutaman palasen ympärille kootaan loput rakennelmasta. Tämä pätee, oli rakenteilla sitten poliisiasema, hevostalli tai avaruusraketti.

Myös päättötyöni rakentaminen lähti liikkeelle parista koko kokoonpanoa määräävistä perusosista. Näitä osia olivat kamera, laser sekä moottorit, jotka valittiin tarkoitukseen sopiviksi. Näiden osien ympärille suunniteltiin loppurakennelma. Sekä kamera, moottorit että laser liittyvät olennaisesti mekaniikan lisäksi päättötyön elektroniseen puoleen, joten osien kuvaukset on hajautettu. Laseria käsitellään elektroniikan yhteydessä. Samoin on moottoreiden laita. Moottorit ja varsinkin hammasrattaiden avulla toteutettu vaihteisto kuuluu kuitenkin kiinteästi mekaniikan yhteyteen.

Kameraa kuitenkin tarkastellaan lähinnä passiivisena, pelkästään mekaanisena, kappaleena, sillä sen elektroniikan toimintaan ei juuri puututa. Tästä syystä olennaiset perustiedot kameran rakenteesta kerrotaan mekaniikan yhteydessä.

Kamera

Etäisyysmittauksessa tärkeää osaa näyttelee digitaalikamera. Kameran valinta ei ollut vaikea, sillä mitään valintaa ei varsinaisesti tarvinnut tehdä. Nurkissani oli pyörinyt jo pitkään Agfan ePhoto kamera, jota oli ikä alkanut pahasti painaa. Päätyminen vanhan kameran käyttöön ei välttämättä ollut kaikkein paras vaihtoehto, mutta toimiva sentään. Agfan ePhoto-sarjan kameroissa on etuna se, että ne ovat niin vanhoja, että kameran ja tietokoneen välinen liikenne hoidetaan RS-232 -portin kautta.

Kamera on vuonna 1999 ostettu Agfa ePhoto 1280. Mallimerkintä 1280 viittaa kameran suurimpaan resoluutioon, joten voidaan laskea kyseessä olevan noin 1,2 megapikiselin kamera. Vertailun vuoksi voidaan todeta, että vuoden 2004 loppuun mennessä digitaalisia kameroita samalla resoluutiolla oli saatavana sisäänrakennettuina lukuisten valmistajien matkapuhelimiin.

Vaikka ePhoto 1280 on teknisesti jäänyt kehityksen jalkoihin, on se kuitenkin kelpo kamera. Sillä saa hyvissä valaistusolosuhteissa mukiinmeneviä kuvia. Osa päättötyön kirjallisen osuuden kuvituksesta on otettu tällä samaisella kameralla. Ongelmia kamerassa ovat surkea hämäränsieto, huima akun-/patterinkulutus ja suuri koko. Suuri koko ei ole eduksi, mutta tukevasti rakennetulla mittalaitteella voidaan koon haitat minimoida.

Fyysiset ominaisuudet:

Mitat:	51 x 156 x 92 mm
Massa:	380 g (ilman pattereita)

Optiset ominaisuudet:

Valovoima (Lens Aperture)	f/2.8 (wide) – 3.5 (telephoto)
Focal Range	10 cm – infinity (wide) 80 cm – infinity (telephoto)
Macro Focal Distance	10 cm – 1 m (wide) 50 cm – 1 m (telephoto)
Polttoväli (Focal Length)	38 – 114 mm (3x zoom) (35 mm equivalent)
Shutter Speed	1/4 – 1/500 seconds, automatic
ISO-luokitus	ISO 100
CCD- kenno (Image Sensor)	1/3" IT-CCD Yellow/Cyan/Magenta/Green
Etsin	Ei etsintä
Flash Range — Normal:	20 cm – 2.6 m, Macro: 40 – 75 cm
4 modes:	AUTO, FILL-IN, REDEYE, OFF
GN:	7.8
LCD-näyttö	Low-temperature 2" poly-silicon TFT LCD, 110,000 pixels

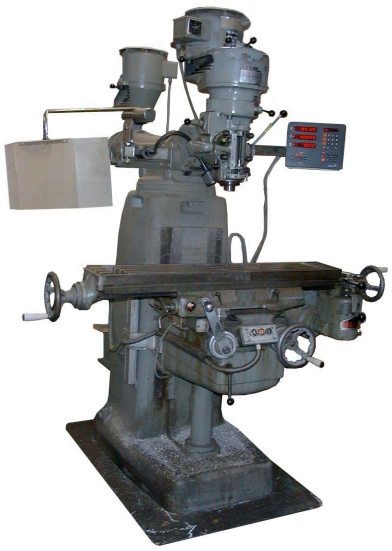
Tärkeitä tietoja etäisyysmittalaitteiston toiminnan kannalta ovat kameran polttoväli (Focal Length) sekä filmikoko (vastaa 35 mm). Mekaanisen osuuden osalta tärkeää on puolestaan kameran mitat ja paino.

Työkalut

Suunnittelua pidetään usein ikävänä ja mielenkiinnottomana osana työntekoa. Usein kiirehditään asioiden edelle ja aletaan toteuttaa puolivalmiita suunnitelmia. Joskus tosin käy päinvastoin. Suuret suunnitelmat mitätöityvät, kun toteutusvaiheessa huomataan suunniteltu ratkaisu mahdottomaksi. Työkalut ovat tärkeä osa projektin toteuttamista, ei vain rakennusvaiheessa vaan jo suunnittelun yhteydessä. Työkalujen ja mahdollisten työstömenetelmien tunteminen auttavat tekemään täsmällisiä, toimivia ja ennen kaikkea realistisia suunnitelmia.

Harvalla on kotonaan kaikkia projektissa tarvittuja työkaluja, kuten sorveja, pylväsporaa (kuvassa) tai jyrsintä. Projektin työstäminen tapahtui Åbo Akademin kiihdytinlaboratorion verstaassa, jossa tarvittava työkaluarsenaali oli edustettuna. Työskentely tapahtui ilta-aikaan koneiden seistessä jouten. Koneiden käyttö ei ollut aivan yksinkertaista, sillä edes ohjaajallani ei ollut kokemusta kaikista työkoneista. Yrityksen ja erehdyksen – kuitenkin lähinnä yrityksen – kautta saimme kuitenkin yhdessä koneet tottelemaan haluamallamme tavalla.

Työstäminen sujui hyvin muutamaa konetta käyttämällä ja yhdistelemällä erilaisia teriä koneisiin. Tärkeimmät yksittäiset koneet olivat sorvi ja jyrsin. **Sorvissa** kappale pyörii ja terä pysyy paikallaan. Näin erilaisten pyöreiden ja säännöllisten kappaleiden tekeminen on yksinkertaista. Esimerkiksi lähes kaikki akseliin liitettävät osat toteutettiin sorvissa. **Jyrsin** (kuvassa) puolestaan antaa työskentelylle toisen lähtökohdan. Jyrsimessä kappale pysyy paikallaan ja terä pyörii. Jyrsimen avulla voikin porata reikiä ja – kuten nimi kertoo – jyrsiä. Jyrsimällä voi saada aikaan erilaisia suoria pintoja ja tasata kappaleiden reunoja.



Sorvin ja jyrsimen avulla saa aikaan ihmeitä. Työskentelytarkkuutena käytettiin sadasosamillimetrin tarkkuutta, johon päästiin laitteisiin kytkettyjen digitaalisten mittausjärjestelmien avulla. Toisin kuin puuta työstettäessä on metallityön yhteydessä äärimmäisen tärkeää saada reiät ynnä muut liitokset sopimaan juuri eikä melkein. Puu antaa aina hieman periksi, mutta metalli ei juuri anna mittavirheille armoa.

Työstökoneiden teriä suojattiin suihkuttamalla niille jatkuvasti denaturoitua etanolia (C_2H_5OH). Tämä esti alumiinin tarttumisen poran- ja jyrsinteriin pidentäen niiden käyttöikää.

Sorvia ja jyrsintä käytettiin etupäässä tarkkaan työskentelyyn. Karkeat sahaukset tehtiin vannesahalla ja eräänlaisella sahauskoneella. **Vannesahassa** terä on pitkä vannemainen metallikappale, joka toimii sahana kiertäessään ympäri suurella nopeudella. **'Sahakone'** puolestaan toimi pikemminkin tavallisen käsisahan tavoin. Laite vaikutti jonkinlaiselta mekaaniselta riemuvoitolta, sillä se näytti lähinnä käsisahan ja höyryveturin risteytykseltä.

Myös pylväsporakone oli ahkerassa käytössä. Lisäksi erilaisia kierretappeja (jenkatappi) käytettiin ahkeraan. Tavallinen jokamiehen työkalupatteristo – ruuvimeisselit, pihdit ym. – oli luonnollisesti myös käytettävissä. Verstaassa oli mukana myös kannettava tietokone, jolta CAD-piirustuksista saattoi hakea apua vitaalin mitan unohduttua.

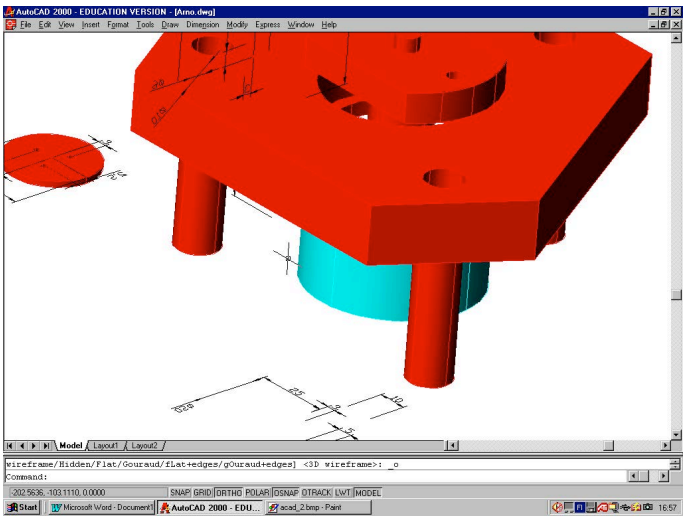
Piirustukset

Suunnitteluun kului runsaasti aikaa, mutta hyvien suunnitelmien tekeminen ennen työkaluihin tarttumista taisi kuitenkin maksaa itsensä takaisin. Varsinkin tarkkuutta vaativissa tai monimutkaisissa projekteissa suunnittelu näyttelee tärkeää osaa. Projektin luonteen huomioiden pidin tärkeänä saada piirustukset tehtyä. Toisaalta myös ohjaajani oli, järjestelmällisenä miehenä, hyvien piirustusten ja tarkan suunnittelun kannalla.

Suunnitteluvaiheessa päätettiin rakennelman pääasialliseksi materiaaliksi alumiini (Al). Tähän päätökseen päädyttiin lähinnä siksi, että alumiini on kevyttä ja sitä on suhteellisen helppo työstää. Lisäksi ohjaajallani oli kokemusta alumiinin kanssa työskentelystä jo ennalta. Alumiinia oli tarjolla erilaisina hukkapaloina, joita saattoi hyödyntää laitetta rakentaessa. Materiaalin lisäksi työstömenetelmät (työkalut) ja rakennelman keskeiset osat määräsivät suunnittelua.

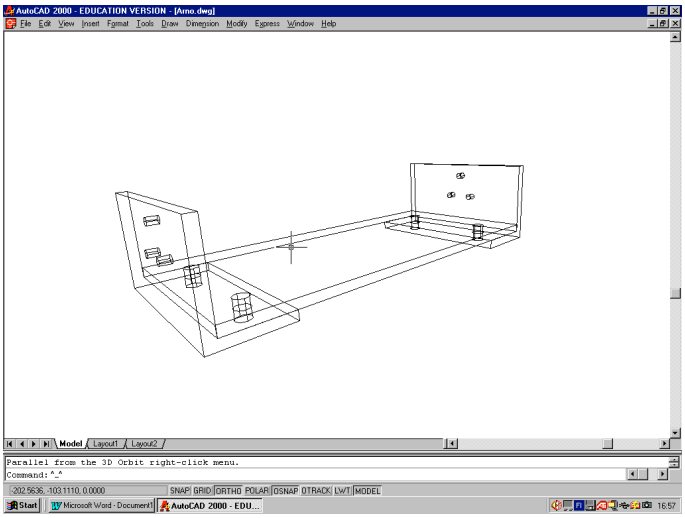
AutoCAD

Laitteen piirustukset toteutettiin Autodeskin AutoCAD-ohjelman avulla. CAD-suunnittelulla on pitkät perinteet ja AutoCAD on yksi käytetyimpiä ohjelmia alalla. AutoCAD-ohjelman historia juontaa juurensa aina 1980-luvun alkuun, jolloin ohjelman ensimmäinen versio tuotiin markkinoille. Tämä Autodesk-yhtiön tuote on laajalti käytetty kaikenlaisen suunnittelun ja piirustusten tekemisen. Autodesk kuuluukin maailman suurimpien 3D-ohjelmistotalojen joukkoon. Sen tuotteisiin kuuluu myös yleisesti käytetty 3ds Max -ohjelma.



Lyhenne CAD tulee englanninkielisistä sanoista Computer-aided design. Muita käytettyjä selityksiä nimelle ovat Computer-assisted design sekä Computer-aided drafting. Joka tapauksessa CAD-sanalla viitataan tietokoneella tapahtuvaan suunnitteluun. Käyttämäni CAD-ohjelma oli AutoCAD 2000. Sen perustoiminnot ja tavallisimmat käskyt tulivat vähitellen tutuiksi. Ohjelman käyttöä hankaloitti tottumattomuus grafiikan piirtoon kirjoitettujen komentojen kautta. Helpottavana tekijänä tosin oli aiempi kokemus 3D-ohjelmista.

AutoCad-ohjelmalla tekniset piirustukset sai toteutettua kolmiulotteisina. Moniulotteisuus helpotti hahmottamaan tiettyjä ongelmakohtia ja luomaan kokonaiskuvaa laitteen rakenteesta. Jo suunnitteluvaiheessa pyrittiin saamaan kaikki mitat mahdollisimman tarkasti kohdalleen ja päättämään osien rakenteesta ja muodoista. Koko rakennelma itse asiassa muotoutui hiljalleen tietokoneen ruudulle suorista viivoista ja ympyröistä. Osien toiminnasta, muodoista ja koosta päätettiin sitä mukaa, kun ruudulla liikkuva häkkyrä muuttui yhä monimutkaisemmaksi.



AutoCad-ohjelma.

Osat ja niiden toteutus

Saan kiittää ohjaajaani Stefan Johanssonia monista teknisistä ratkaisuista. Suunnittelun alkaessa laitteen rakennetta määrääviä osia olivat moottorit, kamera ja laser, joiden massa ja mitat tiedettiin. Suurikokoinen kamera vaati tukevan kiinnityksen, joten kameran kiinnityksestä lähdettiin liikkeelle. Kameran alusta päätettiin kiinnittää halkaisijaltaan 5 millimetrin teräsakseliin. Akseli laakeroitaisiin ja siihen kiinnittyisi myös laser. Akselia liikuttaisi moottori. Sopivat laakerit löytyivät lyhyen etsimisen jälkeen ja varsinainen tarkka suunnittelu saattoi alkaa.

Piirustuksia tehtäessä lähdimme ohjaajani kanssa liikkeelle moottorin ja vaihteiston liittämisestä kameran ympärille rakennettavan telineen akseliin. Tarkoitus oli ensiksi suunnitella vaikein osuus alta pois, toteuttaa se ja sitten palata jälleen suunnittelupöydän ääreen pohtimaan koko hökötyksen kääntymistä vaakatasossa.

Kameraa kannattelevan telineen suunnittelussa oli tärkeää huomioida painon jakaantuminen. Näin saataisiin moottori vaikeuksitta ohjaamaan melko raskasta rakennelmaa. Kameran telineen vastakkaiselle puolelle tuli myös suunnitella vastaava akselinpätkä, joka liittäisi kameraa kannattelevan osan kuulalaakeriin. Pystytason kääntyminen laakeroitiin kolmella pienellä kuulalaakerilla, joiden läpi halkaisijaltaan 5 millimetrin akseli kulki. Laitteen runkona päätettiin käyttää valmiita alumiiniprofiilien kappaleita, joita sattui löytymään ja jotka ovat painoonsa nähden huomattavan vahvoja. Lisäksi ne pystyttiin liittämään yksinkertaisella tavalla toisiinsa erityisten liitososien avulla.

Laserin pidike suunniteltiin vielä lopuksi ennen verstaaseen siirtymistä. Pidikkeessä oli useita kriteerejä. Se piti ensinäkin saada tukevasti kiinnitettyä akseliin. Toiseksi laserin kulmaa piti pystyä säätämään ja lisäksi vielä koko rakennelman piti olla sopivan kevyt.

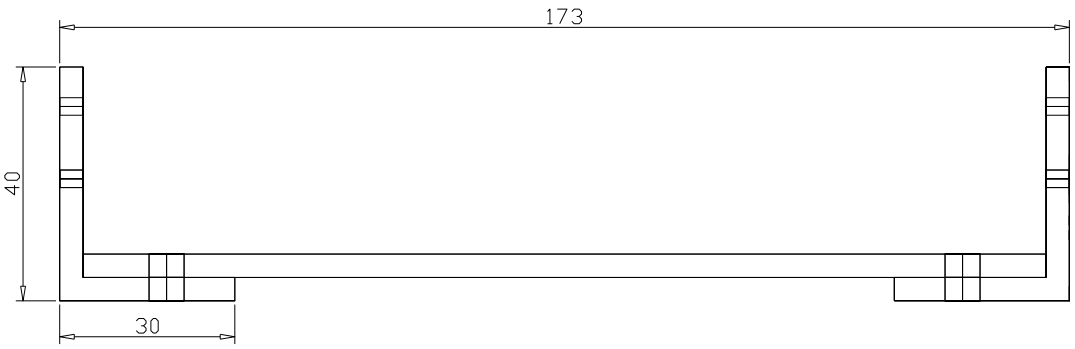
Kun laiteen yläosa oli menestyksekkäästi toteutettu, oli aika palata suunnittelupöydän ääreen. Kovasti viisastuneina (ainakin allekirjoittanut) saatoimme syventyä uuteen ongelmaan: miten saada koko laitteen yläosaa ohjattua? Ratkaisu löytyi tukevasta alumiinialustasta, suuresta kuulalaakerista ja mekanismista, jolla laitteen yläosa saatiin liitettyä hammasrattaaseen.

Olen koonnut tietoa laitteen osista, niiden toteutuksesta sekä erilaisista ongelmista ja ongelmien ratkaisuista tähän. Kuvat piirustuksista eivät aina vastaa valmista mittalaitetta, mutta ovat kaiken kaikkiaan niin tarkkoja kuin suinkin piirustuksilta odotetaan. Lukijan kannattaa huomioida joitakin seikkoja kuvista:

1. Kuvat eivät ole keskenään mittakaavassa
2. Kaikki mittayksiköt ovat millimetrejä (mm)
3. Värityksen tarkoitus on ainoastaan helpottaa eri kappaleiden erottamista toisistaan
4. Osien nimeäminen noudattelee alkuperäisiä nimiä.
5. Merkinnät Right, Left, Front jne. tarkoittavat kuvakulmaa.

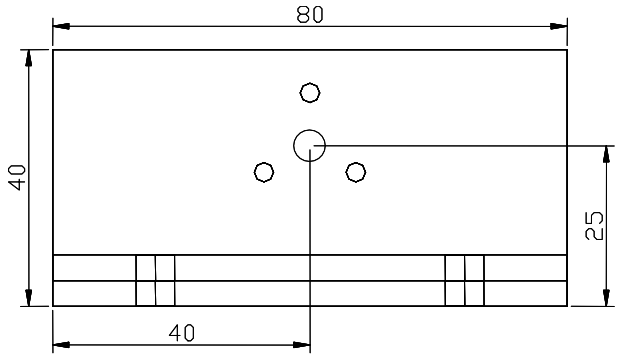
Kameran alusta (L-muoto)

[L, Front]



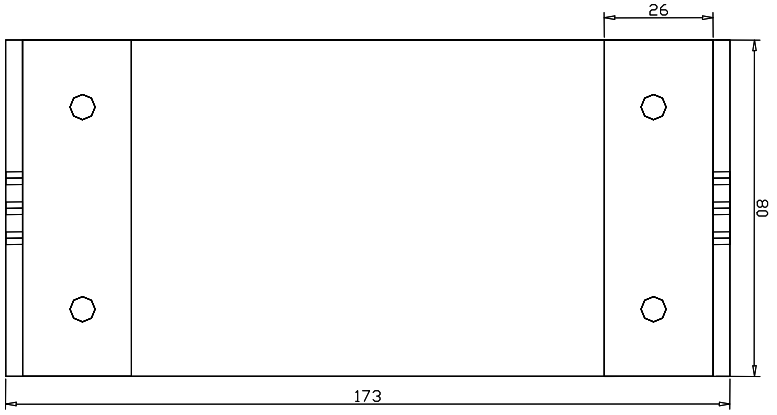
Ensimmäinen versio kameran alustaksi oli yksinkertainen alumiinilevystä taiteltu tuki. Alustavissa testeissä kävi kuitenkin ilmi, että olisi miltei mahdotonta saada akseli kohtisuoraan taitellun alustan kummallekin sivulle. Tämän ongelman vuoksi lopullisessa versiossa päädyttiin käyttämään kahta L-palkkia, joiden väliin kiinnitettiin alumiinilevy. L-palkki on valmista 4 mm paksua palkkia, jonka mitat ovat 30 mm x 40 mm.

[L, Right/Left]



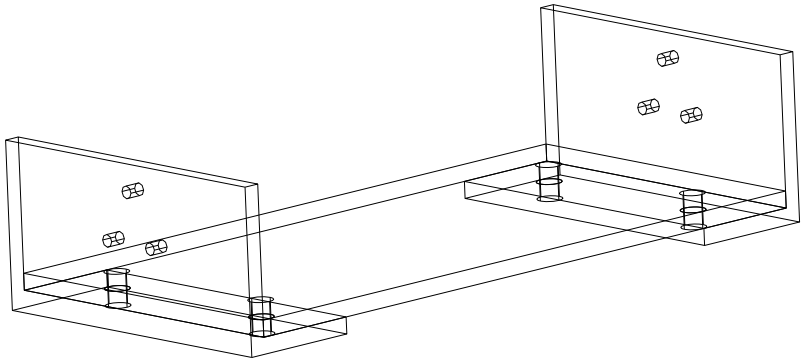
Kameran teline on kiinnitetty kummastakin päästään akseliin. Akselin suoruus on varmistettu tekemällä kummankin pään L-osista identtisiä. Akselin kiinnitysmekanismi perustuu sorvattuun pyöreään kappaleeseen, joka kiinnitetään kolmella ruuvilla kiinni L-palkkiin. 5 mm:n akseli kulkee myös L-palkin läpi (hieman suurempi keskimmäinen reikä). Tämä ohjaa akselin oikeaan paikkaan ja auttaa näin reikiä osumaan oikein kiinnityskappaleeseen. Lisäksi akselin läpivienti lisää tukevuutta.

[L, Top]



Kamera kiinnitetään kuvassa (keskellä) näkyvään alumiinilevyyn. Kiinnityksessä voidaan hyödyntää kameran omaa kiinnitystä, minkä lisäksi kamera täytyy tukea levyyn. Levy kiinnitetään L-palkkeihin kahdella 5 mm:n pultilla kummallakin puolella. Tätä varten kumpaankin L-palkkiin porattiin 5 mm:n kierteelliset reiät. Näiden reikien paikka on piirustuksissa viitteellinen, sillä kameran kiinnityskohdan valinta vaikutti ratkaisevasti reikien paikkaan.

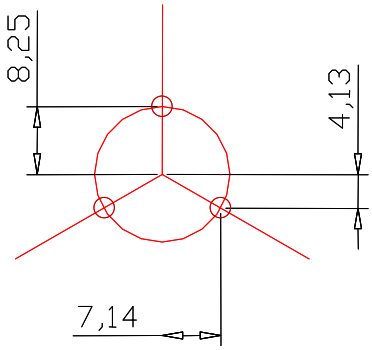
[L, 3D]



Kolmiulotteinen kuva näyttää miltä kameran kiinnitysalusta näyttää. Kuvassa ei ole perspektiivikorjausta, joten osat näyttävät oikean kokoisilta suhteessa toisiinsa.

L-palkit sahattiin suurin piirtein oikean kokoisiksi vannesahalla, minkä jälkeen tarkka mittoihin sovittaminen tapahtui jyrsimellä. Samoin tehtiin pohjan muodostavalle alumiinilevylle. Tarkkoja mittoja vaativat kolmiomuodostelmassa olevat reiät tehtiin niin ikään jyrsimellä. Levyt toisiinsa kiinnittävät reikäparit porattiin terällä, jonka läpimitta oli 5 mm.

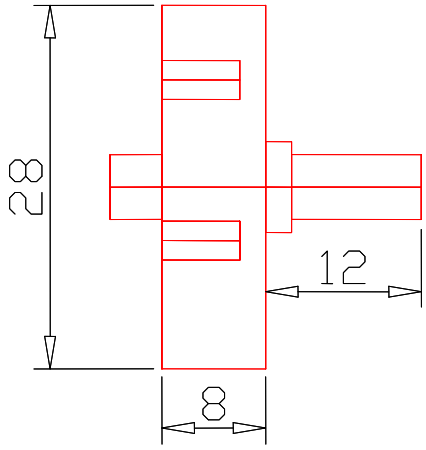
[Mitat, Right]



Kolmiomuodostelmassa olevat 3 mm:n reiät on porattu ympyrän kaarelle niin, että jokaisen reiän väliin jää 120°. Samaa periaatetta on käytetty monessa kohdassa työtä, joten reikien väliset mitat esitetään erikseen, jotta turhaa samojen mittojen toistoa vältettäisiin. Oheinen kuva esittää, hieman pyöristetyt, eri kohtien väliset mitat. Nämä mitat olivat työstövaiheessa erityisen tärkeitä saada kohdalleen, sillä pienikin heitto kolmen reiän ryhmässä olisi tehnyt osien liittämisen yhteen mahdottomaksi.

Päätykiinnitys (End)

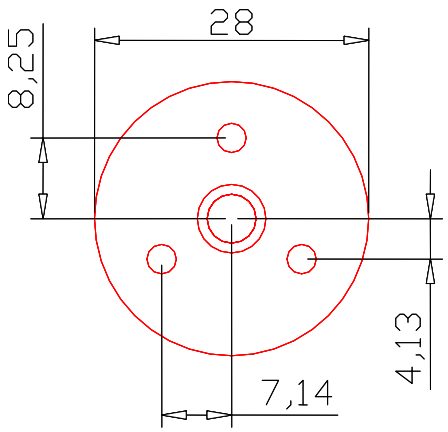
[End, Front]



Oheinen kuva esittää yhdestä ainoasta kappaleesta koostuvaa osaa, jonka avulla kameran teline saadaan liitettyä laakeroituun kiinnekohtaan. Kappaleen mitat olivat suunnitteluvaiheessa lähinnä viitteellisiä, sillä kyseessä oli tyypillisesti sellainen kappale, jonka mitoista saattoi tarpeen mukaan joustaa. Työtä tehdessä kappaleen halkaisija väheni millimetrillä ja osa ulokkeista muutti muotoaan.

Kuvassa vasemmalla oleva kiinnityksen ohjaukseen tarkoitettu pieni akselinpätkä (Ø 5 mm) jätettiin pois. Oikeanpuoleisen akselin pituutta ei tarkasti määrätty, vaan riitti, että se tukevasti asettui kuulalaakeriin.

[End, Right]

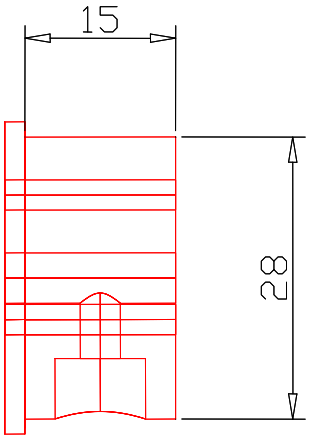


Kuva esittää samaa kappaletta nähtynä sivulta. Koska kiinnitys tapahtuu kolmen 3 mm ruuvin avulla L-palkkiin, on kummassakin kappaleessa kolmiomuodostelmassa olevien reikien oltava aivan kohdakkain. Ohessa on vielä erikseen esitetty reikien tarkat koordinaatit keskiakseliin nähden. Kiinnitystä varten kappaleen reikiin tehtiin kierretapilla kierteet.

Kappaleen työstö aloitettiin alumiinitangosta, josta sorvaamalla saatiin halkaisijaltaan 5 mm:n akselinpätkä esiin. Reiät porattiin jyrsimellä.

Kääntymismekanismi (Vertical)

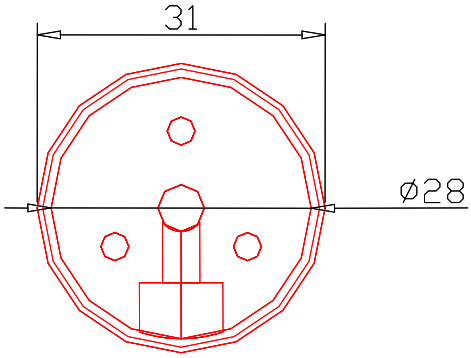
[Vertical, Front]



Yksi työn haastavimpia mekaanisia kohtia oli saada moottori, hammaspyörät ja itse kamera liikkumaan oikein. Haastetta tuotti hammaspyörän liittäminen akseliin. Liimaus ei olisi toiminut alkuunkaan vaan vaadittiin tukevaa ja luotettavaa ratkaisua. Messinkinen hammaspyörä päätettiin kiinnittää kolmella ruuvilla alumiinista sorvattuun tukeen, joka kiinnitettäisiin sekä akseliin että kameran telineeseen.

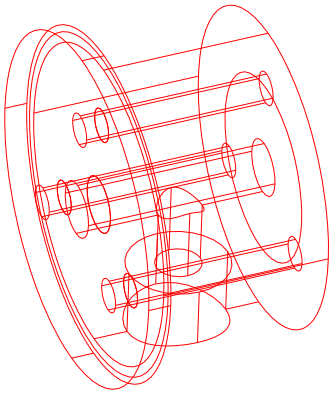
Oheisessa kuvassa on esitetty miltä tämä tuki näyttää. Laitimmaisena vasemmalla on piirretty näkyviin myös hammaspyörä (noin Ø 31 mm x 2 mm). Kiinnitys hammaspyörään (vas.) ja kameran telineeseen (oik.) hoidettiin kolmiomuodostelmassa olevilla 3 mm:n ruuveilla. Reikien paikat noudattavat edellisten osien yhteydessä esitettyjä mittoja.

[Vertical, Right]

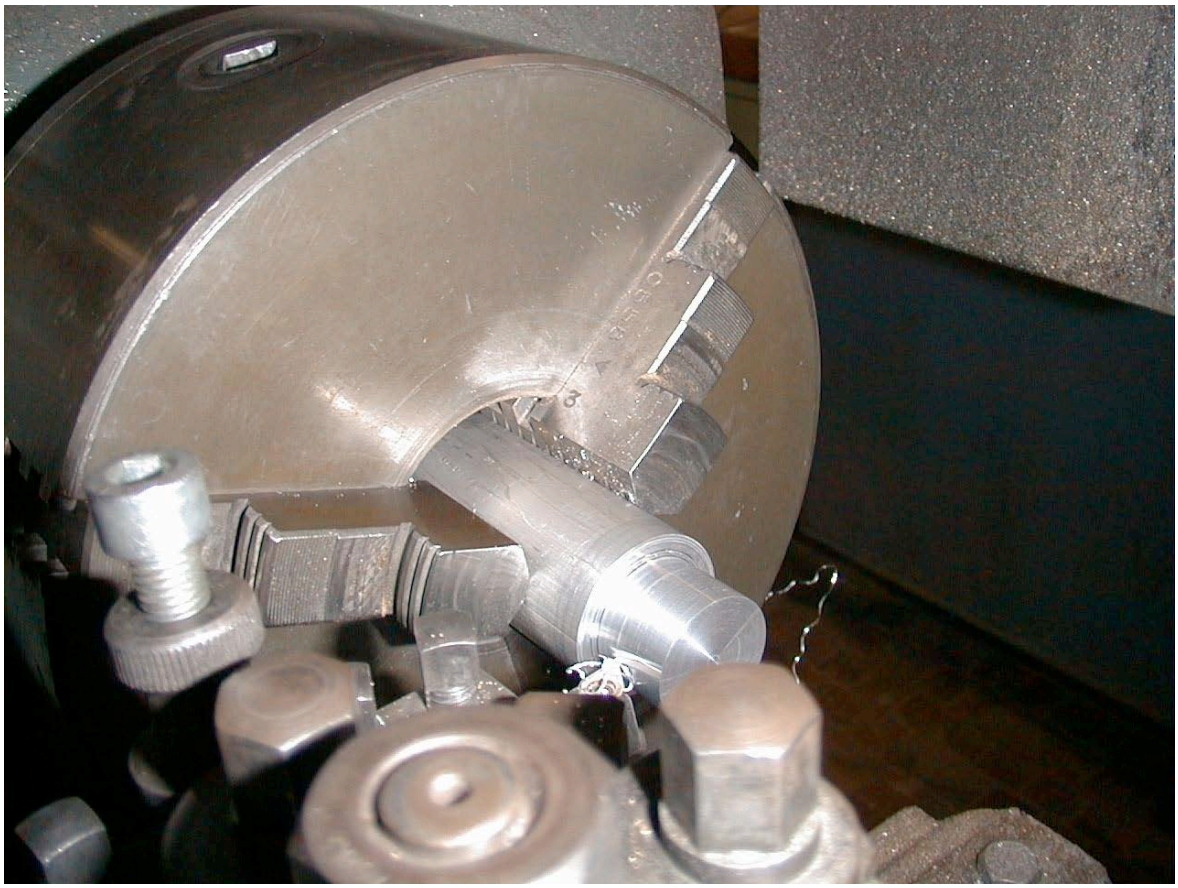


Kappaleen keskellä kulkeva akseli oli tarkoitus kiristää paikalleen kiristysruuvilla (alhaalla). Tätä ruuvia varten oli tarkoitus porata akselin suhteen kohtisuorassa oleva reikä kappaleen keskelle, tehdä reikään kierteet ja kiristää akseli paikalleen sopivalla ruuvilla. Tätä järjestelyä ei loppujen lopuksi tarvittu, sillä akselin kulkiessa kolmen kappaleen läpi kiristyi se pienistä epätarkkuuksista johtuen todella napakasti paikalleen.

[Vertical, 3D]



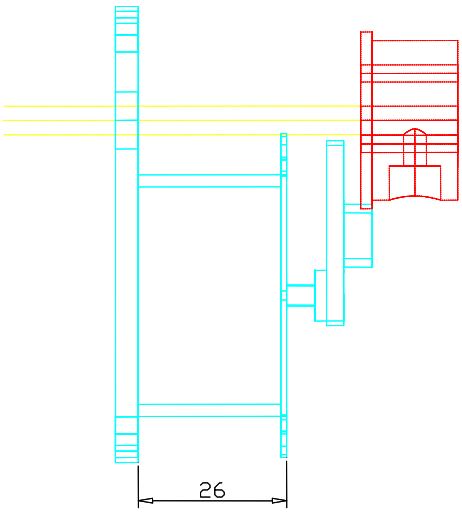
Kappale toteutettiin sorvaamalla. Keskireikä porattiin jo sorvissa ja loput reiät jyrsimessä. Hammasrattaan reikien poraaminen oli aiheuttaa harmaita hiuksia, sillä niiden oli osuttava tarkasti paikalleen. Yksi hammasratas jopa tuhoutui poraamisen yhteydessä hampaiden muututtua sirkkelinterän muotoisiksi. Ongelma ratkaistiin kiinnittämällä hammasratas sorvattuun kappaleeseen ja poraamalla reiät vasta sen jälkeen. Sorvatun kappaleen läpi kulkeviin reikiin tehtiin kierretapilla kierteet. Näin saatiin kiinnitys hoidettua sopivilla 3 mm:n ruuveilla.



Alumiinikappale muotoutumassa sorvissa.

Moottori (Motor)

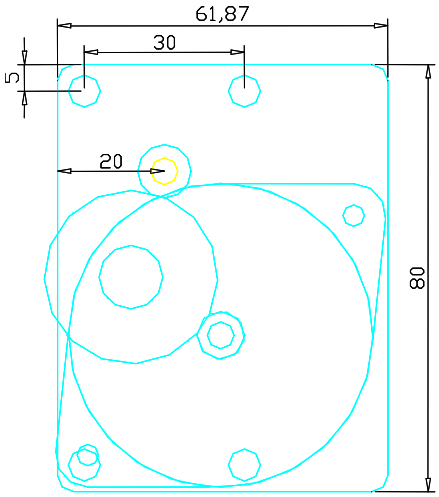
[Motor, Front]



Pelkäsin kovasti moottorin painon (n. 300 g) tuottavan ongelmia laitteen rakenteen suhteen, mutta tämä ongelma ratkesi sijoittamalla moottori mahdollisimman lähelle akselin keskikohtaa. Tälle asetti tiettyjä rajoitteita rakennelmaa tukevat valmiit alumiinirungot, joiden korkeus on 80 mm. Tästä johtuen moottori ei mahtunut helposti akselin kanssa limittäin. Näistä pienistä yhteensovitusvaikeuksista johtuen päätettiin moottorin tarkasta sijainnista ja kiinnityskohdasta vasta kokoamisen edettyä kiinnitysvaiheeseen. Piirustuksissa sijainti ja kiinnitys ovat vain viitteellisesti esitettyinä.

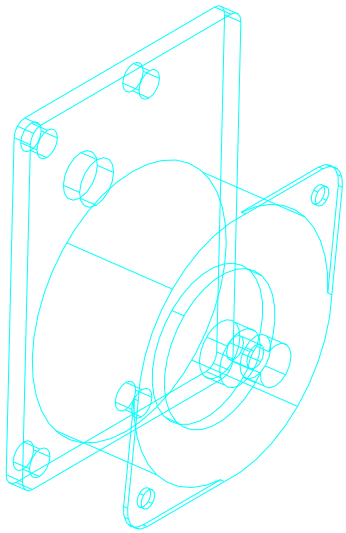
Edellä olevaan kuvaan on jätetty esiin kameran alustan kiinnitysmekanismi (katkoviivoin) kokonaisuuden selkeyttämiseksi. Moottorin mitoista ja hammasrattaan hampaista kerrotaan enemmän muualla. Moottori kiinnitettiin paksuun alumiinilevyyn (4 mm), joka puolestaan kiinnitettiin alumiinirunkoon ruuveilla (6 mm).

[Motor, Right]



Moottorin kiinnitykseen käytetyn alumiinilevyn mitat määräytyivät moottorin asennon mukaan ja myöhemmin päädyttiin hieman kapeampaan levyyn. Levyyn oli porattava läpimeno akselille kohtaan (20 mm, 20 mm) vasemmasta yläkulmasta.

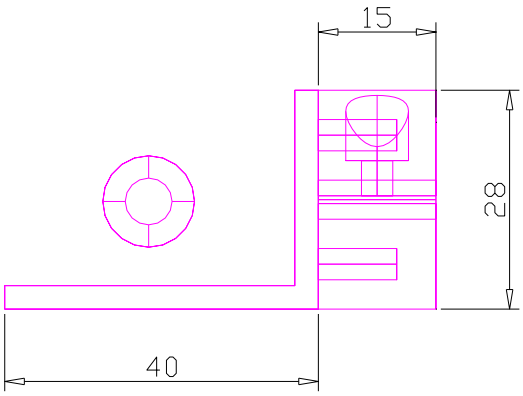
[Motor, 3D]



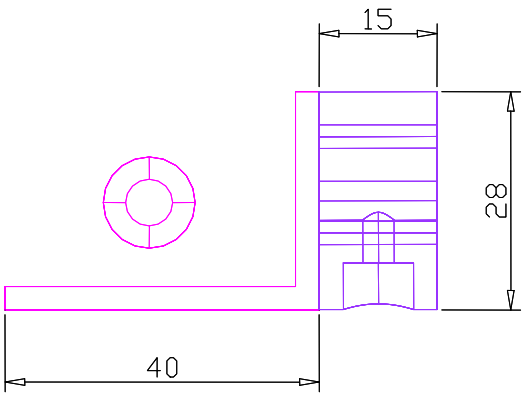
Moottorin kiinnityksessä käytetty alumiinilevy muokattiin mittojen mukaiseksi jyrsimellä. Myös reiät porattiin jyrsintä käyttäen. Moottorin kiinnityskohtia jouduttiin hieman muuttamaan lisättäessä halkaisijaltaan 10 mm tuet kiinnitysruuveille (ei näkyvissä piirustuksissa).

Laserin kiinnitys (Laser)

[LaserA, Front]



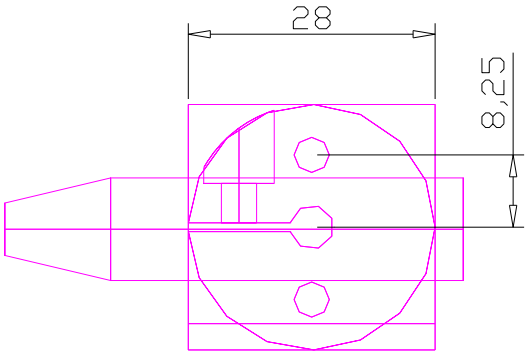
[LaserB, Front]



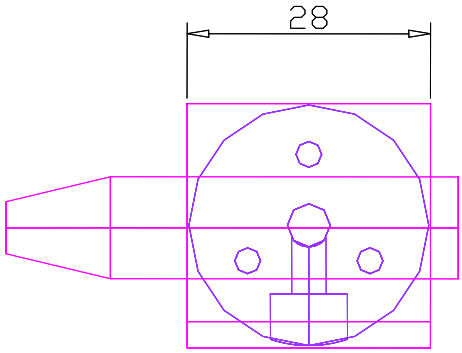
Laserin kiinnittämiskeksi akseliin suunniteltiin alunperin hieman muista kiinnityksistä eroava mekanismi (kuva A). Lopulta toteuttamisvaiheessa suunnitelmia muutettiin ja kiinnitys toteutettiin kuvan B mukaisesti. Tässä on nyt esitetty molemmat vaihtoehtoiset tavat.

Pyöreä kappale, joka kiinnitetään akseliin, noudattaa hyvin pitkälti samaa periaatetta kuin kameratelineeseen kiinnitetty vasemmanpuoleinen kappale. Laser puolestaan on kiinnitetty L-palkkiin (40 mm x 30 mm). Laseria pitelevää kappaletta ei ole merkitty näkyviin. Sen sijaan kuvassa on laseria esittävä sylinterimäinen hahmotelma.

[LaserA, Right]



[LaserB, Right]

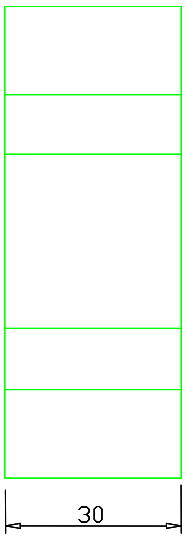


Kuvan A kiristysmekanismi perustuu alumiinin hienoiseen periksiantamiseen. Kuvan B periaate perustuu kiristysruuviin. Kuvassa B pyöreä kappale kiinnitetään L-palkkiin kolmen 3 mm ruuvin avulla samalla periaatteella kuin edellisten osien yhteydessä on selitetty.

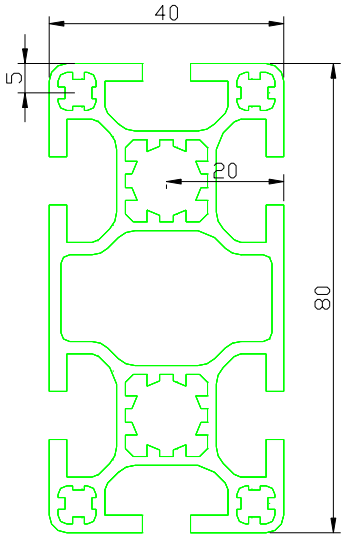
Pyöreä kappale sorvattiin ensin muotoonsa ja reiät porattiin jyrsimellä. L-palkki puolestaan jyrsittiin sahauksen jälkeen siistiksi, minkä jälkeen reiät porattiin siihenkin kohdalleen. Laserille tehtiin myöhemmin kuutiomainen pidike, jonka läpi oli porattu sopiva reikä laser-osoittimelle. Kahdella kiristysruuvilla laser kiinnitettiin alumiinikuutioon ja kuutio palkkiin. Laserin eristys (sähköisesti) muusta laitteesta tehtiin kumimaton ja muoviruuvin avulla (katso elektroniikka).

Alumiinirunko (Profil)

[Profil, Front]



[Profil, Right]



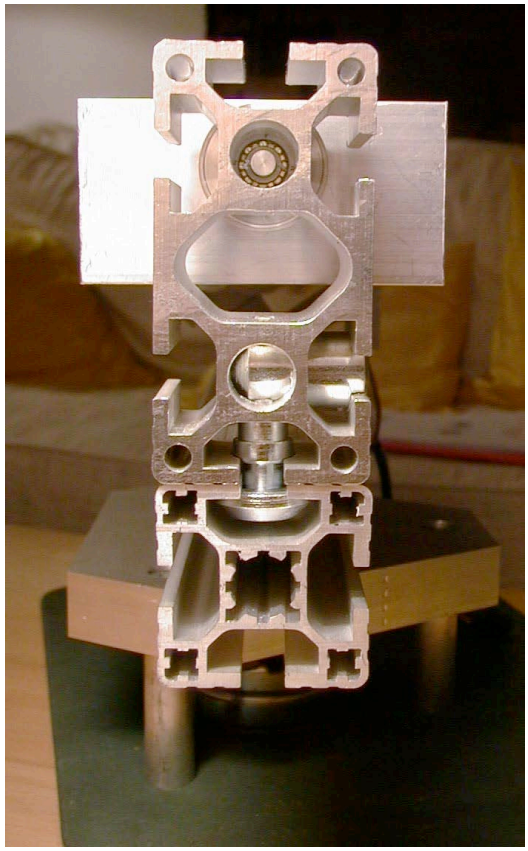
Mittalaitteen koko yläosa on tuettu kahden alumiinirungon varaan. Runko on Maytec-mallista profiilia, joka on tarkoitettu juuri tämäntapaisten kehikkojen kokoamiseen. Monet seikat puhuivat näiden valmiiden kehikkojen käyttämisen puolesta. Ne olivat muihin vaihtoehtoihin nähden tukevia ja niiden kiinnittäminen helppoa. Lisäksi akselin laakerointi voitiin helposti hoitaa upottamalla pieni kuulalaakeri kehikossa valmiiksi olevaan syvennykseen.

Laakerointi tehtiin kolmella samanlaisella kuulalaakerilla. Laitteen päässä olevaan kehikkoon upotettiin niitä yksi ja moottorikehikkoon kaksi, sillä näin varmistettiin akselin kulku suoraan kehikon läpi.

Lisätukevuuden saavuttamiseksi päädyttiin käyttämään piirustuksissa näkyvän kehikkomallin sijaan hieman tukevampaa kyseisen kehikon isoveljeä. Käytetyt kehikot ovat kaikin tavoin mitoiltaan identtisiä kuvan kehikon kanssa paitsi rakenteiden paksuuden osalta. Syy vaihtoon oli itse asiassa se, ettei heppoiseen malliin saanut kunnon kierteitä tehtyä. Kierteitä tarvittiin moottorin tukilevyn kiinnittämiseksi kehikkoon.

Kehikot sahattiin suurin piirtein oikea mittaisiksi ”sahakoneella” (katkaisusaha), minkä jälkeen ne jyrsittiin tarkkaan leveyteensä. Laakereiden upotus tehtiin laajentamalla kehikon reikien halkaisijaa vastaamaan laakereiden tarkkaa halkaisijaa (13 mm).

Koko yläosan rakennelma on kiinnitetty palkkiin (40 mm x 40 mm x 300 mm), joka on samaa sarjaa kuin muutkin käytetyt valmiit elementtiosat. Kehikon, jonka varassa akseli lepää, kiinnittäminen poikittaiseen palkkiin tehtiin käyttämällä juuri tälle elementtisarjalle tarkoitettua kiinnitysmekanismia. Kehikkoihin porattiin kaksi reikää, joista toista käytettiin itse kiinnitykseen ja toinen toimi kiinnityksen kiristämisessä apuna. Poikittaisen palkin uraan asetettu kiinnitin kiristettiin paikalleen vastakappaleen avulla (katso kuva).



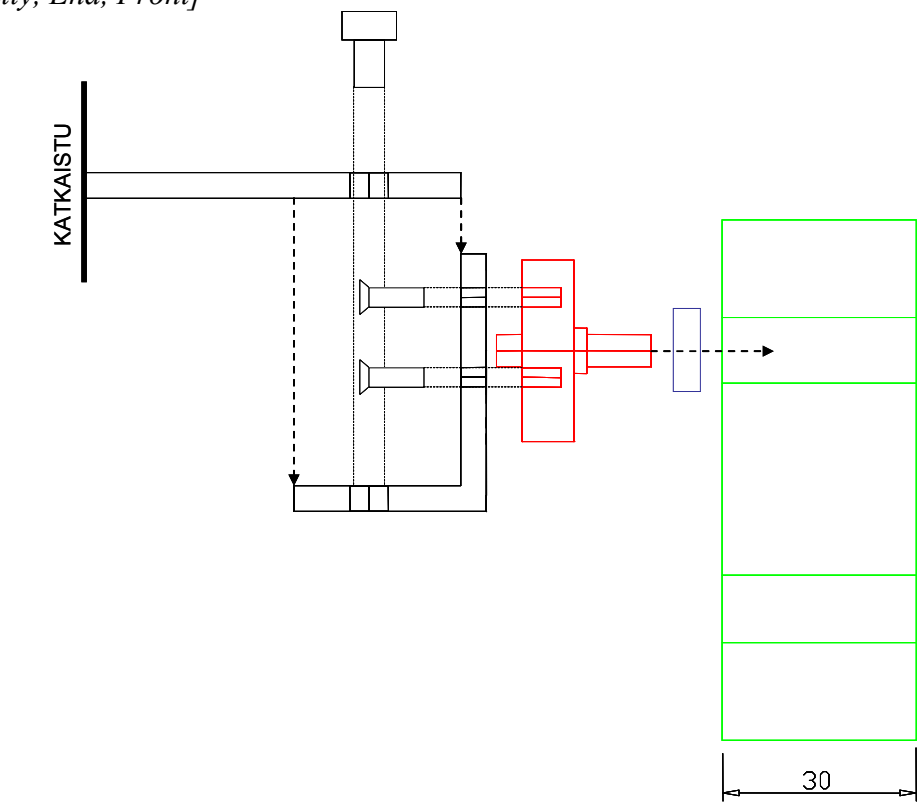
Kuvassa näkyy akselin pään kiinnitys kuulalaakeriin sekä kehikkojen välinen liitos.

Alumiinikehikkoihin oli mahdollisuus laittaa muovisuojukset viimeistelemään rakennelman ulkoasu. Muovisuojukset kiinnitettiin mittalaitteen kumpaankin päähän viimeistelyvaiheessa.

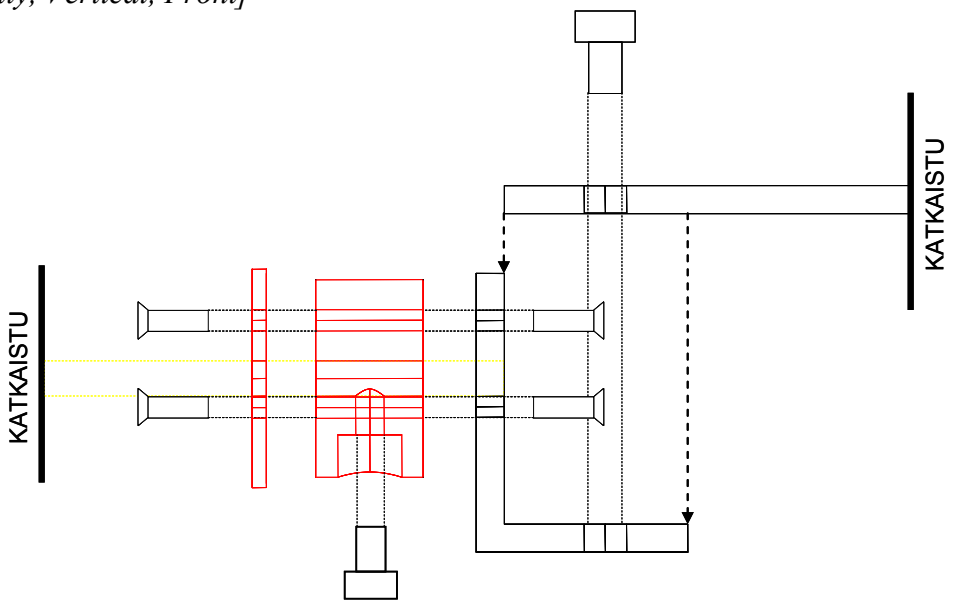
Yläosan kokoaminen

Jottei piirustuksissa esitetyt osat jäisi vain irrallisiksi paloiksi, olen esittänyt osien väliset kiinnitykset seuraavissa kuvissa. Kuvissa näkyvät ruuvit eivät kaikki aivan vastaa todellisuutta. Kuvien tarkoitus ei ole olla eksakteja vaan auttaa hahmottamaan kokonaisuutta. Koska koko rakennelma ei mahdu yhteen kuvaan, olen pätkinyt sen osiin.

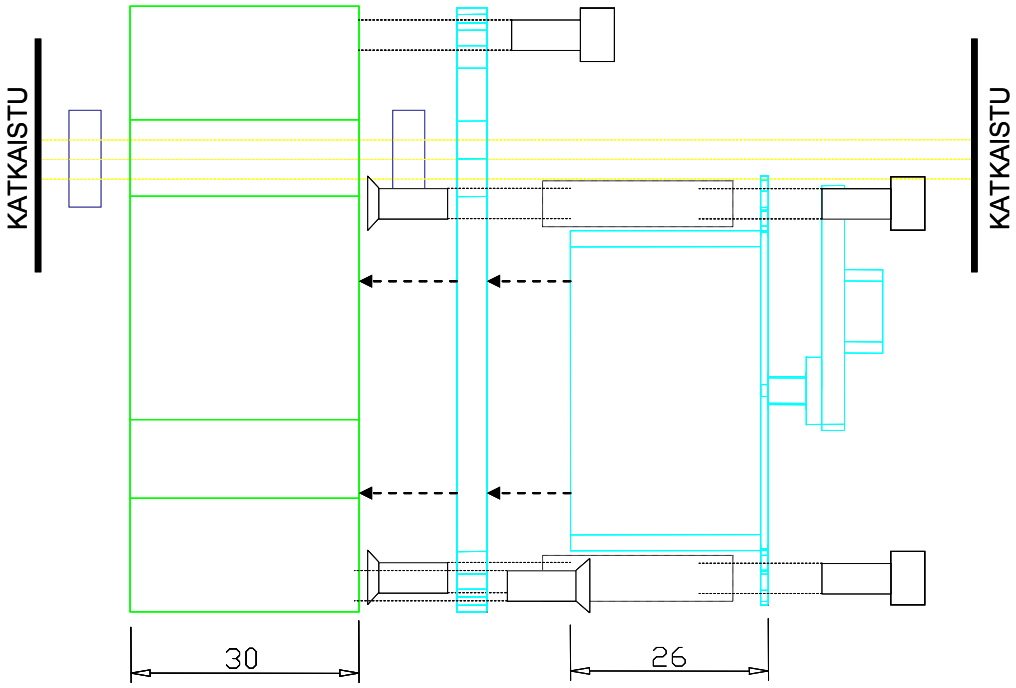
[Yhdistetty, End, Front]



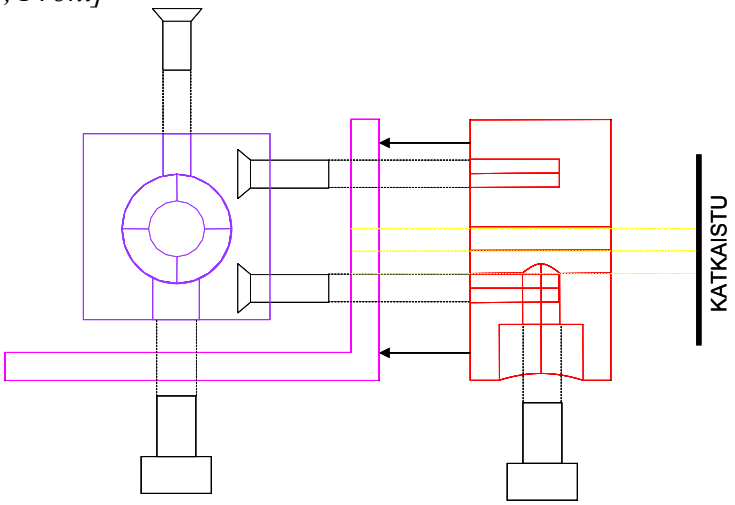
[Yhdistetty, Vertical, Front]



[Yhdistetty, Motor, Front]



[Yhdisetty, Laser, Front]



Alaosa

Mittalaitteen yläosan valmistuttua oli aika kokeilla moottorin toimimista. Moottori jaksoi hyvin kääntää kameraa pystysuunnassa ja järjestelmä toimi hyvin yhdistettynä piirilevyn kautta tietokoneeseen. Hiljalleen tuli kuitenkin aika palata suunnittelupöydän ääreen ja pohtia kääntymistä vaakatasossa.

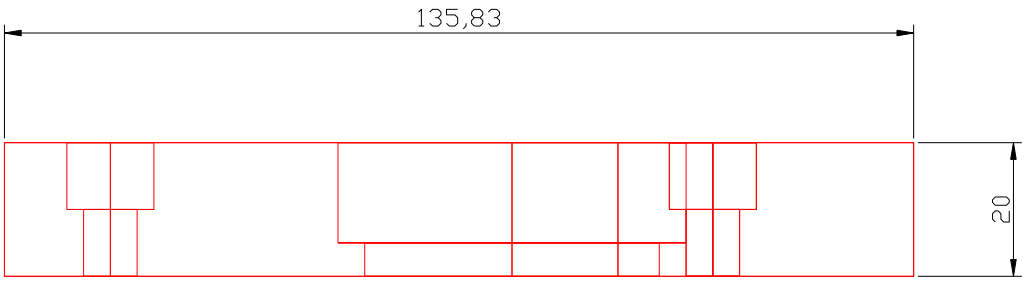
Vaakasuunnassa liikkuminen oli pystysuuntaa hankalampi toteuttaa siinä mielessä, että liikuteltavan rakennelman massa oli huomattavan suuri verrattuna pelkän kameran massaan. Kookas alumiinirakennelma, jossa painoa lisäsivät kamera ja moottori, piti saada pyörimään ympäri mahdollisimman hallitusti. Alustan piti olla tukeva sekä käytännöllinen. Rakennelman epäsäännöllinen muoto haittaisi liikuttelemista.

Pitkällisten pohdintojen jälkeen päädyimme ohjaajani kanssa suunnitelmaan, jossa 20 mm alumiinialustaan upotettaisiin halkaisijaltaan 52 mm:n kuulalaakeri. Laakerin varassa liikuteltava yläosa olisi kytketty alumiinialustan alla roikkuvaan moottoriin. Tämän suunnitelman vahvana puolena oli, että ala- ja ylösosan voisi kätevästi irrottaa toisistaan nostamalla ne erilleen.

Suunnitelma ei luonnollisesti ollut aivan selvä vielä ensimmäisiä piirustuksia tehtäessä, vaan yksityiskohdat muotoutuivat työn edetessä. Alaosan rakenteen kannalta määräääviä osia olivat moottori sekä siihen liitettävä hammaspyörä, kuten myös laakeri ja koko aiemmin koottu rakennelma.

Alusta

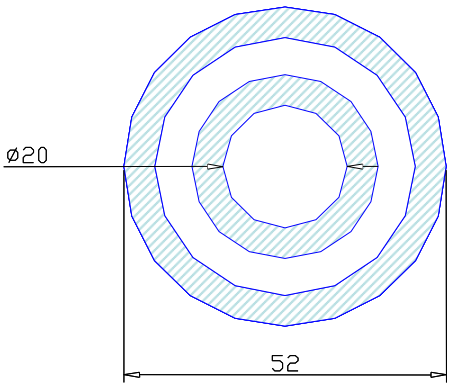
[Alusta, Right]



Idea lähti tukevan 20 mm alustan suunnittelusta. Alustan keskelle tarvittiin halkaisijaltaan 52 mm:n syvennys laakerille. Juuri tämän kokoisia teriä ei jyrsimeen löytynyt. Koko alusta oli saatava sopimaan sorviin jollain ilveellä. Tästä syystä alusta suunniteltiin muodoltaan sopivaksi sorviin, jossa kiinnitys on toteutettu kolmelta puolelta. Alustasta tuli epäsäännöllinen kuusikulmio.

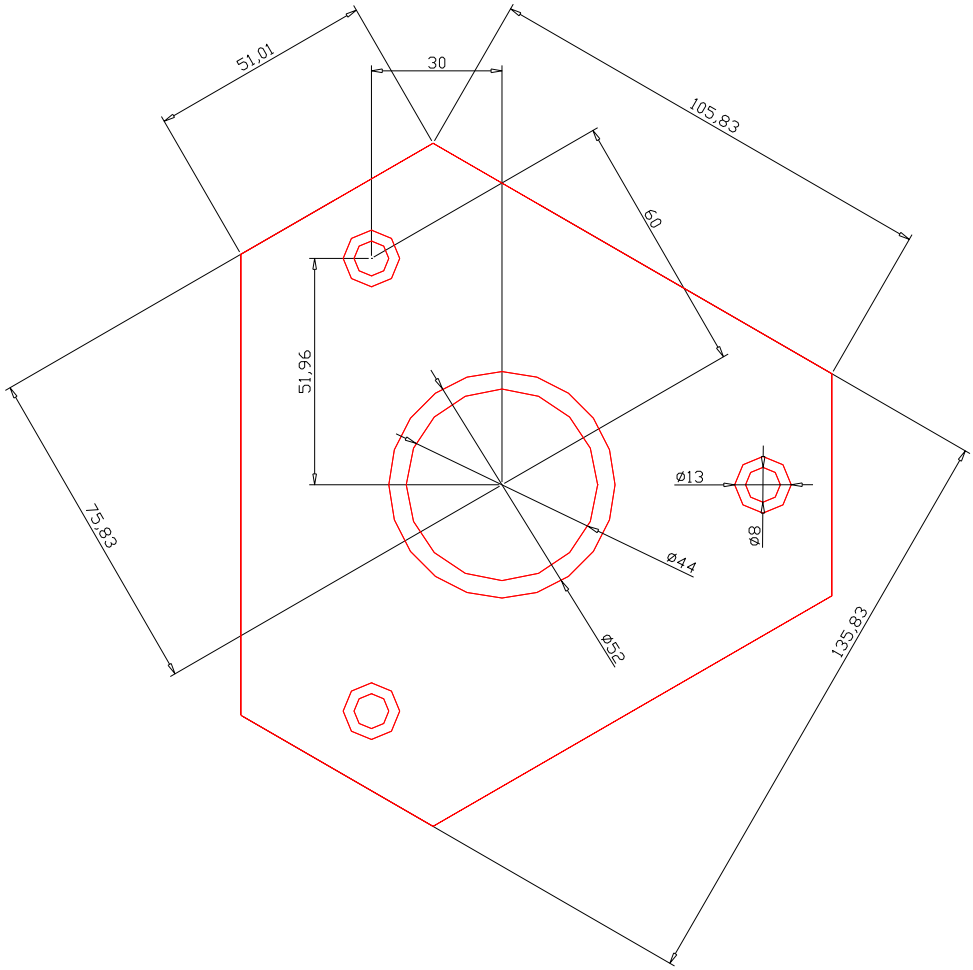
Alustan muotoilu ei ollut helppoa. Jyrsin liikkuu vain kolme akselia pitkin (x, y, z), jotka kaikki ovat toisiaan vasten kohtisuorassa. Ohjaamalla käsin olisi ollut kappaletta välillä irrottamatta mahdotonta saada tehtyä toisiaan vastaan 120° kulmassa olevia sivuja. Edes kappaletta irrottamalla ja siirtämällä ei ollut helppo saada sivuja kohdistettua. Valtavan suuri mutteri osoittautui parhaaksi kohdistusvälineeksi kulmia mitattaessa. Mutterin sivujen välissä oli nimittäin samat 120 astetta.

[Laakeri, Top]



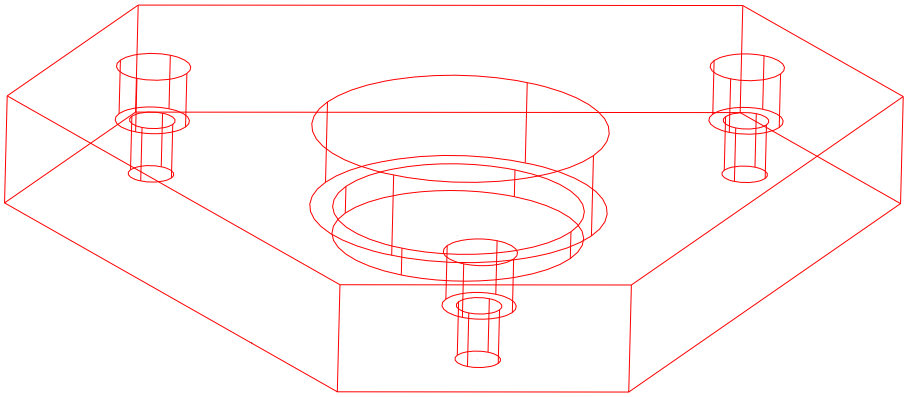
Laakeria varten sorvattava reikä ei ollut helppo toteutettava. Ohjaajani suositteli lämpimästi reiän sorvauttamista asiansa osaavalla sorvitaiturilla. Tämän keskireiän suhteen sain apua verstaan gurulta, jolta sorvaus sujui muiden tekemisten ohessa. Olen todella kiitollinen tästä, sillä reiästä tuli viittä vaille täydellinen: Laakeri sopii siihen kuin valettua, muttei pääse lainkaan liikkumaan sivusuunnassa. Kaiken lisäksi laakeri on helposti nostettavissa pois reiästä. Reiästä teki erikoisen sen porrasmaisuus, sillä jottei laakeri valahtaisi kappaleen läpi, oli reikää kavennettava 15 mm:n kohdalla, jolloin reiän halkaisija vaihtui 45 millimetriin.

[Alusta, Top]



Kappaleen muiden reikien poraaminen ei ollut hankalaa. Alustaa kannattelevat jalat kiinnitettiin 8 mm:n pulteilla, joiden kannat upotettiin kappaleen päälipuolelle 13 mm:n terällä. Alapuolelle porattiin myöhemmin moottorin kiinnitysreiät.

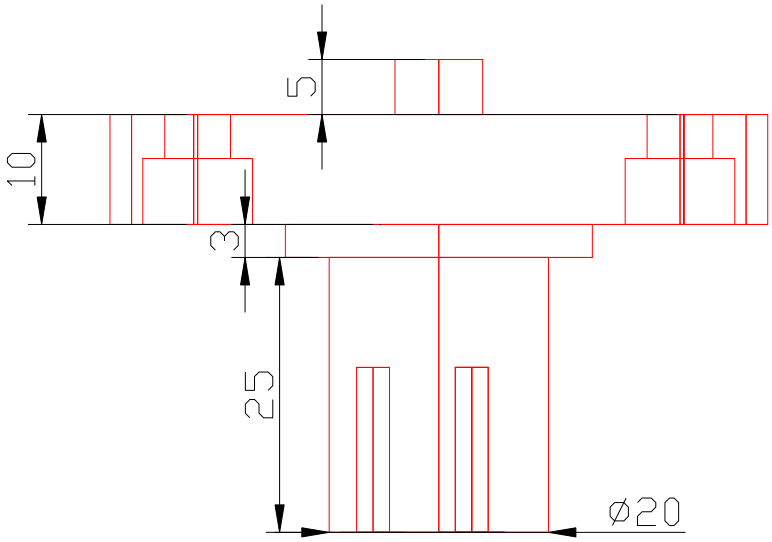
[Alusta, 3D]



Ylhäältä otettu kuva, jossa keskireiän porrastaminen näkyy.
Reiän läpi näkyy alapuolelle kiinnitettävä moottori.

Kääntymismekanismi (Horizontal)

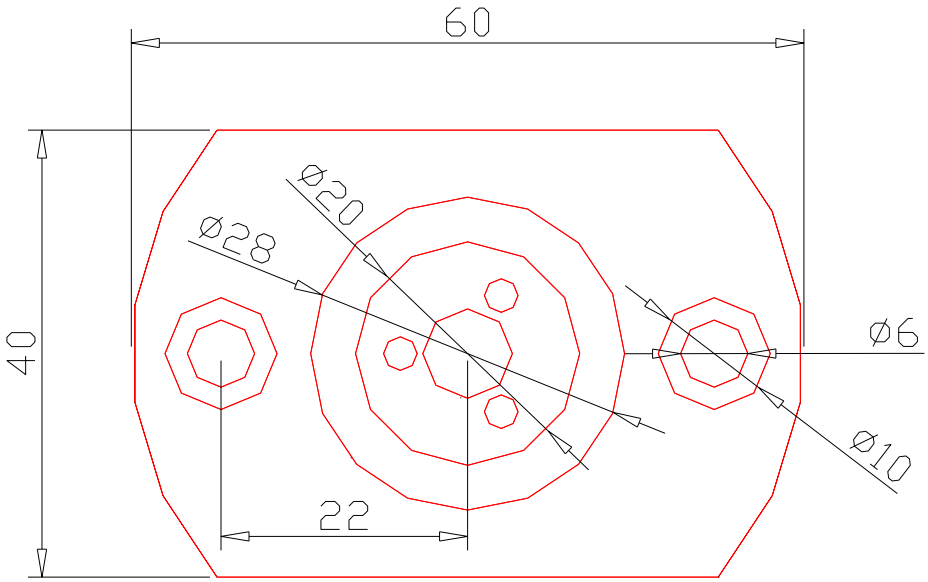
[Horizontal, Front]



Koko laitteen alaosa muodostuu kahdesta tärkeästä osasta. Toinen on alusta, johon laite tukeutuu, ja toinen on laakeriin, hammasrattaaseen sekä yläosaan kiinnittyvä kappale. Tämän laakerin sisään sovitettavan kappaleen tuli sopia tiukasti paikalleen, mutta myös olla helposti liikuteltavissa tarpeen tullen. Lisäksi kappaleeseen tuli pystyä kiinnittämään hammasratas alaosaan. Kappaleen yläosa puolestaan kiinnittyi palkkiin, jonka varaan muu rakennelma oli koottu.

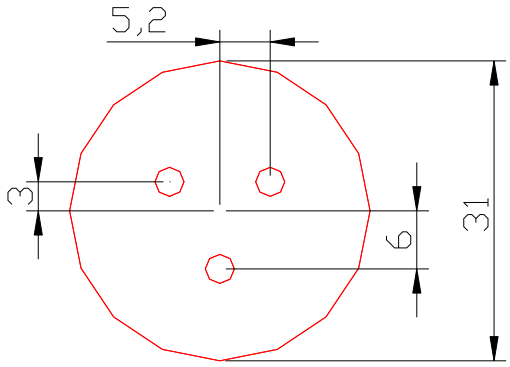
Kappaleen tekeminen alkoi todella paksun umpinaisen alumiinitangon sorvaamisella. Kappaleen yläosa on halkaisijaltaan 60 mm, mutta se kapenee nopeasti, sillä laakerin aukko on halkaisijaltaan 20 mm. Jotta kappaleen yläosa ei makaisi laakerin päällä, on sorvaukseen tehty pieni porrastus (Ø 28 mm x 3 mm).

[Horizontal, Top]



Kaikkein ylinnä on pieni uloke, jonka tarkoitus asettaa kappaleeseen kiinnitettävä palkki tukevasti paikalleen. Kiinnitys tehtiin poraamalla kappaleeseen reiät (Ø 6 mm), joissa ruuvikannat upotettiin. Reikien läpi saatiin palkki kiinnitettyä kappaleeseen palkin uraan asetettavien vastakappaleiden avulla.

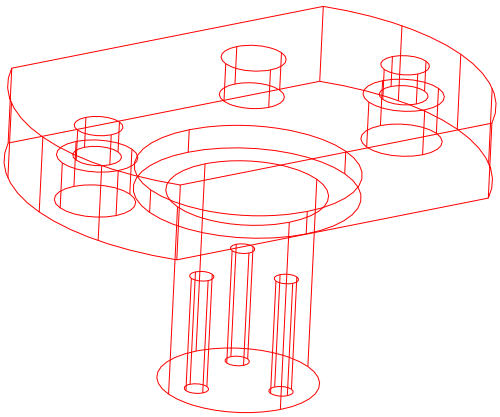
[Hammasrh, Top]

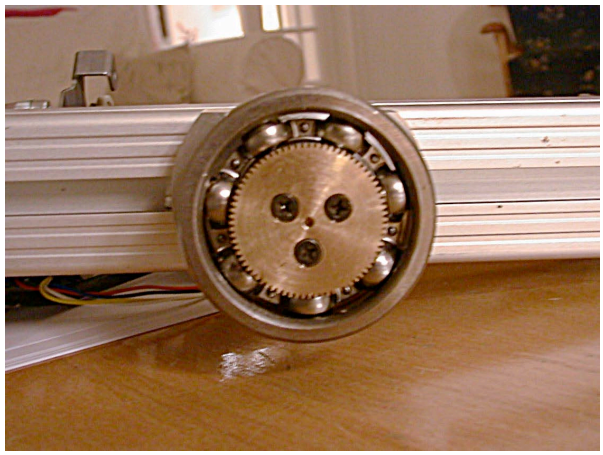


Laakerin läpi kulkevaan osaan kiinnitettiin hammasratas. Koska kappale oli hyvin kapea (Ø 20 mm), jouduttiin aiemmin käytetystä kiinnitysmitoista joustamaan. Hammasrattaan läpi porattiin reiät ja kohdakkain näiden reikien kanssa tehtiin 3 mm:n reiät kiertein kappaleeseen.

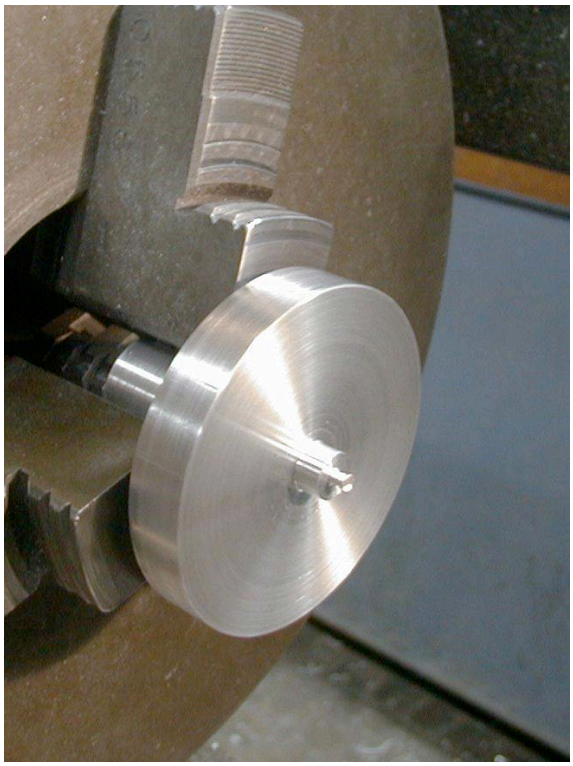
Kappaleen reunojen suoristus sekä erikokoisten reikien poraus tehtiin jyrsimessä, kuten myös hammasrattaan rei'itys. Kappaleen alaosaan tehtiin hammasrattaan kiinnittämisen helpottamiseksi ohjaava syvennys rattaan ulkonevalle keskikohdalle.

[Horizontal, 3D]



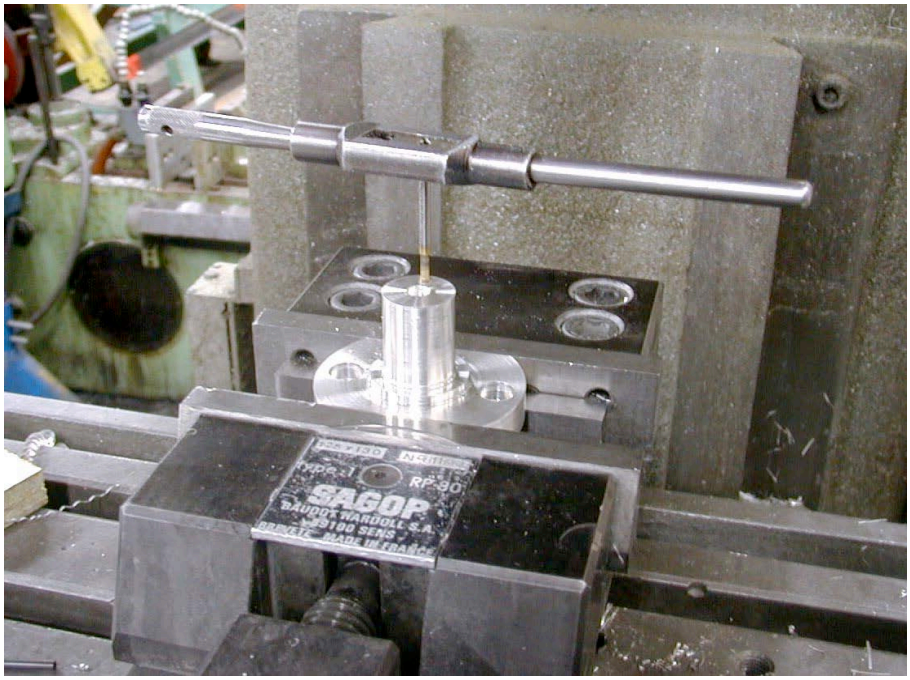


Kappale kuvattuna alhaalta. Laakeri on pujotettu paikalleen ja hammasratas kiinnitetty.



Kappale sorvattuna oikeaan muotoonsa.

Lopullinen muotoilu tehdään jyrsimellä.

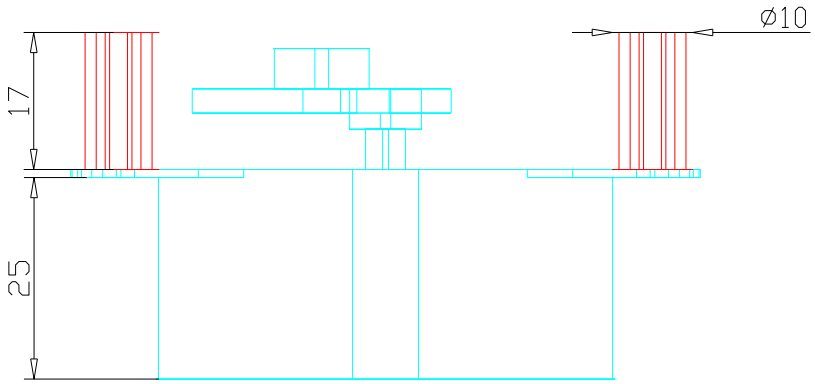


Lopuksi tehdään kierretapilla kierteet hammasrattaan kiinnittämistä varten.

Moottorin kiinnitys

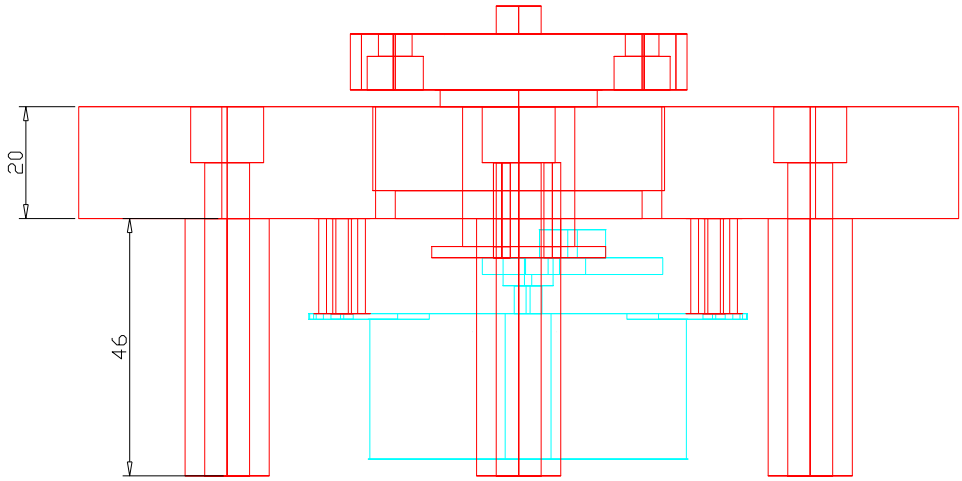
Alun perin moottori piti kiinnittää tukevasti lattiatasossa olevaan levyyn. Lopulta tällainen levy kävi tarpeettomaksi ja moottori ripustettiin ikään kuin roikkumaan alustan alle. Kiinnitysperiaate on yksinkertainen: moottori on ruuvattu kiinni alustaan kahden mittaan sahatun ontoksi poratun putken varassa.

[Moottorih, Front]



Moottoriin itseensä ei tarvinnut tehdä muita muutoksia kuin poistaa moottorin tukirakennelmassa valmiiksi olleet kierteet. Haastavaa oli saada moottori kiinnitysvaiheessa asetettua juuri oikeaan kohtaan. Moottorin hammasratas tuli saada sopivasti koskemaan toista ratasta. Moottorin tarkkaa paikkaa ei ole piirustuksissa esitetty, sillä sopiva paikka määritettiin kokeilemalla.

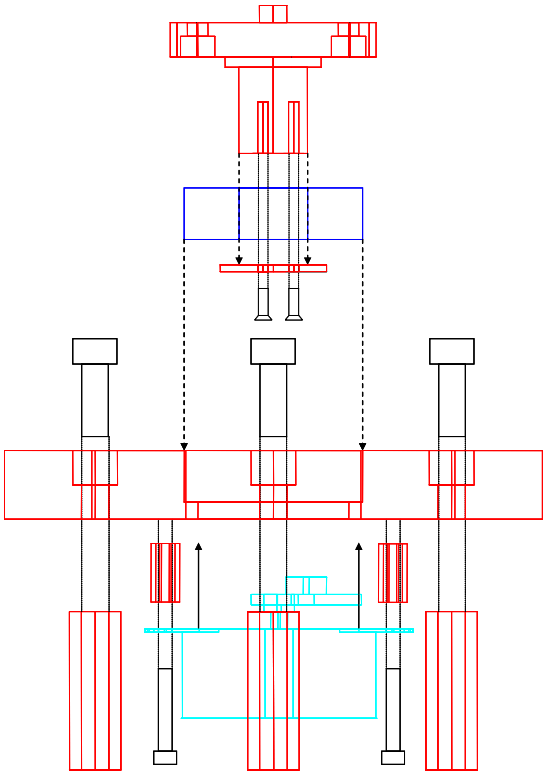
[Alaosa, Front]



Alustaa kannattelevat jalat ovat yksinkertaisesti halkaisijaltaan 15 mm:n metallitankoa, jonka sisään on tehty kierteet (8 mm). Vaikka laite on tarkoitettu seisomaan näiden jalkojen varassa, on jalkoihin kauaskatseisesti tehty kierteet koko pituudelta. Laite voidaan siis pultata kiinni alustaan tarpeen tullen.

Alaosan kokoaminen

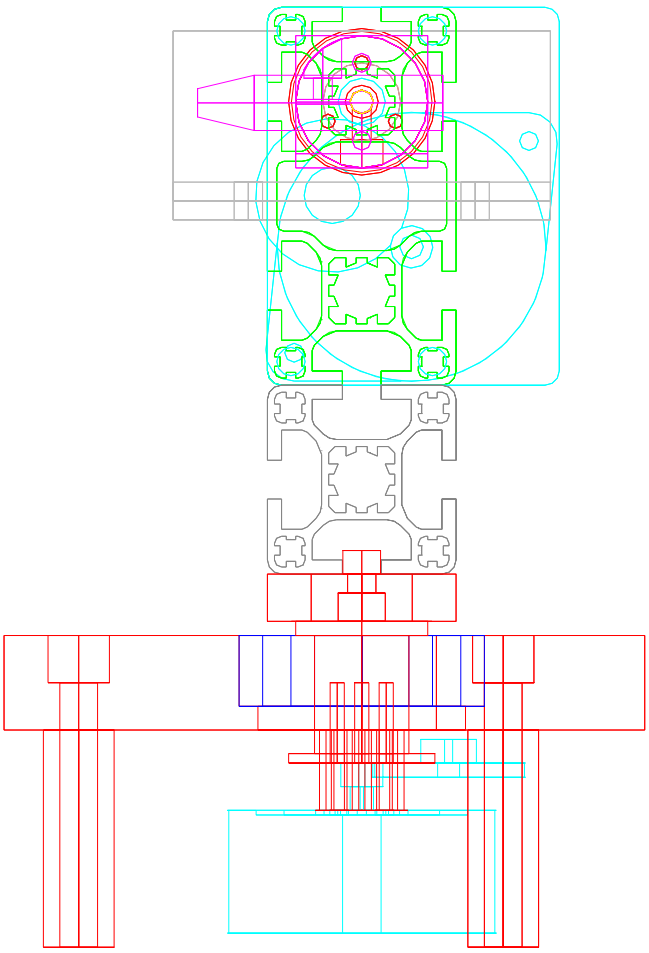
[Alaosa yhdistetty, Front]



Laitteen alaosan kokoaminen tapahtuu liittämällä osat yhteen ja ruuvaamalla ruuvit paikalleen. Huomioitavaa on, että yhtään ruuvia avaamatta voidaan ylinnä näkyvä muuhun laitteeseen kiinnittyvä kappale (johon liitettyinä laakeri ja hammasratas) nostaa erilleen alustasta.

Ohjaajani Stefan Johansson sahan ääressä.

[Koko, Right]



Sivuprofiilikuvasta näkyy, miten ylä- ja alaosa liittyvät toisiinsa.

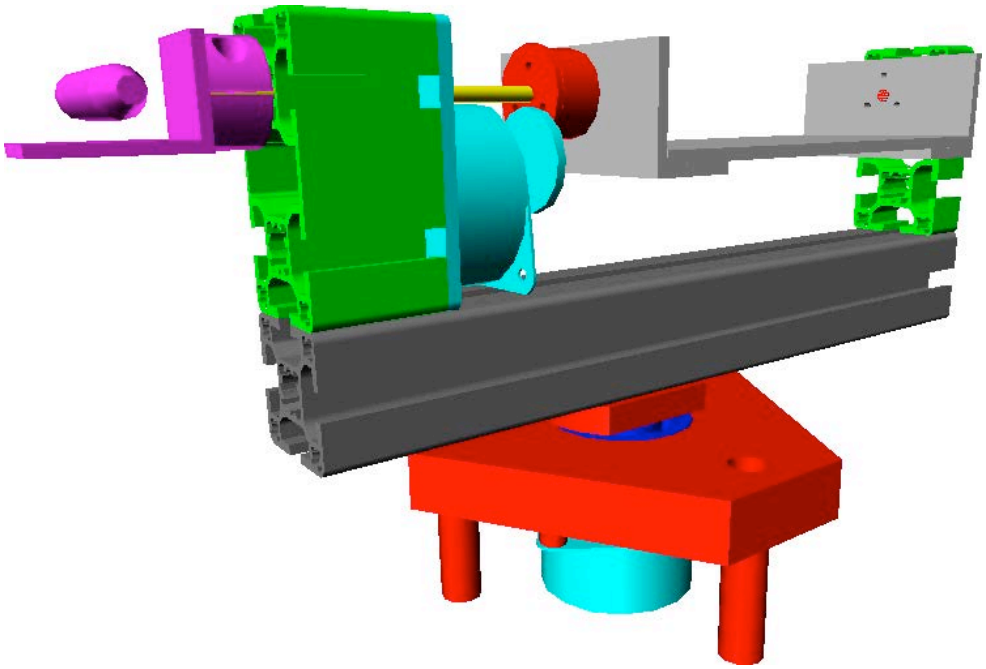
Kokoamisen ja koeajon jälkeen paljastui, että hammasrattaan harvahampaisuus aiheuttaa epätarkkuutta ja tärinää laitteen liikkuessa vaakatasossa ympäri. Vaikka pystysuuntaiseen kääntymiseen käytettiin samanlaista hammasratastoa, ei epätarkkuus siinä haitannut laitteen liikkumista, sillä liikuteltavan kappaleen massa ja ulottuvuudet olivat pieniä. Vaakasuunnassa on momentti ongelma varsinkin ajettaessa laitetta suurella nopeudella (< 10 ms/askel).

Epätarkkuus aiheutti ylimääräistä tärinää, kolinaa sekä huolta mekaanisesta kulumisesta. Ratkaisu löytyi sijoittamalla alustan ja kääntymismekanismin väliin jäävään tilaan pientä kitkaa aiheuttava teflon-levy. Tämä levy jarruttaa tehokkaasti laitteen kääntymistä, mikä näkyy vakautena ja tasaisena kääntymisenä. Teflon-levyn paksuus on 3 mm ja halkaisija 76 mm. Sisäreiän halkaisija on reilut 20 mm. Idea teflonin käytöstä tuli isältäni. Kiitos hänelle siitä.

Valmis mittalaite

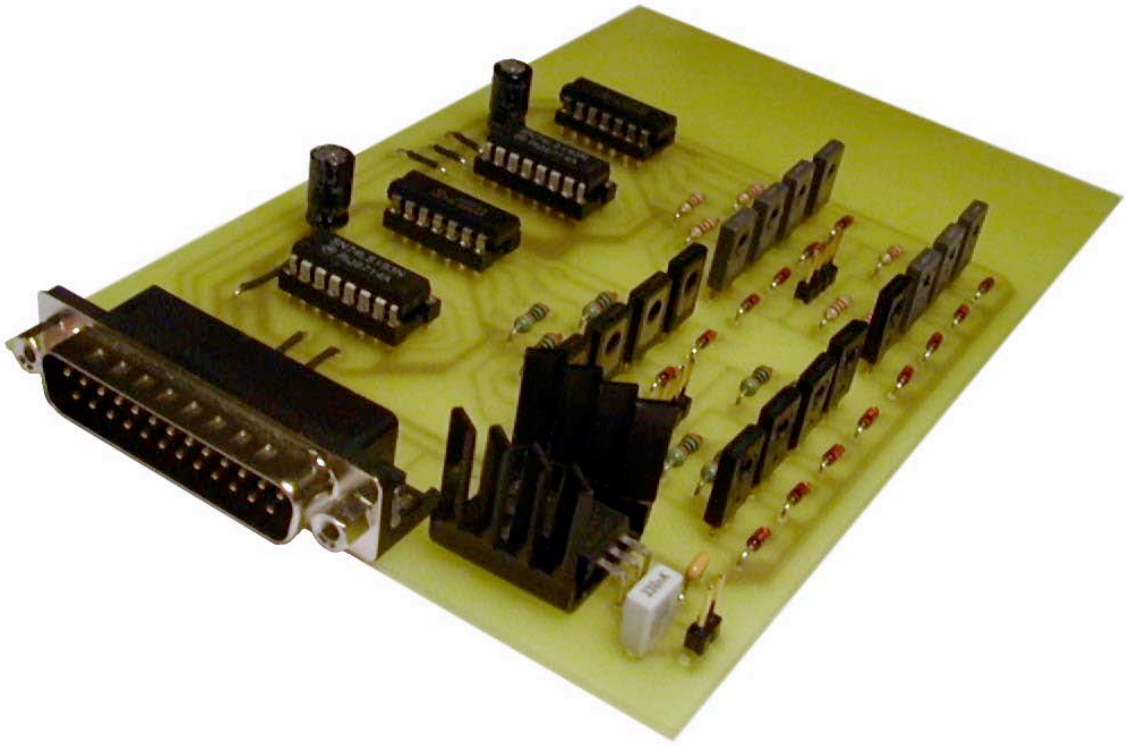
Mittalaitteen viimeistely alkoi perusteellisella laitteen purkamisella osiin ja ylimääräisten tussilla tehtyjen piirrosjälkien poistamisella vahvan etanolin avulla. Tämän jälkeen osat sovitettiin jälleen yhteen ja ruuvikannat kiristettiin huolellisesti. Maytec-profiileihin liitettiin niihin kuuluvat muovisuojukset.

Kameran kiinnitys hoitui vanhasta kamerajalustasta otetulla erityisellä kameran alla oleviin standardikierteisiin sopivalla kiristysruuvilla. Kameran alle asetettu vaahtomuovimatto tuki sitä pysymään vaakatasossa. Kiinnitys varmistettiin kiristämällä nippuside kameran ja alustan ympäri. Laser kohdistettiin osoittamaan samaan suuntaan kameran kanssa. Kaapeleiden kytkemisen jälkeen laite oli mekaanisesti täysin toimintakunnossa.



Mittalaite kokonaisuudessaan.

Elektroniikka



Pohjustus

Nelivuotiaana rakensin hiukkaskiihdyttimiä. Viisivuotiaana vaihdoin avaruusaluksiin. Nyt 18-vuotiaana on aika rakentaa etäisyysmittauslaite. Eräät varhaisimmista muistoistani liittyvät rakentamiseen ja elektroniikan parissa puuhasteluun. Olin pienenä harvinaisen ihastunut erilaisiin monimutkaisiin vekottimiin. Erityisesti niiden purkaminen osiin oli suuri intohimoni.

Vaikka hiukkaskiihdyttimeni ja avaruusalukseni olivat pääosin pahvista tehtyjä, oli niissä aina jotakin oikeasti toimivaa. Yleensä toimiva elektroniikka rajoittui suureen määrään kytkimiä, pariin valodiodiin ja 4,5 V:n paristoon. En muista minkä ikäinen olin, kun vanhat venäläiset hiukkaskiihdyttimen osat, rikkinäiset radiot, johdinkerät ja paristot siirtyivät kaappini kätköihin. Nyt noin kymmenen vuotta myöhemmin olen havainnut lapsuuden kokemukseni hyödyllisiksi ja jälleen kaivanut vanhat romuni esiin kaappien kätköistä.

Pohtiessani aihetta päättötyölle olin lähes alusta asti sillä kannalla, että päättötyöni liittyisi jotenkin elektroniikkaan. Elektroninen osuus osoittautuikin päättötyön rennoimmaksi ja mukavimmaksi osuudeksi. Tämä johtui ehkä osittain siitä, että se on myös huomattavasti muita osioita suppeampi. Tunnen kuitenkin oppineeni todella paljon hyödyllistä juuri elektroniikan parissa.

Elektroniikka on ikään kuin ohjelmoinnin ja mekaniikan välimaastossa. Siinä vaaditaan sekä päättelyä että näppäryyttä. Tämän osuuden lukeminen saattaa vaatia jonkinlaista osa-alueen tuntemista. Toimintaperiaatteet ovat yksikertaisia ja kytkennätkin koostuvat lähes pelkistä peruskomponenteista.

Turun komponenttiliikkeet ovat käyneet tutuiksi projektia tehdessä. Elektroniset osat on hankittu pääosin Turun Komponenttipisteestä, Triopakista, Bebekistä sekä Yleiselektroniikasta. Varsinkin Komponenttipisteen myyjät kävivät tutuiksi, kun vierailukertoja liikkeeseen kertyi toista kymmentä. Onneksi kertaostokset olivat vain muutaman euron luokkaa.

Päättötyön elektroninen osuus käsittelee askelmoottoreiden ohjauskytkennän, piirilevyn tekoa sekä kameran ja laserin virransyöttöä. Varsinkin askelmoottorikytkennän perinpohjainen ymmärtäminen saattaa vaatia askelmoottoreihin ja rinnakkaisporttiin liittyvän tekstin lukemista ohjelmointia käsittelevästä osiosta.

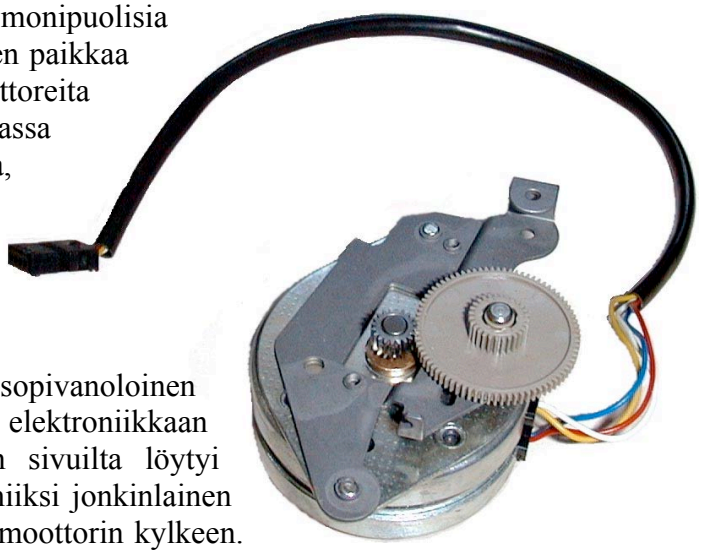
Askelmoottori

Askelmoottori (stepper motor) on elektroninen moottorityyppi, jota käytetään, kun halutaan hallita moottorin liikkeitä tarkasti. Askelmoottorin toiminta perustuu moottorinsisäiseen roottoriin, johon on liitettyjä magneetteja. Roottoria hallitaan kiinteillä sähkömagneeteilla, joiden magneettikenttää muuttelemalla saadaan moottori liikkeeseen. Tämän takia askelmoottoria voidaan pitää tavallisen tasavirtamoottorin (DC) ja solenoidin risteytyksenä.

Askelmoottoreilla on tietty määrä magneettisia napoja, joiden kautta määräytyy askeleiden määrä kierrosta kohti. Yleinen lukumäärä on 200 täyttä askelta/kierros, mikä tarkoittaa, että täyden kierroksen kulkemiseen vaaditaan moottorilta 200 askelta. Kehittyneet askelmoottorit pystyvät lisäksi tekemään eräänlaisia osittaisia askeleita (microsteps).

Askelmoottorit luokitellaan yleensä niiden vääntömomentin mukaan. Eräs ainutlaatuinen ominaisuus askelmoottoreille on niiden kyky pitää vääntömomenttia yllä myös silloin, kun ne eivät ole liikkeessä. Saavuttaakseen maksimaalisen vääntömomentin moottorin käämeihin on johdettava ilmoitettu määrä virtaa joka askeleella. Moottoria ohjaavan virtapiirin on muuteltava virtaa, jotta moottori liikkuisi. Jännite sen sijaan on toisarvoinen seikka.

Tietokoneen ohjaamilla askelmoottoreilla voidaan luoda erittäin tarkkoja ja monipuolisia järjestelmiä, joissa mekaanisten osien paikkaa voi tarkasti hallita. Askelmoottoreita käytetään tästä syystä muun muassa korppuasemissa, tasoskannereissa, tulostimissa ja monissa muissa vastaavissa laitteissa.



Berger RDM57 -askelmoottori

Lyhyen hakemisen jälkeen sopivanoloinen askelmoottori löytyi. Kaikenlaiseen elektroniikkaan liittyvää tavaraa myyvän Bebekin sivuilta löytyi maininta moottorista, jossa oli valmiiksi jonkinlainen hammasratas kiinteästi asennettuna moottorin kylkeen. Posti toi näitä 10 € arvoisia kapineita kaksi jo parissa päivässä tilauksesta.

Pienen ruskean pahvilaatikon uumenista paljastui kaksi kokoonsa nähden painavaa moottoria, joihin oli liitetty metallisella pidikkeellä muovinen hammasratas. Ohessa moottorin teknisiä tietoja, joita en kylläkään tiennyt moottoreita tilattaessa, sillä tekniset tiedot ilmestyivät kaupan sivuille vasta kuukausien päästä ja silloinkin saksaksi.

Berger RDM57 on bipolaarinen (sisältää kaksi käämiä) askelmoottori (saksaksi Scrittmotor). Moottorilta kuluu 24 askelta kierroksen tekemiseen. Koska moottori on bipolaarinen, on siihen kytkettävä neljä johdinta. Moottori aiheuttaa noin 30 Ohmin vastuksen. Käyttöjännite on 10 Volttia ja virta 0,345 Ampeeria.

Tekniset tiedot:

Koko:	Ø 57 mm x 25 mm
Tyyppi:	Bipolaarinen
Askelmäärä:	24/kierros
Askelkulma:	15°
Kulman tarkkuus:	± 3 %
Suurin vääntömomentti:	9,4 Ncm
Pysäytysmomentti	12 Ncm
Hitausmomentti:	28 gcm^2
Paino:	noin 0,3 kg

Hammasrattaat

Moottoreista ei ole yksin mitään hyötyä, vaan moottorit kaipaavat rinnalleen toimivan voimansiirron. Tällaisen järjestelmän voi toteuttaa eri tavoin, mutta hammasrattaat tarjoavat siihen varsin yksinkertaisen ja selkeän keinon.

Moottoreihin on suoraan liitetty hammasrattaita, jotka toimivat eräänlaisena vaihteistona voimansiirrossa moottoreilta itse laitteeseen. Moottoreissa oli alun perin kiinteästi asennettuna hammasrataspari. Haasteeksi muodostui löytää näihin yhteensopivia rattaita. Ensinäkin oli vaikeaa löytää Turunkin kokoisesta kaupungista hammasrattaita myyvää liikettä. Toiseksi oli vaikeaa löytää juuri oikeanlainen hammasmuodoiltaan yhteensopiva ratas.

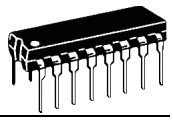
Keskustan Askartelukeskus osoittautui hyväksi paikaksi metsästää näitä pieniä osia, mutta laajoista valikoimista löytyi sielläkin vain yksi yhteensopiva hammasratasmalli, joten valinta oli helppo. Rattaan materiaalina on messinki, jota – toisin kuin muovia – on suhteellisen helppo työstää. Yksi hammasratas tuhoutui kuitenkin työstämisen aikana.

Hammasrattaissa oleellista on hampaiden määrä tai oikeammin kahden vierekkäisen hammasrattaan hampaiden määrien suhteet. Koska yhteensopivien hammasrattaiden hampaiden leveys pysyy vakiona, vaihtelee siis rattaan halkaisija hammasmäärän mukaan.

Käytetty hammasratasto sisälsi neljä erilaista hammasratasta. Rattaissa oli 20, 80, 28 ja 75 hammasta ja kaksi keskimmäistä oli liitetty kiinteästi toisiinsa, jotta järjestelmä toimisi. Laskemalla rattaiden hampaiden määrien väliset suhteet voidaan todeta seuraavaa:

	Hampaita (kpl)	**Moottorin kierros**	**Yksi kierros**	**Yksi askel**
Moottori	20	1	75/7	1/24
	80	1/4	75/28	1/96
	28	1/4	75/28	1/96
Akseli	75	7/75	1	7/1800

Taulukossa hammas-sarake ilmaisee jokaisen rattaan hampaiden lukumäärän. Moottori-merkintä tarkoittaa sitä ratasta, johon moottori on liitetty, ja akseli puolestaan akseliin liitettyä ratasta. Moottorin kierros -sarake näyttää esimerkin omaisesti, että moottorin kiertyessä kokonaisen kierroksen itsensä ympäri liikkuu itse akseli ainoastaan noin 0,1 (7/75) kierrosta ympäri.



Vastaavasti akselin kokonaiseen kierrokseen vaaditaan moottorin kiertymistä noin 10,7 (10 5/7) kierrosta. Koska moottorin tiedetään tekevän 24 askelta kierrosta kohti, voidaan laskea tarvittavan noin 257 (257 1/7) askelta akselin kiertymiseen 360°.

Asteet	**Askeleet (tarkka)**	**Askeleet (kokonaiset)**
1,4°	1	1
45°	32 1/7	32
90°	64 2/7	64
180°	128 4/7	129
360°	257 1/7	257

Askelmoottori liikkuu aina kokonaismäärän askelia. Pienin mahdollinen liike on näin ollen 1,4 asteen muutos varsinaisella akselilla. Ohjelmoinnin yhteydessä hyödynnetään taulukossa esitettyjä pyöristettyjä kokonaislukuarvoja.

Askelmoottoreiden ohjauskytkentä

Askelmoottoreiden askeleiden suhdetta laitteen kiertymiseen tarvitaan vasta ohjelmoinnin puolella. Ensiksi on syytä saada moottorit kytkettyä tietokoneeseen, josta niitä voi ohjata. Moottoreita ohjaavan kytkennän saa tehtyä muutamalla peruskomponentilla erittäin loogisella kytkennällä.

Askelmoottoreiden pyöriminen perustuu magneettisiin napoihin, jotka saavat moottorin akselin kiertymään. Napaisuutta vaihdetaan sisääntulosignaalien kautta. Sisääntulosignaalit saadaan rinnakkaisportista. Yhtä moottoria kohti tarvitaan kaksi signaalia. Rinnakkaisportin datasignaalit saadaan navoista 2-9. Kahden moottorin tarpeisiin riittävät neljä ensimmäistä.

Tarkastellaan moottoreiden ohjauskytkentöjä erillisinä, vaikka ne on juotettu samalle piirilevylle. Kyseessä on lähes identtiset kytkennät ja tarkastelua helpottaa toisen sulkeminen pois. Rinnakkaisportista tulevat kaksi signaalia kulkevat Up/Down Counter -komponenttiin (74LS193), joka muuttaa rinnakkaisportista saadun signaalin kauniin tasaiseksi. Up/Down Counter laskee nimensä mukaisesti portin nousua ja laskua. Linjat voivat olla joko tosia tai epätosia, eivät mitään siltä väliltä. Jos virta kulkee, arvo on tosi; jos ei, arvo on epätosi. Näin tietokoneen bitit muuttuvat sähkövirraksi.

Up/Down Counter -komponentti on liitetty mikropiiriin, joka sisältää neljä poissulkevaa TAI-toimintoa (74LS86). Poissulkeva TAI-toiminto tunnetaan paremmin englanninkielisellä nimellään Exclusive disjunction (XOR). Poissulkeva TAI poikkeaa pelkästä TAI-toiminnosta siinä, että arvo on tosi vain toisen sisääntulon ollessa tosi. Voidaan kirjoittaa totuusarvotaulukko:

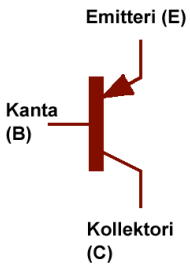
Sisään A	Sisään B	Ulos (A xor B)
0	0	0
0	1	1
1	0	1
1	1	0

Yksi xor-toiminnon symboleista

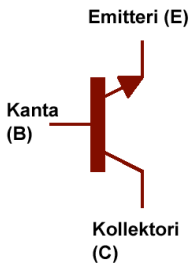
Pelkkä TAI-toiminto ei usein vastaa arkikielessä käytettyä tai-sanaa. Poissulkeva TAI on lähempänä, sillä se vastaa ilmausta ”joko – tai”. Esimerkiksi Hollywood-elokuvissa päähenkilöt ovat joko hyviä tai pahoja, ei kumpaakin, eivätkä kumpaakaan. Samalla tavalla toimii poissulkeva TAI-toimitus. Logiikan dualismille vetää vertoja ainoastaan amerikkalaisten elokuvien maailmankuva.

Neljästä XOR-portista lähtee kustakin johdin. Näiden neljän johtimen arvo vaihtelee niin, että niistä aina kaksi kerrallaan on päällä. Näin saadaan ohjattua transistoreita, jotka vaihtavat moottorin napojen polaarisuutta ja saavat moottorin pyörimään.

pnp-transistori



npn-transistori



Transistorit ovat eniten käytettyjä puolijohteita. Niitä on olemassa monen tyyppisiä. Transistorin toimintaideana on ohjata virran kulkua. Npn-transistoria ohjataan positiivisella ohjausvirralla, pnp-transistoria negatiivisella. Transistorissa on kolme napaa: emitteri (E), kollektori (C) sekä kanta (B). Npn-transistorin kollektori kytketään positiiviseen jännitteeseen, emitteri on kytketty

maapotentiaaliin ja kannalle annetaan ohjausvirta. Ohjausvirralla voidaan ohjata kollektorin ja emitterin välisen virran kulkua.

Yhden npn-transistorin (BD237) kanta on kytketty kuhunkin XOR-ulostuloon. Transistorin kytkeytyessä päälle maadoittaa se yhden moottoriin menevän johtimen. Johtimen kytkeytyessä maapotentiaaliin saa se pnp-transistorin (BD238) kytkeytymään päälle, mikä syöttää virtaa toiseen moottorin johtimeen. Näin saa toinen moottorin käämistöistä virtaa.

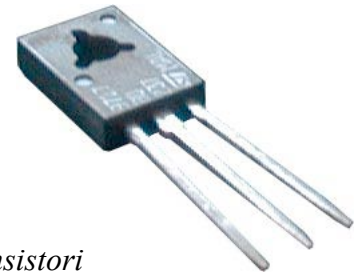
Tällaisia ristiin kytkettyjä npn–pnp-transistoripareja tarvitaan neljä eli yksi kutakin XOR-ulostuloa kohti. Vastaavasti voidaan ajatella moottorin tarvitsevan kaksi syöttöä käämiä kohti ja omat transistorit kumpaakin polaarisuutta varten.

Kannan ohjausvirta on kytketty vastuksen kautta. Vastus suojaa transistoria ylettömiltä virtapiikeiltä. Emitterin ja kollektorin väliin on liitetty diodi. Diodi on puolijohde, jonka läpi virta pääsee kulkemaan vain yhteen suuntaan. Diodin tarkoitus on suojata transistoria askeleen päättyessä syntyvältä käämin virtapiikiltä. Virtapiiriin liitetään kondensaattori suodattamaan jännitevaihteluita.

Käytetyt komponentit

Yhden askelmoottorin ohjaamiseen ei tarvita montaa komponenttia. Tarvittavien komponenttien hankkiminen kävi kivuttomasti. Suurin osa tarvittavista osista on hyllytavaraa, jota löytyy kaikista komponenttiliikkeistä. Turussa kotikulmillani sijaitseva Komponenttipiste paljastui hyväksi komponenttiliikkeeksi, joskin joitain pieniä kompromisseja joutui tekemään. Lisäksi sain joitain komponentteja ohjaajani Stefan Johanssonin kautta. Oheinen lista kuvaa edellisessä kappaleessa kuvattuun kytkentään tarvittavat komponentit.

Komponentti	**Koodi**	**Kpl**
Up/Down ctr	74LS193	1
Exclusive or gate	74LS86	1
Npn-transistori	BD237	4
Pnp-transistori	BD238	4
Diodi	1N4148	8
Vastus	820 Ω	8
Kondensaattori	100 µF, 25 V	1



Transistori

74-sarjan ttl-piirit vaativat 5 V jännitettä. Moottorit toimivat muun muassa tällä jännitteellä, joten koko järjestelmä toteutettiin 5 Voltin jännitteellä. Sopivat muuntajat ovat kiven alla, joten virransyöttö toteutettiin regulaattorilla (7805), joka antaa 5–18 V sisääntulojännitteellä 5 Voltin ulostulojännitteen. Regulaattori kaipaa rinnalleen kaksi kondensaattoria (330 nF ja 100 nF), jotka yhdistävät sen maapotentiaaliin. Regulaattori kuumenee reippaasti ja jäähdytyssiilin käyttö on viisasta.

Pysyin hyvin alkuperäisessä hankintalistassani, mutta joidenkin komponenttien suhteen oli pakko tehdä myönnytyksiä. Kondensaattorin maksimikäyttöjännitteestä jouduin joustamaan ja vaihtamaan 25 Voltin jännitteeseen. Lisäksi aivan samanlaisia vastuksia ei löytynyt kaikkiin kytkentöihin, vaan jouduin käyttämään astetta tarkempia vastuksia osassa kytkentöjä. Tästäkään ei ollut haittaa.

Täydellinen lista komponenteista moottoreiden ohjauskytkennässä:

Komponentti	**Koodi**	**Kpl**
Up/Down counter	74LS193	2
Exclusive or gate	74LS86	2
Holkkikanta	14-napainen	2
Holkkikanta	16-napainen	2
Npn-transistori	BD237	8
Pnp-transistori	BD238	8
Diodi	1N4148	16
Vastus	820 Ω	16
Kondensaattori	100 µF, 25 V	2
Regulaattori	7805	1
Kondensaattori (Ker.)	330 nF	1
Kondensaattori (Ker.)	100 nF	1
D-liitin	M 25-nap.	1
Jäähdytyssiili		1
Piikkirima		1
Hyppylankaa		

Osien ostaminen ei ole suurikaan taloudellinen rasite, sillä yksittäisten komponenttien hinnat liikkuvat senteissä. Yhteishintaa komponenteille on hankala laskea, sillä osan osasista sain lahjoituksena, osan ostin ja loput löysin omista nurkista pyörimästä. Osien yhteishinnaksi kertyi arviolta vajaat parikymmentä euroa.

Kytkentäkaavio

Komponentit ovat kuin palapelin palaset. Jokaisessa on pieni osa kokonaisuutta. Toisin kuin palapelissä voi komponenteista koota lukemattoman määrän yhdistelmiä, joista jokainen on oma kokonaisuutensa ja toimii eri tavalla.

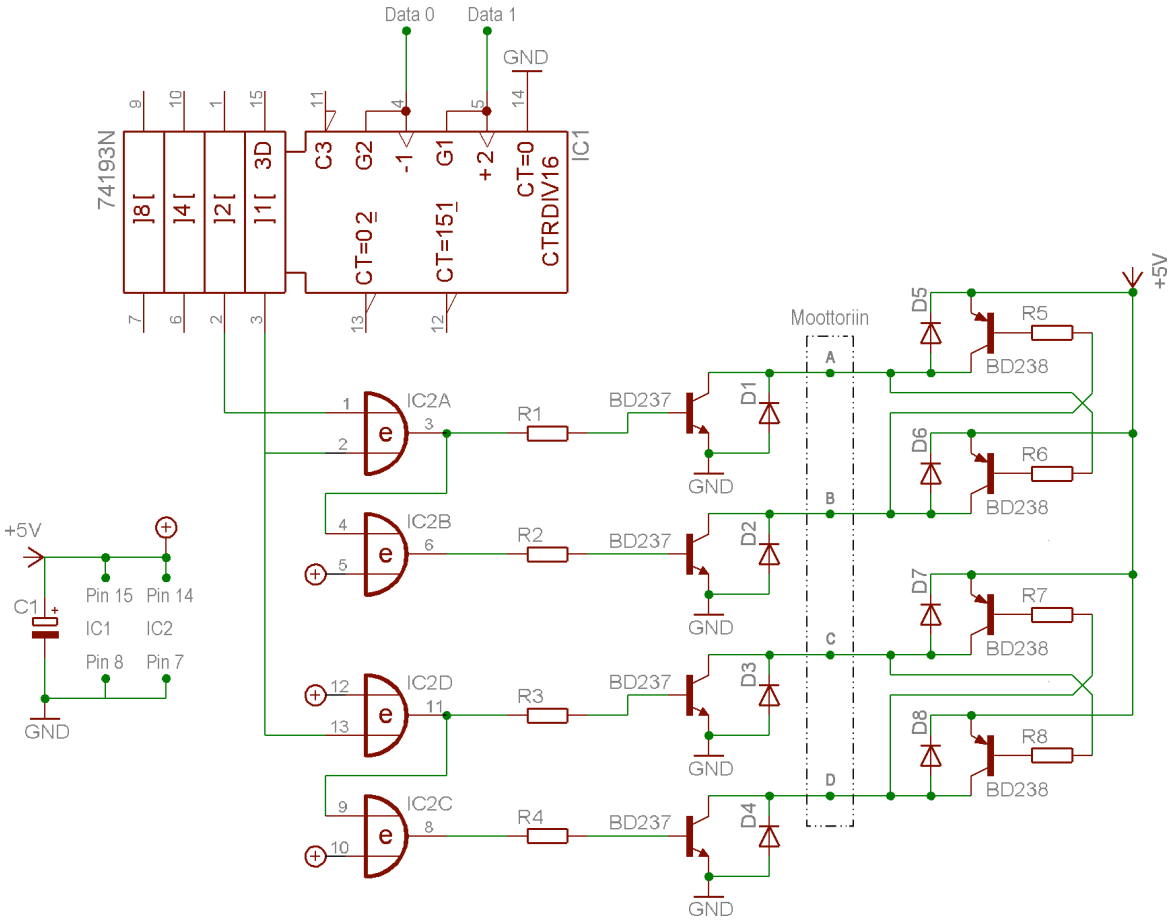
Askelmoottoria voi ohjata monin tavoin, mutta pohjimmiltaan kaikki tavat toimivat saman perusperiaatteen alla. Periaate, jonka olen jo edellä selittänyt, perustuu virran syöttöön askelmoottoreiden käämille. Olen myös kertonut tarpeellisista komponenteista ja niiden toiminnasta sekä seikkaperäisesti selittänyt niiden sovittamisesta yhteen toimivaksi kokonaisuudeksi.

Elektroniikassa esitetään laitteen toimiminen yleensä kytkentäkaavion avulla. Kytkentäkaaviossa on esitetty symbolein laitteen komponentit sekä komponentteja yhdistävät johtimet. Kytkentäkaavion visuaalisella ilmeellä ei yleensä ole mitään tekemistä piirilevyn tai komponenttien asettelun kanssa. Kytkentäkaavion tarkoitus on yksinkertaisesti esittää elektroniikan kielellä miten laite toimii. Kytkentäkaaviossa on jokaiselle komponenttityypille oma symboli. Nämä ovat kansainvälisiä standardeja, joskin jotkut merkinnät vaihtelevat eripuolilla maailmaa (Huomaa poikkeava XOR-merkintä).

Kytkentäkaavion voi piirtää käsin paperille. Kaavion voi tehdä myös tietokoneella, jolloin muutoksi ja korjauksia on huomattavasti helpompi tehdä. Kytkentäkaavioiden tekemiseen käy lähes mikä tahansa piirto-ohjelma. Varta vasten elektroniseen suunnitteluun suunnattuja cad-ohjelmia on runsaasti. Päädyin käyttämään saksalaisen CadSoftin EAGLE-ohjelmaa. EAGLE (Easily Applicable Grapical Layout Editor) sisältää kaikki

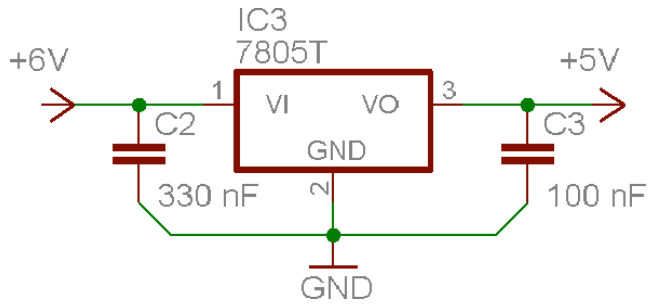
piirilevyn suunnitteluun tarvittavat perustoiminnot. EAGLEn tekee houkuttelevaksi sen ei-kaupalliseen käyttöön tarkoitettu ilmaisversio. Ohjelma sisältää komponenttikirjastoja, josta saa suoraan noukittuja satoja yleisessä käytössä olevia komponentteja.

Piirilevyn suunnittelussa ei tarvinnut aloittaa aivan nollasta, sillä olin saanut askelmoottoriohjauksen peruskytkentäkaavion ohjaajaltani. Tämä paperi oli oletettavasti joskus muinoin Bebekistä saatu; paperin yläkulmassa luki Bebek. Saksankielentaitoni ilahdutti minua suuresti, sillä vähät merkinnät oli kirjoitettu saksaksi. Totesin kavion helppotajuiseksi ja lähdin siitä liikkeelle. Paperista sain suoraan myös tarvittavien peruskomponenttien listan. Minun tarvitsi lähinnä vain ottaa huomioon, että moottori kytkettiin samaan virtalähteeseen muiden osien kanssa.



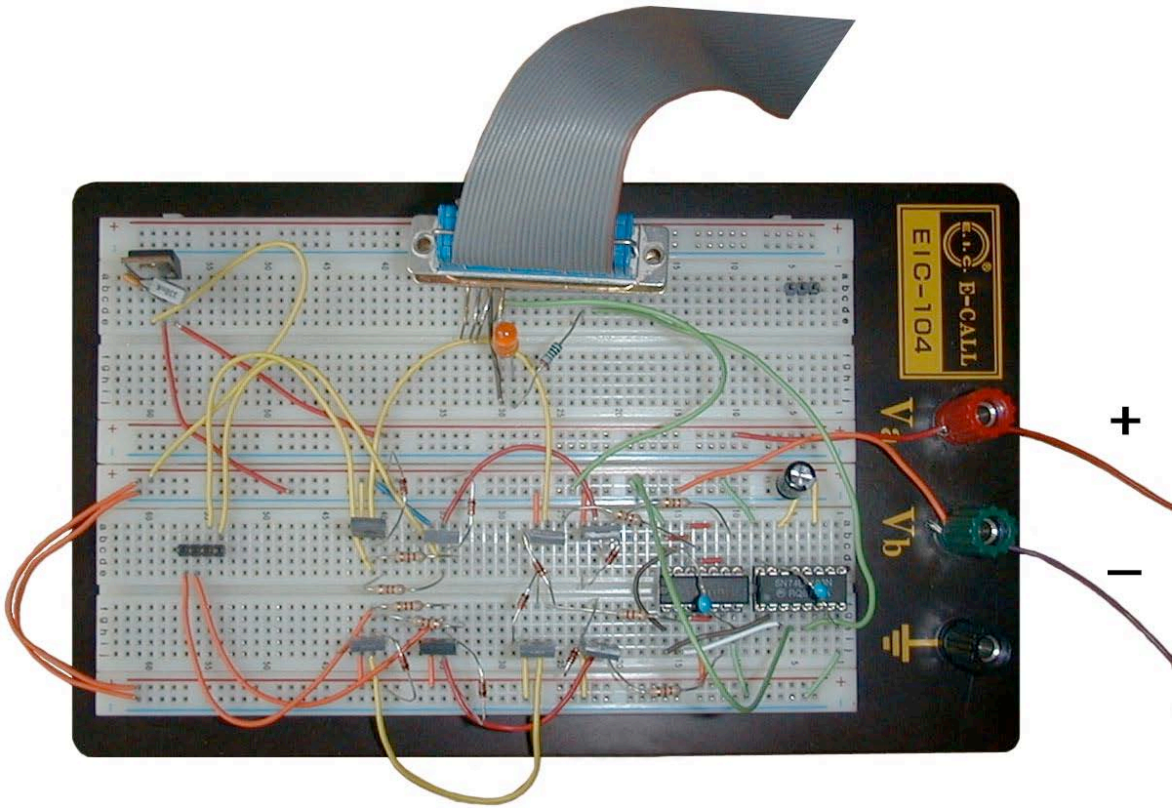
Kytkentäkaavion periaate noudattaa orjallisesti askelmoottoreiden ohjauskytkennän yhteydessä selitettyä periaatetta, jossa rinnakkaisportista tulevat signaalit tulevat Up/Down counterin kautta XOR-portteihin. Saatu signaali määrää transistorien kytkeytymisen ja transistorit puolestaan moottorin käämien napaisuuden.

Merkintä	**Komponentti**	**Koodi**
IC1	Up/Down ctr	74LS193
IC2	Exclusive or gate	74LS86
IC3	Regulaattori	7805
R1–R8	Vastus	820 Ω
D1–D8	Diodi	1N4148
C1	Kondensaattori	100 µF
C2	Kondensaattori	330 nF
C3	Kondensaattori	100 nF



Moottoreiden ja ohjauskytkentöjen virransyöttö on hoidettu regulaattorin avulla. Kamera ja laser käyttävät 6 V jännitettä. Askelmoottorijärjestelmän tarvitsee 5 Voltin jännitettä. Regulaattorista ulos tuleva jännite on kytketty eteenpäin muulle järjestelmälle. Keraamiset kondensaattorit yhdistävät regulaattorin in- ja out-liittimet maapotentiaaliin. Kondensaattorit on asetettava mahdollisimman lähelle regulaattoria.

Kytkentäkaavion valmistuttua olivat tiedossa tarvittavat komponentit sekä niiden väliset kytkennät. Jos on varma asiastaan, voi osat juottaa piirilevylle jo tässä vaiheessa. Itse kaivoin kaapista vuosikausia käyttämättä olleen koekytkentälevyni. Koekytkentälevy on levy täynnä eri tavoin tosiinsa kytkettyjä reikiä, joihin saa kiinnitettyä komponenttien jalkoja väliaikaisesti. Levyllä voi yhdistää komponentit kytkentäkaavion osoittamalla tavalla ja kokeilla kytkentöjen toimivuutta.



Kokosin koekytkentälevylle regulaattorin kytkennän sekä yhtä moottoria ohjaavat komponentit. Saatuani tämän melko sekavaksi käyneen systeemin toimimaan oli aika aloittaa lopullisen piirilevyn suunnittelu.

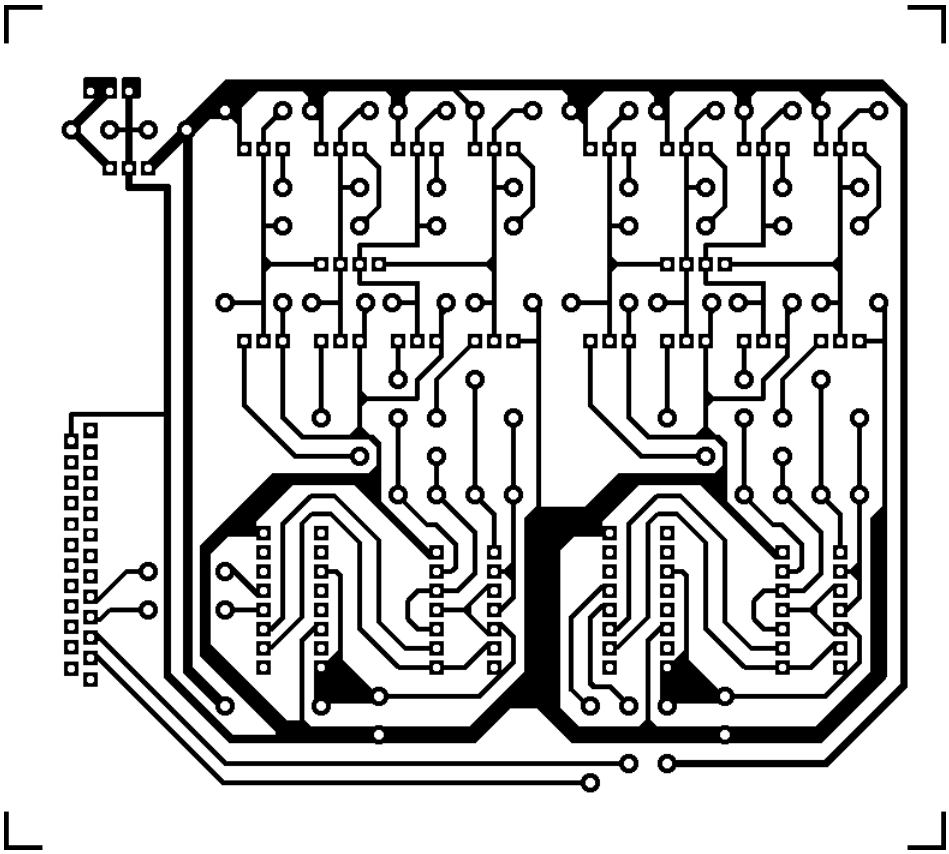
Piirilevyn teko

Kun kytkentäkaavio oli testattu toimivaksi, oli aika juottaa komponentit paikalleen. Komponentit saadaan toimimaan yhteistyössä juottamalla ne piirilevylle. Piirilevyn tekeminen oli seuraava askel. Yksinkertaisissa kytkennöissä kannattaa hyödyntää valmista koekytkentälevyn tapaan toimivaa lasikuitulevyä. Levyn juotospuolen kupariliuskat yhdistävät komponentit toisiinsa ja loput kytkennät toteutetaan hyppylangoilla. Toinen vaihtoehto on suunnitella piirilevy varta vasten juuri kyseiselle kytkennälle.

Piirilevyn valmistus vaikutti mielenkiintoiselta ja palkitsevalta työltä, joten ohjaaja Johanssonin ehdotuksesta päädyin toteuttaman piirilevyn alusta asti. Olisin takuulla päätynyt käyttämään valmista piirilevypohjaa ilman ohjaajani opastusta, sillä en ollut tätä ennen tiennyt piirilevyjä voitavan valmistaa syövyttämällä niitä yksilöllisesti erilaisia kytkentöjä varten.

Ensiksi täytyy kytkentäkaavio muuttaa piirilevymalliksi, jossa komponenttien symbolit on korvattu tarvittavilla rei'illä ja kaikki johtimet kuvattu tarkasti yhdistämään näitä reikiä. Tässä vaiheessa johtimet eivät enää saa risteytyä, sillä tosielämässä tämä aiheuttaisi oikosulun.

Piirilevymallin voi suunnitella Eagle-ohjelmalla, joka jopa osaa reitittää osan komponenteista automaattisesti. Mallin voi piirtää myös millä tahansa piirto-ohjelmalla. Bittikartta- tai vektorigrafiikka käy kumpikin, joskin vektorigrafiikalla saadaan toteutettua kuva hieman näppärämmin, kun pikseleiden koosta ei tarvitse välittää. Luonnollisesti piirilevymallin voi piirtää myös käsin tussilla, mikä kuitenkin ei laajoissa ja monimutkaisissa kytkennöissä ole suositeltava vaihtoehto.



Piirilevymalli valmiina valotettavaksi.

Koin Eaglen piirilevymallin toteuttamiseen tarkoitetun Board-osion suurena pettymyksenä ja päätin toteuttaa piirtämisen muilla ohjelmilla. Photoshop osoittautui kaikkein kätevimmäksi vaihtoehdoksi. Siitä huolimatta, etten saanut tehtyä piirilevyä vektorigrafiikkana, olin tyytyväinen lopputulokseen. Myös Windowsin Paint-ohjelma paljastui – suurena yllätyksenä – käyttökelpoiseksi apuvälineeksi.

Mallin piirtäminen alusta alkaen käsin pikseli kerrallaan oli aikaa vievää, mutta palkitsevaa. Helpotusta urakalle toi Bebekin askelmoottoriohjeessa ollut esimerkinomainen piirilevy, jonka rakennetta hyödynsin. Lopulliseen kuvaan tuli kahta eri moottoria ohjaavan kytkennän lisäksi rinnakkaisportin D-liitin sekä regulaattori kondensaattoreineen. Piirtäessäni pyrin minimoimaan hyppylankojen tarpeen, mutta lopulta päädyin käyttämään kuutta eri hyppyä. Moottoreihin menevät liittimet saivat jäädä levyn keskelle komponenttien joukkoon. Komponenttien jaotus on tavallisesti tehty niin, että vierekkäiset jalat ovat 0,1 tuuman päässä toisistaan. Poikkeuksen teki D-liitin, jonka rivissä olevien liitinten välimatka on 0,108 tuumaa (rivien etäisyys 0,1 tuumaa).

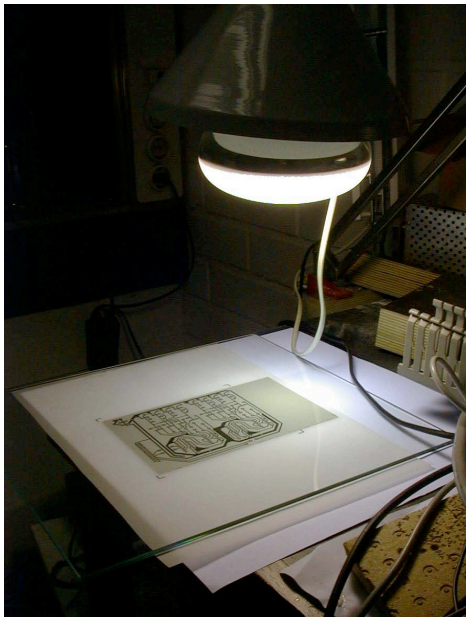
Levyn valmistus

Piirilevy koostuu ohuesta eristinmateriaalista tehdystä levystä (usein lasikuitua), joka on pinnoitettu ohuella kuparikerroksella. Tieto- ym. koneiden piirilevyt ovat teollisesti tuotettuja. Niissä voi olla kymmenenkin erillistä johtavaa kuparikerrosta levyn sisällä. Kuparin tarkoituksena on muodostaa johtimet piirilevylle juotettujen komponenttien välille. Harrastajien piirilevyissä kerroksia on yhdestä kahteen. Kuparin poisto toteutetaan teollisissa levyissä joko jyrsimällä tai syövyttämällä. Kotioloissa taas syövyttäminen on ainoa järkevä vaihtoehto. Harrastajan kannattaa pitäytyä yksinkertaisissa piirilevyissä. Kahdenkin kerroksen tekeminen on yllättävän hankalaa, sillä jo levyn eri puolten kohdistaminen voi tuottaa ongelmia.

Kuparoituja lasikuitulevyjä saa ostaa alan liikkeistä. Ylimääräisen kuparin poistaminen syövyttämällä vaati johtimiin tarvittavan kuparin suojaamisen syövytysvaiheessa. Eristämisen syövytysaineesta voi yksinkertaisimmillaan toteuttaa teipillä tai tussilla. Näppärimmin piirilevyn valmistus sujuu valotusmenetelmällä.

Valotus

Piirilevyn kupari ei ole millään tavalla valoherkkää, eikä kuparipinnasta synny piirilevyä, vaikka kuinka valottaisi. Kupari pitää aluksi pinnoittaa valoherkällä lakalla. Tällaisia lakkoja on olemassa kahta laatua: positiivi- ja negatiivilakkoja. Positiivilakka suojaa valottamattomat osat ja negatiivilakka valottuneet. Lakan voi ruiskuttaa itse levylle tai levyn voi ostaa valmiiksi tasaisella lakkakerroksella.



Valotuksessa tarvitaan maskia, joka suojaa levyllä pimentoon jääviä osia (johtimet ja juotostäplät). Käytimme positiivilakkaa ja maskissa oli johtimet merkitty mustalla. Maskin sai tulostettua erityisvalmisteiselle kalvolle (Laser Film; Farnell), jossa lasertulostimen musta todella on mustaa.

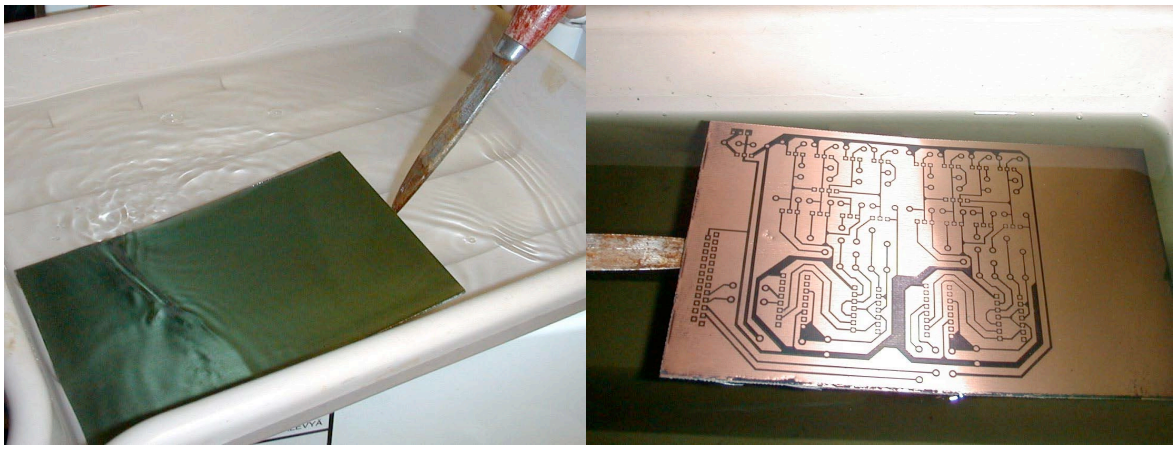
Piirilevyaihion koko oli jo suunnitteluvaiheessa otettu huomioon, joten maskin kuva peitti hyvin 10 cm x 16 cm kokoisen aihion toisen reunan ulottuen reippaasti yli levyn puolivälin. Aihion suojana ollut suojakalvo poistettiin ja maski asetettiin aihion lakkakerroksen päälle. Painoksi laitettiin lasilevy, joka piti maskin paikallaan.

Valotuksessa tarvitaan ultraviolettivaloa. Tavallinen hehkulamppu ei kelpaa ja loisteputkivalossa toimenpide kestäisi luvattoman kauan. Ohjaajani ei ollut ensimmäistä kertaa tekemässä piirilevyä ja hänen varastoistaan löytyi valotuksessa tarvittava UV-lamppu (300 W). UV-valo ei ole terveydelle hyväksi, joten oleilua ultraviolettivalossa tulee välttää.

Valonlähde täytyy sijoittaa melko lähelle piirilevyä niin, että levylle osuu valo tasaisesti. Valotusvaihe on erittäin tärkeä piirilevyn onnistumisen kannalta. Liian lyhyt valotusaika jättää levylle ylimääräistä kuparia, ja liian pitkä valotus puolestaan aiheuttaa johtimien syöpymistä. Epätasainen valotus aiheuttaa kumpaakin vaivaa. Onnistuimme pilaamaan pari piirilevyä, mutta syy tähän saattoi hyvinkin olla kauan sitten umpeutunut lakan parasta ennen -päivämäärä. Sopivaksi valotusajaksi voitaneen kuitenkin sanoa 3 – 8 min.

Kehitys

Valotuksen jälkeen täytyy piirilevyn ylimääräinen lakka saada irtoamaan. Lakka liuotetaan kuparin pinnasta niistä kohdin, mistä kupari halutaan syövyttää pois. Lakan liuottamiseen käytetään kehityksessä voimakkaasti emäksistä natriumhydroksidi- eli lipeäliuosta. Noin litraan vettä sekoitetaan arviolta 10 g natriumhydroksidia. Piirilevy asetetaan liuokseen kuparipuoli ylöspäin, jotta liukenemista olisi helppo seurata. Levyä tulee liikutella ja tarpeen tullen lakkaa hangata liukenemisen edistämiseksi. Levyn toisen pään siirteleminen ylös–alas vaikutti tehokkaalta apukeinolta.



Kehitys eri vaiheissa. Vasemmassa kuvassa lakka ei ole vielä irronnut. Oikealla alkavat piirilevyn piirteet olla näkyvissä.

Levyn kehitys on valmis, kun lakka on poistunut muilta alueilta paitsi johdin- ja juotoskohdista. Jos levyä uitetaan liian kauan, syntyy johtimiin katkoksia. Liian lyhyt kehitys jättää ylimääräistä kuparia levyn pinnalle, jolloin oikosulkujen riski kasvaa ja ulkonäkö kärsii. Valmis levy nostetaan astiasta ja huuhdellaan vedellä.

Tässä vaiheessa voi jo valotuksen onnistumista tarkastella ja mahdollisia virheitä parhaansa mukaan yrittää korjailla. Levylle jäänyt lakka näkyy sopivassa valossa tummina täplinä. Ylimääräisiä täplien poistaminen raaputtamalla ei ole aivan yksinkertaista.

Kehityksessä tulee kuulemma aina virheitä, joten taisimme onnistua melko hyvin. Johtimiin ei tullut katkoksia, eikä oikosulkuja syntynyt. Ylimääräistä lakkaakin jäi vain vähän.

Kehitys kannattaa tehdä tilassa, jossa saa sotkea. Lisäksi lipeäliuosta käsiteltäessä on syytä ottaa huomioon vahva emäksisyys. Lipeän joutuminen silmiin ei ole mukava kokemus. Liuos tulee hävittää kaatamalla se voimakkaasti laimennettuna viemäriin.

Syövytys

Ylimääräinen kupari täytyy vielä syövyttää pois levyltä. Lakan tarkoitus on suojata niitä kupariosia, joiden halutaan jäävän levylle. Syövytyksessä käytettiin ferrikloridia ($FeCl_3$). Ferrikloridin haittapuolena on sottaava vaikutus. Ferrikloridiliemi näyttää lähinnä nestemäiseltä ruosteelta. Sillä on ikävä tapa ruostuttaa lähiympäristön rautaesineet. Rautasotkun sijaan voi käyttää myös muita syövyttäviä aineita. Ainakin natriumpersulfaatti kuuluu toimivan.

Ferrikloridia liuotettiin veteen painosuhteessa 1 : 1. Itse asiassa litku oli vanhaa, sillä kerran käytetty liuos voidaan käyttää uudelleen seuraavaa piirilevyä tehtäessä. Uudelleenkäyttö käy niin pitkään kuin nesteessä on ytyä jäljellä tehokkaaseen syövytykseen. Syövytystä voi tehostaa lämmittämällä allasta noin 50 – 60 °C:een.

Syövytys toteutettiin lämmittävässä vesihauteessa, jossa lämpötila oli pyöreät 60 °C. Syövytyksen toimimista tarkkailtiin silmämääräisesti ja levyä huiluteltiin altaassa. Syövytys kestää ferrikloridiliuoksen vahvuudesta, lämpötilasta ja levyn liikuttelusta riippuen noin 5 minuutista ylöspäin.

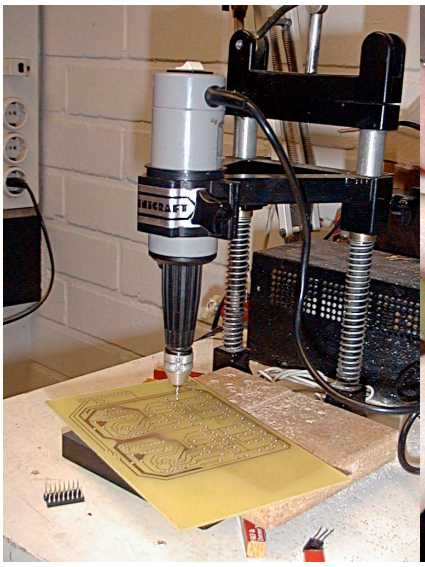
Levyn syövyttyä valmiiksi se huuhdeltiin ja ylimääräinen lika hangattiin pois asetonilla. Tämän jälkeen saattoi tarkastella kauniin vihreäksi paljastuneen piirilevyn yksityiskohtia. Jos tässä vaiheessa olisi johtimissa paljastunut pahoja katkoksia, olisi korjaustoimenpiteenä tarvinnut juottaa ylimääräisiä johdinsäikeitä levylle. Pahoja katkoksia ei onneksi löytynyt.

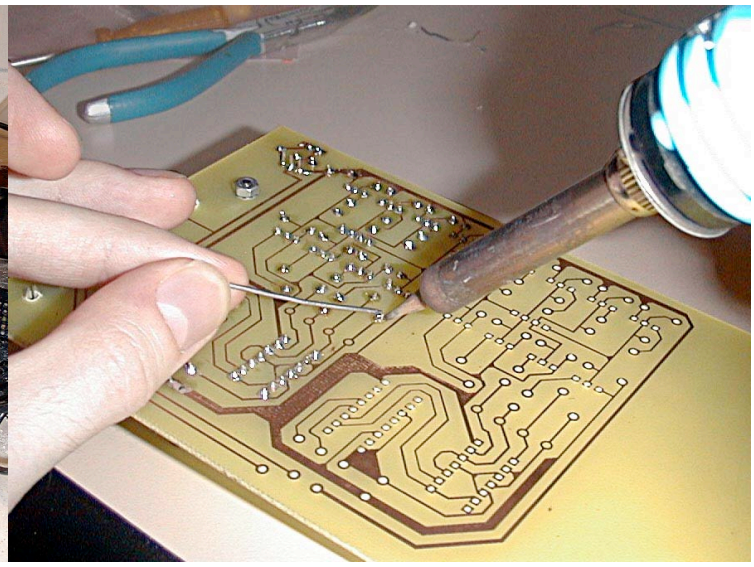
Syövytyksessä käytettäviä liuoksia ei saa kaataa viemäriin. Ne syövyttävät putkistoa ja ovat terveydelle haitallisia. Kun liuosta ei enää halua hyödyntää, pitää se toimittaa ongelmajätehuoltoon.

Reikien poraus

Jotta piirilevylle voisi liittää komponentteja, tarvitsee niitä varten porata sopivat reiät. Levylle porattavat reiät olivat halkaisijaltaan 0,7 mm ja 0,9 mm. Suurempaa reikäkokoa vaativat lähinnä transistorit sekä regulaattori. Vastukset, diodit ym. sopivat hyvin 0,7 millimetrin reikiin. Reikien poraaminen kävi näppärästi telineeseen liitetyllä pienoisporalla.

Piirilevyn pinnan voi suojata tinapinnoitteella. Tämän kalliin toimenpiteen hyöty on rajallinen, enkä itse siihen ryhtynyt. Toinen suojaustapa on juottamisen jälkeen pinnoittaa levy lakalla. Tämäkään ei tuntunut tarpeelliselta, joten levy jäi alkuperäiskuosiinsa.



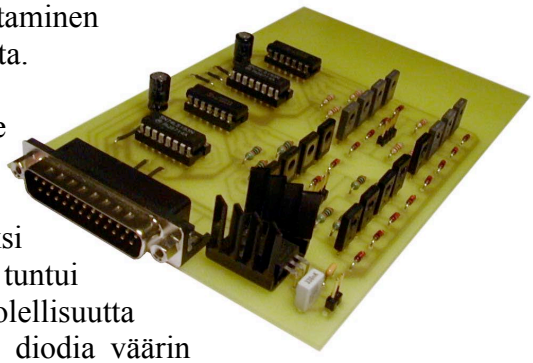


Vasemmalla reikien poraamista pienoisporalla. Oikealla komponenttien juottamista levylle.

Juottaminen

Piirilevyn rungon valmistuttua oli edessä komponenttien juottaminen levylle. Juottaminen tuntui suurelta askeleelta, sillä en ollut käsitellyt kolvia vuosikausiin. Vanha sinertävänturkoosi kolvini löytyi kaapin perältä muuttolaatikkoon pakattuna. Löysin myös sopivan rullan tinaa, tinaimurini ja lajitelman pihtejä. Kun vielä sain kiinnitettyä pöytään pienoisruuvipenkkini, oli koko juotto-operaatioon tarvittava arsenaali koossa.

Juottaminen tarkoittaa komponenttien liittämistä piirilevylle sulatettua tinaa käyttäen. Tinaa lämmitetään kolvilla, joka koostuu kahvasta sekä kuumasta kärjestä. Tina on yleensä ohuena lankana josta sitä saadaan sulatettua komponenttien liitoskohtiin. Juottaminen on oma taiteenlajinsa. Tärkeää on saada juotos onnistumaan niin, että liitos on tukevasti kiinni ja johtaa hyvin. Kylmäjuotosten välttämiseksi tulee varmistaa molempien toisiinsa kiinnitettävien osapuolten lämpeneminen kolvin vaikutuksesta. Sulaneen tinapalleron ujuttaminen kolvilla liitoskohtaan ei tuota haluttua lopputulosta.



Tein viisaasti, kun hankin herkille mikropiireille holkkikannat. Onnistuin nimittäin juottaessa sulattaman yhden kannoista pahaan kuntoon yrittäessäni saada tinaa tarttumaan liiaksi syöpyneeseen juotostäplään. Juottaminen tuntui mukavan rauhalliselta puuhalta. Suurta huolellisuutta noudattaen en onnistunut juottamaan kuin pari diodia väärin päin. Huomasin ja korjasin tämän välittömästi.

Juottaminen onnistui varsin näppärästi huomioiden vähäinen kokemus juotoshommissa. Jäljestä voidaan olla montaa mieltä, mutta puolustukseksi sanon, että piirilevy toimii vallan

mainiosti. Alussa huolettanut regulaattorin lämpeneminen ei ollut enää ongelmana jäähdytyssiilin asennuksen jälkeen. Siili mahtui hyvin D-liittimen kylkeen.

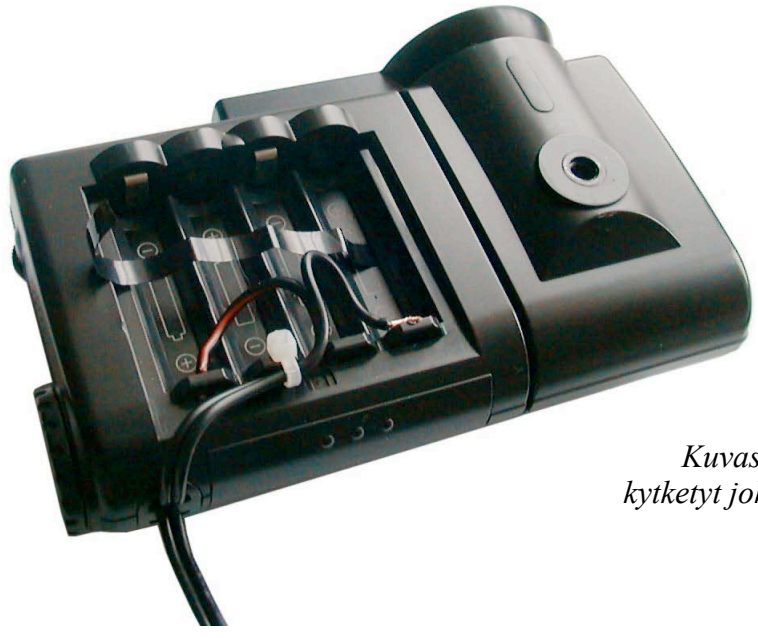
Lopulta kokeillessani piirilevyä se vastoin kaikkia odotuksiani todella toimi. Piirilevyn tekemisestä sanottakoon, että kokemus tuo näissä asioissa taidon. Olisin tuskin saanut minkäänlaista piirilevyä syövytetyksi ilman ohjaajani apua. Kokemus opettaa miltä levyn kuuluu missäkin työvaiheessa näyttää sekä miten pitkään sitä pitää valottaa, liottaa ja syövyttää. Piirilevyn tekeminen oli erittäin opettava ja palkitseva työvaihe kokonaisuudessaan. Varsinkin kun lopulta ymmärsi miten ja miksi kaikki tehtiin.

Myös piirilevyn valmistamisessa ohjaajallani oli suuri rooli. Kuvassa Stefan Johansson verstaassa.

Kameran virransyöttö

Kamera on tarkoitettu käytettäväksi AA-kokoisilla sormiparistoilla. Alun perin kameran mukana toimitetut neljä akkupatteria toimivat 1,2 Voltin jännitteellä. Kameraan sopivat kuitenkin aivan tavalliset paristot. Kamera on aina ollut virtasyöppö ja pattereita on joutunut vaihtamaan tai akkuja lataamaan alle puolen tunnin käytön jälkeen.

Agfa ePhoto 1280 -kamerassa on ulkoiselle virtalähteelle oma liitin takapaneelissa. Tästä ei ole kuitenkaan kummoista hyötyä, sillä liitin ei edusta mitään tunnettua standardia, vaan on erikokoisten liitinten välimuoto. Lisäksi ei vaikuta siltä, että liitin todella olisi kytketty kameran sisällä mihinkään. Tästä viestii myös se, ettei kameraan ole tiettävästi saatavilla edes virallista ulkoista virtalähdettä.



Kuvassa kameran paristotilaan kytketyt johtimet.

Mietittyäni erilaisia vaihtoehtoja päädyin liittämään virtajohdon suoraan kameran paristoliittimiin. Epäsiistein ja samalla luotettavin tapa oli juottaa johto paikalleen. Näin se ei ainakaan pääsisi irtoamaan vahingossa. Kameran neljä paristoa on kytketty sarjaan (4 * 1,5 V = 6 V), joten juotin johdon kiinni ensimmäiseen ja viimeiseen paristokoskettimeen. Johdon fyysisen kuormituksen estämiseksi kiinnitin sen pienellä nippusiteellä kameraan ja vein kaapelin rannelenkin reiän läpi vielä varmuuden vuoksi.

Laser

Laser on käsitteenä tuttu lähes jokaiselle. Elokuvat ovat tehneet lasersäteistä tunnettuja laser-miekkojen, erilaisten tuhosäteiden ja muun kautta. Suurin osa lasereita koskevista totuuksista, joita tiedämme, eivät pidä paikkaansa alkuunkaan.

Ajatuksena laser on varsin yksinkertainen asia. Laserin (*Light Amplification by Stimulated Emission of Radiation*) valo on koheranttia, jolloin kaikki valoaallot ovat samassa vaiheessa. Valo on myös hyvin tarkkaan yksiväristä eli monokromaattista. Tämän takia laserin valon taajuus on vakio. Valon intensiteetti on suuri, ja se on yhdensuuntaista, jonka vuoksi lasersäde etenee pitkiä matkoja hajoamatta.

On olemassa monentyyppisiä lasereita. Etäisyysmittauksessa käytetään ns. osoitinlaseria (Laser pointer). Osoitinlaserin toiminta perustuu yleensä pieneen laser-diodiin. Osoitinlasereita käytetään yleensä karttakepin korvaajana.

Erilaisia osoitinlasereita. Toinen vasemmalta on päättötyössä käytetty laserosoitinmalli. Äärimmäisenä vasemmalla sama laser katkaistuna sopivan mittaiseksi.

Lasereista puhuttaessa mainitaan yleensä myös turvallisuusnäkökohdat. Suomessa *Säteilyturvakeskus* on julkaissut viralliset turvaluokitukset kaikille laserlaitteille. Laserit jakautuvat neljään pääluokkaan (1-4) ja joihinkin alaluokkiin. Lasereiden tehoa merkitään yleensä Wateissa (W).

Luokan I laser on niin heikko, ettei se aiheuta vaaraa ihmissilmälle (esim. CD-soittimissa). **Luokan II** (noin 1-5 mW) laserit puolestaan ovat hieman tehokkaampia, mutta silmän sulkeminen refleksinä auttaa suojaamaan silmää vaurioilta. Luokkaan II

kuuluvat suurin osa osoitinlasereista. **Luokkaan III** kuuluvat tehokkaimmat sallitut osoitinlaserit, ja ne voivat aiheuttaa silmään pahoja vaurioita. **Luokan IV** laserit voivat polttaa ihoa ja sytyttää puun palamaan. Niitä käytetään leikkauslasereina ja joskus valotehosteina. Päättötyössä käytetty laser on osoitinlaser. Se on teholtaan 5 mW ja kuuluu luokkaan IIIa.

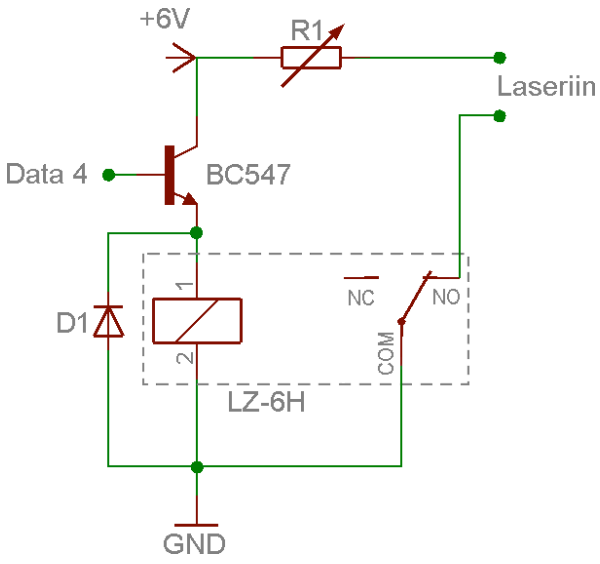
Laserin virransyöttö ja ohjaus

Osoitinlaserit toimivat lähes poikkeuksetta pattereilla. Käyttämäni laser sai tavallisessa käytössä virtanansa kahdesta AAA-paristosta (2 * 1,5 V). Koska halusin ulkoistaa laserin ohjauksen, oli laser liitettävä muun järjestelmän yhteyteen. Järjestelmän perusjännitteeksi valittu 6 Volttia oli saatava lähemmäs laserille tuttua kolmea.

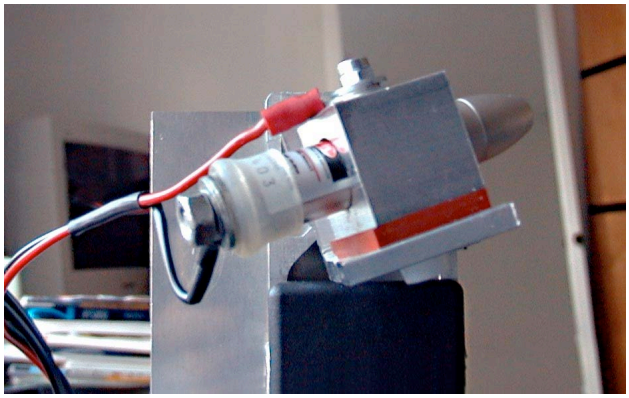
Jännitteen saaminen laserille sopivaksi hoitui yksinkertaisella vastuskytkennällä. Tämä ei välttämättä ole kovin tyylikäs tapa hoitaa virransyöttö, mutta vaikuttaa toimivan. Laserin virrantarpeesta ei ollut mitään käsitystä, joten sopiva vastusarvo määritettiin säätövastuksen avulla. Sopivaksi arvoksi saattoi todeta noin 200 Ω vastuksen.

Lisäksi päätin varsin myöhäisessä vaiheessa haluavani pystyä ohjaamaan laseria tietokoneelta. Tällöin olin jo tehnyt askelmoottoreiden piirilevyn valmiiksi, joten ei auttanut enää mennä lisäämään laserin ohjaamiseen tarvittavia komponentteja levylle. Päätin kokeilla laserin ohjausta rinnakkaisporttiin liitetyn releen avulla. Rinnakkaisportissa oli vielä vapaana neljä datalinjojen ulostuloa.

Löysin kaupasta sopivan releen ja viritin toimivan järjestelmän koekytkentälevylle. Vasta kytkettyäni ohjaussysteemin rinnakkaisporttiin minulle hiljalleen valkeni, että rinnakkaisportin antamat 20 mA eivät riitä releen ohjaamiseen. Tämä ongelma ratkesi käyttämällä transistoria releen ohjaamiseen siten, että npn-transistorin kanta oli kytketty rinnakkaisporttiin. Transistori teki releen oikeastaan tarpeettomaksi, sillä laserin olisi voinut kytkeä suoraan kiinni siihen. Olin kuitenkin jo hankkinut releen, ja toisaalta olisin jäänyt kaipaamaan sen kodikasta naksumista. Päädyin käyttämään relepohjaista järjestelmää, jonka olin voinut todeta toimivaksi. Tehtyäni jo yhden version piirilevystä päätin muuttaa sitä hieman. Löysin pienen säätövastuksen (trimmeri) pyörimästä nurkista ja päätin mahdollistaa laserin säätämisen. Laserin voimakkuutta voi ruuvimeisseliä käyttäen säätää vastuksesta. Sopiva arvo on noin 180–210 Ω.



Merkintä	Komponentti	Koodi
R1	Vastus (trimmeri)	2 kΩ
D1	Diodi	1N4148
	Transistori (npn)	BC547
	Rele	LZ-6H

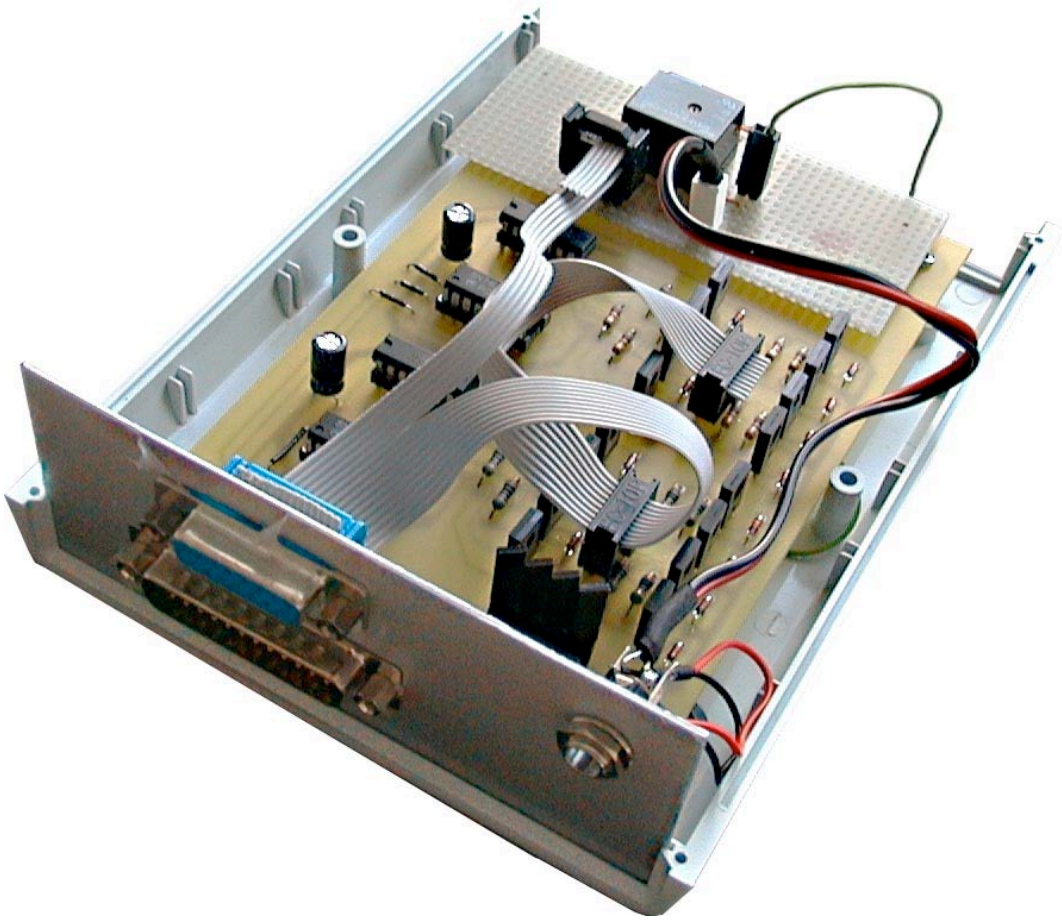


Laserin kiinnityksessä on huomioitu virransyöttö. Laseria pitävä alumiinikuutio on eristetty muusta laitteesta kumimatolla ja muoviruuvilla.

Kotelointi

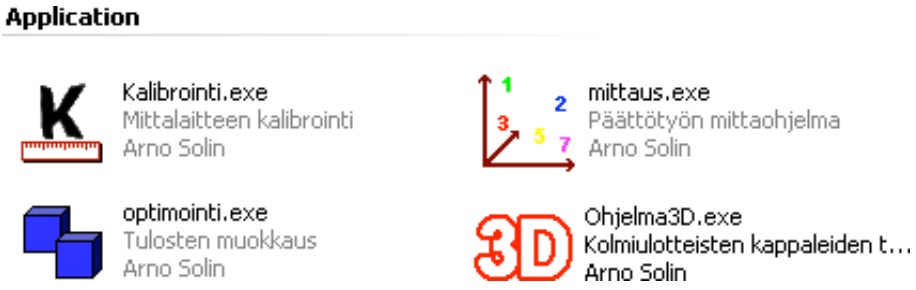
Piirilevyt on aina hyvä suojata koteloon, jotta herkät komponentit eivät vahingossakaan vahingoitu. Sopivan kotelon löytäminen osoittautui yllättäen haastavaksi tehtäväksi. Turun lähialueiden liikkeistä ei löytynyt sopivaa koteloa. Internetissä löytyi YEInternationalin (www.yleiselektroniikka.fi) sivuilta sopiva kotelo. Tukkumyyjän lähimmäksi edustajaksi osoittautui Turun Komponenttipiste ja tilasin heidän kauttaan alumiinisivuin varustetun muovikotelon.

Kotelon saapuminen kesti useita viikkoja johtuen maahantuojan toimitusvaikeuksista. Päättötyön palauttamispäivämäärän lähestyessä tuli kiire saada mittausta hoitava elektroniikka koteloitua. Onneksi kotelon etupaneelin liittimien kiinnittäminen onnistui näppärästi. Kaapelien juottaminen kiinni ja sopivien kytkinten löytäminen osoittautui sekin melko helpoksi. Kotelon sisäiset liitännät on toteutettu lähinnä lattakaapelin avulla, jolloin juottamista saattoi välttää. Mittalaitteeseen menevät signaalit kulkevat 15-napaisen D-liittimen kautta. Mittalaitteen alaosan askelmoottori voidaan irrottaa muusta järjestelmästä. Tämä on tarpeen, jos mittalaitteen haluaa purkaa osiin esimerkiksi kuljetuksen ajaksi.



Laitteen kotelo enää paria ruuvia vailla. Takapaneelin D-liittimet on varattu rinnakkaisportille ja mittalaitteen liitännälle. Oikean reunan liitin menee virtalähteeseen (6 V, DC). Kamera kytketään suoraan tietokoneen sarjaporttiin.

Ohjelmointi



Pohjustus

”Ohjelmointi pelasti minut elämältä.” Suomalainen ohjelmoijille suunnattu nettisivu oli valinnut sloganinsa osuvasti. Lienee kulunut jo vuosia siitä, kun näin tuon ensi kertaa, mutta se nousi jälleen mieleeni päättötyötä tehdessä.

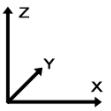
Ohjelmointi todella voi parhaimmillaan tai pahimmillaan ”pelastaa elämältä”. Päättötyön ehdottomasti rankin ja aikaa vievin osuus oli nimittäin juuri ohjelmointi ja kaikki siihen liittynyt tekeminen. Siinä missä mekaniikan ja elektroniikan parissa puuhastelu oli vaihtelevaa ja uutta, oli ohjelmointi verrattain pitkäveteistä. Tästä syystä ohjelmointiin liittyviä kuvia ei paljon ole tarjolla. Kaikki kuvat olisivat suurin piirtein samanlaisia: istumista tietokoneen ääressä. Vaihtelua toisi korkeintaan vaihteleva vuorokaudenaika ja ajoittain vaihtuva tietokone.

Muistan ensimmäiset kirjoittamani ohjelmat. Aloitin ohjelmoinnin ollessani ehkä kymmenvuotias. Parissa vuodessa opin perusasiat ohjelmoinnista. Pidin sitä yleissivistävänä ja hyödyllisenä taitona. Sitten lopetin ohjelmoinnin lähes kokonaan. Oikeastaan syvennyin uudelleen kunnolla ohjelmoinnin maailmaan vasta päättötyön myötä. Saatoin jälleen kokea lapsenomaista innostusta pienen pulman ratkaisemisesta. Ihastuminen pieniin pulmakohtiin on tehnyt päättötyöni tästä osa-alueesta melko atomistisen, hyvin pieniä kokonaisuuksia kerrallaan käsittelevän.

Vaikka olen nimennyt päättötyön kolmannen osuuden yksinkertaisesti ohjelmoinniksi, kattaa tämä osa paljon muutakin kuin pelkkää koodia ja rakenteiden analysointia. Päättötyön mittalaitteen ohjaaminen, tulosten saaminen sekä niiden käsittely on kaikki keskitetty tähän osuuteen. Varsinainen ohjelmointiin liittyvä koodi on suurimmaksi osaksi liitteinä. Koodin sijaan kirjallinen osuus keskittyy vastaamaan lukijalle heränneisiin kysymyksiin miksi ja miten eri asiat on toteutettu.

Tietokoneen ohjelmointi vaatii ohjelmointikieltä. Päättötyön ohjelmat on kirjoitettu Object Pascal (Delphi) -kielellä. Kieli on perusrakenteeltaan selkeää ja siksi melko helppotajuista. Runsaiden kommenttien tarkoitus on auttaa lukijaa hahmottamaan koodin tarkoitus. Lähtökohtanani on ohjelmia kirjoittaessa miltei Stalinin kanssa yhtenevää ajatusmalli resurssien riittävyydestä. Siinä, missä Stalin ei säästellyt kansalaisiaan, en minäkään säästele koneen resursseja. Poikkeuksellisesti lähtökohtana ei ole kirjoittaa mahdollisimman nopeaa koodia, ei mahdollisimman pientä koodia, vaan mahdollisimman selkää koodia. Tämä saattaa näkyä joinakin niin sanottuina purkkaviritelminä.

Asioita käsiteltäessä olen viljellyt sekä suomen- että englanninkielisiä termejä sekaisin. Tähän on useita syitä, joista tärkein on tottumiskysymys. Tietokonemaailma on pitkälti englanninkielinen, vaikka ohjelmistoista on olemassa Matti Meikäläisille suunnattuja suomenkielisiä versioita. Vaikka termistöä on käännetty suomeksi, on suomennoksien viljely joskus kyseenalaista. Usein herää kysymys, mitä järkeä on käyttää suomenkielistä



termiä, jota kukaan ei tunne, kun englanninkielinen vastine on kaikille tuttu. En tiedä kielitoimiston suosittelemaa linjaa, mutta olen suomentanut suuren osan termeistä usein kuitenkin jättäen alkuperäisen muodon näkyviin.

Päättötyön ohjelmoinnin eri osa-alueiden käsittelyssä en noudata mitään tiettyä järjestystä. Jokainen osa-alue on tasa-arvoisen tärkeä laitteen toiminnan kannalta. Käsittelyjärjestys noudattaa pikemminkin järjestystä, jossa olen eri kokonaisuudet toteuttanut. Kaikkia osia ei ole loppuun saakka viimeistelty, mutta tarkoitus onkin antaa laitteen toiminnasta kokonaiskuva. Tästä syystä ohjelmista on turha etsiä mitään ohjaavia velhoja tai ärsyttäviä animoituja tukitoimintoja. Ohjelmat on tehty toimimaan, jos käyttäjä niitä ymmärtää.

Vaikka nyt olen saanut ohjelmoitua eri palaset toimiviksi ja päättötyön ikään kuin valmiiksi tuskin maltan pitää sormiani erossa joistakin koodin osista. Ainakin viimeiseksi osuudeksi jäänyt 3D-ohjelma tulee kokemaan rajuja päivityksiä heti tilaisuuden tullen.

Alan yhä paremmin ymmärtää ohjelmoinnin pelastavaa vaikutusta. Niin taisivat ohjelmointisivuston ylläpitäjätkin, sillä slogan poistettiin seuraavan päivityksen yhteydessä. Taisi osua turhan lähelle käyttäjiä.

Ohjelmointikieli

Jotta tietokone saadaan toimimaan halutulla tavalla, on sille annettava käskyjä. Käskyt ovat konekielisiä. Konekieli on ihmiselle kovin hankalaa, joten on kehitetty erilaisia ohjelmointikieliä edesauttamaan ihmisen ja tietokoneen yhteistoimintaa. Ohjelmointikielillä kirjoitetaan ohjelma, jossa kerrotaan tietokoneelle mitä tehdä. Kääntäjä (compiler) muuttaa koko ohjelman konekieliseksi, minkä jälkeen alkuperäistä koodia ei suorittamiseen enää tarvita (esim. Windowsin EXE-tiedostot).

Ohjelmointikieliä on lukematon määrä. Eri kielet on tarkoitettu erilaisiin tehtäviin ja ne toimivat eri periaattein. Tässä ei ole tarkoitus syventyä tutkimaan ohjelmointia syvällisesti, sillä kyseessä on erittäin laaja ala, joka liittyy kaikkeen tietokoneilla tehtävään. Kielen valinta riippuu aina siitä, mitä ollaan tekemässä.

Kaikkein yleisimpiä ohjelmointikieliä lienevät Basic, Pascal ja C sekä näiden eri johdannaiset. Näistä kolmesta on olemassa erilaisia muunnoksia, mutta rakenne ja toimintaperiaate nojaavat emokieleen. Näillä kaikilla kolmella on hyviä ja huonoja puolia. Basic mielletään usein hyväksi kieleksi aloittajalle, Pascal selkeäksi ja C tehokkaaksi.

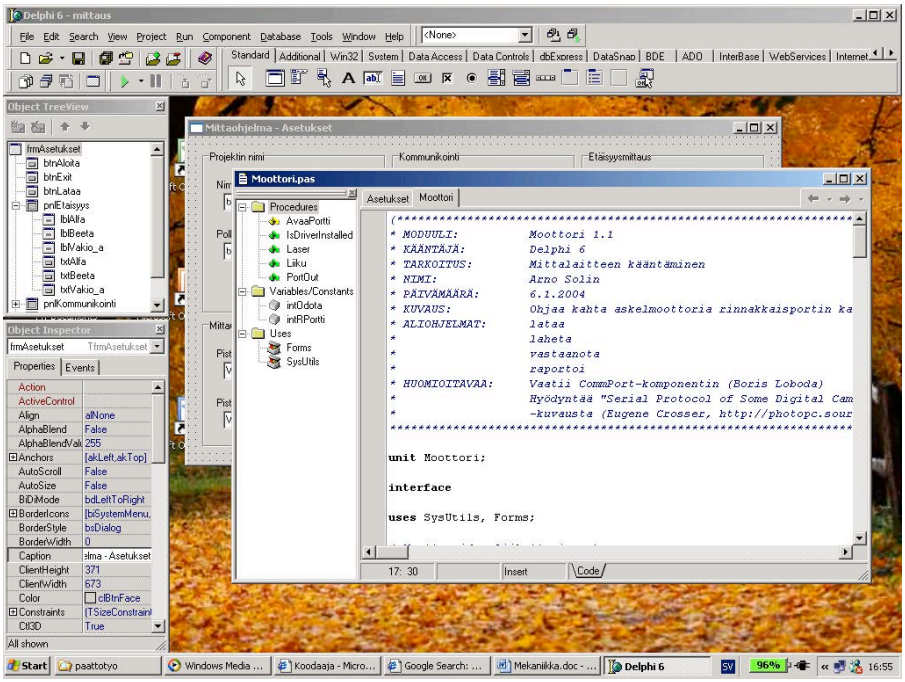
Itse olen aloittanut ohjelmoinnin Basicilla (Visual Basic). Sittemmin olen tarpeen tullen vaihdellut kieliä kuin paitaa ja voin sanoa osaavani alkeet reilusta kymmenestä kielestä. Päättötyötä aloittaessani ajattelin ohjelmoivani työn ainakin osittain Basicilla. Ohjelmoituani muun muassa kuvien siirron kamerasta toimiman Visual Basicissa tulin siihen tulokseen, että kieli oli väärä. Basicin mahdollisimman yksikertaiskiksi tehdyt rakenteet kävivät harvinaisen monimutkaisen näköisiksi koodimäärän kasvaessa. Suurempi ongelma oli hitaus. Basicin rakenne tekee siitä hitaan muun muassa kuvien käsittelyssä ynnä muissa laskentatehoa vaativissa toimenpiteissä.

Pascal

En pitänyt Basicia hyvänä vaihtoehtona, joskin silläkin olisi saanut tehtyä tarvittavat ohjelmat. Vaihtoehtoina pidin C:tä ja Pascalia. C olisi ollut hieman Pascalia nopeampi vaihtoehto, mutta Pascal oli puolestaan C:tä selkeämpää. Pascalia pidetään erittäin selkeänä kielenä, ja sitä onkin käytetty ohjelmoinnin opettamiseen jo pitkään. Lisäksi se ei häviä nopeudessa paljoakaan C:lle. Ottaen huomioon tarkoitukseni tehdä mahdollisimman selkeä ohjelma on Pascal hyvä vaihtoehto. Pascalia on mahdollisuus ymmärtää tuntematta kieltä juuri ollenkaan. Selkeyden kriteerinä oli tehdä koodista niin selkeää, että itse ymmärtäisin vielä parin vuoden kuluttua miksi ja miten olin ratkaissut erilaisia kohtia. Tietysti olisi pelkkää plussaa, jos joku muukin ymmärtäisi koodiani.

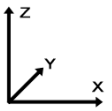
Tarkoitus ei ole tehdä opasta Pascal-ohjelmointiin, mutta pari huomiota kielen rakenteesta haluan tehdä. Pascal voidaan jakaa kahteen leiriin: Pascaliin ja Object Pascaliin. Tämän lisäksi on olemassa vielä Turbo Pascal, joka on laajennettu versio peruspascalista.

Sveitsiläinen Niklaus Wirth kehitti Pascalin vuonna 1970. Tarkoituksena oli luoda mahdollisimman selkeä ohjelmointikieli opetuskäyttöön. Pascal on hyvin helppo kieli. Jos englantia ymmärtää, voi Pascaliakin helposti ymmärtää ilman laajaa perehtymistä kieleen. Tästä syystä sen perusteet on helppo oppia, mutta kieli samalla tehokas ja monipuolinen. Pascalin sanotaan olevan lähempää C/C++:aa kuin Basicia. Pascal on välimuoto: se on sekä tehokas että selkeä.



Borland Delphi 6

Delphi on alun perin Object Pascalille kehitetty ohjelmointiympäristö, joka sisältää kääntäjän lisäksi myös muita ohjelmoinnissa tarpeellisia työkaluja. Delphin Object Pascal on hiljalleen vieraantunut alkuperäisestä Pascalista niin paljon, että kieltä on alettu nimittää yksinkertaisesti Delphiksi. Dephiä kehittää Borland (tunnettu myös Inprisenä). Delphi oli alun perin suunnattu Microsoftin Windows-alustalle, mutta koodia saa käännettyä myös Linuxille (Borlandin Kylix) ja Microsoftin .NET framework -rajapinnalle.



Delphi sisältää useita laajennuksia tavalliseen Pascaliin verrattuna ja sillä saa tehtyä lähes samat asiat kuin C/C++:llakin. Usein sanotaan Delphin olevan Pascalille sama kuin C++ C:lle.

Miten tulkita koodia

Olen pyrkinyt ajan ja jaksamisen mitoissa tekemään koodista mahdollisimman selkeää. Tähän olen pyrkinyt muun muassa kommentoimalla koodia mahdollisimman laajasti. Lukemalla kommentteja ja eri toimenpiteiden selityksiä oletan lukijan ymmärtävän koodin rakenteesta perusasiat.

Kommentteja on kolmenlaisia:

```
{ Tavallinen kommentti on kaarisulkeissa. }
(* Laajat kuvaukset on laitettu alkuperäiseen kommenttiasuun. *)
// Kommentti, joka ei jatku uudelle riville
```

Toinen selkeyttämistä varten tehty toimenpide on ohjelman jakaminen pieniin osakokonaisuuksiin. Sovellusten koodi on jaettu moduuleihin eli eräänlaisiin tiettyyn aihepiiriin liittyviin koodikirjastoihin. Esimerkiksi kameraa ohjaavat funktiot ovat omassa moduulissaan. Moduulit (Delphi Source File, *.pas) ovat liitteenä. Lisäksi sovellusten projektitiedostot (Delphi Project, *.dpr) ovat liitteiden yhteydessä. Sen sijaan lomaketiedostoja (Delphi Form, *.dfm) en ole laittanut liitteiksi. Ne määrittelevät vain ohjelmien ulkoasun, ja sen voi jokainen koodin käyttäjä muokata mieleisekseen.

Kaikkiin tärkeimpiin proseduureihin (funktiot ja aliohjelmat) on alkuun liitetty kuvaus, jossa kerrotaan lyhyesti mm. osion nimi, päivämäärä, tekijä, tarkoitus, parametrit ym. Tämän tarkoituksena on selvittää lukijalle koodipätkän tarkoitus ilman, että välttämättä tarvitsee syventyä itse koodin tutkimiseen.

```
{*****************************************************************************
* FUNKTIO:          Funktion nimi ja versionumero
* TARKOITUS:        Tarkoitus
* NIMI:             Tekijän nimi
* PÄIVÄMÄÄRÄ:       Päivämäärä
* KUVAUS:           Lyhyt kuvaus
* PARAMETRIT:       Parametrit sisään
* PALUUARVO:        Arvo, jonka palauttaa
* HUOMIOITAVAA:     Erityishuomioita
*****************************************************************************}
```

Jotta koodin eri osat erottuisivat toisistaan, on sitä väritetty. Nämä muutokset eivät luonnollisestikaan näy käsiteltäessä koodia tekstimuodossa, mutta ne auttavat hahmottamaan koodin eri osia paremmin. Kielen varatut sanat on lihavoitu, jotta ne erottuisivat paremmin tekstin joukosta. Näitä sanoja ovat muun muassa program, begin ja end. Kommentit ja vakiot on merkitty sinisellä.

Tulin siihen tulokseen, että kirjoitetun koodin värittäminen veisi liikaa aikaa, joten ratkaisin asian kirjoittamalla pienen ohjelman Pascal koodin värittämiseen Php-kielellä säännöllisiä lausekkeita (Regular expressions) käyttäen. Väritys tapahtuu HTML- ja CSS-määrittelyjen avulla.

Muita huomioita

Tämän päättötyön ei ole tarkoitus olla matemaattisesti käsittämätön. Itse asiassa lähes kaiken tässä työssä tarvittavan matematiikan voi ymmärtää pelkällä peruskoulun oppimäärällä. On kuitenkin hyvä huomioida, että tietotekniset merkinnät eroavat joissain kohtia totutuista matemaattisista merkinnöistä, ja tämä voi aiheuttaa sekaannuksia.

Tietotekniikassa käytetään usein muita lukujärjestelmiä kuin tavallista kymmenjärjestelmää. Tässä kohtaa kliseiset ykköset ja nollat astuvat kuvaan. Pienin tietotekniikassa käytettävä arvo on bitti, joka voi saada arvon 1 tai 0.

Bitit voivat muodostaa esimerkiksi 8-bittisiä lukuja. Tällöin bitit ovat kahdeksan sarjoissa, jolloin muodostuu tavu, joka on 0 ja 255 välillä (00000000_2 – 11111111_2). Myös kirjaimia voidaan käsitellä lukuina, sillä merkit ovat vain tietyn merkkikartaston merkkiin viittaavia lukuja.

Binaarijärjestelmän lisäksi tietotekniikassa käytetään yleisesti heksadesimaalijärjestelmää, jossa lukujärjestelmän kantaluku on 16. Tavallisten numeroiden 0–9 lisäksi käytössä ovat kirjaimet A–F, jotka ilmaisevat kymmenjärjestelmän luvut 10–15. Heksadesimaalilukuja merkitään monin eri tavoin tietokoneella. Yleisiä merkintätapoja ovat muun muassa 0x (C), &H (Basic) ja $ (Pascal).

Kymmenjärjestelmä	**Binaari**	**Hex**
1	1_2	1_{16}
8	1000_2	8_{16}
16	10000_2	10_{16}
255	11111111_2	FF_{16}

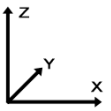
Tietokone varaa kullekin muuttujalle tietyn määrän bittejä muistista, joihin muuttujan arvon on mahduttava. Tästä syystä on muuttujaa määriteltäessä otettava huomioon tarvittavan muistin määrä. Oma lukunsa ovat luvut, jotka eivät ole kokonaislukuja. Murto- ja desimaalilukujen yhteydessä tarvitaan liukulukumuuttujia, joissa luvun merkitsevien numeroiden määrä on sama, mutta desimaalipilkku liikkuu numerosarjaa pitkin.

Suomessa pilkku erottaa desimaalit kokonaislukuosasta. Tietotekniikassa käytetään kuitenkin suuren maailman tyyliin pistettä erottimena. Tämä kannattaa huomioida koodia luettaessa. Jotta lukija menisi sekaisin, olen päättänyt olla uskollinen omalle merkintätavallemme ja viljellyt desimaalipilkkua kommenteissa ja selityksissä.

Kulman yksikkönä toimivat radiaanit, joita käytetään muutenkin matemaattisissa laskutoimituksissa tavallisia asteita enemmän. Käyttäjille ilmoitettavat kulmat ovat kuitenkin asteina. Tässä kohtaa kannattaa lukijan pystytellä hereillä, sillä radiaanien ja asteiden lisäksi myös askelmoottorin askeleet ilmaisevat kääntymiskulmaa.

Etäisyysmittaus

Päättäessäni päättötyöni aiheen loppukeväästä 2004 ajattelin mielessäni etäisyysmittauksen pikkuseikaksi. Olin nähnyt lehdessä mainostettavan parinkymmenen euron hintaista



etäisyysmittaria, joten ajattelin toteuttaa etäisyyden mittaamisen tällaisella halpalaitteella. Bilteman kuvastokin tunsi käsitteen laseretäisyysmitta.

Tarkempi tarkastelu paljasti nämä ”lasermittalaitteet” epätarkoiksi ultraäänimitoiksi, joissa etäisyyden mittaaminen perustuu ultraääneen ja laser toimii vain osoittimena. Ultraääni olisi voinut sekin sopia työhöni, mutta koska ultraäänilaitteiden mittaus perustuu äänen heijastumiseen, en olisi voinut luottaa näiden mittojen antamiin tuloksiin kaltevilta pinnoilta. Päättötyössäni on kyse juuri eri kulmin mitattujen pisteiden hyödyntämisestä.

Ultraäänilaitteet suljettiin pois. Tutkittuani aikani erilaisia mittavaihtoehtoja sain selville oikeitakin lasersäteisiin perustuvia mittalaitteita olevan markkinoilla. Nämä laitteet jakautuvat kaupallisen hyödyntämisen suhteen kahteen ryhmään. Metsästäjille ja armeijan käyttöön tarkoitetut pitkän matkan mittalaitteet ovat ensimmäinen ryhmä. Ne pystyvät mittaamaan parin metrin tarkkuudella monen sadankin metrin päähän. Lyhyillä etäisyyksillä nämä laitteet ovat epätarkkoja.

Toinen pääryhmä ovat rakennusurakoitsijoille ja arkkitehdeille tarkoitetut mitat. Nämä mittalaitteet mittaavat erittäin tarkasti lyhyillä matkoilla. Lisäksi tällaisia mittoja myydään valmiilla tietokoneliitännöillä varustettuina. Täydellinen ratkaisu päättötyöhöni, ajattelin. Ongelmaksi muodostui hinta. Hyvästä mittalaitteesta joutuu pulittamaan satoja euroja. Jos vielä lisäksi kaipaa tietokoneliitännäistä mallia, saa kukkaroaan keventää jo lähes tuhannen euron verran.

Yritin monin tavoin saada käsiini siedettävän hintaista laseretäisyysmittaa siinä kuitenkaan onnistumatta. Lopulta lähetin postia eräälle suomalaiselle lasermitta-alan yritykselle ja kysyin, josko he olisivat suostuvaisia lainaamaan yhtä mittalaitetta minulle.

Vastauksen viipyessä aloin pohtia mitä vaihtoehtoja, ja kun kirjeeseeni vihdoin vastattiin lähes kolmen kuukauden kuluttua, olin jo aikaa sitten haudannut toiveet tällaisen laitteen saamisesta. Eräänä helteisenä heinäkuun päivänä oli mieleeni juolahtanut vallan typerä ajatus mitata etäisyyttä lasersäteen ja digitaalikameran keskinäisellä yhteistyöllä.

En tiedä onko tätä mittakeinoa aikaisemmin hyödynnetty, mutta veikkaisin jonkun muunkin tulleen ajatelleeksi samaa mittausperiaatetta. On kuitenkin syitä, miksi etäisyyttä ei usein mitata kameran ja laserin avulla. Kyseessä on nimittäin kömpelö ja erittäin epätarkka mittaustapa – mutta sitäkin luovempi. Kun huomioidaan päättötyön tarkoitus, lienee käyttämäni etäisyysmittaustekniikka jopa parempi kuin aito lasermittaus. Kameran protokollaa pohtiessa ja suuntakulmia laskiessa opin takuulla enemmän kuin valmista mittalaitetta käyttämällä olisin oppinut.

Etäisyysmittaus laserin ja kameran avulla

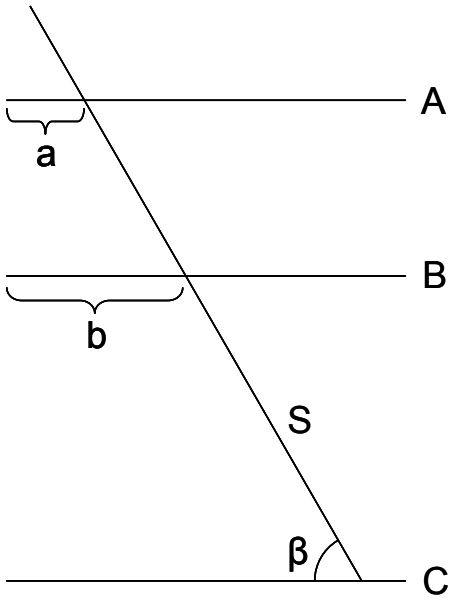
Ihminen hahmottaa etäisyyksiä ympärillään oleviin kappaleisiin lähinnä näkönsä avulla. Meillä on kaksi silmää juuri etäisyyden hahmottamista varten, sillä kumpikin silmä näkee ympäristön hieman eri kulmasta. Aivojen yhdistäessä nämä kaksi kuvaa syntyy mieleemme kolmiulotteinen kuva ympäröivästä maailmasta.

Ihmisen stereonäön simuloimista kahdella digitaalikameralla olisi mielenkiintoista kokeilla, mutta kovin tehokas tapa etäisyyksien määrittämiseen se ei olisi. Jos me ihmisetkin joskus kompastumme arvioituamme etäisyyksiä väärin, voi vain kuvitella minkälaisia epätarkkuuksia ja virheellisiä laskutoimituksia tietokone saisi tuloksiksi.

Periaate

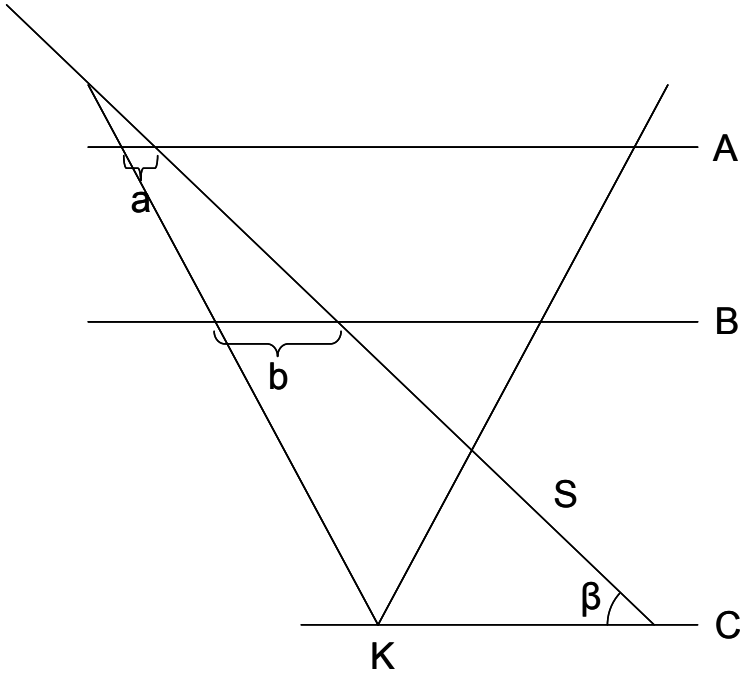
Oletetaan, että seisot taskulampun kanssa pimeässä huoneessa. Osoitat lapulla lattiaa metrin päässä jaloistasi. Lamppusi valokeila on tällöin tietyssä kulmassa lattiaan nähden. Jos muutat tätä kulmaa, siirtyy lattian pinnalla häilyvä valokeila sinusta poispäin tai lähestyy sinua.

Oletetaan, että lamppusi pysyy paikallaan, samassa kulmassa, ja samalla kohoaisit ylöspäin. Nyt liikkuisi myös lattialla oleva valokeila. Tähän samaan periaatteeseen perustuu etäisyysmittaus lasersädettä käyttäen. Mikään ei pakota käyttämään juuri lasersädettä. Valon ei ole pakko olla täysin yksiväristä, eikä valoaaltojen tarvitse olla samassa vaiheessa. Laservalon hienoudet ovat monin tavoin epäolennaisia, mutta eräs lyömätön etu laserilla on: valon intensiteetti. Lasersäde etenee pitkiä matkoja hajoamatta, mikä helpottaa valopisteen löytämistä monen metrin päästä.



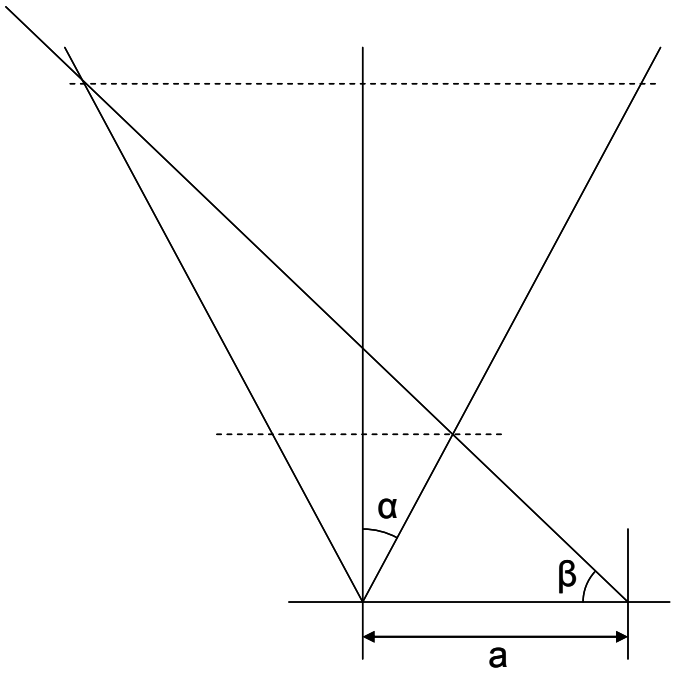
Oheinen kuva esittää mahdollisimman yksinkertaistettuna (laser)säteen paikan muutoksen etäisyyden funktiona. Suora S esittää valonsädettä, joka osuessaan eri etäisyyksillä oleviin kappaleisiin (A ja B) näyttäytyy niissä eri kohtaa. Kuvassa a ja b ovat etäisyyksiä suorien A ja B vasemmasta reunasta sädettä kuvaavan suoran S leikkauskohtiin. Koska suorien S ja C välissä on kulma β (]0, 90[), riippuvat arvot a ja b suorien A ja C sekä B ja C välisistä etäisyyksistä.

Kuvan tapauksessa etäisyys olisi helppo laskea, sillä se saataisiin suoran S kulmakerrointa ($k = \tan \beta$) hyödyntäen. Tilanne hankaloituu hieman liitettäessä kamera järjestelmään.



Kuvassa K tarkoittaa kameran linssiä. Kameran näkökenttä määrää miltä väliltä etäisyyttä voidaan mitata. Laserpisteen osumakohta kameran näkökentässä määrää etäisyyden. Oheisen kuvan perusteella voidaan todeta etäisyyden kasvavan laserpisteen siirtyessä kuvassa oikealta vasemmalle.

Laskutoimitukset – Kameran aukeamiskulma



Etäisyyttä laskettaessa on tiedettävä kolme asiaa:

- Lasersäteen kulma (β)
- Kameran aukeamiskulma (2α)
- Kameran linssin ja laserin välinen matka (a)

Lasersäteen kulma saadaan yksinkertaisesti mittaamalla (myöhemmin laskemalla). Samoin on kameran linssin ja laserin välisen matkan laita. Kameran aukeamiskulma sen sijaan pitää laskea kameran linssin ominaisuuksien perusteella.

Tavallisen kameralinssin (eli ei kalansilmä, fisheye) aukeamiskulma (Angle of View, α) voidaan laskea filmikoon (valokuvafilmin leveys; film dimension, d) ja tehollisen polttovälin (Effective Focal Length, f) avulla:

$$\alpha = 2\arctan\frac{d}{2f}$$

Tehollinen polttoväli (f) voidaan olettaa polttovälin (Focal Length, F) kanssa samansuuruiseksi paitsi niissä tapauksissa missä on kyse makrokuvauksesta (macro photography). Makrokuvauksessa täytyy huomioida suurennusarvo (magnification, m).

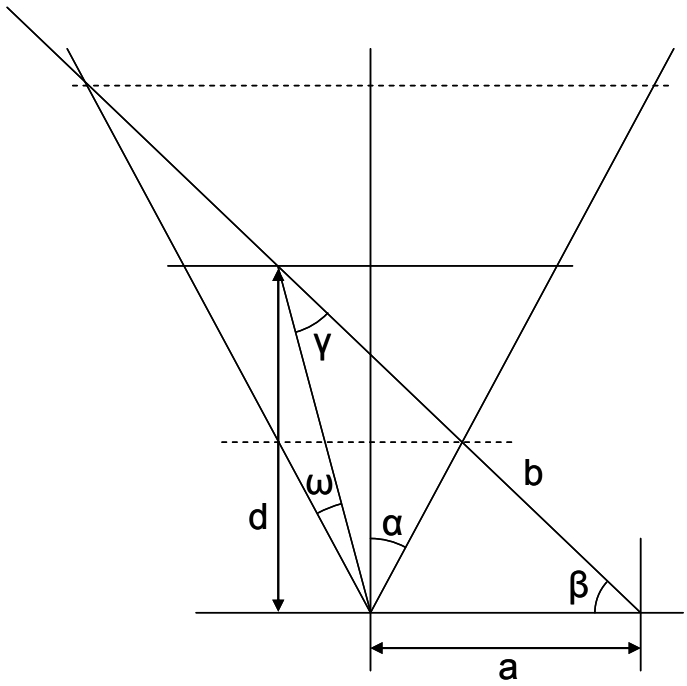
$$f = F \cdot (1+m)$$

Nämä kaavat palauttavat kameran koko aukeamiskulman, mutta olen omissa laskuissani käyttänyt ainoastaan kulman puolikasta, joten kaavat muuttuvat muotoon:

$$\alpha = \arctan\frac{d}{2F}$$

Kaavassa α on aukeamiskulman puolikas, d on filmin koko ja F on kameran polttoväli. Agfa ePhoto1280 -kamera vastaa 35 mm:n kinofilmikameraa filmikooltaan (d = 35 mm). Linssin polttoväli (F) on zoomin ollessa auki 38 mm. Aukeamiskulmaksi saadaan näillä arvoilla noin 49,45° ja kulman puolikkaaksi (α) noin **24,7273°**.

Laskutoimitukset – Etäisyyden laskeminen



Kaikkien etäisyyden laskemisen vaatimien mittojen ja kulmien asteiden ollessa koossa, voi itse laskemisen aloittaa. Yritän mahdollisimman selkeästi selittää miten päädyin käyttämiini yhtälöihin.

Oheinen kuva on tarkoitettu selventämään käytätettyjä merkintöjä. Kulma β on laserin suuntakulma ja α on kameran aukeamiskulman puolikas. Kulma ω riippuu laserpisteen kohdasta digitaalikuvassa. Oomegalle (ω) voidaan johtaa yhtälö:

$$\omega = 2\alpha x$$

Missä 2α on kameran aukeamiskulma ja x on laserpisteen kohta kuvassa. Muuttuja x saa arvon nollan ja yhden välillä ([0,1]). Pieni arvo tarkoittaa pisteen olevan kuvan vasemmassa reunassa, kun suuri arvo puolestaan merkitsee pisteen olevan oikealla.

Kuvan kulmaa γ käytetään vain kaavan johdossa. Voidaan todeta gamman (γ) olevan:

$$\gamma = 180^\circ - (90^\circ + \alpha - \omega) - \beta = 90^\circ - \alpha - \beta + \omega$$

Sinilauseena tunnettua kaavaa käyttäen voi kuvan muuttujan b nyt ratkaista:

$$\frac{a}{\sin\gamma} = \frac{b}{\sin(90^\circ + \alpha - \omega)}$$

$$b = a \cdot \frac{\cos(\alpha - \omega)}{\cos(\omega - \beta - \alpha)}$$

Etäisyys d saadaan nyt helposti laskettua sinin avulla, sillä kolmiosta tiedetään sekä kulma, että hypotenuusa b.

$$d = b \cdot \sin\beta = \frac{a \cdot \sin\beta \cos(\alpha - \omega)}{\cos(\omega - \beta - \alpha)}$$

Oomegakulmat (ω) voidaan muuttaa muotoon 2αx, jolloin saadaan etäisyysfunktio lopulliseen muotoonsa:

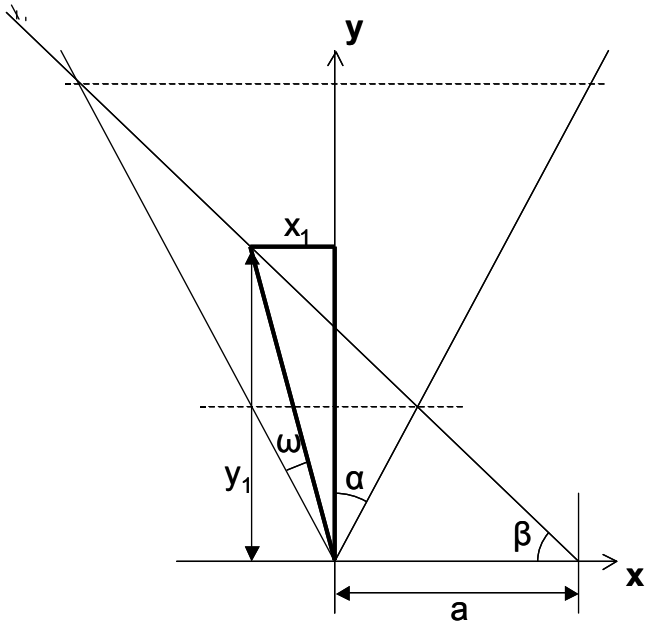
$$d(x) = \frac{a\ \sin\beta\ \cos(\alpha - 2\alpha x)}{\cos(2\alpha x - \beta - \alpha)}$$

Laskutoimitukset – Kalibrointi

”Ketju on yhtä vahva kuin sen heikoin lenkki.” Sama pätee etäisyyden määrittämiseen tarkoitettuun funktioon. Laskennallisesti saatujen tulosten tarkkuus riippuu täysin kaavan vakioiden tarkkuudesta. Tästä syystä on äärimmäisen tärkeää, että nämä vakiot saavat mahdollisimman tarkasti määritellyt lukuarvot.

Kameran avautumiskulman voidaan olettaa pysyvän jatkuvasti samana, joten alfakulman (α) arvo ei muutu. Kulman muutos vaatisi polttovälin muutosta. Polttoväliä voi muuttaa

zoomaamalla. Työssä käytetystä Agfa ePhoto -kamerasta löytyy optinen zoomi, mutta sitä ei käytetä. Kulma α pysyy siis vakiona.



Laserin suuntakulman β määrittämisen voi tehdä laskemalla lasersäteen kulmakertoimen. Kulmakertoimen voi laskea, jos tiedetään kulma α sekä vähintään kaksi eri etäisyyksillä olevaa pistettä. Näistä pisteistä on tiedettävä etäisyys sekä arvo x eli suhteellinen kohta kuvassa. Mitä enemmän pisteitä on, sitä tarkempi arvo saadaan.

Etäisyysmittausta kuvaavan kuvan voidaan nähdä esittävän koordinaatistoa. Lasersädettä kuvaavan suoran kulmakertoimen voi laskea suoran kulmakertoimen kaavalla:

$$k = \tan\beta = \frac{y_2 - y_1}{x_2 - x_1}$$

Laskemisessa tarvittavat etäisyydet (koordinaatistossa y-arvot) saadaan mittaamalla, mutta poikkeama linssin keskikohdan kohtisuorasta (koordinaatistossa x-arvot) joudutaan laskemaan. Oheiseen kuvaan on merkitty tummennettuna kolmio. Kolmion kanta on x_1 ja korkeus y_1. Kantaa vastaavan kulman suuruudeksi voidaan todeta ω – α. Arvo x_1 voidaan laskea tangentin avulla:

$$x_1 = y_1 \tan(\omega - \alpha) = y_1 \tan(2\alpha x - \alpha)$$

Kun vähintään kahden eri etäisyyksillä olevan pisteen koordinaatit voidaan laskea, voidaan määrittää lasersäteen **kulmakerroin β** sekä **vakio a** eli kameran linssin ja laserin välinen etäisyys. Pisteitä ollessa enemmän kuin kaksi saadaan tarkin tulos sovittamalla suora pistejoukkoon. Tällaista suoraa sanotaan regressiosuoraksi. Oheinen regressiosuoran kaava on Maol-taulukoista. Regressiosuoran kaavan merkintöjä on muutettu.

$$y = b_r x + a_r$$

$$b_r = \frac{n\sum x_i y_i - (\sum x_i)(\sum y_i)}{n\sum x_i^2 - (\sum x_i)^2}$$

$$a_r = \frac{\sum y_i - b\sum x_i}{n}$$

$$\tan\beta = -b_r$$

$$a = \frac{a_r}{\tan\beta} = -\frac{a_r}{b_r}$$

Regressiosuoran kaavaa hyödyntäen saadaan kulmakerroin b_r, joka on negatiivinen. Negatiivisuus johtuu siitä, että koordinaatistossa suora on laskeva. Etumerkki vaihdetaan positiiviseksi, sillä halutaan kulman itseisarvo.

Koordinaatistossa vakio a on suoran ja x-akselin leikkauspiste. Vakion a laskeminen voidaan tehdä tangentin avulla kolmiosta, mutta turhilta laskutoimituksilta säästytään, jos vakio a lasketaan jakamalla regressiosuoran vakiotermi a_r sen kulmakertoimella b_r.

Kaavojen käyttö

Laitteen kalibrointikaavoja käyttäen voi laserin kulmakertoimen määrittää aina uudelleen laitteessa tehtyjen pienten muutosten jälkeen. Arvojen määrittäminen helposti uudelleen mahdollistaa myös laitteen säätämisen eri etäisyyksille ja erilaisiin tiloihin sopivaksi.

Seuraavat etäisyydet mitattiin mittanauhalla ja pisteen kohta kuvasta määritettiin pisteentunnistusohjelmalla. Kuvan leveys on 160 pikseliä.

Piste	Etäisyys (m)	Kohta (px)	x (kohta/160)
1	0,72	60	0,375
2	1,35	31	0,19375
3	1,89	22	0,1375
4	2,52	16	0,1
5	3,42	12	0,075

Etäisyys on mitattu kohtisuora etäisyys linssin ja lasersäteen osumiskohdan välillä. Kohta on laserpisteen koordinaatti kuvassa kuvan vasemmasta reunasta aloittaen. Arvo x pisteen kohtaa suhteutettuna kuvan leveyteen.

Lisäksi tiedetään kulma α eli kameran aukeamiskulman puolikas, joka on noin 24,7273°. Regressiosuoran kaavasta tehdyin kalibrointikaavoin voidaan saada selville mittapisteisiin parhaiten sopiva kulma β ja vakio a.

$\beta \approx 65{,}3666°$
$a \approx 0{,}2529$ m

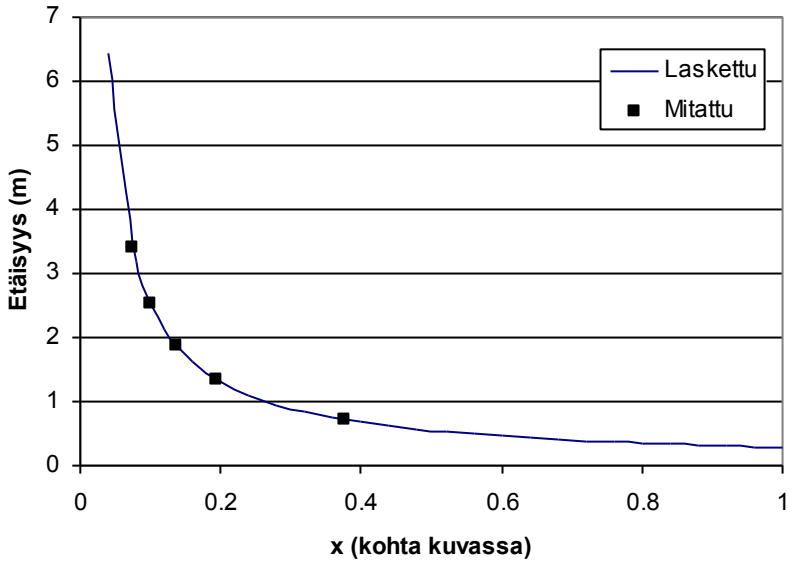
Tulokset annetaan neljän desimaalin tarkkuudella, mikä on ehkä liioittelua. Suuri tarkkuus on kuitenkin merkittävä, sillä kummatkin arvot ovat ratkaisevia laskujen onnistumisen

kannalta. Koska kyseessä on mittausyhtälöihin sijoitettavia arvoja, kannattaa ne pitää mahdollisimman tarkkoina.

Nyt kun kalibrointikaavoilla on laskettu mittapisteitä parhaiten vastaavat lukuarvot, voi niiden avulla laskennallisesti saatavia etäisyysarvoja verrata mitattuihin etäisyyksiin. Etäisyydet lasketaan etäisyyden kaavalla d(x). Etäisyydet ovat metreinä.

Piste	**x**	**Mitattu**	**Laskettu**	**Poikkeama (%)**
1	0.375	0.72	0.7220	0.3
2	0.19375	1.35	1.3459	-0.3
3	0.1375	1.89	1.8727	-0.9
4	0.1	2.52	2.5573	1.5
5	0.075	3.42	3.4027	-0.5

Taulukosta voidaan todeta arvojen osuneen hyvin kohdalleen. Mitattujen pisteiden osuminen nyt laskennallisesti määritettäviin tuloksiin on esitetty oheisessa kuvassa siten, että käyrä kuvaa etäisyyden muutosta x:n funktiona. Mitatut pisteet näkyvät pisteinä.



Kuvassa kuvaaja on katkaistu kohdassa 0,04, sillä tätä pienemmät arvot kasvavat erittäin nopeasti. Kuvasta voi todeta arvojen α, β ja a olevan tarpeeksi tarkkoja etäisyyden laskemiseen.

Etäisyys voidaan nyt vakioarvojen määritysten jälkeen laskea kaavalla d(x). Käytännössä mittaus tapahtuu vaiheittain:

1. Laser kytketään päälle
2. Kamera ottaa kuvan
3. Kuvasta etsitään laserin jättämä piste
4. Pisteen koordinaateista lasketaan arvo x (x = pisteen kohta / kuvan leveys)
5. Etäisyys lasketaan kaavalla d(x).

Aiemmin saatuja α-, β- ja a-arvoja käyttäen voidaan nyt laskea esimerkiksi kuinka kaukana kuvassa kohdassa 0,2 (24 px) oleva piste olisi:

$$d(0{,}2) \approx 1.31 \text{ metriä}$$

Näin voidaan laskea kulmien määrittelyväleillä olevia etäisyyksiä tehokkaasti.

Mittaustavan ongelmia

Laseria ja digitaalikameraa hyödyntävä etäisyysmittaus on osin ongelmallinen tapa mitata etäisyyksiä. Mittaustapana tämä epäpyhä liitto ei ole ainoastaan kömpelö vaan myös epätarkka ja epäluotettava. Seuraavaksi on lueteltu ilmeisiä ongelmia, joita olen kohdannut hyödyntäessäni esiteltyä mittaustapaa.

Ongelma I

Laserin kulmaa (β) sekä linssin ja laserin välistä etäisyyttä (a) säätelemällä voidaan saavuttaa todella vakuuttavan kuuloisia mittausvälejä. Jo pienillä säädöillä voidaan päästä mittausväleihin, jotka alkavat paristakymmenestä sentistä ja jatkuvat satoihin metreihin. Nämä arvot ovat kuitenkin silmänlumetta, sillä mittaustekniikka on tarkkaa ainoastaan lähietäisyydellä. Etäisyyden kasvaessa kasvaa myös muutosnopeus, joten tarkkuus putoaa olennaisesti jo parin metrin etäisyyksiä mitattaessa.

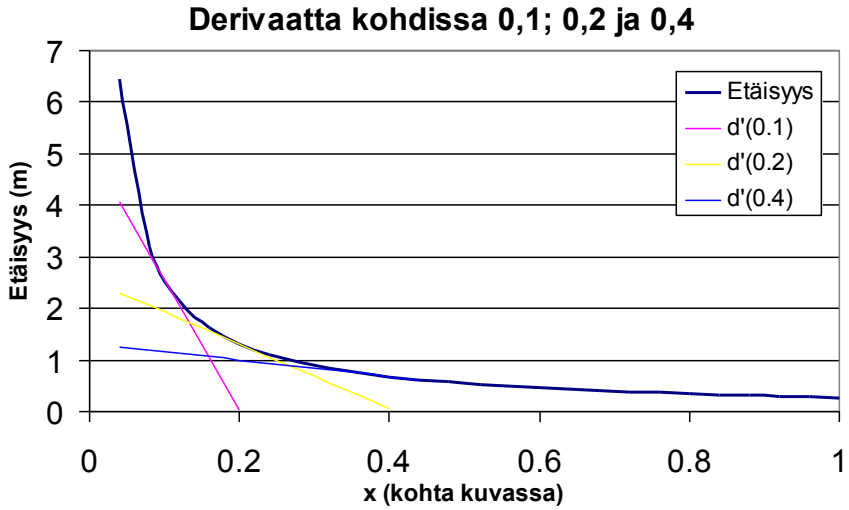
Laskemalla funktion d(x) derivaatta saadaan muutosnopeus, jota voidaan kuvata funktion kulmakertoimena tietyssä kohdassa. Derivoimalla funktio

$$d(x) = \frac{a \sin\beta \cos(\alpha - 2\alpha x)}{\cos(2\alpha x - \beta - \alpha)}$$

saadaan derivaatta:

$$d'(x) = \frac{2\alpha \sin(\beta) a \cos(2\alpha x - \alpha) \sin(2\alpha x - \beta - \alpha)}{\cos^2(2\alpha x - \beta - \alpha)} - \frac{2\alpha \sin(\beta) a \sin(2\alpha x - \alpha)}{\cos(2\alpha x - \beta - \alpha)}$$

Derivaatan avulla voidaan esittää funktion d(x) muutosnopeus kohdissa 0,1; 0,2 ja 0,4.



Kuvasta huomaa, että suurilla etäisyyksillä huononee tarkkuus radikaalisti. Ongelmaa voisi korjata kasvattamalla vakion a suuruutta. Tämä on kuitenkin huono ratkaisu, sillä vakion a kasvaessa kasvaisi mittalaitteen leveys. Suuntakulmalla β on vain vähän merkitystä tarkkuuteen pitkillä etäisyyksillä.

Toinen tapa parantaa tarkkuutta olisi käyttää tarkempaa resoluutiota kamerassa. Mitä enemmän pikseleitä kuvassa on, sitä tarkemmin voidaan laserpisteen paikka määrittää kuvasta. Tarkka kuva helpottaisi laskemista, mutta kuvien siirto kamerasta tietokoneelle veisi enemmän aikaa ja pisteen löytäminen ohjelmallisesti kuvasta vaikeutuisi.

Ongelma II

Toinen mittaustapaan liittyvä ongelma on laserpisteen löytäminen kameran kuvasta. Pisteen löytymiseen vaikuttavat monet seikat, joista kuvan resoluutio, etäisyys ja valaistusolosuhteet ovat merkittävimpiä. Lisäksi jpeg-kuvan häviöllinen pakkaus sekä pinta, johon laser on osunut vaikuttaa mittaukseen.

Pisteen löytämisen vaikeus muodostaa pullonkaulan koko mittaustekniikalle. Tälle ongelmalle ei ole yksinkertaista ratkaisua, ja tulosten tarkkuuteen vaikuttaa lähinnä ohjelmakoodin monipuolisuus. Pisteen löytämisen haastetta käsitellään enemmän etsimistä hoitavan ohjelmakoodin yhteydessä.

Kaavat koodina

Koodi löytyy moduulista Laske.

Kaavojen muuttaminen Pascal-koodiksi ei ole ylivoimainen tehtävä. Pieniä muutoksia kaavojen rakenteisiin tuottavat joidenkin trigonometristen funktioiden puute Delphin peruskirjastoista. Käytössä ovat sini, kosini ja arkustangentti. Näiden kolmen avulla voidaan kuitenkin selvittää kaikki tarvittavat trigonometriset arvot.

Toinen huomioitava tekijä on single-tyyppisten muuttujien tarkkuus. Single on nelitavuinen liukulukumuuttuja, jossa merkitseviä numeroita on 7 – 8. Pyöristys on pieni, mutta näkyy pitkien laskutoimitusten vastauksissa jo viidennessä desimaalissa. Tarkkuus on kuitenkin todettu aivan riittäväksi.

Etäisyyden laskeminen on hyvin suoraviivaista. Kaava d(x) on suoraan muutettu kodiksi. Koodin selkeyttämiseksi on kaavan laskeminen jaettu kahteen osaan. Kalibrointikaavojen muuttaminen koodiksi on hankalampaa kuin etäisyysfunktion. Olen jakanut koodin pieniin osiin selkeyden vuoksi. Kalibrointi tapahtuu *kalibroi*-aliohjelman avulla. Tämä aliohjelma kutsuu puolestaan muita aliohelmia ja funktiota.

Aliohjelma *laske_x* laskee poikkeaman linssin kohtisuorasta. Tämän arvon sekä etäisyyden avulla voidaan funktioilla *regressiosuora_b* ja *regressiosuora_a* sovittaa halutunkokoiseen pistejoukkoon suora. Tuloksina saadaan suoran kulmakerroin sekä leikkauskohta y-akselilla. Nämä arvot siirretään suoraan globaaleihin muuttujiin *beeta* ja *vakio_a*.

Käyttömukavuuden kannalta on tärkeää saada ohjelma tulostamaan ymmärrettävää tietoa. Koodi palauttaa kulmat radiaaneina (rad) ja koska suurin osa ihmisistä lienee tottuneet

tarkastelemaan kulmia asteina, on syytä radiaanit muuttaa asteiksi tulostamista varten. Muunnos asteista radiaaneiksi on tarpeen käyttäjän syöttäessä arvoja ohjelmalle.

Kulman muunnos perustuu siihen, että tan 45° = 1. Tätä ominaisuutta käytetään siksi, että se on niitä harvoja arvoja, jonka saa suoraan perusfunktioista ilman kikkailua. Verrannon avulla muutetaan asteet radiaaneiksi ja toisinpäin.

Kameran ohjaus

Jotta etäisyyttä voidaan mitata, tarvitaan kuva, josta määrittää laserpisteen kohta. Sopivan kuvan ottamiseen tarvitaan digitaalikamera. Kamera on saatava ottamaan käskystä kuva ja tämä kuva on kyettävä siirtämään tietokoneelle jatkokäsittelyä varten.

Digitaaliset kamerat hyödyntävät useimmiten ccd-kennoa (Charge-coupled device) kuvauksessa. Kuvan ottaminen perustuu pieniin kennomaisiin aseteltuihin kondensaattoreihin, joihin syntyy varaus fotonin kulkiessa ohi. Varaus on verrannollinen valon intensiteettiin kyseisessä kohdassa. Kondensaattorit siirtävät varauksensa vahvistimeen, josta kuvadata lähtee eteenpäin jännitevaihteluina. Tämä tieto saadaan säilöttyä digitaaliseen muotoon.

Ccd-kennojen kovin tarkka tuntemus ei ole päättötyön kannalta tarpeen. Pääasia on ymmärtää, että kameran kuva siirtyy kaapelia pitkin tietokoneeseen, jossa kuvaa tutkaillaan.

Päätin jo aikaisessa vaiheessa hyödyntää Agfa ePhoto -kameraani. Tämän kameran maksimiresoluutio on 1280 x 960. Laserpisteen etsiminen tällaisesta kuvasta olisi kuitenkin hyvin hidasta ja päädyin käyttämään alhaisempaa resoluutiota. Pienin kameran kuvausresoluutio on 640 x 480. Tämäkin tuntui turhan hyvältä laadulta ja päädyin käyttämään kameran esikatselukuvaa, joka ei edes tallennu kameran muistikortille. Esikatselukuvan resoluutio on 160 x 120. Kuvadata on jpeg-pakattua.

Kamera on liitetty tietokoneeseen sarjaportin kautta.

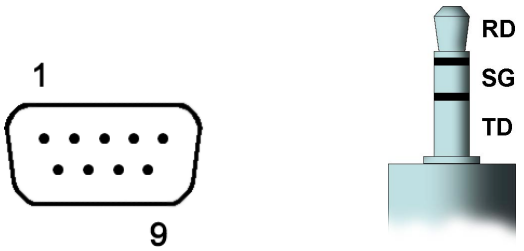
Sarjaportin toimintaperiaate

Sarjaportissa (serial port) tieto liikkuu peräkkäisinä bitteinä, yksi kerrallaan sisään tai ulos. Rinnakkaisporteissa taas bitit liikkuvat rinnakkaisilla linjoilla. Sarjaporttiliikennettä voikin verrata morsetukseen, jossa peräkkäisillä signaaleilla muodostetaan aakkosia. Yleisin sarjaporttiliitäntä on niin sanottu RS-232 -liitäntä, joka on ollut tietokoneissa yleinen liitäntä niin hiirille, näppäimistöille kuin muillekin ulkoisille laitteille.

Alun perin RS-232 -portin liittimenä oli 25-napainen D-liitin, mutta koska tieto välittyy ainoastaan yhtä johdinta pitkin, sarjassa, jäi vielä kontrollisignaalien jälkeen paljon käyttämättömiä napoja. Tästä syystä sarjaporttiliitännällä tarkoitetaan nykyään lähinnä 9-napaista D-liitintä, jonka uros-pää on kiinni tietokoneessa. Käytettyjä liittimiä on kuitenkin valtavia määriä.

USB-liitännät (Universal Serial Bus) ovat viime vuosina vallanneet alaa perinteisiltä sarjaporteilta. Nykyään digitaalikamerat hyödyntävät lähes poikkeuksetta USB:tä. Se on houkutellut käyttäjiä erityisesti helppokäyttöisyydellään. USB-laite ei periaatteessa tarvitse erillistä ajuria kommunikoidakseen tietokoneen kanssa, vaan sitä tuetaan yleensä suoraan.

Sarjaportti puolestaan tarvitsee aina ajurin. Tietokoneen on ilman ajuria mahdoton tietää, mikä laite sarjaporttiin on liitetty ja mitä portista tulevat bitit tarkoittavat. Kommunikointi sarjaportissa tapahtuu, kuten tietokoneissa aina, ykkösien ja nollien avulla. Tämä data saadaan muutettua signaaliksi kerta kaikkiaan nostamalla ja laskemalla portin ulosmenojen jännitettä.



9-napainen D-liitin ja kameran 2,5 mm:n plugi

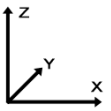
Nasta	Nimi		Tehtävä
2	RD	(Received data)	Vastaanotettu tieto
3	TD	(Transmitted data)	Lähetetty tieto
8	CTS	(Clear to send)	
7	RTS	(Request to send)	
6	DSR	(Data set ready)	
5	SG	(Signal ground)	Maa
4	DTR	(Data terminal ready)	
1	DCD	(Data carrier detect)	

RS-232 -porttien yhteydessä käytetään jos jonkinlaisia asetuksia, mutta yleisimmät ovat nopeus (speed), parity ja stop bits. Nopeus tarkoittaa siirrettyjen bittien määrää sekunnissa (bit per second, bps). Kaikkein tavallisimpia arvoja ovat 300, 1200, 2400, 9600 ja 19200. Portin kommunikaationopeuden määrittelyssä käytetään usein nimitystä Baud, joka viittaa lähetettävän signaalissa tapahtuvien muutoksien määrään sekunnissa.

Parity-arvo on eräänlainen alkeellinen virheentunnistustapa, jonka avulla saadaan tarkistettua ovatko bitit sotkeutuneet matkalla toisiinsa. Parity voi saada arvon N (none), O (Odd, pariton) tai E (Even, parillinen). Parity-bittien virheenkorjaus perustuu ovelaan ideaan lähettää bittijonot aina joko parillisina tai parittomina. Jos bittijono ei vastaa parityasetusta, lisätään jonoon erityinen Parity-bitti, joka tekee kaikista bittijonoista aina samanlaisia, joko parittomia tai parillisia. Jos jono poikkeaa asetuksista, on tieto vääristynyt matkalla. Yleensä Parityä ei käytetä ja tällöin asetuksena on N.

Lopetusbitti (Stop bit) lähetetään jokaisen siirretyn tavun jälkeen. Tämä ominaisuus auttaa kommunikoivat laitteet synkronoitumaan eli tahdittumaan toisiinsa. Yleiskäytössä olevia Stop bits -asetuksia ovat 1 ja 2. Asetukset määritellään tavallisesti muodossa D/P/S, jossa D on databittien määrä, P on Parity ja S on loputusbittien määrä (Stop bits). Yleisin asetus on 8/N/1.

Sisään tulevat tai ulosmenevät bitit muodostavat esimerkiksi 8-bittisiä lukuja. Portin vastaanottamat bittijonot muutetaan yleensä tavuiksi (0 – 255). Tällöin tavut ovat



kahdeksan sarjoissa, jolloin muodostuu luku 0 ja 255 välillä (00000000_2 – 11111111_2). Tämä luku voidaan esittää merkkinä tietokoneiden ANSI-merkkikartaston avulla. Tämä merkkikartasto sisältää kaikki ns. perinteiset merkit, joita tietokoneissa käytetään; aakkoset (a–z ja A–Z), numerot, välimerkit ([space], rivinvaihto, pilkut, sulut ym.), erikoismerkit (ääkköset ååöüÿ ym.) sekä lisäksi tietokoneen käyttämät merkit, joita paperille ei voi kirjoittaa. Toinen yleinen tapa tulkita portin vastaanottamaa dataa on tarkastella sitä heksadesimaalilukuina.

Hyödynnettäessä sarjaporttia ohjelmoinnissa voi porttia toki lähestyä suoraan käyttöjärjestelmän puolelta asm (assembly) -käskyillä. Uusimmissa Windowsin versioissa tämä on kuitenkin estetty, joten yksinkertaisin tapa on kutsua porttia jotakin valmista objektia hyväksikäyttäen. Microsoftin Visual Basic -kääntäjässä tällainen on valmiina, mutta Borland Delphiin on tällainen objekti liitettävä erikseen.

Erilaisia sarjaporttiobjekteja on kuitenkin saatavilla. Suurin osa niistä on harrastajavoimin kokoonkyhättyjä, mikä ei tietenkään rajoita niiden toimivuutta mitenkään. Käyttämäni komponentti on *Boris Lobodan* tekemä TCommPort Component for Delphi 3. Komponentti sisältää varsin monipuoliset funktiot portin lukemiseen ja tietojen lähettämiseen. Informaatio lähetetään ja otetaan vastaan merkkeinä.

Kameran protokolla

Yllätyin, miten hankalaa oli löytää Agfa ePhoto -kameran käyttämän protokollan kuvaus. Pitkällinen nettisurffailu tuotti tulosta ja löysin *Eugene Crosser*in laatiman epävirallisen kuvauksen joidenkin digitaalikameroiden käyttämästä protokollasta. Kävi ilmi, että kameravalmistajat eivät julkaise käyttämiään protokollia, mikä sinänsä on harmittavaa.

Eugene Crosserin protokollakuvauksessa sanottiin protokollan sopivan joihinkin Epsonin, Sanyon, Agfan ja Olympoksen kameramalleihin, joten ilahduin suuresti huomatessani kamerani olevan tuettujen listalla. Kävi ilmi, että tämä protokolla on jonkinlainen epävirallinen digitaalikameroiden käyttämä standardi. Protokolla sopii kaikkiin tiettyä piirilevyä käyttäviin kameroihin.

Monimutkaisissa ja hienoissa kameroissa on todella paljon ominaisuuksia ja suurinta osaa näistä pystyy hallitsemaan myös tietokoneen kautta. Jätän kuitenkin suurimman osan protokollaan käskyistä huomioitta, sillä ne eivät ole projektini kannalta olennaisia ja käyttämäni kamera ei edes tue niitä kaikkia.

Lyhennetty versio **Serial Protocol of Some Digital Cameras** -protokollakuvauksesta:

Tietokone ja kamera kommunikoivat datapakettien ja yksittäisten bittien avulla. Sarjaportissa käytettävät asetukset ovat: 8bit, no parity, no flow control, kaikki aritmeettinen tieto siirretään vähiten merkitsevä bitti ensin (”little endian”).

Protokollan perusosat ovat:

Initialization Byte	NUL	0x00	Aloitus
Action Complete Notification	ENQ	0x05	Valmis
Positive Acknowledgement	ACK	0x06	Onnistui
Unable to Execute Command	DC1	0x11	Suoritus mahdoton
Negative Acknowledgement,			

also Camera Signature	NAK	0x15	Epäonnistui
Packet	Variable length sequence of bytes		
Termination Byte		0xff	Lopetus

Paketin rakenne:

Kohta	Pituus	Kuvaus
0	1	Packet type (Paketin tyyppi)
1	1	Packet subtype/Sequence (Alatyyppi/tunniste)
2	2	Length of data (Dataosuuden pituus)
4	määräämätön	Data
-2	2	Checksum (Tarkistussumma)

Paketteja on kolmea tyyppiä:

0x02 Data packet that is not last in sequence (Datapaketti, joka ei ole viimeisenä)
0x03 Data packet that is last in sequence (Joukon viimeinen datapaketti)
0x1b Command packet (Komentopaketti)

Datapaketit, jotka palaavat yhden ja saman komennon pyynnöstä, on numeroitu nollasta alkaen. (Sequence). Jos palautettava data mahtuu yhteen pakettiin, on palaavalla paketilla alatyyppi (Packet subtype) 0x03 ja tunniste (Sequence) 0. Komentopaketeilla on alatyyppi (Packet subtype) 0x43 tai 0x53. Vain ensimmäisen komennon alatyyppi on 0x53.

Dataosuuden maksimipituus on 2048 bittiä, mikä tarkoittaa koko paketin maksimipituudeksi 2054 bittiä. Tarkistussumma on yksinkertainen 16-bittinen dataosuuden bittien summa. Dataosuuden pituus ja tarkistussumma lähetetään vähiten merkitsevä bitti ensin.

Komennot ja niiden muotoilu

Komento on osa paketin dataosuutta komentopaketissa. Komento muotoillaan seuraavasti:

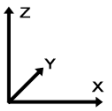
Kohta	Pituus	Kuvaus
0	1	Command code (Komennon tyyppi)
1	1	Register number or subcode (Rekisterin numero tai alatyyppi)
2	määräämätön	Optional argument (Valinnainen)

Komentoja on viittä tyyppiä:

Merkintä	Tyyppi	Kuvaus
0	int32	Set value of integer register (Tallenna luku integerrekisteriin)
1	none	Read value of integer register (Lue luku integerrekisteristä)
2	vdata	Take action unrelated to registers (Suorita toiminto)
3	vdata	Set value of vdata register (Tallenna dataa vdatarekisteriin)
4	none	Read value of vdata register (Lue dataa vdatarekisteristä)

Komentoihin 0 ja 3 kamera vastaa ACK (0x06) signaalilla. Komentoon 2 kamera vastaa ACK (0x06) ja "Action Complete Notification" (0x05) -signaaleilla. Komentoihin 1 ja 4 vastataan datapaketeilla, joista jokainen tulee hyväksyä ACK (0x06) -signaalilla.

Rekisterit



Kameran rekistereitä on noin sata ja tässä on esitelty ainoastaan tärkeimmät. On hyvä huomata, että rekisterinumero on ilmoitettu kymmenjärjestelmässä.

Numero	Tyyppi	Luku/kirjoitus	Kuvaus
1	int32	R/W	Resoluutio: 1 - Std, 2 - Hi, 3 - Ext
4	int32	W	Aktivoidun kuvan numero
12	int32	R	Valitun kuvatiedoston pituus *
13	int32	R	Valitun esikatselukuvan pituus *
14	vdata	R	Valitun kuvan data *
15	vdata	R	Valitun esikatselukuvan data *
17	int32	R/W	Nopeus 1 - 9600, 2 – 19 200…

* Huomaa, että rekisterit 12,13,14 ja 15 ovat käytettävissä vasta neljännen rekisterin saatua arvon. Jos rekisteriin 4 asetetaan luku 0 ja viides toiminto (action) on suoritettu ennen tätä, palauttaa kamera ”livekuvaa”.

Toiminnot (Actions)

Komentoa 2 käytettäessä ei käytetä rekisterinumeroita vaan toimintoja (action). Nämä toiminnot eivät ole riippuvaisia rekisteristä vaan ne suoritetaan itsenäisinä.

Merkintä	Tyyppi	Kuvaus
0	single zero byte	Poista viimeisin kuva
1	single zero byte	Poista kaikki kuvat
2	single zero byte	Ota kuva
4	single zero byte	Keskeytä kommunikointi
5	single zero byte	Ota esikatselukuva (saatavilla kuvana 0)
7	single zero byte	Poista valittu kuva *

* Huomaa, että toiminto 7 on käytettävissä vasta neljännen rekisterin saatua arvon.

Kameraa ohjaavat komennot

Etäisyysmittauksen kannalta olennaista on saada kamera ottamaan kuva haluttuna ajankohtana ja siirtämään tämä kuva tietokoneelle käsiteltäväksi. Nämä yksinkertaiset toimenpiteet vaativat kuitenkin paljon työtä ja toimintoja. Kameran protokollaa hyväksikäyttäen voidaan tehdä yksinkertainen toimintakaavio:

Aloitetaan:

	TIETOKONE	
0x00	NUL	Intialization byte

	KAMERA	
0x15	NAK	Camera signature

Asetetaan portin nopeus:

TIETOKONE
Komentopaketti:
0x1B, 0x53, 0x03, 0x00, 0x00, 0x11, 0x02, 0x13, 0x00

0x1B	Type	Komentopaketti
0x53	Subtype	Ensimmäinen komentopaketti
0x03	Length	Dataosuuden pituus
0x00	Length	
0x00	Data; Command code	Tallenna luku integerrekisteriin
0x11	Data; Register number	Rekisterin numero (Nopeus)
0x02	Data; Optional argument	Uusi arvo (2 eli 19 200)
0x13	Checksum	Tarkistussumma: $11_{16} + 02_{16} = 13_{16}$
0x00	Checksum	

KAMERA

0x06	ACK	Positive Acknowledgement

Otetaan Preview-kuva:

TIETOKONE

Komentopaketti:
0x1B, 0x43, 0x03, 0x00, 0x02, 0x05, 0x00, 0x07, 0x00

0x1B	Type	Komentopaketti
0x43	Subtype	Muu kuin ensimmäinen komentopaketti
0x03	Length	Dataosuuden pituus
0x00	Length	
0x02	Data; Command code	Suorita toiminto
0x05	Data; Register number	Toiminnon numero (Ota esikatselukuva)
0x00	Data; Optional argument	
0x07	Checksum	Tarkistussumma: $02_{16} + 05_{16} = 07_{16}$
0x00	Checksum	

KAMERA

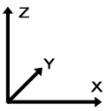
0x06	ACK	Positive Acknowledgement
0x05	ENQ	Action Complete Notification

Kerrotaan halutun kuvan numero:

TIETOKONE

Komentopaketti:
0x1B, 0x43, 0x06, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x00, 0x04, 0x00

0x1B	Type	Komentopaketti
0x43	Subtype	Muu kuin ensimmäinen komentopaketti
0x06	Length	Dataosuuden pituus
0x00	Length	
0x00	Data; Command code	Tallenna luku integerrekisteriin
0x04	Data; Register number	Rekisterin numero (Aktivoidun kuvan numero)
0x00	Data; Optional argument	Uusi arvo (0 eli äsken otettu esikatselukuva)
0x00	Data	
0x00	Data	
0x00	Data	
0x04	Checksum	Tarkistussumma: $04_{16} = 04_{16}$



0x00 Checksum

KAMERA

0x06 ACK Positive Acknowledgement

Tilataan halutun kuvan data:

TIETOKONE

Komentopaketti:
0x1B, 0x43, 0x03, 0x00, 0x04, 0x0¤, 0x00, 0x12, 0x00

0x1B	Type	Komentopaketti
0x43	Subtype	Muu kuin ensimmäinen komentopaketti
0x03	Length	Dataosuuden pituus
0x00	Length	
0x04	Data; Command code	Lue dataa vdatarekisteristä
0x0E	Data; Register number	Rekisterin numero (Valitun kuvan data)
0x00	Data; Optional argument	
0x12	Checksum	Tarkistussumma: $04_{16} + 0E_{16} = 12_{16}$
0x00	Checksum	

KAMERA

Datapaketti:
0x02 tai 0x03, 0x??, 0x??, (korkeintaan 2048 bittiä), 0x??, 0x??

0x02 tai 0x03	Type	Komentopaketti
0x??	Subtype	Paketin tunniste (0, 1, 2...)
0x??	Length	Dataosuuden pituus (alemmat tavut)
0x??	Length	Dataosuuden pituus (ylemmät tavut): maksimipituus 2048
0x??	Data	JPEG-tiedoston dataa
...		
0x??	Checksum	Tarkistussumma (alempi tavu): korkeintaan FF_{16}
0x??	Checksum	Tarkistussumma (ylempi tavu): korkeintaan FF_{16}

Jollei kaikki data mahdu yhteen pakettiin, lähetetään paketteja kunnes kaikki data on siirretty. Viimeisen datapaketin tyyppi on 0x03. Tietokone vastaa jokaiseen onnistuneeseen siirtoon ACK (0x06) -signaalilla. Jos paketin siirto kestää kohtuuttoman kauan tai paketti on vaurioitunut (tarkistussumma ei täsmää), voi tietokone pyytää paketin lähettämistä uudelleen NAK (0x15) -signaalilla.

TIETOKONE

0x06 ACK Positive Acknowledgement

Lopetetaan yhteistoiminta:

TIETOKONE

0xFF Termination Byte

KAMERA (Asettuu nollatilaan)

Kuvien siirto ohjelmallisesti

Kuvia siirtävä ohjelmakoodi toimii hyvin pitkälti edellä esitettyjen kameraa ohjaavien komentojen avulla. Kameran toiminnoille on tehty oma moduuli (Kamera), joka hoitaa kommunikoinnin kameran kanssa.

Moduulissa on kolme tärkeää osaa: lataa, laheta ja vastaanota. Lisäksi moduulissa on aliohjelma raportoi, joka luo lokitietoa käyttäjälle latauksen edistymisestä. Laheta-funktio lähettää tavuittain tietoa sarjaportista ulos. Vastaanota-funktio puolestaan ottaa vastaan kameran lähettämää tietoa.

Aliohjelma Lataa koordinoi koko kommunikointia. Se asettaa kameralle yhteysnopeuden, ottaa kuvan, lataa sen kamerasta ja tallentaa kiintolevylle halutulla nimellä. Aliohjelman toiminnasta kannattaa huomioida pari seikkaa:

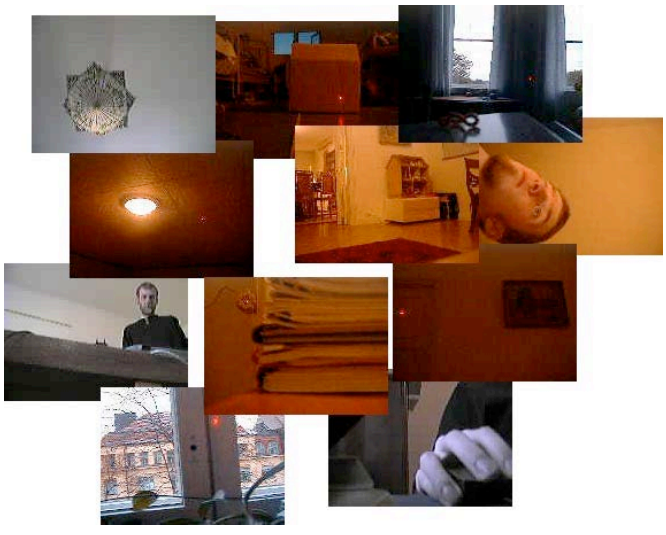
Kameran lähettämiä vastauksia ei tarkasteta. Kamera vastaa jokaiseen komentoon erilaisilla viesteillä. Koodi on tyytyväinen, kunhan saa kameralta minkä tahansa vastauksen. Tulleen tavun arvoa ei tarkasteta, sillä jos paluuarvo on väärä, ilmenee se jatkossa muutenkin virheenä. Nämä kohdat, joissa kameran vastaus oletetaan, on kommentein merkitty ol.-merkinnällä.

Tarkistussumma tarkastetaan. Vaikken ole vielä kertaakaan törmännyt virheelliseen pakettiin, tein koodiin ominaisuuden, joka tarkistaa siirretyn paketin tarkistussumman. Jos summa ei täsmää kameran lähettämän summan kanssa, pyydetään kameraa lähettämään viallinen paketti uudelleen.

Tarkistussumma on yksinkertainen 16-bittinen dataosuuden bittien summa. Tämä tarkoittaa, että siirretyt bitit summataan keskenään seuraavasti:

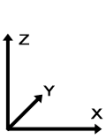
```
    10111
+    1011
=  100010
+  100110
= 1001000
+...
```

Kamerasta siirrettyjä kuvia



Paketin maksimipituus on 2048-tavua, mikä tarkoittaa, että summan maksimiarvo on reilut puoli miljoonaa. 16-bittinen luku voi kuitenkin suurimmillaan olla ainoastaan 2^{16}-1. Summasta otetaan ainoastaan 16 viimeistä bittiä. 16-bittisen summan voi hahmottaa myös siten, että aina kun summa kasvaa yli 2^{16}-1 (65 536), aloitetaan laskeminen taas nollasta. Olen ratkaissut tarkistussumman muuttamisen 16-bittiseksi jakamalla luvun 65536:lla ja ottamalla jakojäännöksen talteen.

Kuva tallennetaan tekstinä. Käytetty sarjaporttia hoitava komponentti palauttaa vastaanotetut tavut merkkeinä. Koska siirtovaiheessa ei vielä olla kiinnostuneita itse



kuvasta, vaan pelkästä informaatiosta, tallennetaan bitit kylmästi tekstitiedostona. Tekstitiedoston pääte muutetaan .jpg:ksi. Näin on entinen tekstitiedosto nyt muutettu kauniiksi kuvatiedostoksi.

Kuvadatan käsittely

Jpeg-kuvaformaatti

Tietotekniikassa kuvien tallentaminen on aina ollut hanakalaa tekstin tallentamiseen verrattuna. Tekstin saa näppärästi tavuittain talteen. Kuvissa joudutaan tallentamaan erilaisia väriarvoja jokaista pikseliä kohti. Tämä vie paljon tilaa. Esimerkiksi yhden megapikselin pakkaamaton värillinen (24-bit) bittikarttakuva vie tilaa 3 megatavua.

Kuvien suuren tallennuskapasiteettitarpeen takia on kehitelty erilaisia pakkausmenetelmiä kuvien tallentamiseksi pienempään tilaan. Yleisesti käytetty pakkausstandardi on JPEG. JPEG:llä pakattujen kuvien tiedostotyyppiä kutsutaan usein myös JPEG:ksi; käytetyimmät tiedostopäätteet tällaisilla kuvilla ovat .jpeg, .jfif, .jpg, .JPG ja .JPE. Pääte .jpg on kaikkein yleisin pääte JPEG-kuville.

Nimi on lyhenne ilmaisusta Joint Photographic Experts Group. JPEG itsessään määrittelee ainoastaan tavan, jolla kuva muutetaan bittivirraksi, muttei miten tämä tieto tallennetaan. JFIF (JPEG File Interchange Format) -standardi puolestaan määrittelee miten tuottaa tallennukseen ja siirtoon soveltuva tiedosto JPEG-virrasta. Tavallisesti puhuttaessa “JPEG-tiedostosta” tarkoitetaan JFIF-tiedostoa.

JPEG/JFIF on kaikkein tavallisin kuvaformaatti varsinkin kuvien tallennuksessa ja siirrossa internetin kautta. Nämä kuvat eivät sovellu kaikkeen, esimerkiksi tekstin ja viivapiirrosten tallennukseen, sillä kuvien pakkaustapa tuottaa näissä rumaa jälkeä. Tekstiä ja viivoja sisältävät kuvat toimivat paremmin PNG- ja GIF-formaateissa. JPEG/JFIF sopii mainiosti valokuville ja vastaaville kuville, joissa kuvan vaikutelma ei ole riippuvainen yksittäisistä pikseleistä.

JPEG-standardi on tunnettu häviöllisenä kuvien pakkauksen suhteen. Tämä tarkoittaa, että pakatusta kuvasta ei enää saa koostettua alkuperäisen kuvan laatuista kuvaa. Tietoa häviää aina pakkauksen yhteydessä. Standardi sisältää monia tapoja koodata (encode) kuvadataa, mutta vain osaa näistä käytetään yleisesti. Tavallisesti koodaus tehdään 24-bittiselle kuvalle (kahdeksan bittiä punaiselle, vihreälle ja siniselle).

Kuvan koodaus on kehitelty juuri ihmissilmää ajatellen. Ihminen ajattelee kuvaa eri tavalla kuin kone ja erottelee kuvasta eri asioita. Pakkauksessa hyödynnetään muun muassa ihmisen tarkkuutta kirkkauksien suhteen ja samalla huonontaen värien erottelua.

Kuvan värit muutetaan (Color Space Transformation) koodattaessa RGB-arvoista omaan YUV-formaattiin. Tämä vastaa televisiokuvan värien esittämisessä käytettyä tapaa. Y-komponentti ilmaisee värin valomäärän, joka on kahta muuta tärkeämpi, sillä ihmissilmä erottaa tarkemmin valon vaihtelun kuin pienet värimuutokset. Muita pakkausvaiheita ovat muun muassa Downsampling ja DCT (Discrete Cosine Transformation), joissa kuvan data jaetaan osiin ja tiedot yhtenäistetään.

Kvantisaatio (Quantization) on eniten kuvan kokoa pienentävä – ja laatua huonontava – operaatio pakkauksessa. Ihmissilmän ominaisuutta olla tunnistamatta vaihtelua korkean taajuuden kirkkauksissa hyödyntämällä voidaan pyöristää kirkkausarvoja samoiksi. Lopuksi kuvaa muokataan vielä häviöttömällä entropiapakkauksella (Huffman-koodaus), joka yhdistää samanlaiset jonot.

Pakkauksen purku (Decode) tapahtuu vastaavalla tavalla, mutta käänteisessä järjestyksessä. Pakkaussuhdetta säätelemällä saadaan aikaan erikokoisia ja -laatuisia kuvia. Suhteessa 10:1 pakattua kuvaa on vaikea erottaa alkuperäisestä. Teoriassa on mahdollista saavuttaa jopa 100:1 pakkaussuhde, mutta tällöin kuva olisi hyvin kärsinyt ja vioittunut.

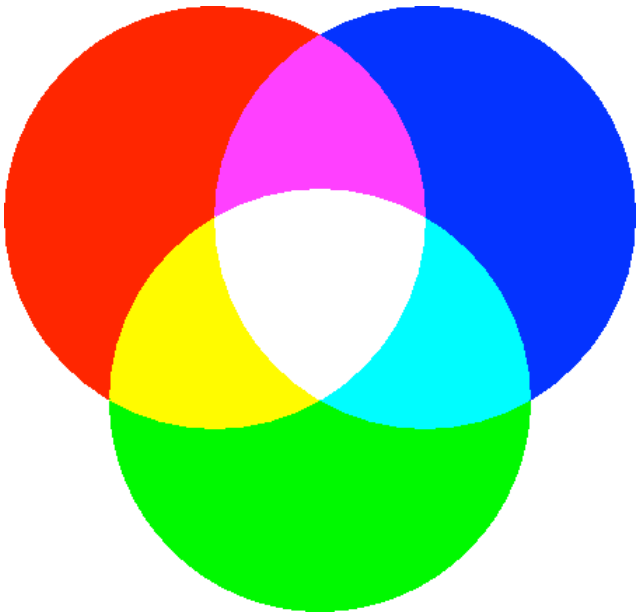
Näihin pakkausmenetelmiin ei ole tarvetta enempiä syventyä, sillä kameran pakkaama kuva voidaan muuttaa Delphin jpeg-kirjastolla bittikarttakuvaksi.

Tietokoneen värikoodaus

Kuvien käsittely ohjelmallisesti ei onnistu ilman kuvien rakenteen tuntemista. Kuvat on tallennettu tietokoneen muistiin pikseleinä, jotka muodostavat rivejä ja sarakkeita. Pikselit ovat erivärisiä ja värierot muodostavat näkyvän kuvan. Pikseleiden värit voidaan ilmoittaa monin eri tavoin. Yleisiä väriavaruuksia ovat muun muassa RGB, CMYK, HSV, HSL sekä Lab. Päättötyössäni olen hyödyntänyt RGB- ja HSV-/HSL-väriavaruuksia niiden erilaisten ominaisuuksien vuoksi.

RGB

Tietotekniikassa värit ilmoitetaan yleisesti RGB-koodina. RGB on lyhennys väreistä Red (Punainen), Green (Vihreä) ja Blue (Sininen). Näitä arvoja yhdistelemällä eri suhteissa saadaan aikaan muita värejä. Tällaista värillisiä valoja yhdistelevää värimallia sanotaan additiiviseksi.



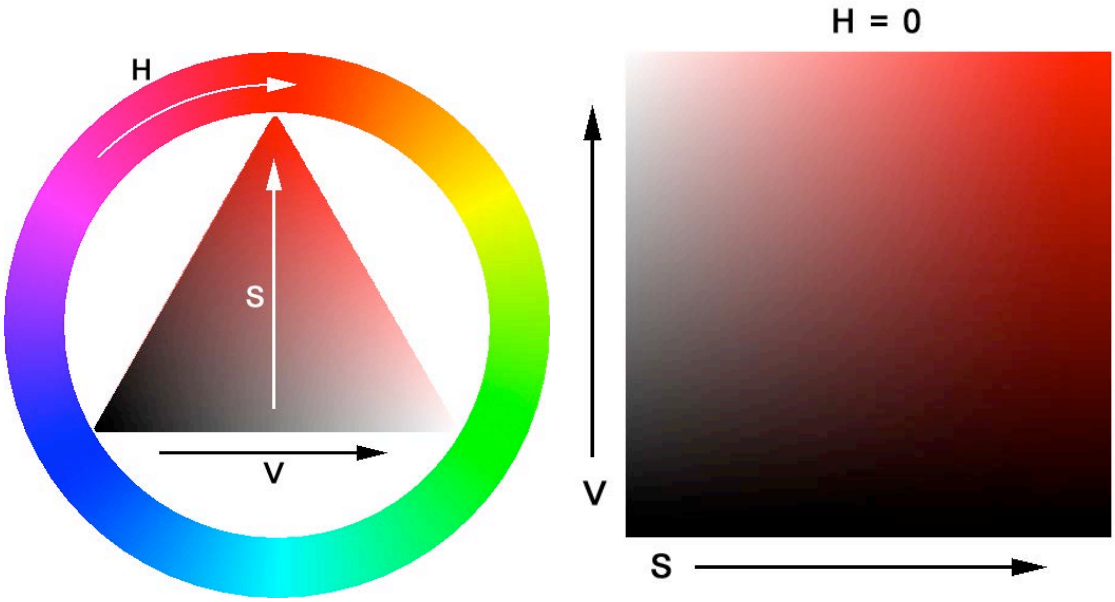
Tietokoneelle tallennetut kuvat kuten myös näytöllä näkyvä kuva perustuvat yleensä RGB-väriavaruuteen. Tavallisesti jokaista pikseliä kohti tallennetaan 24 bittiä tietoa, jossa on 8 bittiä jokaista kolmea väriä kohden. Näin punainen, vihreä ja sininen voivat kukin saada 256 eri arvoa. 24-bittisessä kuvassa mahdollisia värejä on reilut 16,7 miljoonaa erilaista.

RGB	Hex	Nimi
(0, 0, 0)	#000000	Musta
(255, 255, 255)	#FFFFFF	Valkoinen
(255, 0, 0)	#FF0000	Punainen
(0, 255, 0)	#00FF00	Vihreä
(0, 0, 255)	#0000FF	Sininen
(255, 255, 0)	#FFFF00	Keltainen
(0, 255, 255)	#00FFFF	Syaani
(255, 0, 255)	#FF00FF	Magenta

RGB-arvot voidaan ilmoittaa monella tavalla. Esimerkiksi html (Hypertext Markup Language) esittää värit peräkkäisinä heksadesimaalilukuina (esim. #00FF00 – Vihreä). Tallennettaessa värejä tiedostoon voidaan värit tallentaa järjestyksessä RGB tai BGR riippuen miten värit luetaan tiedostosta.

HSV

HSV on lyhenne sanoista Hue, Saturation ja Value. Nämä kolme arvoa määräävät pikselin värin. Hue on väriarvo, joka saadaan eräänlaisesta väripyörästä, jossa jokainen väri sulautuu seuraavaan. Hue-arvo ilmoitetaan joko asteina 0-360 tai prosentteina 0-100 %. Saturation puolestaan ilmoittaa värin ”puhtauden”; mitä pienempi on Saturation-arvo, sitä harmaampi on väri. Saturaatio ilmoitetaan prosentteina 0-100 %. Value on värin kirkkaus (0-100 %). Valuen ollessa 100 % on väri valkoinen ja arvon ollessa 0 % on väri musta.



HSV-värejä käytetään yleensä kuvankäsittelyssä, kun halutaan nähdä tietyn värin suhde toiseen tai hahmottaa värejä värikartalla. HSV on myös kätevä ominaisuus värisokealle värien tarkasteluun (kirj. huom.). HSV-arvona pystyy helposti hahmottamaan värin kirkkauden sekä perussävyn, vaikka väri olisi kovin haalea.

RGB-arvon muuttaminen HSV-arvoksi on melko yksinkertaista. Esittelen nyt yleisesti käytetyn muuntokaavan, jonka olen ottanut Wikipedian sivuilta.

R, G ja B arvot muutetaan kaikki muotoon 0,0–1,0 jakamalla 255:llä. Seuraavat laskutoimitukset palauttavat HSV-arvon, joka on samaa muotoa (0,0–1,0). Määritellään arvoksi MAX suurin (R, G, B) arvoista ja arvoksi MIN pienin näistä arvoista. Voidaan kirjoittaa yhtälö:

$$H = \begin{cases} \left(0 + \frac{G-B}{MAX-MIN}\right) \times 60, \text{ jos } R = MAX \\ \left(2 + \frac{B-R}{MAX-MIN}\right) \times 60, \text{ jos } G = MAX \\ \left(4 + \frac{R-G}{MAX-MIN}\right) \times 60, \text{ jos } B = MAX \end{cases}$$

$$S = \frac{MAX-MIN}{MAX}$$

$$V = MAX$$

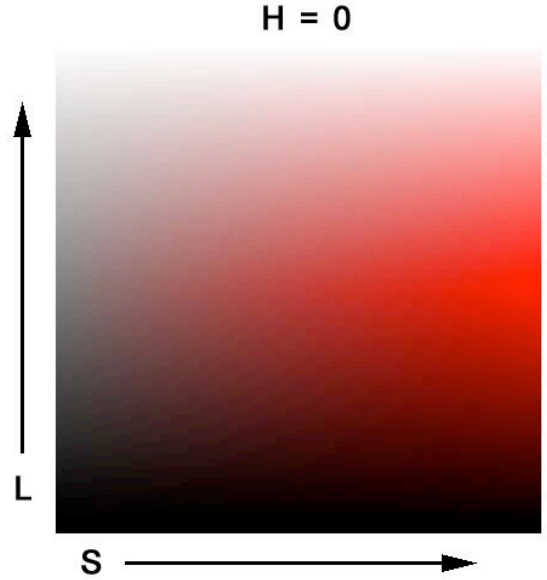
Saatu arvo H vaihtelee välillä 0 ja 360 ilmaisten värin kohdan ympyrässä asteina. S ja V ovat välillä 0,0 ja 1,0. Koska H on kulma asteina voi se kiertyä kokonaisen kierroksen ympäri siten, että esimerkiksi arvo 360 vastaa arvoa 0, 480 arvoa 120 ja -30 arvoa 330.

Lisäksi on hyvä huomioida pari erikoistapausta:

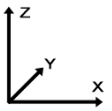
- Jos **MAX = MIN** (eli S = 0), H:lla ei ole arvoa. Tällöin S arvo määrää kuvan harmaaksi ja väriarvolla (H) ei ole merkitystä.
- Jos **MAX = 0** (eli V = 0), S:llä ei ole arvoa. Tällöin V arvo määrää kuvan mustaksi, jolloin H ja S ovat merkityksettömiä.

HSL

HSL (Hue, Saturation, Lightness) on hyvin samankaltainen värijärjestelmä kuin HSV (Hue, Saturation, Value). Molemmat määrittävät Hue-arvon samalla tavalla, mutta Saturation ja Value/Lightness poikkeavat toisistaan. HSL voidaan kirjoittaa myös muodossa HLS.



Kuvassa näkyy HSL- ja HSV-järjestelmien erot.
Lightness-arvo kulkee HSL:ssä aina mustasta valkoiseen.



HSL-väreissä Lightness on kirjaimellisesti värin valoisuus. Tämän ominaisuuden avulla voidaan määrittää kuvan tummuus tai vaaleus. Saturaatio määritellään hieman eri tavalla kuin HSV:ssä, mutta se viittaa myös HSL-värimallissa värin puhtauteen (harmaan määrään).

RGB-arvon muuttaminen HSL-arvoksi tapahtuu pitkälti samalla tavalla kuin HSV-arvon laskeminen. Ainoat erot ovat S- ja L-arvojen laskemisessa.

$$S = MAX - MIN$$

$$L = \frac{MAX + MIN}{2}$$

Pistetunnistus

Etäisyysmittauksessa tarvittavien pisteiden tunnistaminen bittikarttakuvasta ei ole yksinkertaista. Tietokoneelle on hankala selittää, että kuvasta pitäisi löytää hailakoita punaisia pisteitä. Erilaisia konenäköhankkeita on maailmalla tekeillä sadoittain, jollei tuhansittain. Tarkoitus ei ole saada tietokone ymmärtämään kuvien sisältöä paljoakaan, vaan vain paikallistamaan punaisen laserpisteen kuvasta.

Pistetunnistusta on hankala lähestyä ihmisen näkökulmasta. Me katsomme kuva ja näemme siinä pisteen sen kummemmin pohtimatta tai käymättä kuvaa rivi riviltä läpi. Jos emme löydä pistettä heti, tutkailemme kuvaa tarkemmin eri kohdista. Tietokone puolestaan ei näe värejä tai kuvioita. Tietokone näkee ykkösiä ja nollia. Kuva on peräkkäinen rivi tavuja, jossa luvut esittävät värejä. Tältä pohjalta on pistetunnistusta lähdettävä ratkomaan.

Toimintaperiaate

Kuvainformaatiota luetaan pikseli kerrallaan. Jokaisen pikselin koordinaatti (x, y) tiedetään. Lisäksi saadaan luettua jokaiselle pikselille kolme väriarvoa RGB-väriavaruudessa esitettyinä. Käyttämäni pistetunnistustapa perustuu todella yksinkertaiseen logiikkaan. Tämä tietty kunnianhimottomuus johtuu siitä, että totesin tämän yksinkertaisen tavan olevan kaikkein luotettavin ja paras erilaisissa tilanteissa ja olosuhteissa.

Kuvan käsittely aloitetaan käymällä läpi kuva pikseli kerrallaan vasemmalta oikealle ylhäältä alaspäin. Kohdattaessa tietyn punaisuusarvon ylittävä pikseli lasketaan tätä pistettä ympäröivien pikseleiden punaisuus. Jos etsitty alue on tarpeeksi punainen, tallennetaan se muistiin. Siirrytään seuraavaan pikseliin ja etsintä jatkuu samalla tavalla. Lopuksi, kun koko kuva on käyty läpi, otetaan muistiin tallennetuista alueista punaisuusarvoltaan suurin ja päätellään sen olevan etsitty piste.

Tämä tapa etsiä kuvasta pisteitä on epäluotettava valoisissa kuvissa. Tämä tekniikka on luotettava ainoastaan kuvissa, joissa punaisen pisteen ja taustan välinen kontrasti on tarpeeksi suuri. Pisteiden löytäminen helpottuu suuresti, jos kuva on mahdollisimman pimeä. Tästä syystä johtuen mittausta auttaa kuvaustilan valon vähentäminen.

Funktiot

Yksinkertaiselta vaikuttava pistetunnistus on koodina monimutkaisemman näköinen. Koodatessa olen ottanut huomioon erilaisia erityistilanteita, jotka vaativat huomiota. Pistetunnistusfunktio on kirjoitettu kameran esikatselukuvia varten. Näiden kuvien resoluutio on 160 x 120 pikseliä. Koodi löytyy moduulista Etsi.

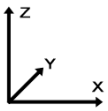
JPEG-kuvan muuntaminen bittikartaksi. Kameran tallentamat JPEG-kuvia ei voi suoraan käsitellä Delphin kuvanlukukäskyillä. Kuvien avaamiseen tarvitaan erillinen JPEG-moduuli, joka liitetään uses-osioon. Tämän laajennuksen avulla voidaan JPEG-pakkaus purkaa tavalliseen bitmap-muotoon.

Pikseleiden luku. Pikseleiden väri voidaan lukea yksitellen TCanvas.Pixel-ominaisuudella. Tämä on kuitenkin todella hidas tapa. Paljon nopeammin saadaan ohjelma toimimaan, kun luetaan kerrallaan kokonainen vaakarivi pikseleitä muistiin ja käydään ne läpi. Tämä tapahtuu TBitmap.ScanLine-ominaisuudella. Kuvan vaakariviä voi käsitellä taulukkona tavuja, jossa värit ovat aina kolmen arvon ryppäissä. Väriarvot ovat järjestyksessä BGR, sillä punaiselle värille on varattu alimmat kahdeksan bittiä.

Värien tunnistus. RGB-arvoista on hankala hahmottaa pikselin väriä. Pikselin punaisuuden määrittäminen tehdään muuttamalla väriavaruus HSV-arvoiksi, joista määritellään punaiseksi värit välillä -30° – 30°. Punaisen määrä lasketaan Pythagoraan lauseella kolmesta arvosta (särmiön halkaisija).

Kuvan valomäärä tunnistetaan HSL-väriavaruuden Lightness-arvon perusteella. Tätä ominaisuutta hyödynnetään kuvan kokonaisvalomäärän saamisessa. Kaikkien kuvan pikseleiden keskiarvosta määritetään valoisuus, joka auttaa tuloksien luotettavuuden punnitsemisessa.

Kuvissa pistetunnistusfunktion palauttamia arvoja. Valkoinen pystyviiva merkitsee löydetyn pisteen x-koordinaattia kuvassa. Pystyviiva on selkeyden vuoksi katkaistu pisteen kohdalta. Oikean alakulman kuvasta funktio ei ole löytänyt pistettä, koska kuva on liian valoisa ja.



Askelmoottoreiden ohjaus

Koko laitteen toiminnan kannalta on elintärkeää, että kameraa ja laseria saa liikuteltua hallitusti sekä pysty- että vaaka-akselin ympäri. Mekaanisesta liikkumisesta huolehtivat kaksi askelmoottoria (katso mekaaninen osuus). Askelmoottoreita puolestaan ohjaa piirilevy, joka huolehtii askelmoottoreille lähetettävistä pulsseista (katso elektroninen osuus). Kyseinen piirilevy on liitetty tietokoneen rinnakkaisporttiin (Parallel Port).

Rinnakkaisportin toimintaperiaate

Rinnakkaisportti tunnetaan myös englanninkielisellä nimellään Parallel Port. Tämän lisäksi monet tuntevat sen tulostinliitäntänä (kirjoitinportti). Ennen USB (Universal Serial Bus) -aikaa suurin osa tulostimista oli varustettu rinnakkaisporttiliitännällä. Yhä edelleen useimmista koneista löytyy rinnakkaisporttiliitäntä. Se on jopa sarjaporttiliitäntää yleisempi.

Rinnakkaisportin liitin on tietokoneessa 25-napainen naaraspuolinen D-liitin. Rinnakkaisporttiliitäntä on klassisesti yhdistetty IBM:n 36-napaiseen Centronics-liitäntään, joita yhä edelleen näkee tulostimissa.

Rinnakkaisportissa (nimensä mukaan) bitit kulkevat rinnakkain, kun taas sarjaportissa sarjassa, peräkkäin. Rinnakkaisportissa voidaan käyttää jopa kahdeksaa linjaa tiedon siirtämiseen, minkä takia se onkin ollut suosittu liitäntä esimerkiksi tulostimissa ja ulkoisissa levyasemissa aikoinaan.

Linjan ollessa elektronisesti aktiivissa tilassa (electrical high) sen jännite on +2,4 – +5 V. Passiivissa tilassa (electrical low) jännite on 0 ja +0,8 V:n välillä. Looginen arvo 1 saadaan jännitteen ollessa +2,4 – +5 V ja looginen 0 jännitteen ollessa 0 – +0,8 V. Rinnakkaisportissa on useita nastoja, jotka kertovat muun muassa tulostimen tilasta ja joita voidaan käyttää eri laitteiden kättelysignaaleina. Näillä ei kuitenkaan ole työni kannalta merkitystä, joten mainitsen ainoastaan niiden olemassaolon.

IBM määritteli aikoinaan kolme perusosoitetta rinnakkaisportille (80x86 IO -osoiteavaruudessa). Kirjoittimen tuli saada osoite 0x378 tai myöhemmin 0x278, kun taas tietokoneen näyttö- ja kirjoitinadapterin (Monochrome Display and Printer Adapter) tuli saada osoite 0x3BC. Bootauksen aikana BIOS etsii rinnakkaisportteja osoitteista 0x3BC, 0x378 ja 0x278. Löydetyt portit määritellään tässä järjestyksessä. Ensimmäinen löydetty saa nimekseen LPT1, toinen LPT2 ja kolmas LPT3 (jos kolmatta löytyy).

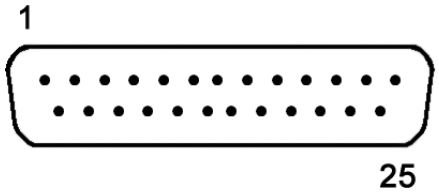
Hankalaksi porttien osoitteiden määrittämisen tekee, että MDPA:ta (Monochrome Display and Printer Adapter) ei löydy kaikista koneista. Tästä syystä osoitteet vaihtelevat koneesta toiseen.

Osoite	**MDPA**	**ei MDPA:ta**	
0x3BC	LPT1	-	Monochrome Display and Printer Adapter
0x378	LPT2	LPT1	Ensisijainen tulostin
0x278	LPT3	LPT2	Toissijainen tulostin

Nimi	**MDPA**	**ei MDPA:ta**
LPT1	0x3BC	0x378
LPT2	0x378	0x278
LPT3	0x278	-

Signaalit ja nastojen bitit

Nasta	**Signaali**	**Kuvaus**
PIN 1:	/Strobe	
PIN 2:	Data Bit 0	Vähiten merkitsevä bitti (Least Significant Data)
PIN 3:	Data Bit 1	…
PIN 4:	Data Bit 2	…
PIN 5:	Data Bit 3	…
PIN 6:	Data Bit 4	…
PIN 7:	Data Bit 5	…
PIN 8:	Data Bit 6	…
PIN 9:	Data Bit 7	Eniten merkitsevä bitti (Most Significant Data)
PIN 10:	/Acknowledge	
PIN 11:	Busy	
PIN 12:	Page End	
PIN 13:	Select	
PIN 14:	/Auto Feed	
PIN 15:	/Error	
PIN 16:	/Initial Printer	
PIN 17:	/Select Input	
PIN 18 – 25:	Ground	Maa

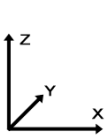


Moottoreiden ohjaus rinnakkaisportin avulla

Näistä monista nastoista ja ominaisuuksista askelmoottoreiden ohjaukseen käytetään ainoastaan neljää Data Bit -nastaa sekä maadoitukseen nastaa 25. Data Bit 0 ja 1 ovat kytketty Moottorin 1 ohjaukseen ja Data Bit 2 ja 3 ohjaavat Moottoria 2. Laserin ohjaus on kytketty nastaan 6. Siksi voimme keskittyä lähinnä rinnakkaisportin nastoihin 2 – 6 ja 25.

Nastojen loogiset arvot ilmoitetaan lukuna. Binäärimuodossa tämä luku ilmaisee databittien arvot. Esimerkiksi luku 10101010_2 (171) kertoo joka toisen linjan olevan loogisesti 1 eli toisin sanoen niissä kulkee virta. Luku 0 kytkee kaikki linjat pois päältä ja 11111111_2 (255) tarkoittaa kaikkien linjojen olevan päällä. Koska askelmoottoreita ohjaava piirilevy hyödyntää ainoastaan neljää vähiten merkitsevää databittiä, ei neljällä eniten merkitsevän bitin arvoilla ole merkitystä. Arvot vaihtelevat siis 0 ja 1111_2:n (15) välillä.

Kumpikin moottori pystyy toimimaan toisistaan riippumatta, joten moottorin pyörimisen kannalta ainoastaan kahdella databitillä on merkitystä. Ensimmäinen moottori hyödyntää databittejä 0 ja 1, kun taas toinen moottori bittejä 2 ja 3. Moottori saadaan liikkumaan antamalla ensiksi molemmille moottorin biteille loogiset arvot 1. Muuttamalla tämän jälkeen toinen biteistä nollaksi, saadaan moottori liikkumaan yhden askeleen valitun bitin mukaiseen suuntaan.



	Lähtötilanne	**Suunta +**	**Suunta –**
Moottori 1	xxxxxx11	xxxxxx10	xxxxxx01
Moottori 2	xxxx11xx	xxxx10xx	xxxx01xx

Biteillä, jotka on merkitty x:llä, ei ole merkitystä kyseisen moottorin liikkumisen kannalta. Moottoreiden liikkeen suuntaan käytännössä vaikuttaa lisäksi se, miten päin niiden liittimet on liitetty piirilevyyn. Tästä syystä konkreettista suuntaa ei anneta. Käytetty merkintätapa (+ ja –) on vain suuntaa-antava.

Edellä ollut ohjausselitys näyttää vain, miten moottorit saa ottamaan yhden ainoan askeleen haluttuun suuntaan. Jatkuvaan pyörimisliikkeeseen vaaditaan useita peräkkäisiä askelia, jotka saadaan aikaan vuorottelemalla lähtötilanteen (xxxx1111) ja halutun suunnan arvoa. Moottoreita voi myös ajaa samanaikaisesti, sillä kuten taulukosta käy ilmi eri moottoreita ohjaavat arvot eivät vaikuta toisiinsa. Ohessa taulukko, joka näyttää kymmenjärjestelmän luvut, jotka ovat moottorin pyörimisen suhteen samanarvoisia. Ensiksi on aina ilmoitettu yksin kyseessä olevaan moottoriin vaikuttava luku. Esitettävät luvut ovat ainoastaan väliltä 0000_2 – 1111_2 (0 – 15).

	Lähtötilanne	**Suunta +**	**Suunta –**
Moottori 1	3 (15, 11, 7)	2 (14, 10, 6)	1 (13, 9, 5)
Moottori 2	12 (13, 14, 15)	8 (9, 10, 11)	4 (5, 6, 7)

Näitä arvoja tarkasteltaessa saattaa hyvinkin herätä kysymys, miksi moottoreiden lähtöarvona on, että linjat ovat loogiselta arvoltaan 1. Tähän ei ole vastausta, sillä monin tavoin voisi olla selkeämpää pitää nollatilana (lähtöarvona) nollaa. Jo hyväksi todettua järjestelmää tuntui kuitenkin ikävältä mennä sorkkimaan ja kytkentöjä muuttelemaan, joten moottoreiden ohjaus toimii esitellyillä arvoilla.

Moottoreita liikuttavat funktiot

Rinnakkaisporttiin on perinteisesti ollut helppo päästä käsiksi tietokoneohjelmoinnissa. Vielä vanhat 9x Windowsit (95, 98, ME) tukivat rinnakkaisportin hallintaa suoraan. Windows NT, 2000 ja XP ovat puolestaan hankalia tapauksia. Tiukentuneet turvamäärittelyt estävät porttien käsittelyn ilman välikäsiä.

Aikoina ennen Windowsia useimmissa ohjelmointikielissä tuettiin I/O (In/Out) -porttien hallintaa suoraan. Windowsin ilmestyttyä asumaan tietokoneiden sydämeen hankaloitui porttien hallinta. Tämä johtui siitä, että Windows emuloi suoritettavalle ohjelmalle tietokoneen osia niin, että kukin ohjelma kuvittelee keskustelevansa suoraan näppäimistön, näytönohjaimen tai muun koneen osan kanssa. Tästä syystä useiden ohjelmien samanaikainen suoritus on Windowsissa mahdollista.

Tästä fyysisten (hardware) osien emuloinnista johtuen Windows ei voi hyväksyä kommunikointia suoraan I/O-porttien kanssa, sillä käyttöjärjestelmä uskoo olevansa ainoa portteja käyttävä tekijä koneessa. Itse asiassa ns. 9x Windowsit 95/98 hyväksyvät porttien hallinnan sovellustasolla. Tosin ani harvat ohjelmointikielet tukevat tätä ominaisuutta suoraan. Yksinkertaisin tapa hallita portteja kyseisissä järjestelmissä lienee upottaa oman koodin joukkoon portteja hoitava assembly-koodipätkä (konekieli).

Verrattain hankalan assemblyn käyttö tuntuu luonnottoman yksinkertaiselta vaihtoehdolta, kun porttien ohjauksen hanakluutta vertaa uudemman polven Windowseihin. Turvallisina

tunnetut Windows NT/2000/XP eivät hyväksy porttien ohjausta sovellustasolta lainkaan. Toisin sanoen Windows 95 kirjoitettu portteja hyödyntävä ohjelma ei toimi XP:ssä.

Windows NT/2000/XP sallii kuitenkin porttien hallinnan kernel-tason ajureille (Kernel mode driver). Kernel-ajuri suoritetaan järjestelmän kaikkein alhaisimmalla tasolla eli ne ovat prosessien suoritusjärjestyksessä etusijalla. Kernel-tasolla koodilla on täydet oikeudet kaikkiin koneen osiin ja nämä ajurit voivat tehdä ihan mitä lystäävät, myös tuhota koko järjestelmän. Juuri tämän takia kernel-ajurin kirjoittaminen ei ole ihan maailman yksinkertaisimpia asioita.

Tällaisen ajurin tekeminen ei kuitenkaan ole aivan mahdotonta, työlästä kuitenkin. Se vaatisi kahlaamista käyttöjärjestelmän dokumentaation läpi huolellisesti. Tuloksena olisi pieni ajuri, joka toimisi linkkinä sovellustasolla pyörivän ohjelman ja halutun portin välillä. Tässä järjestelyssä on kuitenkin ongelmana nopeus. Käskyn kulkeminen sovelluksesta ajurin kautta porttiin vie aikaa useita kertoja enemmän kuin portin suora hallinta. Kernel-ajurilla on, kuten mainittu, täydet oikeudet muuttaa järjestelmää ja tästä syystä tällainen ajuri pystyy muuttamaan Windowsin rajoitusta olla sallimatta porttien hallintaa.

Tässä kohtaa on helpotus todeta, ettei jokaisen ohjelmoijan, joka harkitsee sarjaportin ohjausta tarvitse uppoutua Windowsin – varsin laajaan – dokumentaatioon ja kirjoittaa itse tällainen kernel-ajuri ohjelmalleen. Onneksi maailmassa on sellaisia laupeita sieluja, jotka ovat hoitaneet tämän työn jo valmiiksi ja vieläpä ilmaiseksi.

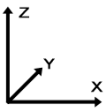
Texasilaisen *Fred Bulbackin* kirjoittama io.dll pystyy hoitamaan rinnakkaisportin hallinnan uusissa Windowseissa. Ajuri tukee seuraavia käyttöjärjestelmiä Windows 95/98/ME/NT/2000/XP. Tämä ajuri toimii, kuten edellä on kuvattu. DLL-tiedosto (Dynamic Link Library) tunnistaa käytettävän käyttöjärjestelmän ja havaitessaan käyttöjärjestelmäksi NT:n, 2000:n tai XP:n asentaa se järjestelmään tarvittavan kernel-ajurin.

Io.dll ei suinkaan ole ainoa saatavilla oleva ajuri. Maksullisten ajureiden lisäksi netistä löytyy etsimällä myös Freeware- ja Shareware-ajureita. Io.dll ei, ikävä kyllä, ole avoimen lähdekoodin ohjelma, mutta vaakakupissa painaa enemmänkin luotettavuus, joten päädyin käyttämään kyseistä ajuria. Muitakin hyviä vaihtoehtoja on ja totesin muun muassa avoimeen lähdekoodiin perustuvan inpout32.dll-ajurin toimivaksi vaihtoehdoksi.

Io.dll

Io.dll on dynaaminen linkkikirjasto (Dynamic Link Library), jonka voi linkittää ohjelmakoodin yhteyteen. Ohjelmakoodista voi tällöin kutsua dll-tiedoston funktioita aivan kuin ohjelman omia funktioita. Io.dll sisältää lukuisia erilaisia funktioita, mutta moottoreiden ohjaamiseen tarpeen oli ainoastaan PortOut-aliohjelma, joka hoitaa bittien lähettämisen. Delphissä dll-kirjaston linkittäminen tapahtuu seuraavanlaisesti. Varsinainen io.dll-tiedosto pitää tallentaa samaan hakemistoon ohjelman kanssa tai Windowsin system-kansioon.

```
procedure PortOut(Port : Word; Data : Byte); stdcall; external 'io.dll';
```



Aliohjelman parametri “Port” tarkoittaa käytettävän portin osoitetta (katso Rinnakkaisportin toimintaperiaate) ja ”Data” lähetettävää lukua (katso Moottoreiden ohjaus rinnakkaisportin avulla).

Miten aliohjelma toimii?

Moottoria ja laseria koskevat funktiot ovat moduulissa Moottori.

Moottoreita liikuttava aliohjelma on toteutettu hyvin yksinkertaisesti. Liikkuminen perustuu toistettavaan while-silmukkaan, jota toistetaan kunnes haluttu määrä askelia (intMoottori1 ja intMoottori2) on liikuttu. Silmukan sisällä suoritetaan muuttujien arvoista riippuen byteOut-muuttujan arvon asettaminen ja lopulta lähetys rinnakkaisportin kautta moottoreille. Sopiva viive eli moottorin pyörimisnopeus määritellään millisekunteina (intOdota). Viive toteutetaan sleep-funktiolla.

Lähempään tarkasteluun on poimittu Moottorin 1 ehtolause, jossa kyseistä moottoria käännetään suuntaan – eli byteOut-muuttujan arvo vaihtelee välillä xx11 ja xx10. Nämä arvot pätevät ainoastaan intMoottori-muuttujan ollessa nollaa suurempi.

```
// Moottori 1
{ arvo vaihtelee välillä ......11 ja ......10 }
if intMoottori1 > 0 then
```

Jos bittioperaattori and palauttaa luvun 1, sisältää muuttuja byteOut jo bitin xxx1. Tällöin byteOut-muuttujasta vähennetään 1.

```
if byteOut and 1 = 1 then
    dec(byteOut,1)
```

Jos bittioperaattori and ei löydä bittiä 1, lisätään muuttujaan 1. Koska moottori liikkuu aina nousevan pulssin myötä, voidaan päätellä moottorin nyt liikkuneen yhden askeleen. Siksi intMoottori1-muuttujan arvoa vähennetään yhdellä.

```
else
begin
    inc(byteOut,1);
    dec(intMoottori1);
end;
```

Moottorin 1 liikkuminen toiseen suuntaan on toteutettu samalla tavoin, kuten myös Moottorin 2 liikkuminen.

En väitä, että tämä olisi nopein, paras tai edes selkein tapa toteuttaa moottoreiden ohjaus. Tämän aliohjelman näyttää kuitenkin selkeästi moottoreiden ohjaamiseen käytettävien komentojen eri variaatiot. Tämä koodi ei varsinaisesti takaa moottorin liikkumista tasaisella nopeudella. Vaihtoehtoinen tapa olisi jättää sleep-aliohjelma pois ja tarkistaa kulunut määrä millisekunteja silmukansisäisellä ehtolauseella. Lisäksi ehtolauseiden määrää olisi voinut vähentää muun muassa muuttamalla bitin arvo takaisin alkuperäiseen arvoon jo heti samalla silmukan kierroksella.

Rinnakkaisportin ohjauksen yhteydessä ilmeni joillakin koneilla ongelmia alkuun. Välillä portti oli ikään kuin nukkumassa eikä halunnut totella sille lähetettyjä käskyjä. Ongelma korjaantui, kun otin käyttöön rinnakkaisportin ohjaussignaalit. Vaikka itse mittalaitetta ei

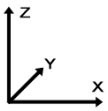
kiinnosta signaalien arvot, herättää niiden asettaminen portin ainakin eloon. Asettamalla linjat päälle taataan laitteen toimiminen.

Koska datalinjat varaavat kaikki vapaat bitit perusosoitteesta, on ohjauslinjojen osoite aina kahta numeroa suurempi kuin perusosoite. Esimerkiksi, jos perusosoite on 0x3BC, on ohjauslinjojen osoite 0x3BE (956 + 2). Linjojen kytkeminen tapahtuu yksinkertaisella AvaaPortti-funktiolla.

Laserin ohjaus

Laseria ohjataan, moottoreiden tapaan, rinnakkaisportin avulla. Moottorit varasivat datalinjoista neljä, joten viides on sopivasti vapaa käytettäväksi laseria varten. Rinnakkaisportin linja ohjaa transistoria, jonka avulla rele saadaan kytkettyä päälle ja pois. Kun linja on päällä, kulkee virta ja laser on päällä. Vastaavasti linjan ollessa nollassa ei virta kulje, eikä laser ole päällä.

Laseria ohjaava aliohjelma on niputettu yhteen moottoreita ohjaavan koodin kanssa samaan moduuliin, sillä kaikki moduulin osat hyödyntävät rinnakkaisporttia. Laserin aliohjelma on erittäin yksinkertainen. Ainoa huomioitava seikka laserin ohjauksessa on, että moottoreita ohjaavien linjojen pitää antaa olla tosia, sillä muuten moottori tulkitsee laserin sytyttämisen ja sammuttamisen ylimääräisenä askeleena. Laser sytytetään lähettämällä portille luku 31 (00011111_2) ja sammutetaan luvulla 15 (00001111_2).

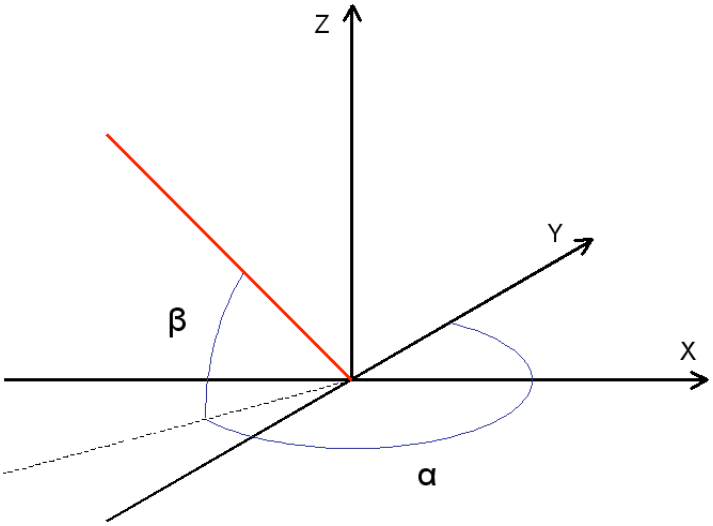


Kolmiulotteisuuden luonti

Kun laite on moottoreiden avulla siirretty oikeaan asentoon ja kamera on ottanut halutun kuvan, on kaikki tarvittava data kerätty. Seuraavaksi pistetunnistuskoodi hakee kuvasta laserpisteen kohdan. Jos pistettä ei löydy, näytetään kuvaa myöhemmin käyttäjälle, joka mahdollisesti tietokonetta paremmalla menestyksellä löytää pisteen kuvasta. Kun kaikki nämä toiminnot on käyty läpi, on vuorossa kolmiulotteisen mallin luominen saadusta informaatiosta.

Kolmiulotteisuus saadaan aikaan sijoittamalla pisteitä kolmiulotteiseen koordinaatistoon. Koordinaatiston akselit ovat X, Y ja Z. Ne on tässä tapauksessa määritelty siten, että Z-akseli suuntautuu ylös ja esittää näin korkeutta. X ja Y määrittelevät leveyden ja syvyyden. Laitteen paikka on origossa.

Koordinaatistoon sijoitettavat pisteet vaativat kolme arvoa. Nämä saadaan laskettua, kun tiedetään etäisyys pisteeseen origosta sekä laitteen kääntymiskulma pysty- ja vaakatasossa. Nimitetään vaakatason kääntymiskulmaa alfaksi (α) ja pystytason kulmaa beetaksi (β). Etäisyys pisteeseen olkoon d.



Trigonometrisia funktioita käyttäen voidaan laskea x-, y- ja z-koordinaatit:

$$X = \cos\alpha \ \cos\beta \ d$$
$$Y = \sin\alpha \ \cos\beta \ d$$
$$Z = \sin\beta \ d$$

Saadut tulokset eivät ole tarkkoja, sillä matkan varrella on monta epätarkkuustekijää. Pistetunnistuksen epävarmuuden ja epätarkkojen kulma-arvojen lisäksi epätarkkuutta aiheuttaa oletus, että saatu etäisyys on kohtisuorassa kameran linssiä vasten. Tämä yksinkertaistus on tietoinen, sillä laskemalla koordinaattien poikkeama kohtisuorasta ei saavuteta merkittävää parannusta laitteen tarkkuuteen. Lisäksi kannattaa huomioida, ettei kamera suinkaan ole lattiatasossa. Kolmiulotteisuuden luontia hyödynnetään Tiedosto-moduulin AvaaMittaus-aliohjelmassa.

Laitetta ohjaava ohjelma

Toimintaperiaate

Etäisyysmittalaitetta ohjaava ohjelma joutuu koordinoimaan monia tehtäviä. Laitteen kääntäminen, laserin kytkeminen, kuvan ottaminen ja lataaminen tietokoneelle, pisteiden etsiminen ja etäisyyden laskeminen on saatava hoidettua oikeassa järjestyksessä. Lisäksi erilaisten ongelmakohtein ilmenemisestä olisi suoriuduttava kunnialla.

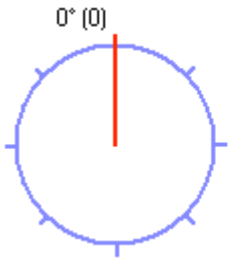
Mittalaite vaatii kalibrointia, jotta tulokset olisivat lähelläkään todellisuutta. Kalibrointi on muusta mittaohjelmasta erillään omana sovelluksenaan. Ennen mittauksen aloittamista on kuitenkin tehtävä joitakin säätöjä. Näitä ovat:

Mittaustapa tarkoittaa mittauksen tarkkuutta. Käyttäjä voi määritellä kuinka monta erillistä mittausta laite tekee vaakatasossa ja pystytasossa. Vaaka-arvo voi vaihdella kahden ja 257:n välillä. Pystytasossa pisteitä voi enimmillään olla 57. Enimmillään mittapisteitä saadaan siis 257 * 57 = 14 649 kpl. Näin monen pisteen tallentaminen ei ole suositeltavaa.

Porttien osoitteet kertovat ohjelmalle mistä kameran ja mittalaitteen tavoittaa. Kameralle voi valita sarjaportin COM1 tai COM2. Mittalaite voi olla liitettynä rinnakkaisporttiin LPT1, LPT2 tai LPT3. Joillakin koneilla rinnakkaisportti LPT1 löytyy valikosta LPT2:n kohdalta. Muistutuksena tästä nimiin on sulkeissa merkitty vaihtoehtoinen nimi. Rinnakkaisportin toimivuutta voi kokeilla asetettaessa *laite nolla-asemaan.*

Mittausvakiot ovat kalibroinnin kautta saatuja arvoja. Näitä ovat laserin beeta-kulma sekä linssin ja laserin välinen etäisyys (vakio a). Lisäksi tarvitaan kameran aukeamiskulman puolikas alfa (noin 24,7273).

Kun tarvittavat säädöt on tehty, voi laite aloittaa mittaamisen. Mittaus tapahtuu siirtämällä laitetta käyttäjän määrittelemän matkan kerrallaan myötäpäivään, minkä jälkeen laite mittaa yhden pisteen vaakatasossa. Tämän jälkeen laite mittaa halutun määrän pisteitä pystytasossa.



Laite liikkuu, oheisen kuvan mukaisesti, tietyn matkan kerrallaan aina seuraavaan pysähdyspisteeseen. Kuvassa laitteelle on annettu vaakatasossa kahdeksan ja pystytasossa neljä mittauskohtaa. Punaisella merkitty vaihe esittää laitteen sijaintia.

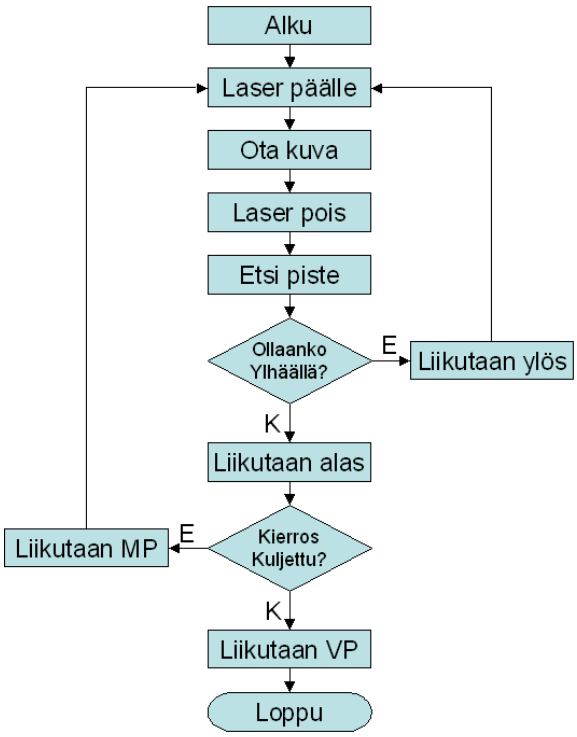
Ennen kuin ohjelma alkaa käännellä laitetta askelmoottoreiden avulla. On tarpeen laskea kerralla liikuttavat askelmäärät. Käyttäjä on syöttänyt, kuinka monta erillistä mittauskohtaa hän haluaa vaaka- ja pystytasossa (Pysty ja Vaaka). Nämä arvot ovat kokonaislukuja.

Koska laite kääntyy jokaista askelta kohti 1,4°, on kokonainen kierros (360°) noin 257 askelta. Pystysuunnassa laitteen ei ole tarkoitus kääntyä kokonaista kierrosta, eikä se edes ole mahdollista, sillä kameran kaapelit ovat esteenä. Kamera pääsee kääntymään pystytasossa 56 askelta (78,4°) ylös vaakatasosta. Tämä riittää mainiosti, sillä kuvien ottaminen suoraan ylöspäin ei ole mielekästä. Kaapeleiden liittimiä muuttamalla on tämäkin mahdollista.

Laite liikkuu mittapisteiden välillä tietyn määrän kokonaisia askelia. Nämä kokonaisluvut voidaan laskea:

```
DeltaV := 56 div Pysty;
DeltaH := 257 div Vaaka;
```

DeltaV viittaa pystytasossa (Vertical) liikuttavaan määrään askelia ja DeltaH vaakatason (Horizontal) askelmäärään. Ohjelma olettaa lisäksi, että käyttäjä haluaa mitata arvot nimenomaan kääntymiskulman ääripäistä, joten Pysty-muuttujasta vähennetään yksi. Näin saadaan laskut täsmäämään. Laitteen liikkuessa myötäpäivään (MP) tai ylös liikutaan aina delta-muuttujien antaman määrän askelia kerralla. Liikuttaessa alas liikutaan aina muuttujista välittämättä takaisin vaakatasoon. Mittaustapahtuman aikana laite liikkuu vain kerran vastapäivään (VP). Tämä tehdään mittauksen lopuksi, kun laite palaa takaisin alkuasentoon.



Vuokaavio esittää mittausta ohjaavan aliohjelman rakennetta. Kaikki toiminnot on jaettu omiin moduuleihin, josta niitä lyhyesti ja ytimekkäästi voi kutsua tarpeen tullen. Näin muita osioita koordinoivasta koodista saa mahdollisimman selkeää. Laitetta ohjaava koodi toimii hyvin pitkälti vuokaavion esittämällä tavalla. Lisäksi käyttäjälle ilmoitellaan, missä vaiheessa kulloinkin ollaan, mikä tuo oman lisänsä koodin joukkoon.

Käyttäjän kannalta väliaikatietojen saaminen mittauksesta on mukavaa. Vielä enemmän käyttömukavuutta tuo mittausprosessin hallittu keskeyttäminen tarpeen tullen. Parisataa erillistä mittapistettä käsittävän mittauksen haluaa mielellään pystyä keskeyttämään

tarvittaessa. Prosessin lopettaminen väkisin CTRL-ALT-DEL -yhdistelmällä on ikävä tapa keskeyttää mittaus. Tästä syystä halusin upottaa ohjelmaan mahdollisuuden keskeyttää mittaus. Lisäksi mahdollisuus keskeytyneen mittauksen jatkamiseksi oli ominaisuus, jota pidin tarpeellisena.

Mittauksen keskeyttäminen ja jatkaminen perustuu siihen, että silmukoista on mahdollisuus poistua suorittamatta niitä loppuun saakka. Kaikki mittausta käsittelevät muuttujat on määritelty moduulin sisällä globaaleiksi, joten aliohjelman suoritusta voi jatkaa siitä, mihin jäätiin.

Mitatut arvot tallennetaan yksiulotteiseen taulukkoon. Tieto siitä, mikä arvo kuuluu mihinkin, on laskettavissa muiden muuttujien avulla. Mittauksen valmistuttua ohjelma siirtyy varmentamaan tuloksia.

Tulosten varmentaminen perustuu yksinkertaisesti siihen, että tietokoneen saamat tulokset pitää käyttäjän toimesta vielä varmistaa, jotta optimaalinen tulos voidaan taata. Varmistaminen tapahtuu niin, että kone näyttää mittauksessa saatuja kuvia ja esittää omat päätelmänsä pisteen sijainnista. Käyttäjä joko hyväksyy tuloksen tai muuttaa sen mieleisekseen. Varmentamisen lopuksi tulokset pitää vielä tallentaa myöhempää käyttöä varten.

Tallennusmuoto

Käyttäjän on pystyttävä tallentamaan mittauksessa saadut tulokset. Tulosten tallentaminen suoraan johonkin yleisesti tunnettuun kolmiulotteisen vektoridatan tallennusformaattiin oli yksi vaihtoehto, mutta halusin pystyä tallentamaan tulokset aivan omassa muodossa. Syy tähän oli, että näin tuloksia saattoi tarkastella myös mittauksen jälkeen tulkitsematta niitä vaikeaselkoisesta koordinaateista ja polygoneista koostuvasta informaatiosta. Lisäksi mittaustiedostosta voi ladata asetuksia seuraavaa mittausta varten, jolloin käyttäjä säästyy vaivalta syöttää arvoja käsin.

Helpoin tapa tallentaa informaatiota erilaisista muuttujista tiedostoksi on käyttää verta vasten tähän tarkoitettuja funktioita, joilla muuttujien tyypit ja arvot siirtyvät tiedostoon binaarimuodossa. Tällaisen tiedoston avaaminen käy näppärästi, kunhan tietää missä järjestyksessä ja miten tavaraa on tiedostoon tallennettu. Koneelle erittäin selkeä tarkoittaa ihmiselle epäselvää ja tällä tavalla tallennettujen arvojen tarkastelu tai muuttaminen ilman tarkoitukseen räätälöityä ohjelmaa on hankalaa.

Tarkoitukseni oli tehdä tallennusmuodosta ennen kaikkea selkeä. Päätin tallentaa muuttujien arvot tekstinä tiedostoon. Näin tulosten tarkastelu ja muokkaaminen käsin helpottui huomattavasti. Kun tiedosto kerran on puhtaasti tekstiä, on turha alkaa tiedostopäätteen kanssa pelleilemään; päätin antaa tiedostoille päätteen txt.

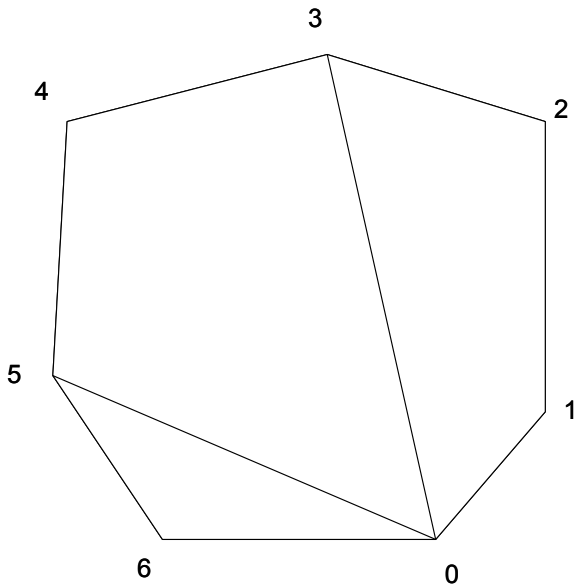
Tiedostot noudattavat ohessa esitettyä rakennetta. Risuaita (#) rivin alussa tarkoittaa kommenttia ja tyhjät rivit jätetään myös huomioimatta luettaessa. Alun määrittelyosuus noudattaa tiukasti esitettyä järjestystä, eikä arvoja saa jättää välistä pois. Mittapisteet puolestaan ovat juuri tietylle mittaukselle ominaisia saatuja arvoja. Kuvan nimen, joka kertoo kääntymiskulmat askeleina, ja etäisyyden väliin ei voi kirjoittaa kommenttia tai jättää tyhjää riviä. Tiedostonkäsittelyfunktiot löytyvät moduulista Tiedosto.

```
#Etäisyysmittaus
#Arno Solin, Päättötyö 2004-2005, versio 1.0
```

```
#Nimi
Mittausesimerkki
#Alfa
24.7273
#Beeta
65.0
#Vakio_a
25
#Sarjaportin osoite
1
#Rinnakkaisportin osoite
1
#Pisteet pystytasossa; Vertical
5
#Pisteet vaakatasossa; Horizontal
16

Mittapisteet:
000_00.jpg
  0.000
000_14.jpg
  0.955
000_28.jpg
  27.389
000_42.jpg
  6.369
000_56.jpg
  8.599

#Pisteet jatkuvat...
```



Moduulin Tiedosto AvaaMittaus-funktio luo edellä esitettyjen arvojen perusteella kolmiulotteisen kappaleen laskemalla pisteiden koordinaatit. Erilliset pisteet yhdistetään polygoneiksi. Sivuilla rinnakkaiset pisteet yhdistetään nelikulmioiksi, mutta mallin pääliosan polygonit lasketaan kuvan osoittamalla tavalla.

Tulosten optimointi

Ohjelman toimintaperiaate

Tulosten optimointi viittaa mittaustulosten muokkaamiseen järkevään muotoon. Saadut mittauspisteet ovat enemmän tai vähemmän hajanaisia, eikä välttämättä näytä järkeviltä alkuunkaan. Tulokset voi luonnollisesti tallentaa sellaisinaan VRML-muotoon, jolloin niiden tarkastelu ja muokkaus ulkopuolisilla 3D-ohjelmilla on mahdollista.

Usein on kuitenkin tarpeen seuloa mittatiedoista esiin olennainen. Tällöin kaivataan toimintoa, joka pystyy tarkastelemaan dataa ja löytämään siitä suuret linjat. Olennaisen löytäminen helpottuu mitä enemmän mittapisteitä on käytettävissä, jolloin voidaan laskea keskiarvoja ja suhteuttaa muotoja dataan. Perusperiaatteena on, että vaikka yksi mittapiste voi erehtyä paljon, vähenee virhemarginaali suurta pistejoukkoa tarkasteltaessa. Tällaista optimointia suorittavan ohjelman voi tehdä erittäin älykkääksi ja hahmottamaan monenlaisia muotoja ja epäsäännöllisyyksiä. Tämä työn osa-alue kuuluu niihin, joista yksin olisi voinut tehdä päättötyön pelkästään sen laajuuden vuoksi.

Tekemäni optimointivaihtoehto perustuu karkeaan oletukseen. Se olettaa mitatun tilan olevan suorakulmaisen särmiön muotoinen. Optimointia ei siis pidä käyttää, jos mitattu tila ei muistuta suorakulmaista särmiötä.

Ohjelma sovittaa mittaustulosten kaikkiin vaakasuoraan mitattuihin arvoihin suorakulmiota. Sovitus tapahtuu kömpelösti regressiosuorien avulla. Regressiosuorille määritetään korrelaatiokertoimet, joiden perusteella määritetään sivujen pituus ja suorakulmion asento. Lopuksi suorat asetetaan oikeaan kohtaan, suorakulmaisesti toisiinsa nähden. Empiirinen tutkimus osoitti tämän tavan kaikkein tehokkaimmaksi sovituskeinoksi.

Optimoinnissa käytetään pisteiden sovittamista samalle suoralle. Ohessa on Maol-taulukoista otettu regressiosuoran (y = bx + a) sekä suoran korrelaatiokertoimen kaavat.

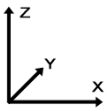
$$b = \frac{n\sum x_i y_i - (\sum x_i)(\sum y_i)}{n\sum x_i^2 - (\sum x_i)^2}$$

$$a = \frac{\sum y_i - b\sum x_i}{n}$$

Pearsonin korrelaatiokerroin (tulomomenttikerroin):

$$r = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{[n\sum x_i^2 - (\sum x_i)^2][n\sum y_i^2 - (\sum y_i)^2]}}$$

Jos tulokset eivät edelleenkään optimoinnin jälkeen näytä järkeviltä voi käyttäjä miettiä, josko jokin yksittäinen mittavirhe vääristää tuloksia olennaisesti. Tällöin arvon muuttaminen käsin saattaa olla hyvä vaihtoehto. Optimointi ei ole pakollista, mutta siistii tuloksia huomattavasti tehden niistä samalla ikävän selkeitä. Koodi sekä lisätietoja optimoinnin toiminnasta löytyy moduulista Optimoi.



Tallennus VRML-muotoon

VRML (Virtual Reality Modeling Language) on yleisesti käytetty tiedostomuoto kolmiulotteisten kappaleiden tallentamiseen. VRML on suunniteltu käytettäväksi 3D-tiedostojen levittämiseen ja esittämiseen internetin välityksellä. Se on tekstipohjainen tiedostomuoto, jossa komiulotteiset kappaleet voidaan määritellä erilaisten pisteiden ja polygonien kautta. Kappaleille voidaan määritellä pintaväri ja -materiaali, kirkkaus, heijastusarvo, läpinäkyvyys ynnä muita ominaisuuksia. Kappaleiden animoiminen, linkittäminen ja erilaisten skriptien upottaminen ovat myös osa VRML-maailmaa.

VRML-tiedostoja kutsutaan maailmoiksi (World) ja niiden pääte on .wrl. Vaikka tiedostot ovat tekstiä, saatetaan niitä joskus oletusarvoisesti pakata tilan säästämiseksi. Ensimmäinen VRML-versio määriteltiin vuonna 1994. Nykyään VRML-määrittelyn kehittelyä hoitaa Web3D Consortium, joka virallisesti suosittelee uuteen X3D-standardiin siirtymistä.

Yhä yleisesti käytetty ja perusominaisuudet sisältävä versio 2.0 on päättötyöhöni sopiva. Itse asiassa en hyödynnä VRML-muodon ominaisuuksia kuin nimeksi. Ideana on kuitenkin vain saada mittaustulokset siirrettyä yleisesti tuettuun tiedostomuotoon, jotta niitä olisi mahdollisuus hyödyntää.

Ohessa on VRML-moduulin Tallenna-aliohjelmalla luotu wrl-tiedosto. Tiedosto sisältää yksikertaisen kuution, joka koostuu kahdeksasta kulmapisteestä ja näiden välille pingotetuista kuudesta tahkosta, polygonista.

Kommentit on eroteltu risuaitamerkillä (#). Alussa kerrotaan olevan kyse VRML 2.0 -tiedostosta. Lisäksi, hyvän tavan mukaan, kerrotaan millä tiedosto on luotu ja varmuuden vuoksi vielä päivämäärä. Päivämäärä on suomalaisittain järjettömässä muodossa. Annettakoon tämä kuitenkin anteeksi, sillä tulin siihen tulokseen, että tämä on kansainvälisesti kaikkein selkein esitysmuoto.

Tämä tiedosto sisältää ainoastaan yhden komiulotteisen kappaleen (nimi: mittaus). Translation määrittelee kappaleen nollapisteen poikkeaman keskipisteestä. DiffuseColor tarkoittaa kappaleen väriä (RGB muotoa 0.00 – 1.00 kutakin väriä kohti). Coord DEF -osio määrittelee kappaleen pisteiden x-, y- ja z-arvot. Arvo x on leveys, y korkeus ja z viittaa syvyyteen. Pisteet nimetään nollasta aloittaen. CoordIndex puolestaan määrittelee mitkä pisteistä ovat minkäkin polygonin kulmia.

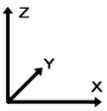
```
#VRML V2.0 utf8

# Arno Solin, Päättötyö, VRML exporter 1.0
# Date: feb 5 2005 22:40

DEF mittaus Transform {
  translation 0 0 0
  children [
    Shape {
      appearance Appearance {
        material Material {
          diffuseColor 1.0 0.5 0.5
        }
      }
      geometry DEF mittaus-FACES IndexedFaceSet {
        ccw TRUE
```

```
            solid TRUE
            convex TRUE
            coord DEF mittaus-COORD Coordinate { point [
              50.00 50.00 -50.00,
              50.00 50.00 50.00,
              -50.00 50.00 50.00,
              -50.00 50.00 -50.00,
              50.00 -50.00 -50.00,
              55.00 -50.00 50.00,
              -50.00 -50.00 50.00,
              -50.00 -50.00 -50.00]
            }
            CoordIndex [
              3, 2, 1, 0, -1,
              4, 5, 6, 7, -1,
              0, 1, 5, 4, -1,
              1, 2, 6, 5, -1,
              2, 3, 7, 6, -1,
              3, 0, 4, 7, -1]
          }
        }
      ]
    }
```

VRML-moduulin Avaa-aliohjelma olettaa tiedostojen mukailevan edellä esitettyä rakennetta. Päättötyön ulkopuolisia tiedostoja avattaessa kannattaa kiinnittää huomiota tiedostojen rakenteeseen ja sisennyksiin.



Tulosten 3D-tarkastelu

Toimintaperiaate

Mittauksen valmistuttua ja vapaaehtoisen optimoinnin jälkeen olisi mukava vielä saada selville, mitä tuloksia oikeastaan on saatu. Jos on tottunut kolmiulotteisten kappaleiden tarkasteluun tekstimuodossa, voi arvoja selaamalla saada käsityksen mittaustuloksista. Paljon mukavampaa kuitenkin on nähdä työn hedelmät visuaalisessa muodossa.

VRML-tiedostoja tuetaan miltei kaikissa kolmiulotteista dataa käsittelevissä ohjelmistoissa. Kaikki eivät kuitenkaan omista tällaisia suhteellisen vaikeakäyttöisiä ja perehtymistä vaativia ohjelmia. Toisaalta, jos tarkoituksena on pelkästään katsoa tuloksia, ovat tarpeen vain perusominaisuudet.

Päätin tehdä vielä neljännen ohjelmaosion päättötyöhöni. Sen tarkoitus poikkesi olennaisesti muista, sillä kalibrointi, mittaus ja optimointi muokkasivat kaikki tuloksia tavalla tai toisella. Viimeinen osio on tarkoitettu tuloksista nauttimiseen, niiden tarkasteluun. Kyseessä on yksinkertainen 3D-ohjelma, jossa yksinkertaisia kappaleita voi tarkastella kolmiulotteisesti.

Delphi ei perusominaisuuksineen tarjoa moniakaan valmiita funktioita kolmiulotteisuuden luontiin ja valtavien omien rajapintojen ohjelmointi ei tuntunut mielekkäältä. Mahdollisuutena oli käyttää valmiita rajapintoja, kuten OpenGL- tai DirectX-rajapintaa. Hyvin optimoidun ja yleisesti käytetyn rajapinnan käyttö ollut fiksua, mutta päädyin jälleen omille poluilleni. Yksinkertaisen kolmiulotteisen kappaleen piirtäminen kaksiulotteiselle näytölle käy näppärästi trigonometristen funktioiden ja tavallisten piirtokäskyjen avulla.

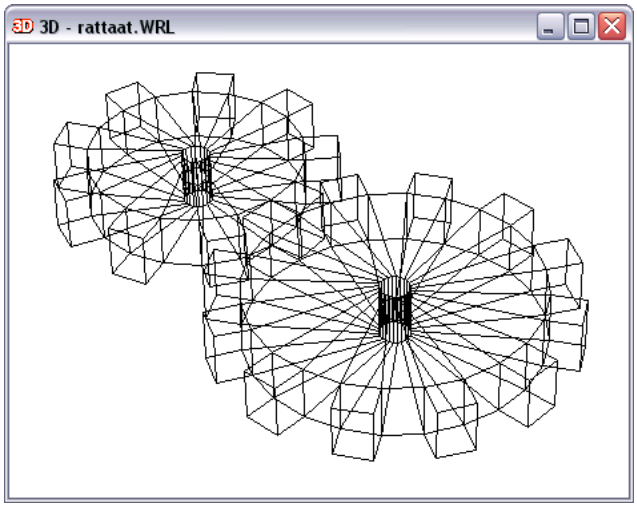
Itse asiassa tämän 3D-ohjelman kohdalla oli kyse jonkinlaisesta ympyrän sulkeutumisesta, sillä päättötyön tekeminen oikeastaan alkoi tästä ja loppui ohjelmoinnin osalta tähän. Idea ohjelmaan juontaa juurensa joskus ohjelmointiputka.net-ohjelmointisivuilla näkemääni Antero Korteilan tekemään kuution pyörittelyä esitelleeseen esimerkkiin. Jos kyseinen sivusto edelleen on pystyssä, löytynee tuo Visual Basicissa toteutettu lyhyt esimerkki vielä sieltä.

Tutkittuani Korteilan esimerkkiä johdin siinä käytetyt yhtälöt. Kyseistä tapaa laskea kolmiulotteisen pisteen paikkaa näkee siellä täällä. Kyseessä on helppotajuinen ja ymmärrettävä tapa esittää miten pisteen paikka voidaan laskea, kun tiedetään alkuperäinen koordinaatti ja kahden kääntymisakselin kulmat.

$$x_{piirto} = x\ \cos\alpha + y\ \sin\alpha$$
$$y_{piirto} = (y\ \cos\alpha - x\ \sin\alpha)\ \cos\beta - z\ \sin\beta$$
$$z_{piirto} = (y\ \cos\alpha - x\ \sin\alpha)\ \sin\beta + z\ \cos\beta$$

Kaavassa x, y ja z ovat pisteen tallennettu koordinaatti, x_{piirto}, y_{piirto} ja z_{piirto} ovat uusi koordinaatti, α on haluttu kääntymiskulma vaakatasossa ja β on kääntymiskulma pystytasossa.

Päätettyäni päättötyöni aiheen mieleeni nousi jälleen nämä yhtälöt. Yhdistämällä lasketut pisteet oikeassa järjestyksessä saadaan luotua rautalankamalli (wireframe), mutta jos halutaan saada kappale näyttämään aidolta, pitää polygonit (monikulmiot) piirtää esiin niin, että kappale näyttää kiinteältä.



Kuvassa mekaanista osuutta varten tehty 3d-malli hammasrattaista

Polygonin voi Delphissä piirtää valmiiksi peittävällä värillä. Tämä on suoraan käyttöjärjestelmän puolelta tuettu toiminto. Jotta kappale näyttäisi aidolta, on polygonit piirrettävä oikeassa järjestyksessä niin, että etualalla olevat polygonit piirretään taaempana olevien päälle. Idioottivarmin tapa toteuttaa polygonien piirto oikeassa järjestyksessä on yksinkertaisesti laskea jokaiselle polygonille syvyysarvo (y) ja järjestää polygonit tämän jälkeen oikeaan järjestykseen.

Polygonien järjestäminen

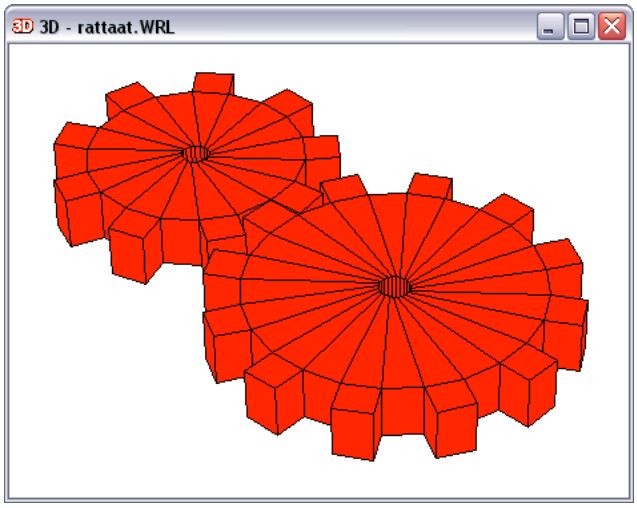
Lajittelualgoritmit ovat osa klassista ohjelmointia. Ne ovat mielenkiintoinen ja paljon tutkittu ohjelmoinnin osa-alue. Yksi yksinkertaisimpia lajittelualgoritmeja on Bubble Sort, joka perustuu järjestettävän taulukon läpikäymiseen verraten aina kahta lukua toisiinsa ja tarpeen tullen vaihtamaan lukujen paikkaa, jos luvut ovat väärässä järjestyksessä. Taulukon läpikäyntiä jatketaan yhä uudelleen kunnes se on järjestetty.

Bubble Sort on esimerkki erittäin yksinkertasiesta lajittelualgoritmista. Se ei kuitenkaan sovellu polygonien järjestämiseen, sillä se on erittäin hidas. Taulukko käydään läpi turhan monta kertaa ja arvojen paikkaa vaihdetaan turhan useasti. Algoritmien tehokkuus voidaan esittää niin sanotun ison O-merkinnän (Big O notation) avulla. Se kuvaa algoritmin yhtälön yksinkertaistettua asymptootin yhtälöä. Bubble Sort joutuu tekemään $O(n^2)$ vertailua, missä n on taulukon solujen lukumäärä.

Lajittelualgoritmeja on lukuisia ja ne ovat mielenkiintoinen ja haastava ohjelmoinnin osa-alue. On mahdotonta valita mitään yksinkertaisesti parasta algoritmia, sillä moni lajittelualgoritmi sisältää sekä heikkoja että vahvoja puolia. Käytettävä algoritmi määräytyy käyttökohteen ja käyttötavan mukaan. Usein ovat myös vastakkain nopeus ja vaaditun muistin määrä.

Polygonien järjestämisen on ennen kaikkea toimittava nopeasti, sillä käyttäjä haluaa kappaleen pyörivän sulavasti ruudulla. Quick Sort ei ehkä ole kaikkein parhaiten tarkoitukseeni sopiva algoritmi, mutta kiinnostava se joka tapauksessa on. Quick Sortin ymmärtäminen on hieman Bubble Sortia vaikeampaa. Quick Sort käyttää menetelmää, jota kutsutaan nimellä hajota ja hallitse (divide and conquer). Se valitsee taulukosta luvun (ns. pivot-luku), johon verraten se jakaa listan pivot-lukua pienempiin ja suurempiin arvoihin. Tässä vaiheessa algoritmi kutsuu itseään ja järjestää jaetun taulukon samalla periaatteella

ikään kuin kahtena taulukkona. Koska algoritmi kutsuu itseään, on se rekursiivinen. Algoritmi jatkaa toimintaansa jakaen taulukkoja yhä pienempiin osiin ja järjestäen näitä.



Quick Sort on erittäin tehokas. Se suorittaa keskimäärin O(n log(n)) vertailua, missä n on taulukon koko. Sillä on kuitenkin omat varjopuolensa. Huonoin mahdollinen tulos noudattaa yhtälöä $O(n^2)$. Erityisen tärkeää on valita sopiva pivot-luku, sillä pieleen mennyt valinta voi ajaa algoritmin pyörimään syvään suohon suorittamaan paljon ylimääräistä työtä. Erityisen ongelmallinen algoritmi on sellaisille taulukoille, jotka ovat jo valmiiksi järjestyksessä. Pivot-luku voidaan valita satunnaisesti, jolloin huonoin mahdollinen tulos on erittäin epätodennäköinen.

Toinen huomioitava seikka on rekursiivisuus, joka vaatii paljon muistia. Funktion kutsuessa itseään se varaa jokaiselle kopiolle itsestään muistia. Tästä syystä Quick Sortia on pitkään pidetty epäkäytännöllisenä ja voikin sanoa, että vasta muistimäärien pyöriessä sadoissa megatavuissa on Quick Sort varteenotettava vaihtoehto lajittelualgoritmiksi.

Valitsin Quick Sortin syystä, että se on mukavan monipuolinen ja selviytyy niistä korkeintaan muutamasta tuhannesta järjestettävästä polygonista aivan tarpeeksi nopeasti. Tutkin erilaisia Quick Sortin toteutustapoja internetissä ja löysin jopa pari valmiiksi Pascalissa toteutettua funktiota. Ne olivat kuitenkin valtavan pitkiä ja monimutkaisia, vaikeaselkoisesti toteutettuja. Ajattelin toteuttaa oman, tiivistetyn, version Quick Sort -lajittelualgoritmista.

```
procedure qsort(var a: array of integer; alku, loppu: integer);
var
  l,r,luku: integer;
begin
  if loppu > alku then
  begin
    luku := a[loppu];
    l := alku -1;
    r := loppu;
    while r > l do
    begin
      repeat inc(l) until a[l] >= luku;
      repeat dec(r) until a[r] <= luku;
      vaihda(a[l],a[r]);
    end;
    vaihda(a[l],a[loppu]);
    vaihda(a[loppu],a[r]);

    qsort(a, alku, l-1);
    qsort(a, l+1,loppu);
  end;
end;
```

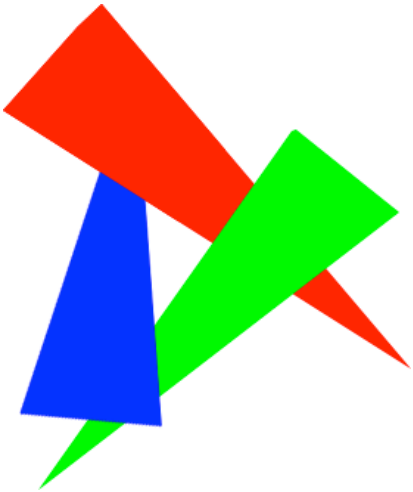
Ohjelmointi osoittautui vaikeaksi ja vasta kahden pitkän illan jälkeen sain algoritmin tuottamaan haluttuja tuloksia. Työ näyttää kannattaneen, sillä vaikka pivot-lukua ei arvota,

toimii funktio kiitettävällä nopeudella. Se itse asiassa pesi mennen tullen valmiit Pascal-esimerkit.

Silkasta mielenkiinnosta päätin testailla algoritmin suoritusrajoja. Järjestettävien pisteiden määrien pyöriessä muutamissa tuhansissa kului testikoneelta (Pentium 1,7 GHz, 1024 MB ram, Win-XP) niin vähän aikaa lajitteluun, ettei tuloksia saanut talteen. Miljoonan kokonaisluvun järjestäminen suuruusjärjestykseen kesti testikoneelta 157 ms (noin 0,16 s) kymmenen kerran keskiarvona. Kymmenen miljoonaa arvoa järjestyi 1616 millisekunnissa. Vertailun vuoksi voi todeta, että toisella Quick Sortia hyödyntävällä algoritmilla kymmenen miljoonan arvon järjestäminen kesti noin puolitoista minuuttia.

Tarpeeksi suuri määrä järjesteltäviä arvoja tuo näkyviin Quick Sortin huonot puolet. Kymmenen miljoonaa järjestettävää lukua varaa muistia käyttöön 40 MB. Sata miljoonaa arvoa haukkaa jo 400 megatavun leijonaosan käyttöönsä.

Todeta voi, että Quick Sortin nopeus riittää muutaman tuhannen polygonin järjestämiseen huolimatta siitä, että osutaanko pahimpaan mahdolliseen tulokseen tai ei. Polygonien järjestämiseen tarkoitettu koodi eroaa edellä esitetystä yksinkertaisesta algoritmikoodista vain vaihdettavien lukuarvojen osalta. Polygonien numerot ja etäisyydet vaihdetaan aina rinnakkain.



Kulmiot ristikkäin. Näiden polygonien piirtojärjestys on aina väärä.

Polygonien piirtäminen järjestämällä ne syvyysarvon mukaiseen järjestykseen ei aina tuota haluttua tulosta. On olemassa muutama poikkeustapaus, jolloin kappale piirtyy näytölle väärin. Esimerkiksi polygonien ollessa limittäin toistensa päällä (katso kuva). Tällöin oikeaa järjestystä ei ole ja kuva piirtyy väärin joka tapauksessa. Polygonit saattavat piirtyä väärin myös joskus, kun samalla alueella on useita polygoneja, ja kun kärkipisteiden koordinaattien keskiarvot eivät palautakaan oikeaa järjestystä polygoniryppäästä. Tästä syystä juuri mikään oikea 3D-ohjelma ei piirrä polygoneja järjestelemällä niitä käyttämälläni tavalla. Järjestelytekniikkaa voikin sanoa köyhän miehen renderöinniksi.

Värittämällä polygonit niiden normaalien ja valon suunnan välistä kulmaa käyttäen voidaan saada aikaan parempi illuusio moniulotteisuudesta.

Ohjelmien käyttö

Mittalaitteen käyttäminen ei ole ylivoimaisen vaikea tehtävä. Laite on kuitenkin tarkoitettu jonkin verran asiaan paneutuneen ihmisen käytettäväksi. Tässä kirjallisessa osuudessa käsiteltyjen käsitteiden ymmärtäminen riittää mainiosti ohjelmien käyttämiseen. Ohjelmien yhteyteen ei ole liitetty tarkkoja käyttöohjeita. Lisäksi kaikkia mahdollisesti eteen sattuvia ongelmatilanteita ei ole voitu luonnollisestikaan ennakoida tai huomioida työtä tehdessä.

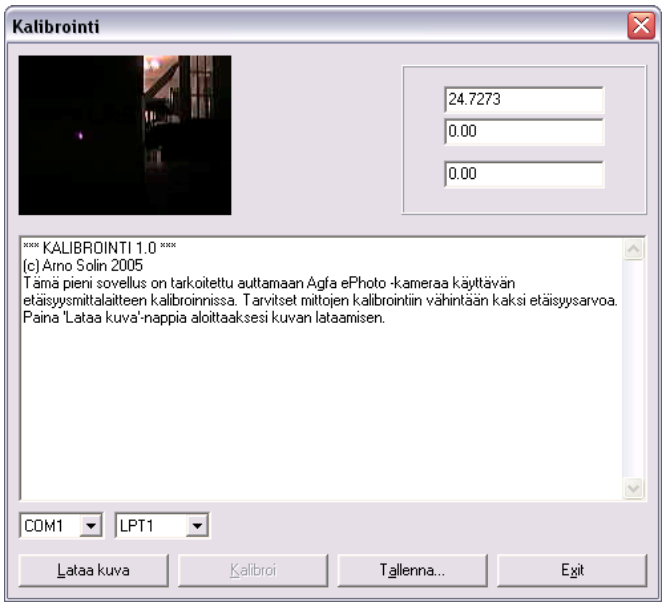
Ohjelmat on testattu toimiviksi ainakin käyttöjärjestelmissä Windows 2000 ja XP. Periaatteessa mitään estettä ohjelmien toimimiselle Windows 95, 98, NT ja ME ympäristöissä ei ole, mutta ohjelmia ei ole testattu kyseisillä alustoilla. Mitta- ja kalibrointiohjelmien käyttö vaatii, että tietokoneesta löytyy yksi vapaa sarjaportti ja rinnakkaisportti.

Ohjelmille ei ole määritelty laitevaatimuksia, joten toimivuus kannattaa yksinkertaisesti selvittää kokeilemalla. Hyvän käyttömukavuuden saavuttamiseksi kannattaa koneessa olla vähintään parinsadan megahertsin prosessori ja 32 MB keskusmuistia. Vapaata kovalevytilaa tarvitaan pari megatavua, minkä lisäksi mittaus vaati valitusta tarkkuudesta riippuen tilaa kuvien tallentamiseen.

Laitteen kytkeminen

Mittalaiteen D-liitin (15 napaa) kytketään piirilevykotelon naarasliittimeen. Seuraavaksi kytketään kotelon 25-napainen D-liitin tietokoneen rinnakkaisporttiliittimeen. Kameran sarjaportti kytketään 9-napaisen D-liittimen avulla tietokoneen sarjaporttiin. Lopuksi kytketään mittalaitteeseen virta (6 V, DC).

Laitteen kalibrointi



Mittalaitteen kalibrointi on ehdottoman tärkeää. Jollei sinulla ole voimassaolevaa kalibrointitiedostoa, on sellainen luotava. Jos kalibrointi on tarpeen, avaa Kalibroi-ohjelma. Varmista, että porttiasetukset ovat oikein. Jollet tiedä miten tietokoneesi portit on määritelty, voit joko tarkistaa asetukset ohjauspaneelista (Control Panel) tai

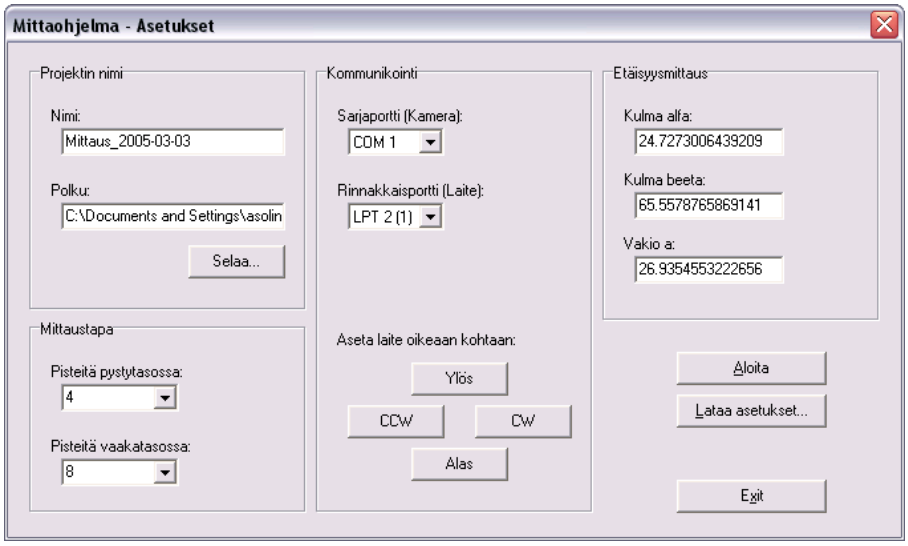
yksinkertaisesti kokeilemalla selvittää oikeita asetuksia. Rinnakkaisportin asetusten selvittäminen on yksinkertaista Mittaohjelmasta käsin.

Kun asetukset ovat kunnossa, paina ”Lataa kuva”-nappia ladataksesi kuvan. Kuvan ilmestyttyä näkyviin kannattaa luotettavien tulosten toivossa käsin määrittää kuvan laserpisteen paikka. Jos pistettä ei näy kuvassa, saattaa vikana olla väärin valittu rinnakkaisportti. Huoneen pimentäminen auttaa luotettavien tulosten saamisessa. Ohjelma pyytää vielä käsin syöttämään mittauspisteelle etäisyyden. Muista käyttää joka pisteen kohdalla samaa mittayksikköä (suositellaan: cm).

Ladattuasi mahdollisimman monta erilaista etäisyysarvoa, voit painaa ”Kalibroi”-nappia. Saadut tulokset näkyvät tekstikehyksessä ja vakioarvot päivittyvät lomakkeen oikeassa yläkulmassa. Voit edelleen jatkaa arvojen syöttämistä. Kalibroinnin valmistuttua paina ”Tallenna” ja valitse sopiva polku. Kalibrointitiedosto nimetään oletusarvoisesti päivämäärän mukaan.

Vinkki: Hyväksi havaittu kalibrointimenetelmä on käyttää kevyttä liikuteltavaa esinettä, jota voi siirtää eri etäisyyksille laitteesta. Etäisyyden määrittäminen käy näppärästi mittanauhan avulla.

Mittaus

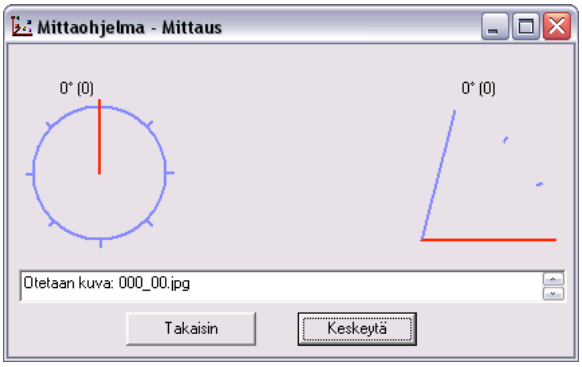


Varsinainen mittaus aloitetaan avaamalla ohjelma Mittaus. Ensimmäinen askel on määritellä ohjelmalle käytettävät asetukset. Asetusten määrittely kannattaa aloittaa lataamalla kalibrointitiedosto, joka asettaa paikalleen etäisyysmittauksen vakioarvot sekä porttien asetukset. Käyttäjän on kuitenkin mahdollisuus muokata arvoja haluamikseen. Ohjelma tarjoaa oletusarvoisesti päivämäärää projektin nimeksi ja tallennuspolkuna ohjelman oman kansion polkua. Mittauksen tarkkuuden käyttäjä voi määritellä kahdesta pudotusvalikosta lomakkeen vasemmassa reunassa. Lukemat viittaavat pysty- ja vaakatasossa mitattavien pisteiden määrään. Arvot voivat olla pystysuunnassa 2–56 ja vaakasuunnassa 4–257.

Käyttäjän täytyy varmistua vielä laitteen alkusijainnista. Itse mittalaite kannattaa sijoittaa keskelle mitattavaa tilaa lattiatasoon niin, että sen välittömässä läheisyydessä ei ole ylimääräisiä esteitä. Mittaohjelman asetuslomakkeen keskivaiheilla olevilla suuntaa merkitsevillä painonapeilla voi laitetta kääntää sopivaan kohtaan. Ennen mittauksen alkua

laite tulee sijoittaa pystysuunnassa täysin vaakatasoon. Vaakasuunnassa laite tulee kääntää sellaiseen asentoon, että kaapelit riittävät keskiakselin kiertymiseen 360°.

Mittaus aloitetaan painamalla ”Aloita”, jolloin ohjelma tarkistaa syötetyt asetukset ja siirtyy mittaustilaan. Painamalla jälleen ”Aloita” alkaa mittaus. Mittauksen voi keskeyttää painamalla kesken mittauksen ”Keskeytä”. Keskeytettyä mittausta voi myös jatkaa, mutta ylimääräisiä keskeytyksiä kannattaa välttää.



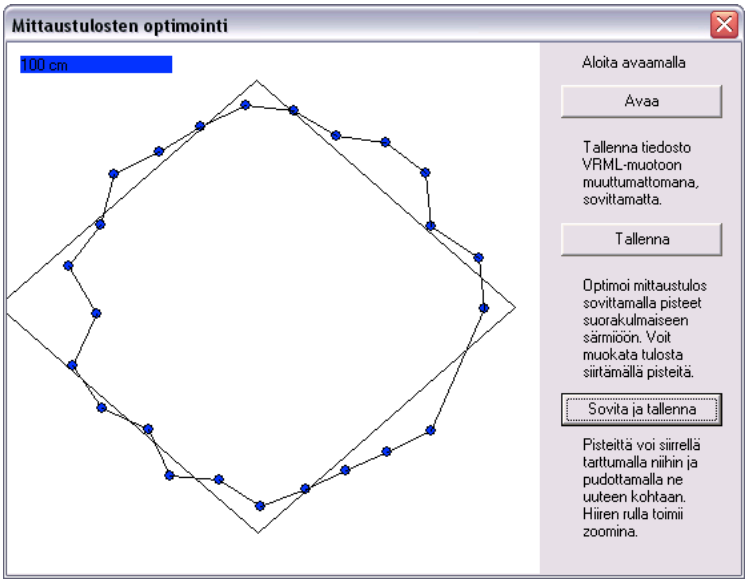
Jos mittaus ei kaikesta huolimatta pääse käyntiin, kannattaa se keskeyttää ja tarkistaa valittu sarjaportti. Jos mittaus kesken kaiken ilmoittaa kameran virheestä, kannattaa mittalaite käynnistää uudelleen (virta pois, virta päälle) ja jatkaa sen jälkeen mittausta. Kyse on tällöin kameran yhteyden keskeytymisestä.

Mittauksen edistymistä voi seurata näytöltä. Kaikkien pisteiden läpikäynnin jälkeen laite siirtyy takaisin alkupisteeseen ja ohjelma pyytää käyttäjää tarkistamaan saatuja tuloksia. Automaattinen pistetunnistus toimii hyvin vain poikkeusolosuhteissa (selkeät pinnat ja pimeä huone), minkä takia tulosten varmistaminen käsin takaa onnistuneen lopputuloksen. Saatujen kuvien läpikäynnin jälkeen ohjelma tallentaa tulokset (pääte txt) ja sulkee itsensä.

Optimointi

Jos mittaustuloksia on tarkoitus käyttää jossakin 3D-ohjelmassa, täytyy ne tallentaa muiden ohjelmien ymmärtämään muotoon. Optimointi-sovellus pystyy muuttamaan mittaustiedoston VRML-muotoon (pääte wrl). Pelkän muunnoksen lisäksi sovelluksella voi muokata saatuja tuloksia.

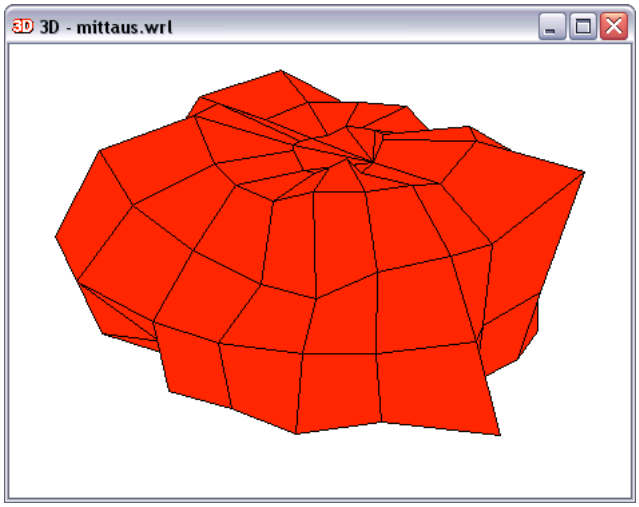
Optimointi aloitetaan avaamalla mittaustiedosto. Avaamisen yhteydessä sovellus laskee mittadatan pohjalta kolmiulotteisen mallin muodon ja polygonien järjestyksen. Tämän tiedon voi tallentaa muuttumattomana painamalla ”Tallenna”. Koordinaattipisteet voidaan sovittaa suorakulmaisen särmiön muotoiseen kappaleeseen. Tällöin painetaan ”Sovita ja tallenna”.



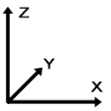
Pohjan sovitus tehdään alimpien mittapisteiden mukaan. Ne näkyvät jatkuvasti ruudulla. Sovitustuloksiin voi vaikuttaa siirtämällä pisteitä. Hiiren rulla (scroll) toimii zoomina ja suhde näkyy yläkulmassa. Mitat on ilmaistu senttimetreinä. Sovittamiseen vaaditaan yli neljä vaakapistettä. Sovitus ei aina tuota suorakulmaista särmiötä, vaikka särmiö syntyisikin. Näissä tilanteissa kannattaa tiettyjä ongelmapisteitä siirrellä.

Tulosten tarkastelu

Saatujen tulosten tarkastelu käy VRML-muunnoksen jälkeen lukuisissa eri ohjelmissa. Esimerkiksi 3ds Max -ohjelma tukee VRML-tiedostoja ja sisältää runsaasti ominaisuuksia tulosten hyödyntämiseen.



Jos tarpeen on vain tarkastaa mittauksen onnistuminen tai haluaa vain katsella tuloksia, voi mallin avata myös päättötyön yhteyteen kuuluvalla Ohjelma3D-sovelluksella. Sovellus ymmärtää sekä mittaustiedostoja että rakenteeltaan yksinkertaisia VRML-tiedostoja. Tiedoston voi avata antamalla tiedoston nimi (polku) ohjelman parametriksi. Tämä käy joko komentorivillä tai raahaamalla tiedosto ohjelman kuvakkeen päälle. Lisäksi parametriominaisuutta voi hyödyntää Windowsin Avaa sovelluksessa -toiminnolla.



Luonnollisesti tiedoston avaaminen käy myös ohjelmasta käsin. Näppäinoikotiellä (Ctrl + O) voi tiedoston avata. Tuetut tiedostopäätteet ovat txt ja wrl. Muut mahdolliset näppäinoikotiet ja hiiren toiminta selviävät painamalla sovelluksessa F1.

F1	Näytä ohje.
F2	Rautalankamalli (Wireframe)
F3	Värjätyt polygonit
F4	Varjostus
Ctrl + O	Avaa
Hiiri (vasen)	Kappaleen pyörittäminen
Hiiri (oikea)	Kappaleen siirtäminen
Hiiri (rulla)	Zoom-toiminto

Lopuksi

Jokaisen oppilaan luokallani piti keväällä 2004 pitää luokkatovereilleen puhe valitsemastaan aiheesta. Keksin tuolloin tehdä puheen ”Päättötyötään aloittavalle luokalle”. Tarttuminen päivänpolttavaan aiheeseen tuolloin osoittautui hyväksi ideaksi. Se oli aihe, josta oli sanottavaa. Se oli aihe, joka kiinnosti kuulijoita. Ja se oli aihe, jota voisin kommentoida oman päättötyöni loppukappaleessa.

Puheessa julistin melko mahtipontisesti, että ”Päättötyö on kohdattava haasteena. Haasteena, josta saatu voitto tulee tuntumaan hyvältä vielä vuosien kuluttua.” Nyt päättötyön valmistuttua voi vain olla tyytyväinen, että projekti on saatettu päätökseen. Haasteita ja haastavuutta olen saanut kokea enemmän kuin olisi ollut tarpeen, mutta se ei vähennä voiton arvoa. Itse asiassa päinvastoin. Projektin laajuus kattaa lähes sanonnan ”Kaikki, mikä ei tapa, vahvistaa” mittasuhteet.

Minulta on projektia tehdessä kysytty, mitä järkeä koko touhussa on. Sen saa minun puolestani jokainen lukija päätellä itse. Järkevyysaspektin lisäksi esiin on usein noussut ”etäisyysmittahärvelin” tarkoitus. Mitä hyötyä on laitteesta, joka kartoittaa ympäröivää tilaa?

Vaikka tarkoitus – rehellisesti sanottuna – ei ollut rakentaa mitään hyödyllistä, voi käyttötarkoituksia kyllä löytää. Ensinäkin voi laitetta käyttää tilan mallintamiseen pohjapiirustuksia varten. Tällaisia piirustuksia tarvitsevat esimerkiksi arkkitehdit ja graafikot. Saatua informaatiota voi hyödyntää konenäköhankkeissa. Tilan hahmotusta ja etäisyystietoa saattavat tarvita myös esimerkiksi sokeat ihmiset. Mittauksista voi laskea myös mitatun tilan tilavuuden tai seinäpinta-alan ynnä muuta vastaavaa. Näiden tietojen hyödyntäminen on yksin omasta mielikuvituksesta kiinni.

Pystyykö rakentamani mittalaite kaikkeen tähän? Saako sokea silmät, arkkitehti mittausvälineen ja fyysikko uuden lelun? Ei, ei pysty. Rakentamani laite on pelkkä prototyyppi, joka osaa luoda karkeita malleja ympäristöstään. Laite on rakennettu lähes teoreettisiin ideaalioloihin. Mittausohjelma ei erota milloin on kyse seinästä, milloin tuolista ja milloin ihmisestä. Tuloksia tärkeämpää on itse toimintaperiaate, joka on osoittautunut toimivaksi.

Projektin loppupuolella heräsi kysymys siitä, mitä olisin voinut tehdä toisin. Moni asia olisi ratkennut muullakin tavalla kuin sillä, jota olen käyttänyt. Mittalaitteen materiaali, kamera, moottorit ja laser olisivat kaikki voineet poiketa käytetyistä osista. Mekaniikka on itse asiassa räätälöity juuri käytettäville osille. Olen projektia tehdessä useasti miettinyt, että minkälainen mittalaitteesta olisi tullut, jos olisin käyttänyt aitoa laseretäisyysmittaa. Olisi mielenkiintoista verrata kameran avulla saatuja tuloksia aidon mittalaserin tuloksiin. Ohjelmoinnin puolella olisin mieluusti syventynyt parantelemaan pistetunnistusta, jos aikaa olisi liiennyt.

Näitä viimeisiä rivejä kirjoitellessa pohdin aihevalintani onnistumista: Jos saisin mahdollisuuden palata ajassa taaksepäin, valitsisinko uudelleen tämän aiheen vai vaihtaisinko sen johonkin muuhun? Tulin siihen tulokseen, että valitsisin jonkin muun aiheen. Aiheen vaihtamiseen on selkeä syy: tämä on nyt koettu.

Lähteet

Niiden internetlähteiden kohdalla, joilla ei ole annettu päivämäärää, on päivämäärä se, jona tiedot ovat olleet esillä. Wikipedian artikkeleihin on jokaiseen viitattu erikseen johtuen Wikipedian erikoisluonteesta.

Ahdenkari Jari, Majander Olli, *Piirilevyjen valmistus.* http://www.mbnet.fi/rakentelunurkka/perusteet. Sanoma Magazines Finland, 2005.

Bebek electronic, *Berger hammaspyörällä RDM57-6G*. http://www.bebek.fi. Joulukuu 2004.

Bulback Fred, *io.dll*. Geek Hideout, http://www.geekhideout.com/iodll.shtml, 2003.

Crosser Eugene, *Digital Cameras on Fujitsu chipset*. http://photopc.sourceforge.net, 2001.

Seppänen, Tiihonen, Wuolijoki (matematiikka), *MAOL-taulukot.* Matemaattisten Aineiden Opettajien Liitto ry, Keuruu 2001.

Stewart Zhahai, *IBM Parallel Port FAQ/Tutorial.* http://www.timgoldstein.com/CNC/ParallelPortPrimer.htm, 1994.

Säteilyturvakeskus, *Laserit*. http://www.stuk.fi/sateilytietoa/sateilevat_laitteet/fi_FI/laser, Säteilyturvakeskus, 2005.

Wikipedia, the free encyclopedia, *Angle of view.* http://en.wikipedia.org/wiki/Angle_of_view. Sisältö lisenssillä GNU, Tammikuu 2005.

Wikipedia, the free encyclopedia, *Category: Color space.* http://en.wikipedia.org/wiki/Category:Color_space. Sisältö lisenssillä GNU, Tammikuu 2005.

Wikipedia, the free encyclopedia, *Charge-coupled device.* http://en.wikipedia.org/wiki/CCD. Sisältö lisenssillä GNU, Maaliskuu 2005.

Wikipedia, the free encyclopedia, *JPEG.* http://en.wikipedia.org/wiki/Jpeg. Sisältö lisenssillä GNU, Helmikuu 2005.

Wikipedia, the free encyclopedia, *Stepper motor*. http://en.wikipedia.org/wiki/Stepper_motor. Sisältö lisenssillä GNU, Helmikuu 2005.

Wikipedia, the free encyclopedia, *VRML*. http://en.wikipedia.org/wiki/VRML. Sisältö lisenssillä GNU, Maaliskuu 2005.

LIITTEET

Moduulit

Moottori

```
(*****************************************************************************
* MODUULI:          Moottori 1.1
* KÄÄNTÄJÄ:         Delphi 6
* TARKOITUS:        Mittalaitteen kääntäminen
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       6.1.2004
* KUVAUS:           Ohjaa kahta askelmoottoria rinnakkaisportin kautta
* ALIOHJELMAT:      Liiku
*                   AvaaPortti
*                   Laser
* HUOMIOITAVAA:     Vaatii io.dll-tiedoston (Fred Bulback,
*                   http://www.geekhideout.com/iodll.shtml)
*****************************************************************************)

unit Moottori;

interface

uses SysUtils, Forms;

{ Moottoreiden liikuttaminen }
procedure Liiku(intMoottori1, intMoottori2 : integer);

{ Portin avaus ja testaus }
function AvaaPortti: boolean;

{ Laserin ohjaus }
procedure Laser(paalla : boolean);

var
  intOdota,              { Askelten välissä odoteltava aika (millisekuntteja) }
  intRPortti : integer;  { Rinnakkaisportin osoite }

implementation

{*****************************************************************************
* ALIOHJELMA:       PortOut [io.dll]
* TARKOITUS:        Rinnakkaisportin hallinta (Kernel mode driver)
* NIMI:             Fred Bulback
* PÄIVÄMÄÄRÄ:       2003
* KUVAUS:           Ohjaa rinnakkaisporttia Win 95/98/NT/2000/XP
* PARAMETRIT:       Port       : Portin osoite (yleensä $3BC tai $378)
*                   Data       : Lähetettävä data [0-255]
* HUOMIOITAVAA:     Vaatii io.dll-tiedoston
*                   Funktion toimintaa ei ole dokumentoitu, sillä kyseessä on
*                   Fred Bulbackin koodi. Koodin käyttö on täysin ilmaista ja
*                   koodin ohjelmoija ei ota vastuuta mistään.
*                   http://www.geekhideout.com/iodll.shtml
*****************************************************************************}

procedure PortOut(Port : Word; Data : Byte); stdcall; external 'io.dll';

{*****************************************************************************
* FUNKTIO:          isDriverInstalled [io.dll]
* TARKOITUS:        Rinnakkaisportin ajurin (Kernel mode driver) tarkistus
* NIMI:             Fred Bulback
* PÄIVÄMÄÄRÄ:       2003
* KUVAUS:           Ajurin tarkistus Win 95/98/NT/2000/XP
* PALUUARVO:        Tosi, jos ajuri asennettu ja toimii; Epätosi, jos ei.
* HUOMIOITAVAA:     Vaatii io.dll-tiedoston
```

```
*                   Funktion toimintaa ei ole dokumentoitu, sillä kyseessä on
*                   Fred Bulbackin koodi. Koodin käyttö on täysin ilmaista ja
*                   koodin ohjelmoija ei ota vastuuta mistään.
*                   http://www.geekhideout.com/iodll.shtml
*****************************************************************************}

function IsDriverInstalled : Boolean; stdcall; external 'io.dll';

{*****************************************************************************
* ALIOHJELMA:       Liiku 1.0
* TARKOITUS:        Ohjaa rinnakkaisportin kautta kahta askelmoottoria
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       19.11.2004
* PARAMETRIT:       intMoottori1        : Moottori1:n askeleet
*                   intMoottori2        : Moottori2:n askeleet
*                   intOdota            : Määrä millisekuntteja, jotka odotetaan
*                   intRinnakkaisportti : Käytettävän portin numero
* PALUUARVO:        Ei palauta mitään.
* HUOMIOITAVAA:     Ei sisällä virheenkäsittelyä.
*                   Vaatii 'io.dll'-tiedoston
*****************************************************************************}

procedure Liiku(intMoottori1, intMoottori2 : integer);
var
  byteOut : byte;
begin
  { Kaikkien linjojen loogiset arvot todeksi }
  byteOut := 15; // eli 00001111

  while (intMoottori1 <> 0) or (intMoottori2 <> 0) do
  begin
    // Moottori 1
    { Arvo vaihtelee välillä ......11 ja ......10 }
    if intMoottori1 > 0 then
        if byteOut and 1 = 1 then
            dec(byteOut,1)
        else
        begin
            inc(byteOut,1);
            dec(intMoottori1);
        end;

    { Arvo vaihtelee välillä ......11 ja ......01 }
    if intMoottori1 < 0 then
        if byteOut and 2 = 2 then
            dec(byteOut,2)
        else
        begin
            inc(byteOut,2);
            inc(intMoottori1);
        end;

    // Moottori 2
    { Arvo vaihtelee välillä ....11.. ja ....10.. }
    if intMoottori2 > 0 then
        if byteOut and 4 = 4 then
            dec(byteOut,4)
        else
        begin
            inc(byteOut,4);
            dec(intMoottori2);
        end;

    { Arvo vaihtelee välillä ....11.. ja ....01.. }
    if intMoottori2 < 0 then
        if byteOut and 8 = 8 then
            dec(byteOut,8)
        else
        begin
```

```
            inc(byteOut,8);
            inc(intMoottori2);
        end;

    { Lähetetään dataa portille }
    PortOut(intRPortti,byteOut);

    { Odotetaan }
    sleep(intOdota);

    { Suoritetaan muita komentoja, jottei ohjelma hyydy }
    Application.ProcessMessages;
  end;
end;

{*******************************************************************************
* FUNKTIO:          AvaaPortti 1.0
* TARKOITUS:        Avata rinnakkaisportti toimintakuntoon
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       6.1.2005
* KUVAUS:           Antaa oletusosoitteen, avaa portin ja tarkistaa toiminnan
* PALUUARVO:        Tosi, jos toimii; Epätosi, jos ei.
* HUOMIOITAVAA:     Vaatii 'io.dll'-tiedoston
*******************************************************************************}

function AvaaPortti: boolean;
begin
  { Oletusarvoinen portti }
  if intRPortti = 0 then intRPortti := $3BC;

  { Kaikki ohjauslinjat päälle }
  PortOut(intRPortti+2,0);

  { Toimiiko ajuri? }
  result := IsDriverInstalled;
end;

{*******************************************************************************
* ALIOHJELMA:       Laser 1.0
* TARKOITUS:        Kytkeä laser päälle ja pois
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       29.1.2005
* KUVAUS:           Ohjaa rinnakkaisportin kautta relettä
* PARAMETRIT:       paalla : tosi (päällä) / epätosi (pois päältä)
* HUOMIOITAVAA:     Vaatii 'io.dll'-tiedoston
*******************************************************************************}

procedure Laser(paalla : boolean);
begin
  if paalla then
    { Kyteketään laseria ohjaava rele päälle. }
    { Lähetetään 00011111 }
    PortOut(intRPortti,31)
  else
    { Kyteketään laseria ohjaava rele pois. }
    { Lähetetään 00001111 }
    PortOut(intRPortti,15);
end;

end.
```

Kamera

```
(*****************************************************************************
* MODUULI:          Kamera 1.2
* KÄÄNTÄJÄ:         Delphi 6
* TARKOITUS:        Agfa ePhoto-kameran ohjaus
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       12.1.2005
* KUVAUS:           Ottaa kuvan, lataa sen koneelle ja talleentaa sen.
* ALIOHJELMAT:      lataa
*                   laheta
*                   vastaanota
*                   raportoi
* HUOMIOITAVAA:     Vaatii CommPort-komponentin (Boris Loboda)
*                   Hyödyntää "Serial Protocol of Some Digital Cameras"
*                   -kuvausta (Eugene Crosser, http://photopc.sourceforge.net).
*****************************************************************************)

unit Kamera;

interface

uses
  Windows, SysUtils, Forms, CommPort;

  { Kuvan ottaminen, lataaminen ja tallentaminen }
  procedure lataa(strFilename: string);

  { Tavun lähetys kameralle }
  function laheta(bOut: byte): boolean;

var
  { Sarjaportti }
  CommPort1 : TCommPort;

  { Globaali muuttuja. Tarkoitettu ohjelman keskeyttämiseen; hätäjarru }
  bToimi : boolean;

  { Mittauksen lokitieto }
  strLoki : string;

implementation

{*****************************************************************************
* ALIOHJELMA:       raportoi 1.1
* TARKOITUS:        Latauksen lokitiedon luominen
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       31.12.2004
* KUVAUS:           Lähettää tekstiä näytettäväksi käyttäjälle
* PARAMETRIT:       strTeksti : raportoitava teksti
* HUOMIOITAVAA:     Linkitetty frmMain-formille
*****************************************************************************}

procedure raportoi(strTeksti: string);
begin
  strLoki := strLoki + strTeksti + #13#10;
end;

{*****************************************************************************
* FUNKTIO:          laheta 1.0
* TARKOITUS:        Tavun lähetys sarjaportin kautta
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       28.11.2004
* KUVAUS:           Lähettää tavun sarjaportista ulos
* PARAMETRIT:       bOut : Lähtevä tavu
* PALUUARVO:        Palauttaa arvon tosi, jos onnistui. Muuten epätosi.
* HUOMIOITAVAA:     Vaatii CommPort-componentin. (Boris Loboda)
*****************************************************************************}
```

```
function laheta(bOut: byte): boolean;
begin
  try
    CommPort1.PutString(chr(bOut));
    result := true;
  except
    result := false;
  end;
end;

{*****************************************************************************
* ALIOHJELMA:       vastaanota 1.0
* TARKOITUS:        Tavujen vastaanotto sarjaportin kautta
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       28.11.2004
* KUVAUS:           Lähettää tavun sarjaportista ulos
* PARAMETRIT:       intLength : Saapuvien tavujen määrä
* PALUUARVO:        Palauttaa tavujonon string-tyypin merkkijonona
* HUOMIOITAVAA:     Vaatii CommPort-componentin. (Boris Loboda)
*****************************************************************************}

function vastaanota(intLength: integer): string;
var
  strBuffer : string;  // Puskuri siiretylle datalle
  n         : integer; // Puskurin pituus
begin
  try
    { Odotetaan kunnes haluttu määrä tietoa on siirtynyt }
    while (CommPort1.InBuffUsed < intLength)
     and (bToimi = true) do application.ProcessMessages;

    n := CommPort1.InBuffUsed;
    setLength(strBuffer, n);
    CommPort1.GetBlock(strBuffer[1], n);

    result := strBuffer;
  except
    result := '';
  end;
end;

{*****************************************************************************
* ALIOHJELMA:       Kamera_lataa 1.2
* TARKOITUS:        JPEG-kuvan otto ja lataus Agfa ePhoto 1280 -kamerasta
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       28.11.2004
* KUVAUS:           Ottaa kuvan ja lataa yksitellen kameran muistista kuvadataa
* PARAMETRIT:       strFilename : Tallennettavan jpeg-tiedoston nimi
* HUOMIOITAVAA:     Hyödyntää "Serial Protocol of Some Digital Cameras"
*                   -kuvausta (Eugene Crosser, http://photopc.sourceforge.net).
*                   Koodi ei sisällä virheenkäsittelyä.
*                   Vaatii: laheta-funktion
*                           vastaanota-funktion
*****************************************************************************}

procedure lataa(strFilename: string);
var
  strBuffer,                    // Puskuri siiretylle datalle
  strOutput         : string;   // Tallennettava tieto

  n,i,                          // Väliaikaismuuttujia
  intPacketLength,              // Paketin pituus
  intChecksumA,                 // Laskettu tarkistussumma
  intChecksumB      : integer;  // Kameran tarkistussumma

  bViimeinenPaketti : boolean;  // Paketin tyyppi

  F                 : TextFile; // Tiedostomuuttuja
begin
```

```
{ Tyhjennetään loki ja varmistetaan toimiminen }
strLoki := '';
bToimi := true;

raportoi('*** KUVAN SIIRTO AGFA ePHOTO -KAMERASTA ***');

{ Aloitetaan kommunikointi }
raportoi('[Kommunikointi aloitettu]');

{ Tyhjennetään komponentin puskuri }
CommPort1.FlushInBuffer;

{ Aloitetaan }
laheta($00);        //NULL    (Intialization byte)

{ Odotetaan kameran vastausta }
vastaanota(1);      //NAK                          (Camera signature, 0x15) ol.

{ Aseta nopeus }
laheta($1B); //type           Komentopaketti
laheta($53); //subtype        Ensimmäinen komentopaketti
laheta($03); //length         Dataosuuden pituus
laheta($00); //length
laheta($00); //data           Tallenna luku integerrekisteriin
laheta($11); //data           Rekisterin numero (Nopeus)
laheta($02); //data           Uusi arvo (2 eli 19 200)
laheta($13); //cs             Tarkistussumma
laheta($00); //cs

{ Odotetaan kameran vastausta }
vastaanota(1);      //ACK     (Positive Acknowledgement, 0x06) ol.

raportoi('[Lataus aloitettu]');

{ Otetaan Preview-kuva }
laheta($1B); //type           Komentopaketti
laheta($43); //subtype        Muu kuin ensimmäinen komentopaketti
laheta($03); //length         Dataosuuden pituus
laheta($00); //length
laheta($02); //data           Suorita toiminto
laheta($05); //data           Toiminnon numero (Ota esikatselukuva, 5)
laheta($00); //data
laheta($07); //cs             Tarkistussumma
laheta($00); //cs

{ Odotetaan kameran vastausta }
vastaanota(2);      //ACK     (Positive Acknowledgement, 0x06) ol.
                    //ENQ     (Action Compelte Notification, 0x05) ol.

{ Kerrotaan halutun kuvan numero }
laheta($1B); //type           Komentopaketti
laheta($43); //subtype        Muu kuin ensimmäinen komentopaketti
laheta($06); //length         Dataosuuden pituus
laheta($00); //length
laheta($00); //data           Tallenna luku integerrekisteriin
laheta($04); //data           Rekisterin numero (Aktivoidun kuvan numero)
laheta($00); //data           Uusi arvo (0 eli äsken otettu esikatselukuva)
laheta($00); //data
laheta($00); //data
laheta($00); //data
laheta($04); //cs             Tarkistussumma
laheta($00); //cs

{ Odotetaan kameran vastausta }
vastaanota(1);      //ACK     (Positive Acknowledgement, 0x06) ol.

{ Tilataan halutun kuvan data }
laheta($1B); //type           Komentopaketti
laheta($43); //subtype        Muu kuin ensimmäinen komentopaketti
```

```
  laheta($03); //length          Dataosuuden pituus
  laheta($00); //length
  laheta($04); //data            Lue dataa vdatarekisteristä
  laheta($0E); //data            Rekisterin numero (Valitun kuvan data)
  laheta($00); //data (opt)
  laheta($12); //cs              Tarkistussumma
  laheta($00); //cs

  { Kamera palauttaa vaihtelevan määrän paketteja, joissa varsinainen
    kuva on. Pakettien rakenne on yhtenäinen muiden pakettien kanssa. }

{ Haetaan paketit }
  repeat
  begin
    { Nollataan nollausta kaipaavat muuttujat }
    intChecksumA := 0;

    { Haetaan paketin alkuosa }
    strBuffer := vastaanota(10);
    n := length(strBuffer);

    { Tarkistetaan onko kyseessä viimeinen paketti }
    if ord(strBuffer[1]) = 3 then bViimeinenPaketti := true
      else bViimeinenPaketti := false;

    { Luetaan paketin pituus }
    intPacketLength := ord(strBuffer[3]) + ord(strBuffer[4]) * 256;

    { Käydään läpi alun dataosuus }
    for i := 5 to n do
    begin
      strOutput := strOutput + strBuffer[i];
      inc(intChecksumA,ord(strBuffer[i]));
    end;

    { Haetaan paketin jäljellä oleva osa }
    strBuffer := vastaanota(4 + intPacketLength - n);
    n := length(strBuffer);

    { Käydään loppupaketti läpi }
    for i := 1 to n - 2 do
    begin
      strOutput := strOutput + strBuffer[i];
      inc(intChecksumA,ord(strBuffer[i]));
    end;

    { Tarkastetaan tarkistussumma (Checksum) }
    intChecksumB := ord(strBuffer[n-1]) + ord(strBuffer[n])*256;

    { intChecksumA:sta 16-bittinen versio eli jakojäännös }
    intChecksumA := intChecksumA mod 65536;

    { Tarkistetaan siirron onnistuminen }
    if intChecksumA = intChecksumB then
    begin
      { Lähetetään ACK }
      laheta($06);    //ACK     (Positive Acknowledgement, 0x06)
    end
    else { Tarkistussumma ei täsmää --> uudelleen }
    begin
      { Ilmoita virheestä }
      raportoi('Tiedonsiirrosa tapahtui tarkistusummavirhe.');

      { Poistetaan virheellinen paketti }
      SetLength(strOutput, Length(strOutput)-intPacketLength);

      { Lähetetään NAK }
      laheta($15);    //NAK     (Negative Acknowledgement, 0x15)
```

```
      { Uusitaan haku }
      bViimeinenPaketti := false;
    end;
    { Jos kyseessä ei ole viimeinen paketti tai tarkistussumma ei
      täsmää, palataan silmukan alkuun. }
  end until (bViimeinenPaketti = true);

{ Kuvan tallennus JPEG-tiedostoksi }

  { Avataan, luodaan, tiedosto kirjoitettavaksi }
  AssignFile(F, strFilename);
  ReWrite(F);

  { Kirjoitetaan tiedostoon }
  Write(F, strOutput);

  { Suljetaan tiedosto }
  CloseFile(F);

  { Kerrotaan, että tallennettu }
  raportoi('Kuva tallennettu: ' + strFilename);

  { Lopetetaan komunikointi kameran kanssa }
  laheta($FF);        //Lopetus (Termination Byte, 0xFF)

  { Kamera vastaa vielä, mutta siitä ei tarvitse välittää. }
end;

{*****************************************************************************
*
* INITIALIZATION:    Sarjaportti luodaan samalla, kun moduuli avataan käyttöön.
*
*****************************************************************************}

initialization
  { Luodaan sarjaportti, joka tuhoutuu sovelluksen mukana }
  CommPort1 := TCommPort.Create(Application);

  { Asetetaan oletusasetukset }
  with CommPort1 do
  begin
    ComNumber := 1;
    Baud      := 19200;
    Databits  := 8;
    Parity    := paNone;
    Stopbits  := sb1_0;
    InSize    := 2048;
    OutSize   := 2048;
    Open      := true;
  end;
end.
```

Etsi

```
(*****************************************************************************
* MODUULI:          Etsi 1.2
* KÄÄNTÄJÄ:         Delphi 6
* TARKOITUS:        Agfa ePhoto-kameran ohjaus
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       14.11.2004
* KUVAUS:           Etsii pisteen kuvasta
* ALIOHJELMAT:      EtsiPiste
*                   AvaaJPEG
* HUOMIOITAVAA:     Vaatii JPEG-lisäosan (katso uses).
*****************************************************************************)

unit Etsi;

interface

uses
  Windows, SysUtils, Graphics, Forms, Jpeg;

{ Pisteen etsiminen }
function EtsiPiste(strFilename: string): integer;

{ Kuvan näyttäminen }
function AvaaJPEG(strFilename: string; kohta: integer = -1): TBitmap;

implementation

(*****************************************************************************
* FUNKTIO:          RGB_to_HSV 1.0
* TARKOITUS:        Laskee RGB-arvosta HSV-järjestelmän kautta värin punaisuuden
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       14.11.2004
* KUVAUS:           Palauttaa annetun värin punaisuusarvon
* PARAMETRIT:       R  :  punainen [0..255]
*                   G  :  vihreä [0..255]
*                   B  :  sininen  [0..255]
* PALUUARVO:        Palauttaa punaisuusarvon väliltä [0..100]
* HUOMIOITAVAA:     Ei sisällä virheenkäsittelyä.
*                   Alisteinen EtsiPiste-funktiolle
*****************************************************************************)

function RGB_to_HSV(R,G,B: byte): byte;
var
  v0, v1, v2      : single;
  max, min, delta : single;
  H, S, V         : single;
  raja            : single;
begin

  v0 := r / 255; // Punainen
  v1 := g / 255; // Vihreä
  v2 := b / 255; // Sininen

  { max-arvo }
  max := v0;
  if v1 > max then max := v1;
  if v2 > max then max := v2;

  { min-arvo }
  min := v0;
  if v1 < min then min := v1;
  if v2 < min then min := v2;

  V := max;

  if (max = 0) or (max = min) then
  begin
```

```
    H := 0;
    S := 0;
  end
  else
  begin
    S := (max - min) / max;
    delta := max - min;
    if v0 = max then H := (v1 - v2) / delta
    else if v1 = max then H := 2 + (v2 - v0) / delta
    else H := 4 + (v0 - v1) / delta;
  end;

  { Muutetaan lukuväliä; H [0..360] }
  H := H * 60;
  if H < 0 then H := H + 360;

  S := S*100; // S [0..100]
  V := V*100; // V [0..100]

  { Määritellään HSV-arvosta punaisuusarvo }

  if (H > 330) or (H < 30) then
  begin
    if H > 330 then H := 360 - H; // Normalisoidaan etäisyys
    H := H / 30 * 250; // Painoarvo (100 täysi)

    raja := sqrt(sqr(H) + sqr(100 - S) + sqr(100-V));
    raja := 100 - raja;
    if raja < 0 then raja := 0;
  end
  else
    raja := 0;

  result := round(raja); // Palautetaan arvo
end;

(*****************************************************************************
* FUNKTIO:          Lightness 1.0
* TARKOITUS:        Laskee vastaavien RGB-arvojen L-arvon (HSL-järjestelmä)
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       14.11.2004
* KUVAUS:           Palauttaa annetun värin valoarvon.
* PARAMETRIT:       R  :  punainen [0..255]
*                   G  :  vihreä [0..255]
*                   B  :  sininen  [0..255]
* PALUUARVO:        Palauttaa Lightness-arvon väliltä [0..100]
* HUOMIOITAVAA:     Ei sisällä virheenkäsittelyä.
*                   Alisteinen EtsiPiste-funktiolle
*                   Sisältää keskiarvolaskennan
*****************************************************************************)

function Lightness(R, G, B: Longint): byte;
var
  max, min: integer;
begin
  { Lasketaan keskiarvot väreille }
  R := R div (160*120);
  G := G div (160*120);
  B := B div (160*120);

  { Haetaan suurin arvo }
  max := R;
  if G > max then max := G;
  if B > max then max := B;

  { Haetaan pienin arvo }
  min := R;
  if G < min then min := G;
  if B < min then min := B;
```

```
  { Lasketaan L ja muutetaan prosenteiksi (0..100) }
  Lightness := (max + min)*10 div 51;
end;

(*****************************************************************************
* FUNKTIO:          EtsiPiste 1.1
* TARKOITUS:        Etsiä jpeg-kuvasta punainen laserpiste
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       16.11.2004
* KUVAUS:           Etsii kuvasta punaisen pisteen
* PARAMETRIT:       strFilename  :  tiedosto (jpeg)
* PALUUARVO:        Palauttaa pisteen X-koordinaatin:
*                     -Jos hyvä tulos [0..159]
*                     -Jos tulos epävarma [0..-159]
*                     -Jos virhe [ei mitään]
* HUOMIOITAVAA:     Vaatii: jpeg-kirjaston
*                           RGB_to_HSV-funktion
*                           Lightness-funktion
*                   Rakenne osittain ontuu
*****************************************************************************)

function EtsiPiste(strFilename: string): integer;
var
  { Väliaikaiset kuvat }
  jpeg  : TJpegImage;
  bmp   : TBitmap;

  { Kuvahaun perusmuuttujat }
  X,Y   : integer;          // skannattavat koordinaatit
  r,g,b : byte;             // puna-, viher- ja siniset arvot
  P     : PByteArray;       // ScanLine:n palauttama bittijoukko

  X2, Y2,                   // väliaikainen X ja Y
  intAlue, intPaino,        // hakualueen punapisteiden lukumäärä ja painoarvo
  intTmp                   // apumuuttuja
             : integer;     // löydetyn pisteen koordinaatit
  X0, Y0, P0 : array of integer;  // löydetyt pisteet; X, Y, painoarvo

  kaR, kaG, kaB : Longint; // yhteenlasketut väriarvot
  i             : integer; // silmukkamuuttuja
const
  intHakualue   = 6;        // Pisteiden läheisyys
label Jatka;
begin
  { Alustetaan muuttujat. }
  SetLength(X0,1);
  SetLength(Y0,1);
  SetLength(P0,1);
  intTmp := 0;
  kaR := 0; kaG := 0; kaB := 0;

  { Luodaan kuvapuskurit }
  jpeg:=TJpegImage.Create;
  bmp:=TBitmap.Create;

  try
    try
      { Avataan jpeg-kuva }
      jpeg.LoadFromFile(strFilename);
      bmp.Assign(jpeg);

      for Y := 0 to bmp.Height-1 do
      begin
        { Luetaan vaakarivillinen pikseleitä }
        P := bmp.ScanLine[Y];

        for X := 0 to bmp.Width-1 do
        begin
```

```
        { Luetaan pikselin väri (RGB) }
        r := P[X*3+2];
        g := P[X*3+1];
        b := P[X*3];

        { Keskiarvojen laskemista varten }
        inc(kaR,r); Inc(kaG,g); Inc(kaB,b);

        if RGB_to_HSV(r, g, b) > 3 then
        begin
          { Tarkistetaan onko alueella jo piste }
          for i := Length(X0)-1 downto 0 do
            if (X > X0[i]-intHakualue) and (X < X0[i]+intHakualue) then
              if (Y > Y0[i]-intHakualue) and (Y < Y0[i]+intHakualue) then
                GoTo jatka; // Hypätään seuraavaan pikseliin

          { Varsinaisen pisteen tunnistus }
          intPaino := 0; intAlue := 0;
          try
            for Y2 := Y - 1 to Y + 1 do
            begin
              P := bmp.ScanLine[Y2];
              for X2 := X - 1 to X + 1 do
              begin
                { Luetaan pikselin väri (RGB) }
                r := P[X2*3+2];
                g := P[X2*3+1];
                b := P[X2*3];
                intTmp := RGB_to_HSV(r, g, b);
                inc(intPaino,intTmp);
                if intTmp > 3 then inc(intAlue);
              end;
            end;

            { Jos alanurkan pikseli on kuvan keskiarvoa punaisempi,
              oletetaan punaisuuden jatkuvan ja korostetaan pistettä. }
            if intTmp > (intpaino*100) div 900 then
            begin
              intPaino := intPaino*2; intAlue := intAlue*2;
            end;
          except
            { Virhe, jos luetaan vahingossa kuvan ulkopuolelta. }
          end;

          if (intPaino >= 50) or (intAlue >= 6) then
          begin
            { Tallenetaan piste, jos arvot tarpeeksi merkittäviä }
            intTmp := Length(X0);
            SetLength(X0,intTmp+1);
            SetLength(Y0,intTmp+1);
            SetLength(P0,intTmp+1);

            { Piste tallennetaan alueen keskelle (+1) }
            X0[intTmp] := X+1;
            Y0[intTmp] := Y+1;
            P0[intTmp] := intPaino;
          end;
        end;
        { Jatketaan seuraavaan kohtaan GoTo-käskyn kautta. }
        Jatka:
      end;
    end;
  except
    Application.MessageBox('Ohjelman suorituksessa ilmeni virhe. '+
    'Onko tiedosto olemassa?', 'Virhe suorituksessa', MB_OK);
    { Ei suoriteta loppuun. }
    Abort;
  end;
```

```
    { Etsitään piste, jolla suurin painoarvo. }
    intTmp := 0;
    for i := 1 to length(X0)-1 do
      if P0[i] > P0[intTmp] then intTmp := i;

    { Lightness-arvon perusteella palautetaan tulos }
    if Lightness(kaR,kaG,kaB) < 30 then
      result := X0[intTmp]
    else
      result := -X0[intTmp];

    { Muita hyödynnettäviä arvoja ovat:
      Length(X0)-1 : Löydettyjen pisteiden määrä.
      kaR,kaG,kaB  : Kuvan värien keskiarvot. }
  finally
    { Vapautetaan kuvat }
    bmp.Free;
    jpeg.Free;
  end;
end;

{*******************************************************************************
* FUNKTIO:          AvaaJPEG 1.0
* TARKOITUS:        JPEG-kuvan muuttaminen bittikartaksi
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       24.2.2005
* KUVAUS:           Avaa JPEG-kuvan ja merkisee siihen pisteen näkyviin
* PARAMETRIT:       strFilename  :  tiedosto (jpeg)
*                   kohta        :  kohta kuvassa
* PALUUARVO:        Palauttaa bittikartan
*******************************************************************************}

function AvaaJPEG(strFilename: string; kohta: integer = -1): TBitmap;
var
  { Väliaikainen kuvat }
  jpeg  : TJpegImage;
begin
  { Luodaan kuvapuskurit }
  result := TBitmap.Create;

  { Jos kuvaa ei ole, poistutaan }
  if not FileExists(strFilename) then Exit;

  try
    jpeg:=TJpegImage.Create;
    try
      { Avataan jpeg-kuva }
      jpeg.LoadFromFile(strFilename);
      result.Assign(jpeg);

      { Merkitään pisteen kohta }
      with result.Canvas do
      begin
        Pen.Color := clWhite;
        MoveTo(kohta, 0);
        LineTo(kohta, 119);
      end;
    finally
      jpeg.Free;
    end;
  except
    result.Free;
  end;
end;

end.
```

Laske

```
(*************************************************************************
* MODUULI:          Laske 1.0
* KÄÄNTÄJÄ:         Delphi 6
* TARKOITUS:        Etäisyyden laskemista
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       24.2.2005
* KUVAUS:           Laskee etäisyyttä ja kalibroi
* ALIOHJELMAT:      rad2deg
*                   deg2rad
*                   etaisyys
*                   kalibroi
* HUOMIOITAVAA:     Muuttaa moduulin sisällä olevia muuttujia.
*************************************************************************)

unit Laske;

interface

{ Radiaanit asteiksi }
function rad2deg(radians: single): single;

{ Asteet radiaaneiksi }
function deg2rad(degrees: single): single;

{ Etäisyyden laskeminen }
function etaisyys(x: single): single;

{ Vakioiden kalibroiminen }
procedure kalibroi(x,y: array of single);

var
  alfa, beeta, vakio_a: single;

implementation

{*************************************************************************
* ALIOHJELMA:       rad2deg 1.0
* TARKOITUS:        Muuttaa radiaanit asteiksi
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       29.12.2004
* PARAMETRIT:       radians :  kulma radiaaneina
* PALUUARVO:        Kulma asteina
*************************************************************************}

function rad2deg(radians: single): single;
begin
  { Radiaanit (0..2*Pii) muutetaan asteiksi (0..360).
    Perustuu verrannollisuuteen, missä 45 astetta = arctan 1 }
  result := radians / ArcTan(1) * 45;
end;

{*************************************************************************
* ALIOHJELMA:       deg2rad 1.0
* TARKOITUS:        Muuttaa asteet radiaaneiksi
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       29.12.2004
* PARAMETRIT:       degrees :  kulma asteina
* PALUUARVO:        Kulma radiaaneina
*************************************************************************}

function deg2rad(degrees: single): single;
begin
  { Asteet (0..360) muutetaan radianeiksi (0..2*Pii).
    Perustuu verrannollisuuteen, missä 45 astetta = arctan 1 }
  result := degrees / 45 *  ArcTan(1);
end;
```

```
{*****************************************************************************
* ALIOHJELMA:        laske_x 1.0
* TARKOITUS:         Laskea poikkeama linssin keskikohdan kohtisuorasta.
* NIMI:              Arno Solin
* PÄIVÄMÄÄRÄ:        29.12.2004
* KUVAUS:            Laskee regressiosuoran laskemisessa tarvittavan x-arvon
* PARAMETRIT:        x :  suhteellinen kohta kuvassa
*                    y : (käsin) mitattu etäisyys
* PALUUARVO:         Poikkeama linssin keskikohdan kohtisuorasta
* HUOMIOITAVAA:      Arvon palautus tehdään muokkaamalla parametria x
*                    Tarvitaan globaalia muuttujaa alfa.
*****************************************************************************}

procedure laske_x(var x,y: single);
var
  Tmp : single; // Väliaikaismuuttuja
begin
  { Kaavasta: x1 = y1 * tan(2*alfa*x - alfa) }

  Tmp := 2*alfa*x - alfa;

  { Tangenttifunktion käyttö vaatii erillistä kirjastoa, joten oikastaan hieman
    tan(x) = sin(x) / cos(x) }

  x := y * sin(Tmp)/cos(Tmp);
end;

{*****************************************************************************
* FUNKTIO:           regressiosuora_b 1.0
* TARKOITUS:         Suoran kulmakertoimen laskeminen
* NIMI:              Arno Solin
* PÄIVÄMÄÄRÄ:        29.12.2004
* KUVAUS:            Sovittaa suoran pistejoukkoon
* PARAMETRIT:        x,y : taulukoidut x- ja y-arvot
* PALUUARVO:         Palauttaa beetakulman kulmakertoimen (rad)
* HUOMIOITAVAA:      Single-muuttujat pyöristävät hieman tuloksia.
*****************************************************************************}

function regressiosuora_b(x,y: array of single): single;
var
  i                : integer; // Silmukassa käytettävä muuttuja
  Tmp1, Tmp2, Tmp3 : single;  // Väliaikaismuuttuja
begin
  { y = bx + a
    Regressiosuoran kulmakertoimen laskeminen
    b = (n * sum(xi*yi)-(sum(xi)*sum(yi)))/(n*sum(xi^2) - sum(xi)^2) }

  { Muuttujien nollaus }
  Tmp1 := 0; Tmp2 := 0; Tmp3 := 0;

  { Summien laskeminen }
  for i := 0 to length(x)-1 do
  begin
    Tmp1 := Tmp1 + x[i]*y[i];
    Tmp2 := Tmp2 + x[i];
    Tmp3 := Tmp3 + y[i];
  end;

  Tmp1 := length(x)*Tmp1 - Tmp2*Tmp3;

  { Muuttujan nollaus }
  Tmp3 := 0;

  { Summan laskeminen }
  for i := 0 to length(x)-1 do
    Tmp3 := Tmp3 + sqr(x[i]);

  { Paluuarvon asettaminen }
  result := Tmp1 / (length(x)*Tmp3 - sqr(Tmp2));
```

```
end;

{*****************************************************************************
* FUNKTIO:          regressiosuora_a 1.0
* TARKOITUS:        Suoran vakiotermin laskeminen
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       29.12.2004
* KUVAUS:           Sovittaa suoran pistejoukkoon
* PARAMETRIT:       x,y : taulukoidut x- ja y-arvot
*                   b   : kulmakerroin (regressiosuora_b)
* PALUUARVO:        Palauttaa suoran ja y-akselin leikkauskohdan
* HUOMIOITAVAA:     Single-muuttujat pyöristävät hieman tuloksia.
*                   Kulmakerroin laskettava ensin.
*****************************************************************************}

function regressiosuora_a(x,y: array of single; b: single): single;
var
  i                : integer; // Silmukassa käytettävä muuttuja
  Tmp1, Tmp2       : single;  // Väliaikaismuuttuja
begin
  { y = bx + a
    Regressiosuoran vakiotermin laskeminen
    a = (sum(yi) - b*sum(xi)) / n }

  { Muuttujien nollaus }
  Tmp1 := 0; Tmp2 := 0;

  { Summien laskeminen }
  for i := 0 to length(x)-1 do
  begin
    Tmp1 := Tmp1 + y[i];
    Tmp2 := Tmp2 + x[i];
  end;

  { Paluuarvon asettaminen }
  result := (Tmp1 - b*Tmp2)/length(x);
end;

{*****************************************************************************
* FUNKTIO:          etaisyys 1.0
* TARKOITUS:        Etäisyyden laskeminen
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       29.12.2004
* KUVAUS:           Laskee etäisyyden laserin osumakohtaan
* PARAMETRIT:       x       : kohta kuvassa (0..1)
* PALUUARVO:        Palauttaa pisteen etäisyyden (yks. sama kuin vakio_a:lla)
* HUOMIOITAVAA:     Single-muuttujat pyöristävät hieman tuloksia.
*                   Globaalit muuttujat:
*                   alfa    : (aukeamiskulman puolikas)
*                   beeta   : (laserin kulma)
*                   vakio_a : (laserin ja linssin välinen etäisyys)
*****************************************************************************}

function etaisyys(x: single): single;
begin
  { d(x) = a*sin(beeta)*cos(alfa-2*alfa*x)/cos(2*alfa*x-beeta-alfa) }
  result := vakio_a*sin(beeta)*cos(alfa-2*alfa*x);
  result := result / cos(2*alfa*x-beeta-alfa);
end;

{*****************************************************************************
* ALIOHJELMA:       kalibroi 1.0
* TARKOITUS:        Sovittaa beeta- ja vakio_a-arvot sopiviksi
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       30.12.2004
* KUVAUS:           Kalibroi beeta- ja vakio_a-muuttujan
* PARAMETRIT:       x :  suhteellinen kohta kuvassa (taulukko)
*                   y : etäisyys (taulukko)
* HUOMIOITAVAA:     Käyttää globaaleja muuttujia väliarvoihin.
```

```
*                    Globaalit muuttujat:
*                    alfa    : (aukeamiskulman puolikas)
*                    beeta   : (laserin kulma)
*                    vakio_a : (laserin ja linssin välinen etäisyys)
*                    Vaatii: laske_x
*                            regressiosuora_b
*                            regressiosuora_a
*******************************************************************************}

procedure kalibroi(x,y: array of single);
var
  i : integer; // Silmukassa käytettävä muuttuja
begin
  { Lasketaan kaikki x-arvot }
  for i := 0 to length(x)-1 do laske_x(x[i], y[i]);

  { Sovitetaan suora pistejoukkoon (x, y)
    saadaan kulmakerroin beeta }
  beeta := regressiosuora_b(x,y);

  { Etsitään suoran vakiotermi a }
  vakio_a := regressiosuora_a(x,y,beeta);

  { Lasketaan vakio_a suoran vakiotermistä ja muutetaan etumerkki }
  vakio_a := -vakio_a / beeta;

  { Lasketaan kulma kulmakertoimesta ja muutetaan samalla etumerkki }
  beeta := -ArcTan(beeta);
end;


end.
```

Tiedosto

```
(*****************************************************************************
* MODUULI:          Tiedosto 1.0
* KÄÄNTÄJÄ:         Delphi 6
* TARKOITUS:        Luoda kolmiulotteisia malleja tuloksista
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       9.2.2005
* KUVAUS:           Avaa ja muuntaa tulokset kolmiulotteiseksi malliksi
* ALIOHJELMAT:      AvaaMittaus : Mittaustiedoston avaaminen
*                   LaskeXYZ    : Koordinaattien laskeminen
*****************************************************************************)

unit Tiedosto;

interface

uses
  SysUtils, Type3D;

var
  Pysty : integer; { Mahdollisuus lukea avatun tiedoston pystypisteiden määrä }

{ Mittaustiedoston avaaminen }
procedure AvaaMittaus(strFilename : string; var Piste    : TKoordinaattiArray;
                                            var Polygoni : TPolygoniArray);
{ Koordinaattipisteiden laskeminen }
function LaskeXYZ(H,V: integer; d: single): TKoordinaatti;

{ Mittaustiedon tallentaminen }
procedure Tallenna(strFilename: string; etaisyys: array of single;
                   Pysty, Vaaka, Sarjap, Rinnakkaisp : integer;
                   strNimi : string; alfa, beeta, vakio_a: single);

implementation

{*****************************************************************************
* FUNKTIO:          LaskeXYZ 1.1
* TARKOITUS:        Laskea koordinaattipiste
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       6.3.2005
* KUVAUS:           Laskee x-, y- ja z-arvon kahdesta kulmasta ja etäisyydestä
* PARAMETRIT:       H : Kulma (askeleina, vaaka)
*                   V : Kulma (askeleina, pysty)
*                   d : Etäisyys
* PALUUARVO:        Palauttaa arvon TKoordinaatti(x,y,z)
* HUOMIOITAVAA:     Palauttaa arvon muodossa TKoordinaatti (katso Type3D)
*****************************************************************************}

function LaskeXYZ(H,V: integer; d: single): TKoordinaatti;
  { Askeleet radiaaneiksi }
  function StepsToRad(intSteps: integer): single;
  begin
    result := intSteps * 1.4 / 360 * 2 * 3.141592654;
  end;
begin
  { X = cos H * cos V * d}
  result.X := cos(-StepsToRad(H)) * cos(StepsToRad(V)) * d;

  { Y = sin H * cos V * d}
  result.Y := sin(-StepsToRad(H)) * cos(StepsToRad(V)) * d;

  { Z = sin V * d}
  result.Z := sin(StepsToRad(V)) * d;
end;

{*****************************************************************************
* ALIOHJELMA:       AvaaMittaus 1.0
* TARKOITUS:        Muuntaa arvot yleiseen muotoon jatkokäsittelyä varten
```

```
* NIMI:              Arno Solin
* PÄIVÄMÄÄRÄ:        9.2.2005
* KUVAUS:            Avaa mittaustiedoston ja muuntaa sen 3D-malliksi
* PARAMETRIT:        strFilename : Tiedoston nimi
*                    Piste       : Koordinaatit
*                    Polygoni    : Polygonit
* HUOMIOITAVAA:      Palauttaa muokatut Piste- ja Polygoni-taulukot
*****************************************************************************}

procedure AvaaMittaus(strFilename : string; var Piste    : TKoordinaattiArray;
                                            var Polygoni : TPolygoniArray);
var
  F                  : TextFile;                                  { Tiedosto }
  Lue                : boolean;                                   { Luetaanko? }
  i, j, n, kohta,                          { Silmukoiden ja kohtien muuttujat }
  Vaaka              : integer;                                   { 3D-mallin mitat }
  strTmp, strLuku    : string;                          { Väliaikaismerkkijonot }
  mitat              : array of array [0..1] of integer; { Väliaikaiset mitat }
  etaisyys           : array of single;                           { Etäisyydet }
begin
  { Onko tiedostoa olemassa? }
  if not FileExists(strFilename) then Abort;

  { Ei lueta pisteitä vielä }
  Lue := false;
  kohta := 0;
  Pysty := 0;
  Vaaka := 0;

  { Avataan tiedosto }
  AssignFile(F, strFilename);
  Reset(F);

  while not Eof(F) do
  begin
    { Luetaan rivi tiedostosta }
    ReadLn(F, strTmp);
    strTmp := Trim(strTmp);

    { Onko tyhjä rivi tai kommentti }
    if strTmp = '' then continue;
    if strTmp[1] = '#' then continue;

    { Haetaan tiedostosta pysty- ja vaakapisteiden määrät }
    if kohta = 6 then
      Pysty := StrToInt(strTmp);
    if kohta = 7 then
      Vaaka := StrToInt(strTmp);
    inc(kohta);

    { Lue piste, jos ollaan pisteiden kohdalla }
    if Lue then
    begin
      { Asetetaan koko }
      n := length(mitat);
      SetLength(mitat, n+1);
      SetLength(etaisyys, n+1);

      { Luvut muotoa "000_00.jpg" }
      strLuku := strTmp[1] + strTmp[2] + strTmp[3];
      mitat[n, 0] := StrToInt(strLuku);
      strLuku := strTmp[5] + strTmp[6];
      mitat[n, 1] := StrToInt(strLuku);

      { Luetaan ja tallennetaan etäisyys }
      ReadLn(F, strTmp);
      strTmp := Trim(strTmp);
      Etaisyys[n]:= StrToFloat(strTmp);
    end;
```

```
    { Onko jo ohitettu pisteiden alkukohta }
    if strTmp = 'Mittapisteet:' then Lue := true;
  end;

  { Suljetaan tiedosto }
  CloseFile(F);

  { 3D-PISTEET:
    Nollataan tulokset ja määritellään uusi koko }
  Piste := nil;
  Polygoni := nil;
  SetLength(Piste, Length(mitat));

  { Lasketaan etäisyyden ja kulmien avulla x-, y- ja z-koordinaatit }
  for i := 0 to Length(Piste)-1 do
    Piste[i] := LaskeXYZ(Mitat[i,0], Mitat[i,1], Etaisyys[i]);

  { POLYGONIT: }

  { Esimerkki: tuloste, kun pysty = 5 ja vaaka = 3
  0,1,6,5  |  5,6,11,10  |  10,11,1,0  |  Pohja:
  1,2,7,6  |  6,7,12,11  |  11,12,2,1  |  0,5,10,10
  2,3,8,7  |  7,8,13,12  |  12,13,3,2  |  Katto:
  3,4,9,8  |  8,9,14,13  |  13,14,4,3  |  4,9,14,14 }

  { Polygonien määrä }
  n := (Vaaka-1) div 2;           { Pohja }
  n := 2*n + (Pysty - 1) * Vaaka; { Seinät }
  SetLength(Polygoni, n);

  { n on polygonin numero }
  n := 0;

  { Luodaan polygonit seiniin }
  for i := 0 to Vaaka-2 do
    for j := 0 to Pysty-2 do
      with Polygoni[n] do
      begin
        a := Pysty * i + j;
        b := Pysty * i + j + 1;
        c := Pysty * (i+1) + j + 1;
        d := Pysty * (i+1) + j;

        { Seuraava polygoni }
        inc(n);
      end;

  { Yhdistävä sivu }
  i := Vaaka-1;
    for j := 0 to Pysty-2 do
      with Polygoni[n] do
      begin
        a := Pysty * i + j;
        b := Pysty * i + j + 1;
        c := j + 1;
        d := j;

        { Seuraava polygoni }
        inc(n);
      end;

  { Luodaan lattian ja katon polygonit. Polygonit luodaan nelikulmaisina aina,
    kun mahdollista. Kaikkien polygonien yhtenä pisteenä on linjan nolla piste.
    Jos kyseessä on viimeinen polygoni, ja vaakapisteitä on pariton määrä,
    luodaan kolmikulmainen polygoni. Katon ja lattian polygonit ovat toistensa
    peilikuvia. }

  for i := 0 to ((Vaaka-1) div 2)-1 do
```

```
    with Polygoni[n] do
    begin
      a := 0;
      b := i*2*Pysty + Pysty;
      c := i*2*Pysty + 2*Pysty;
      d := i*2*Pysty + 3*Pysty;
      { Viimeisen polygonin muodon ja tarpeen tarkistava ehtolause: }
      if i = ((Vaaka-1) div 2)-1 then
        d := d - Pysty*(Vaaka mod 2);

      { Seuraava polygoni }
      inc(n);

      { Vastaavat katon polygonit }
      Polygoni[n].a := a + Pysty - 1;
      Polygoni[n].b := b + Pysty - 1;
      Polygoni[n].c := c + Pysty - 1;
      Polygoni[n].d := d + Pysty - 1;

      { Seuraava polygoni }
      inc(n);
    end;
end;

{*****************************************************************************
* ALIOHJELMA:       Tallenna 1.1
* TARKOITUS:        Tallentaa mittaustiedoston (.txt)
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       27.2.2005
* KUVAUS:           Tallentaa mittausprojektin tekstitiedostoon
* PARAMETRIT:       strFilename : Tiedoston nimi
*                   etaisyys    : Mitatut etäisyydet (talukko)
*                   Pysty       : Pystypiseteiden määrä
*                   Vaaka       : Vaakapiseteiden määrä
*                   Sarjap      : Sarjaportti
*                   Rinnakkaisp : Rinnakkaisportti
*                   strNimi     : Projektin nimi
*                   alfa        : Aukeamiskulman puolikas
*                   beeta       : Laserin suuntakulma
*                   vakio_a     : Linssin ja laserin välinen etäisyys
* HUOMIOITAVAA:     Tiedoston olemassaolon aiheuttaman ongelman korjaamiseksi
*                   Kääntäjän automaattinen tiedotsofunktiotarkistus käännetään
*                   hetkeksi pois päältä.
*****************************************************************************}

procedure Tallenna(strFilename: string; etaisyys: array of single;
                   Pysty, Vaaka, Sarjap, Rinnakkaisp : integer;
                   strNimi : string; alfa, beeta, vakio_a: single);
var
  F               : TextFile;
  DeltaV, DeltaH,
  n, H, V         : integer;
begin
  try
    {$I-} { Kääntäjän virheentarkistus pois }
    { Avataan, luodaan, tiedosto kirjoitettavaksi }
    AssignFile(F, strFilename);
    Reset(F);
    if IOResult <> 0 then ReWrite(F);
    {$I+} { Kääntäjän virheentarkistus takas }

    { Kirjoitetaan tiedostoon }
    Writeln(F, '#Etäisyysmittaus');
    Writeln(F, '#Arno Solin, Päättötyö 2004-2005, versio 1.0');
    WriteLn(F, '');

    Writeln(F, '#Nimi');
    Writeln(F, strNimi);
```

```
    Writeln(F, '#Alfa');
    Writeln(F, FloatToStr(alfa));

    Writeln(F, '#Beeta');
    Writeln(F, FloatToStr(beeta));

    Writeln(F, '#Vakio_a');
    Writeln(F, FloatToStr(vakio_a));

    Writeln(F, '#Sarjaportin osoite');
    Writeln(F, IntToStr(Sarjap));

    Writeln(F, '#Rinnakkaisportin osoite');
    Writeln(F, IntToStr(Rinnakkaisp));

    Writeln(F, '#Pisteet pystytasossa; Vertical');
    Writeln(F, IntToStr(Pysty+1));

    { Ei tiedetä; oletusarvo }
    Writeln(F, '#Pisteet vaakatasossa; Horizontal');
    Writeln(F, IntToStr(Vaaka));

    { Pisteiden tallennus }
    WriteLn(F, #13#10, 'Mittapisteet:');
    { Lasketaan delta-arvot }
    DeltaV := 56 div Pysty;
    DeltaH := 257 div Vaaka;

    for H := 0 to Vaaka-1 do
    begin
      for V := 0 to Pysty do
      begin
        WriteLn(F, Format('%.3d_%.2d.jpg', [H*DeltaH, V*DeltaV]));
        n := H*(Pysty+1)+V;
        WriteLn(F, Format('  %.3f', [etaisyys[n]]));
      end;
      WriteLn(F, '');
    end;
  finally
    { Suljetaan tiedosto }
    CloseFile(F);
  end;
end;

end.
```

Type3D

```
(*****************************************************************************
* MODUULI:          Type3D 1.0
* KÄÄNTÄJÄ:         Delphi 6
* TARKOITUS:        3D-laskennan muuttujatyyppien koordinointi
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       7.2.2005
* KUVAUS:           Sisältää tarpeellisten tietuetyyppien määritelmät
* ALIOHJELMAT:      -
* HUOMIOITAVAA:     Linkitetty moniin yhteyksiin.
*****************************************************************************)

unit Type3D;

interface

type
  { 3D-koordinaattien pyhä kolminaisuus }
  TKoordinaatti = record
      X : single;
      Y : single;
      Z : single;
  end;
  { Polygonit esitetään tasoina, jotka määritellään neljän arvon avulla }
  TPolygoni = record
      a : integer;
      b : integer;
      c : integer;
      d : integer;
  end;
  { Taulukoiden luomista varten on luotava omat tietueet, sillä muuten arvoja ei
    pysty kopioimaan suoraan muihin muuttujiin }
  TKoordinaattiArray = array of TKoordinaatti;
  TPolygoniArray = array of TPolygoni;

implementation

end.
```

VRML

```
(*****************************************************************************
* MODUULI:          VRML 1.1
* KÄÄNTÄJÄ:         Delphi 6
* TARKOITUS:        VRML-tiedostojen luonti ja avaus
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       19.2.2005
* KUVAUS:           Tallentaa ja avaa taulukoituja polygoneja
* ALIOHJELMAT:      Avaa
*                   Tallenna
*                   Suhteuta
* HUOMIOITAVAA:     VRML (Virtual Reality Modeling Language)-tiedostojen rakenne
*                   on hieman huteralla pohjalla. Todettu toimivaksi ainakin 3D
*                   Studio MAX VRML97 exporter 4.0:n kanssa.
*****************************************************************************)

unit VRML;

interface

uses
  SysUtils, Type3D;

var
  piste : TKoordinaattiArray; { Pisteet }
  polygoni : TPolygoniArray;  { Polygonit }

{ Pisteen suhteuttaminen }
procedure Suhteuta(koko : single; var piste : TKoordinaattiArray);

{ VRML-tiedoston avaaminen }
procedure Avaa(strFilename: string;
               var PisteOut : TKoordinaattiArray; var PolygoniOut : TPolygoniArra
y);

{ VRML-tiedoston tallentaminen }
procedure Tallenna(strFilename: string;
                   piste : TKoordinaattiArray; polygoni : TPolygoniArray);

implementation

{*****************************************************************************
* ALIOHJELMA:       Etsi 1.0
* TARKOITUS:        Etsiä tietty kohta tiedostosta
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       3.2.2005
* KUVAUS:           Etsii halutun kohdan tiedostosta
* PARAMETRIT:       strIn    : Merkkijono, josta etsitään
*                   strFind  : Merkkijono, jota etsitään
*                   intKohta : Kohta, josta aloitetaan
* HUOMIOITAVAA:     Alisteinen Avaa-aliohjelmalle
*****************************************************************************}

procedure Etsi(strIn, strFind: string; var intKohta: integer);
var
  i      : integer;
  strTmp : string;
begin
  { Pätkitään oikea osa tekstistä esiin ja tutkitaan tekstiä }
  strTmp := copy(strIn, intKohta, length(strIn)-intKohta);
  i := Pos(strFind, strTmp);
  intKohta := intKohta+i+length(strFind);

  { Valitetaan, jollei etsittyä löydy }
  if i = 0 then Abort;
end;

{*****************************************************************************
```

```
* ALIOHJELMA:       LueKoordinaatteja 1.1
* TARKOITUS:        Luoda taulukoidut koordinaattiarvot
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       20.2.2005
* KUVAUS:           Lukee X-, Y- ja Z-arvoja tiedostosta
* PARAMETRIT:       strIn : Merkkijono, jota luetaan
*                   i     : Kohta, josta aloitetaan
* HUOMIOITAVAA:     Alisteinen Avaa-aliohjelmalle
*                   Koordinaattien on oltava sisennettyjä.
*****************************************************************************}

procedure LueKoordinaatteja(strIn : string; var i : integer);
var
  code,                { Virhekoodi, ei hyödynnetä }
  index    : integer;  { Tallennettavan pisteen numero }
  strTmp   : string;   { Väliaikaismerkkijono }
  valiarvo : single;   { Väliaikainen liukulukuarvo }

  { Aliohjelma pisteiden tallentamista varten }
  procedure Tallenna(luku : single; var index : integer);
  var
    n : integer;
  begin
    { Muuta kokoa }
    n := Length(piste);
    if index mod 3 = 0 then
      SetLength(piste, n+1);

    { Annetaan arvot }
    case index mod 3 of
      0 : piste[index div 3].X := luku;
      1 : piste[index div 3].Z := luku;
      2 : piste[index div 3].Y := luku;
    end;
    inc(index); { Ei muutella vanhoja arvoja, vaan siirrytään eteenpäin }
  end;
begin
  { Koordinaatit ovat muodossa "0.00 0.00 0.00,"
    Pätkitään tekstistä oikea osa esiin ja nollataan muuttujia }
  strIn := copy(strIn, i-1, length(strIn)-i);
  strIn := TrimLeft(strIn);
  i := 0; index := 0;

  { Liikutaan tekstin läpi merkki kerrallaan ja suoritetaan kriittisissä
    kohdissa (" ", ",", "]") toimenpiteet }
  repeat

    strTmp := strTmp + Trim(strIn[i]);
    inc(i);
    if (strIn[i] = ' ') or (strIn[i] = ',') or (strIn[i] =  ']') then
    begin
      if strTmp = ',' then strTmp := '';
      if strTmp = '' then continue;

      { Muutetaan teksti luvuksi ja lähetetään tallennettavaksi }
      val(strTmp, valiarvo, code);
      Tallenna(valiarvo, index);
      strTmp := '';
    end;
  until strIn[i] = ']';
end;

{*****************************************************************************
* ALIOHJELMA:       LuePolygoneja 1.0
* TARKOITUS:        Luoda taulukoidut polygoniarvot
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       3.2.2005
* KUVAUS:           Lukee monikulmion kulmapisteitä tiedostosta
* PARAMETRIT:       strIn : Merkkijono, jota luetaan
```

```
*                   i     : Kohta, josta aloitetaan
* HUOMIOITAVAA:     Alisteinen Avaa-aliohjelmalle
*****************************************************************************}

procedure LuePolygoneja(strIn: string; var i: integer);
var
  n      : integer;  { Muutettava lukuarvo, väliaikaismuuttuja }
  strTmp : string;   { Väliaikaismerkkijono }

  { Aliohjelma polygonien erittelyä ja tallentamista varten }
  procedure Tallenna(strIn : string);
  var
    n, a, i : integer;
    strTmp : string;
  begin
    { Muuta kokoa }
    n := Length(polygoni);
    SetLength(polygoni, n+1);

    { Arvo muotoa: "0, 0, 0, 0," }
    for i := 0 to 2 do
    begin
      a := pos(',', strIn);
      strTmp := copy(strIn, 0, a-1);
      case i of
        0: polygoni[n].a := StrToInt(strTmp);
        1: polygoni[n].b := StrToInt(strTmp);
        2: polygoni[n].c := StrToInt(strTmp);
      end;
      strIn := copy(strIn,a+1, length(strIn)-a);
      strIn := Trim(strIn);
    end;

    { Piste d luodaan vain, jos tiedoston polygonit ovat nelikulmaisia. Muussa
      tapauksessa pisteet c ja d saavat saman arvon. }
    a := pos(',', strIn);
    if a > 0 then
    begin
      strTmp := copy(strIn, 0, a-1);
      polygoni[n].d := StrToInt(strTmp);
    end
    else
      polygoni[n].d := polygoni[n].c;
  end;
begin
  { Polygoni muodossa "0, 0, 0, 0, -1"
    Teksti erilleen ja muuttujien nollaus }
  strIn := copy(strIn, i-1, length(strIn)-i);
  n := pos(']', strIn);
  strIn := copy(strIn, 0, n);
  strIn := Trim(strIn);
  i := 0;

  while length(strIn) > 8 do
  begin
    { Eritellään erilleen yksi polygoni }
    n := Pos('-1', strIn);
    strTmp := copy(strIn, 0, n-1);
    strIn := copy(strIn, n+3, length(strIn)-n);

    { Lähetetään eriteltäväksi ja tallennettavaksi }
    Tallenna(Trim(strTmp));
  end;
end;

{*****************************************************************************
* ALIOHJELMA:       Avaa 1.1
* TARKOITUS:        VRML-tiedoston avaaminen ja arvojen lukeminen
* NIMI:             Arno Solin
```

```
* PÄIVÄMÄÄRÄ:       19.2.2005
* KUVAUS:           Lukee VRML-tiedostosta pisteitä ja polygoneja
* PARAMETRIT:       strFilename : Tiedoston nimi (*.wrl)
* HUOMIOITAVAA:     Ei ole kovin tarkka oikeasta tiedostorakenteesta.
*                   Ei sisällä virheenkäsittelyä.
*                   Tukee koordinaattimuotoisia tiedostoja, joista ladataan
*                   näkyviin ainoastaan ensimmäinen kappale.
*****************************************************************************}

procedure Avaa(strFilename: string;
               var PisteOut : TKoordinaattiArray; var PolygoniOut : TPolygoniArra
y);
var
  i        : integer;  { Kohta tiedostossa }
  strRivi,             { Yksi rivi }
  strTmp   : string;   { Tiedoston sisältö }
  Tiedosto : TextFile; { Tiedosto }
begin
  { Luetaan koko tiedoston sisältö }
  AssignFile(Tiedosto, strFilename);
  Reset(Tiedosto);

  while not Eof(Tiedosto) do
  begin
    ReadLn(Tiedosto, strRivi);
    strTmp := strTmp + strRivi;
  end;

  CloseFile(Tiedosto);

  { Poistetaan vanhat polygonit ja pisteet }
  piste := nil;
  polygoni := nil;

  { Kohta tiedostossa, josta aloitetaan }
  i := 0;

  { Kaikki kirjaimet pieniksi helpottamaan tunnistusta }
  strTmp := LowerCase(strTmp);

  { Siirrytään oikeaan kohtaan ja varmistutaan tiedostorakenteesta }
  Etsi(strTmp, 'def ',i);
  Etsi(strTmp, 'transform {', i);
  Etsi(strTmp, 'def', i);
  Etsi(strTmp, 'coordinate {', i);
  Etsi(strTmp, 'point [', i);

  { Luetaan tiedostossa olevat koordinaatit }
  LueKoordinaatteja(strTmp, i);

  { Seuraava kohta }
  Etsi(strTmp, 'coordindex [', i);

  { Luetaan polygonit }
  LuePolygoneja(strTmp, i);

  { Yhdistetään moduulin globaalit koordinaatit ulosmeneviin arvoihin }
  PisteOut := piste; PolygoniOut := polygoni;
end;

{*****************************************************************************
* ALIOHJELMA:       Suhteuta 1.0
* TARKOITUS:        Muokata moniulotteisia kappaleita sopivaan kokoon
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       3.2.2005
* KUVAUS:           Suhteuttaa kappaleen tiettyyn kokoon
* PARAMETRIT:       koko  : suurin arvo, johon suhteutetaan muut
*                   piste : Taulukko, jonka arvoja muutetaan
* HUOMIOITAVAA:
```

```
*****************************************************************************}

procedure Suhteuta(koko : single; var piste : TKoordinaattiArray);
var
  i   : integer;  { Silmukkamuuttuja }
  max : single;   { Maksimiarvo ja laskettu suhde }
begin
  { Etsitään suurin itseisarvo }
  max := 0;
  for i := 0 to length(piste)-1 do
  begin
    if Abs(piste[i].X) > max then max := Abs(piste[i].X);
    if Abs(piste[i].Y) > max then max := Abs(piste[i].Y);
    if Abs(piste[i].Z) > max then max := Abs(piste[i].Z);
  end;
  { Lasketaan arvo, johon suhteutetaan }
  max := koko / max;

  { Annetaan uudet arvot }
  for i := 0 to length(piste)-1 do
  begin
    piste[i].X := max * piste[i].X;
    piste[i].Y := max * piste[i].Y;
    piste[i].Z := max * piste[i].Z;
  end;
end;

{*****************************************************************************
* ALIOHJELMA:       Tallenna 1.1
* TARKOITUS:        VRML-tiedoston luominen taulukoiduista arvoista
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       19.2.2005
* KUVAUS:           Luo VRML-tiedostoston pisteistä ja polygoneista
* PARAMETRIT:       strFilename : Tiedoston nimi (*.wrl)
*                   Piste       : Tallennettavat koordinaatit (X, Y, Z), taul.
*                   Polygoni    : Tallennettavat monikulmiot (a, b, c, d), taul.
* HUOMIOITAVAA:     Tiedoston rakenne mukailee tavallisen VRML 2.0 -tiedoston
*                   rakennetta.
*                   Ei sisällä virheenkäsittelyä.
*                   Y-arvo on korkeus ja Z on syvyys.
*****************************************************************************}

procedure Tallenna(strFilename: string;
                   Piste : TKoordinaattiArray; Polygoni : TPolygoniArray);
var
  F : TextFile;
  i : integer;
begin
  { Avataan, luodaan, tiedosto kirjoitettavaksi }
  AssignFile(F, strFilename);
  ReWrite(F);
                                                        { Tiedoston rakenne }
  WriteLn(F, '#VRML V2.0 utf8' + #13#10);                  { VRML-tiedosto }
  WriteLn(F, '# Arno Solin, Päättötyö, VRML exporter 1.0');
  WriteLn(F, '# Date: ' + FormatDateTime('mmm d yyyy hh:nn', Now) + #13#10);

  WriteLn(F, 'DEF mittaus Transform {');                          { Kappale }
  WriteLn(F, '  translation 0 0 0');
  WriteLn(F, '  children [');
  WriteLn(F, '    Shape {');
  WriteLn(F, '      appearance Appearance {');
  WriteLn(F, '        material Material {');
  WriteLn(F, '          diffuseColor 1.0 0.5 0.5');
  WriteLn(F, '        }');
  WriteLn(F, '      }');

  WriteLn(F, '      geometry DEF mittaus-FACES IndexedFaceSet {'); { Geometria }
  WriteLn(F, '        ccw TRUE');
  WriteLn(F, '        solid TRUE');
```

```
  WriteLn(F, '        convex TRUE');
  WriteLn(F, '        coord DEF mittaus-COORD Coordinate { point [');

  for i := 0 to Length(piste)-1 do                                    { Pisteet }
    if i < Length(piste)-1 then
      WriteLn(F, '          ' +
        Format('%.2f %.2f %.2f,', [piste[i].X, piste[i].Z, piste[i].Y]))
    else
      WriteLn(F, '          ' +
        Format('%.2f %.2f %.2f]', [piste[i].X, piste[i].Z, piste[i].Y]));

  WriteLn(F, '        }');
  WriteLn(F, '        coordIndex [');

  for i := 0 to Length(polygoni)-1 do                               { Polygonit }
    if i < Length(polygoni)-1 then
      WriteLn(F, '          ' +
        Format('%d, %d, %d, %d, -1,',
        [polygoni[i].a, polygoni[i].b, polygoni[i].c, polygoni[i].d]))
    else
      WriteLn(F, '          ' +
        Format('%d, %d, %d, %d, -1]',
        [polygoni[i].a, polygoni[i].b, polygoni[i].c, polygoni[i].d]));


  WriteLn(F, '      }');                                             { Loppuosa }
  WriteLn(F, '    }');
  WriteLn(F, '  ]');
  WriteLn(F, '}');

  { Suljetaan tiedosto }
  CloseFile(F);
end;

end.
```

Optimoi

```
(*****************************************************************************
* MODUULI:          Optimoi 1.0
* KÄÄNTÄJÄ:         Delphi 6
* TARKOITUS:        Sovittaa suorakulmion pistejoukkoon
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       19.2.2005
* KUVAUS:           Sisältää tarpeellisten tietuetyyppien määritelmät
* ALIOHJELMAT:      Suorakulmio : Suorakulmion luominen
*****************************************************************************)

unit optimoi;

interface

uses Type3D, SysUtils;

type
  TSuora = record { y = bx + a }
    b: single;    { Kulmakerroin }
    a: single;    { Vakio }
    r: single;    { Korrelaatio }
  end;

{ Koordinaattien sovittaminen suorakulmioon; 2D }
procedure Suorakulmio(var piste: TKoordinaattiArray);

implementation

uses Unit1;

{*****************************************************************************
* FUNKTIO:          regressiosuora_b 1.0
* TARKOITUS:        Suoran kulmakertoimen laskeminen
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       29.12.2004
* KUVAUS:           Sovittaa suoran pistejoukkoon
* PARAMETRIT:       x,y : taulukoidut x- ja y-arvot
* PALUUARVO:        Palauttaa beetakulman kulmakertoimen (rad)
* HUOMIOITAVAA:     Single-muuttujat pyöristävät hieman tuloksia.
*****************************************************************************}

function regressiosuora_b(x,y: array of single): single;
var
  i               : integer; // Silmukassa käytettävä muuttuja
  Tmp1, Tmp2, Tmp3 : double;  // Väliaikaismuuttuja
begin
  { y = bx + a
    Regressiosuoran kulmakertoimen laskeminen
    b = (n * sum(xi*yi)-(sum(xi)*sum(yi)))/(n*sum(xi^2) - sum(xi)^2) }

  { Muuttujien nollaus }
  Tmp1 := 0; Tmp2 := 0; Tmp3 := 0;

  { Summien laskeminen }
  for i := 0 to Length(x)-1 do
  begin
    Tmp1 := Tmp1 + x[i]*y[i];
    Tmp2 := Tmp2 + x[i];
    Tmp3 := Tmp3 + y[i];
  end;

  Tmp1 := Length(x)*Tmp1 - Tmp2*Tmp3;

  { Muuttujan nollaus }
  Tmp3 := 0;

  { Summan laskeminen }
```

```
  for i := 0 to length(x)-1 do
    Tmp3 := Tmp3 + sqr(x[i]);

  { Paluuarvon asettaminen }
  try
    result := Tmp1 / ((Length(x)*Tmp3) - sqr(Tmp2));
  except
    result := 0;
  end;
end;

{*****************************************************************************
* FUNKTIO:          regressiosuora_a 1.0
* TARKOITUS:        Suoran vakiotermin laskeminen
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       29.12.2004
* KUVAUS:           Sovittaa suoran pistejoukkoon
* PARAMETRIT:       x,y : taulukoidut x- ja y-arvot
*                   b   : kulmakerroin (regressiosuora_b)
* PALUUARVO:        Palauttaa suoran ja y-akselin leikkauskohdan
* HUOMIOITAVAA:     Single-muuttujat pyöristävät hieman tuloksia.
*                   Kulmakerroin laskettava ensin.
*****************************************************************************}

function regressiosuora_a(x,y: array of single; b: single): single;
var
  i                : integer; { Silmukassa käytettävä muuttuja }
  Tmp1, Tmp2       : single;  { Väliaikaismuuttuja }
begin
  { y = bx + a
    Regressiosuoran vakiotermin laskeminen
    a = (sum(yi) - b*sum(xi)) / n }

  { Muuttujien nollaus }
  Tmp1 := 0; Tmp2 := 0;

  { Summien laskeminen }
  for i := 0 to length(x)-1 do
  begin
    Tmp1 := Tmp1 + y[i];
    Tmp2 := Tmp2 + x[i];
  end;

  { Paluuarvon asettaminen }
  result := (Tmp1 - b*Tmp2) / Length(x);
end;

{*****************************************************************************
* FUNKTIO:          korrelaatio 1.0
* TARKOITUS:        Suoran korreloimisen määrittäminen
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       28.1.2005
* KUVAUS:           Laskee Poearsonin korrelaatiokertoimen
* PARAMETRIT:       x,y : taulukoidut x- ja y-arvot
* PALUUARVO:        Palauttaa korrelaatiokertoimen (-1 - 1)
*****************************************************************************}

function korrelaatio(x,y: array of single): single;
var
  i                : integer; { Silmukassa käytettävä muuttuja }
  Tmp1, Tmp2,
  Tmp3, Tmp4       : double;  { Väliaikaismuuttuja }
begin
  { Pearsonin korrelaatiokerroin (tulomomenttikerroin)
    r = (n*sum(x*y)-sum(x)*sum(y))/
        sqrt((n*sum(x^2)-sum(x)^2)(n*sum(y^2)-sum(y)^2))}

  { Muuttujien nollaus }
  Tmp1 := 0; Tmp2 := 0; Tmp3 := 0; Tmp4 := 0;
```

```
  { Summien laskeminen }
  for i := 0 to length(x)-1 do
  begin
    Tmp1 := Tmp1 + x[i]*y[i];
    Tmp2 := Tmp2 + x[i];
    Tmp3 := Tmp3 + y[i];
  end;

  { Lasketaan osoittaja }
  result := length(x)*Tmp1 - Tmp2*Tmp3;
  Tmp1 := 0;

  { Summien laskeminen }
  for i := 0 to length(x)-1 do
  begin
    Tmp1 := Tmp1 + sqr(x[i]);
    Tmp4 := Tmp4 + sqr(y[i]);
  end;

  { Lasketaan neliöjuuren alle jäävä luku }
  Tmp1 := (length(x)*Tmp1 - sqr(Tmp2)) * (length(x)*Tmp4 - sqr(Tmp3));

  { Lasketaan tulos }
  try
    result := result / sqrt(Tmp1);
  except
    result := 0;
  end;
end;

{*****************************************************************************
* FUNKTIO:          Leikkauspiste 1.0
* TARKOITUS:        Kahden suoran leikkauspisteen määrittäminen
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       18.2.2005
* KUVAUS:           Laksee kahden suoran leikkauspisteen
* PARAMETRIT:       s1,s2 : kaksi suoraa (muotoa: y = bx + a)
* PALUUARVO:        Palauttaa koordinaatin
* HUOMIOITAVAA:     Vain 2D. Z-akselilla koordinaatti aina 0.
*****************************************************************************}

function Leikkauspiste(s1, s2: TSuora): TKoordinaatti;
begin
  result.X := (s2.a - s1.a) / (s1.b - s2.b);
  result.Y := s1.b * result.X + s1.a;
  result.Z := 0;
end;

{*****************************************************************************
* FUNKTIO:          Sovita 1.0
* TARKOITUS:        Suorien sovittaminen oikeisiin arvoihin
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       18.2.2005
* KUVAUS:           Valikoi taulukosta oikeat arvot sovittamiseen
* PARAMETRIT:       piste    : Taulukoidut koordinaatit
*                   intKohta : Lähtöpiste
*                   intMaara : Pisteiden määrä
* PALUUARVO:        Palauttaa suoran kulmakeroimen, vakiotermin ja korrelaatiok.
*****************************************************************************}

function Sovita(piste: TKoordinaattiArray; intKohta, intMaara : integer): TSuora;
var
  n, i  : integer;          { Silmukoita ja väliarvoja varten }
  sx,sy : array of single; { Taulukot arvoille }
begin
  { Tarvittavan taulukon koko }
  setLength(sx,intMaara);
  setLength(sy,intMaara);
```

```
  for i := 0 to length(sx)-1 do
  begin
    { Erikoiskohta, jossa menty koko kierros }
    n := (intKohta+i) mod Length(piste);
    sx[i] := piste[n].x;
    sy[i] := piste[n].y;
  end;

  { Sovitetaan suora arvoihin }
  result.b := regressiosuora_b(sx, sy);
  result.a := regressiosuora_a(sx,sy,result.b);

  { Korrelaatio }
  result.r := Korrelaatio(sx,sy);
end;

{*****************************************************************************
* ALIOHJELMA:       Suorakulmio 1.0
* TARKOITUS:        Suorakulmion sovittaminen pistejoukkoon
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       18.2.2005
* KUVAUS:           Sovittaa kaksiulotteisesti suorakulmion pistejoukkoon
* PARAMETRIT:       piste : Taulukoidut koordinaatit (myös paluuarvo)
* HUOMIOITAVAA:     Palauttaa neljä pistettä (suorakulmion kulmat).
*                   Tulos ei aina vastaa odotuksia.
*****************************************************************************}

procedure Suorakulmio(var piste: TKoordinaattiArray);
var
  i, intLeveys, maara, intPi, intPl : integer;
  r,paras : single;
  s1,s2,s3,s4 : TSuora;
begin
  maara := Length(piste);
  intPi := 0; intPl := 0; paras := 0;

  for i := 0 to (maara div 4) do
  begin
    for intLeveys := 2 to maara div 2 - 1 do
    begin
      { Sovitus suorakulmioon }
      s1 := Sovita(piste, i,                          intLeveys);
      s2 := Sovita(piste, i+intLeveys-1,              maara div 2 - intLeveys+2);
      s3 := Sovita(piste, i+maara div 2,              intLeveys);
      s4 := Sovita(piste, i+maara div 2+intLeveys-1, maara div 2 - intLeveys+2);

      r := s1.r * s2.r * s3.r * s4.r;

      if paras < abs(r) then
      begin
        intPi := i;
        intPl := intLeveys;
        paras := abs(r);
      end;
    end;
  end;

{ Paras sovitus }
  i := intPi; intLeveys := intPl;
  s1 := Sovita(piste, i,                          intLeveys);
  s2 := Sovita(piste, i+intLeveys-1,              maara div 2 - intLeveys+2);
  s3 := Sovita(piste, i+maara div 2,              intLeveys);
  s4 := Sovita(piste, i+maara div 2+intLeveys-1, maara div 2 - intLeveys+2);

{ Paras suorakulmio }
  s1.b := (s1.b - s2.b + s3.b - s4.b) / 4;
  s2.b := -s1.b; s3.b := s1.b; s4.b := -s1.b;
```

```
  SetLength(piste, 4);
  piste[0] := Leikkauspiste(s1,s2);
  piste[1] := Leikkauspiste(s2,s3);
  piste[2] := Leikkauspiste(s3,s4);
  piste[3] := Leikkauspiste(s4,s1);
end;


end.
```

Kalibrointi (Sovellus)

```
(*******************************************************************************
* OHJELMA:          Kalibrointi (kalibrointi 1.0.0)
* KÄÄNTÄJÄ:         Delphi 6
* TARKOITUS:        Mittalaitteen kalibrointi
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       27.1.2005
* KUVAUS:           Antaa käyttäjän määrittää mittaamalla vakioarvoja
* HUOMIOITAVAA:     Kalibrointi tapahtuu regressiosuoran avulla sen jälkeen, kun
*                   käyttäjä on syöttänyt vähintään kaksi arvoa.
*******************************************************************************)

program Kalibrointi;

uses
  Forms,
  Main in 'Main.pas' {frmMain};

{$R *.res}

begin
  Application.Initialize;
  Application.Title := 'Kalibrointi';
  Application.CreateForm(TfrmMain, frmMain);
  Application.Run;
end.
```

Main

```
(*****************************************************************************
* MODUULI:          Main 1.1 (Kalibrointi - frmMain)
* KÄÄNTÄJÄ:         Delphi 6
* TARKOITUS:        Kalibroinnin koordinoiminen
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       5.1.2005
* KUVAUS:           Mittalaitteen vakioiden kalibrointi
* HUOMIOITAVAA:     Lomakkeen moduuli.
*****************************************************************************)

unit Main;

interface

uses
  Windows, Messages, SysUtils, Variants, ExtCtrls, StdCtrls, Controls, Classes,
  Forms, Dialogs,
  Kamera,    { Kameran ohjaus }
  Moottori, { Laserin ohjaus }
  Etsi,      { Pisteiden etsiminen }
  Laske;     { Etäisyys ja kalibrointi }

type
  TfrmMain = class(TForm)
    pnlPaneeli  : TPanel;
    txtAlfa     : TEdit;      { Arvot näkyvissä }
    txtBeeta    : TEdit;
    txtVakio_a  : TEdit;
    cbCom       : TComboBox; { Sarjaportin valinta }
    cbPar: TComboBox;        { Rinnakkaisportin valinta }
    btnLataa    : TButton;    { Kuvan lataus }
    btnKalibroi : TButton;    { kalibroinnin aloitus }
    btnTallenna : TButton;    { Tulosten tallentaminen }
    btnExit     : TButton;    { Sovelluksesta poistuminen }
    Memo: TMemo;              { Tekstikehys }
    procedure btnExitClick(Sender: TObject);
    procedure btnLataaClick(Sender: TObject);
    procedure cbComChange(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure btnKalibroiClick(Sender: TObject);
    procedure FormMouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure btnTallennaClick(Sender: TObject);
    procedure cbParChange(Sender: TObject);
  end;

var
  frmMain: TfrmMain;

  { Globaalit muuttujat }
  bManuaali: boolean;

  { Kalibrointiin tarvittavat arvot }
  mitat_x : array of single;
  mitat_y : array of single;
implementation

{$R *.dfm}

(*****************************************************************************
* ALIOHJELMA:       raportoi 1.0
* TARKOITUS:        Lokitiedon ja opasteiden luominen
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       1.1.2005
* KUVAUS:           Lähettää tekstiä näytettäväksi käyttäjälle
* PARAMETRIT:       strTeksti : raportoitava teksti
* HUOMIOITAVAA:
```

```
*****************************************************************************)

procedure raportoi(strTeksti : string);
begin
  frmMain.Memo.Text := frmMain.Memo.Text + strTeksti + #13#10;

  { Siirrytään alaspäin }
  frmMain.Memo.SelStart := Length(frmMain.Memo.Text);
  SendMessage(frmMain.Memo.Handle, EM_SCROLLCARET, 0,0);
end;

(*****************************************************************************
* ALIOHJELMA:       Piste 1.0
* TARKOITUS:        Tallentaa pisteen käyttöä kalibrointia varten
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       1.1.2005
* KUVAUS:           Lähettää tekstiä näytettäväksi käyttäjälle
* PARAMETRIT:       x    : kohta kuvassa
*                   uusi : luodaan uusi/muokataan vanhaa
* HUOMIOITAVAA:     Tallentaa pisteet globaaleihin dynaamisiin taulukoihin
*                   mitat_x ja mitat_y
*****************************************************************************)

procedure Piste(x: integer; uusi: boolean);
var
  numero: integer;
  strInput: string;
begin
  if uusi = true then
  begin
    numero := Length(mitat_x);
    SetLength(mitat_x,numero+1);
    SetLength(mitat_y,numero+1);

    repeat
      strInput := InputBox('Mitta',
        'Anna pisteen etäisyys (cm) mahdollisimman tarkasti.', '0.00');
      mitat_y[numero] := StrToFloat(strInput);
    until mitat_y[numero] <> 0;
  end
  else
    numero := Length(mitat_x)-1;

  mitat_x[numero] := x / 159;
end;

(*****************************************************************************
* ALIOHJELMA:       HankiMittatiedot 1.0
* TARKOITUS:        Hankkia kaikki tarvittavat tiedot pisteen luomista varten
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       1.1.2005
* KUVAUS:           Ohjaa moduulien funktioita
* HUOMIOITAVAA:     Virhetilanteessa toiminta epävarma
*****************************************************************************)

procedure HankiMittatiedot();
var
  x: integer;
begin
  Kamera.bToimi := true;

  try
    { Laser päälle }
    Moottori.Laser(true);

    { Ladataan kuva kamerasta. }
    Kamera.lataa('kalibrointi_tmp.jpg');

    { Laser pois }
```

```
    Moottori.Laser(false);

    { Etsitään kuvasta laserpiste. }
    x := Etsi.EtsiPiste('kalibrointi_tmp.jpg');

    { Tallenetaan saatu kohta kuvasta. }
    Piste(x, true);

    { Näytetään tulos }
    frmMain.Canvas.Draw(50,20,AvaaJPEG('kalibrointi_tmp.jpg', x));

    { Pyydetään käyttäjää varmistamaan tulos. }
    bManuaali := true;

    raportoi(#13#10+'*** VARMISTA PISTEEN KOHTA ***');
    raportoi('Paina hiirellä haluamaasi kohtaa kuvassa.');
    raportoi('Jos olet tyytyväinen automaattiseen tulokseen, '
      +'paina hiirellä lomaketta kuvan ulkopuolella.');
  except
    { Laser pois }
    Moottori.Laser(false);

    raportoi('VIRHE: Kameran toiminta häiriintynyt!');
    MessageDlg('Kameran toiminta häiriintynyt.'+ #13#10 +'Lataus keskeytetty.',
      mtError, [mbOk], 0);
  end;
end;

(*****************************************************************************
* FUNKTIO:          Tallenna_I 1.0
* TARKOITUS:        Tallentaa keskeisten muuttujien arvot
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       5.1.2005
* KUVAUS:           Tallentaa (kalibroimalla saadut) arvot käyttöä varten
* PARAMETRIT:       strFilename : Tiedoston nimi
* HUOMIOITAVAA:     Käsittelee ainoastaan tiedoston alkuosaa.
*****************************************************************************)

procedure Tallenna_I(strFilename: string);
var
  F : TextFile;
begin
  { Avataan, luodaan, tiedosto kirjoitettavaksi }
  AssignFile(F, strFilename);
  ReWrite(F);

  { Kirjoitetaan tiedostoon }
  Writeln(F, '#Kalibroimalla luotu tiedosto.');

  Writeln(F, '#Nimi');
  Writeln(F, 'Kalibroidut arvot');

  Writeln(F, '#Alfa');
  Writeln(F, frmMain.txtAlfa.Text);

  Writeln(F, '#Beeta');
  Writeln(F, frmMain.txtBeeta.Text);

  Writeln(F, '#Vakio_a');
  Writeln(F, frmMain.txtVakio_a.Text);

  Writeln(F, '#Sarjaportin osoite');
  Writeln(F, intToStr(Kamera.CommPort1.ComNumber));

  Writeln(F, '#Rinnakkaisportin osoite');
  Writeln(F, intToStr(frmMain.cbPar.ItemIndex));

  { Ei tiedetä; oletusarvo }
  Writeln(F, '#Pisteet pystytasossa; Vertical');
```

```
  Writeln(F, '0');

  { Ei tiedetä; oletusarvo }
  Writeln(F, '#Pisteet vaakatasossa; Horizontal');
  Writeln(F, '0');

  { Suljetaan tiedosto }
  CloseFile(F);
end;

{                   TAPAHTUMAKÄSITTELY (frmMain) }

(*****************************************************************************
* ALIOHJELMA:       btnExitClick 1.0
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       5.1.2005
* KUVAUS:           Sovelluksen sulkeminen
*****************************************************************************)

procedure TfrmMain.btnExitClick(Sender: TObject);
begin
  { Suljetaan sovellus. }
  Application.Terminate;
end;

(*****************************************************************************
* ALIOHJELMA:       btnLataaClick 1.0
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       5.1.2005
* KUVAUS:           Lataa uuden kuvan / Keskeyttää latauksen
*****************************************************************************)

procedure TfrmMain.btnLataaClick(Sender: TObject);
begin
  if btnLataa.Caption = 'Stop' then
  begin
    { Keskeytetään lataus }
    Kamera.bToimi := false;

    { Lopetetaan komunikointi kameran kanssa }
    laheta($FF);        // Lopetus (Termination Byte, 0xFF)

    btnLataa.Caption := '&Lataa kuva';
    btnLataa.Cancel := false;
    btnExit.Cancel := true;
  end
  else
  begin
    btnLataa.Caption := 'Stop';
    btnExit.Cancel := false;
    btnLataa.Cancel := true;
    Memo.Text := '';

    HankiMittatiedot;

    btnLataa.Caption := '&Lataa kuva';
    btnLataa.Cancel := false;
    btnExit.Cancel := true;
  end;
end;

(*****************************************************************************
* ALIOHJELMA:       cbComChange 1.0
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       5.1.2005
* KUVAUS:           Vaihtaa sarjaportin osoitteen
*****************************************************************************)

procedure TfrmMain.cbComChange(Sender: TObject);
```

```
begin
  { Kytketään sarjaportti pois päältä. }
  Kamera.CommPort1.Open := false;

  { Vaihdetaan käytettävää porttia. }
  if cbCom.Text = 'COM1' then
    Kamera.CommPort1.ComNumber := 1
  else
    Kamera.CommPort1.ComNumber := 2;

  { Avataan sarjaportti jälleen. }
   Kamera.CommPort1.Open := true;
end;

procedure TfrmMain.FormCreate(Sender: TObject);
begin
  txtAlfa.Text        := '24.7273';
  txtBeeta.Text       := '0.00';
  txtVakio_a.Text     := '0.00';
  btnKalibroi.Enabled := false;

  raportoi('*** KALIBROINTI 1.0 ***');
  raportoi('(c) Arno Solin 2005');
  raportoi('Tämä pieni sovellus on tarkoitettu auttamaan Agfa ePhoto -kameraa '
  +'käyttävän etäisyysmittalaitteen kalibroinnissa. Tarvitset mittojen '
  +'kalibrointiin vähintään kaksi etäisyysarvoa. Paina ''Lataa kuva''-nappia '
  +'aloittaaksesi kuvan lataamisen.');
end;

(*****************************************************************************
* ALIOHJELMA:       btnKalibroiClick 1.0
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       5.1.2005
* KUVAUS:           Koordinoi kalibrointia
*****************************************************************************)

procedure TfrmMain.btnKalibroiClick(Sender: TObject);
var
  i : integer;
  beeta, vakio_a, poikkeama : single;
begin
  try
    { Asetetaan alfa-arvo moduuliin. }
    Laske.alfa := Laske.deg2rad(strToFloat(txtAlfa.Text));

    { Etsitään sopivat arvot regressiosuoran kautta }
    Laske.kalibroi(mitat_x, mitat_y);

    { Haetaan tulokset }
    beeta := Laske.beeta;
    vakio_a := Laske.vakio_a;

    { Näytetään käyttäjälle tulokset }
    txtBeeta.Text := FloatToStr(Laske.rad2deg(beeta));
    txtVakio_a.Text := FloatToStr(vakio_a);

    raportoi('TULOKSET:');
    raportoi('Etäisyys'#9\'Kohta'#9'Poikkeama');
    for i:=0 to length(mitat_x)-1 do
    begin
      poikkeama := 1 - (mitat_y[i] / Laske.etaisyys(mitat_x[i]));
      poikkeama := round(poikkeama * 100);
      raportoi(Format('%.2f', [mitat_y[i]]) +#9+
        Format('%.2f', [mitat_x[i]]) +#9+ Format('%.f', [poikkeama])+' %');
    end;
  except
    MessageDlg('Kalibroinnissa tapahtui virhe.', mtError, [mbOk], 0);
  end;
end;
```

```
(*****************************************************************************
* ALIOHJELMA:        FormMouseDown 1.0
* NIMI:              Arno Solin
* PÄIVÄMÄÄRÄ:        5.1.2005
* KUVAUS:            Antaa käyttäjän asettaa piste kohdalleen
*****************************************************************************)

procedure TfrmMain.FormMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  if bManuaali then
  begin
    if (x > 50) and (x < 210) and
       (y > 20) and (y < 140) then
    begin
      caption := intToStr(X-50);
      Piste(X-50, false);
    end;
    bManuaali := false;
    btnKalibroi.Enabled := true;
  end;
end;

(*****************************************************************************
* ALIOHJELMA:        FormMouseDown 1.0
* NIMI:              Arno Solin
* PÄIVÄMÄÄRÄ:        5.1.2005
* KUVAUS:            Esittää käyttäjälle tallennusvalikon
*****************************************************************************)

procedure TfrmMain.btnTallennaClick(Sender: TObject);
var
  strFilename : string;
begin
  { Määritellään oletusarvo nimelle. }
  strFilename := 'Kalibrointi_'+ FormatDateTime('yyyy-mm-dd', Date);
  { Pyydetään käyttäjää valitsemaan tiedostonimi. }
  if PromptForFileName(strFilename,
                       'Teksti (*.txt)|*.txt|Kaikki (*.*)|*.*',
                       'txt',
                       'Tallenna kalibrointitiedosto',
                       GetCurrentDir,
                       true)
  then
    { Tallennetaan valitulla nimellä. }
    Tallenna_I(strFilename);
end;

(*****************************************************************************
* ALIOHJELMA:        cbParChange 1.0
* NIMI:              Arno Solin
* PÄIVÄMÄÄRÄ:        5.1.2005
* KUVAUS:            Vaihtaa rinnakkaisportin osoitetta
*****************************************************************************)

procedure TfrmMain.cbParChange(Sender: TObject);
begin
  { Asetetaan rinnakkaisportin osoite }
  case cbPar.ItemIndex of
    0: Moottori.intRPortti := $3BC;
    1: Moottori.intRPortti := $378;
    2: Moottori.intRPortti := $278;
  end;
  { Portti auki }
  Moottori.AvaaPortti;
end;

end.
```

Mittaohjelma (Sovellus)

```
(*************************************************************************
* OHJELMA:          Mittaohjelma (mittaus 1.0.0)
* KÄÄNTÄJÄ:         Delphi 6
* TARKOITUS:        Mittatiedon kerääminen
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       27.1.2005
* KUVAUS:           Koordinoi mittalaitetta ja kerää informaatiota
* HUOMIOITAVAA:     Tämä ohjelma on osa Arno Solinin päättötyötä.
*                   Tarkoituksena on erityisen mittalaitteen ohjaaminen.
*************************************************************************)

program mittaus;

uses
  Forms,
  Asetukset in 'Asetukset.pas' {frmAsetukset},
  Main      in 'Main.pas'      {frmMain},
  Tarkistus in 'Tarkistus.pas' {frmTarkistus},
  Moottori  in 'Moottori.pas',
  Kamera    in 'Kamera.pas',
  Etsi      in 'Etsi.pas',
  Tiedosto  in 'Tiedosto.pas',
  Type3D    in 'Type3D.pas',
  Laske     in 'Laske.pas';

{$R *.res}

begin
  Application.Initialize;
  Application.Title := 'Mittaohjelma';
  Application.CreateForm(TfrmAsetukset, frmAsetukset);
  Application.CreateForm(TfrmMain, frmMain);
  Application.CreateForm(TfrmTarkistus, frmTarkistus);
  Application.Run;
end.
```

Asetukset

```
(*****************************************************************************
* MODUULI:          Asetukset 1.0 (Mittaohjelma - frmTarkistus)
* KÄÄNTÄJÄ:         Delphi 6
* TARKOITUS:        Asetusten hallinta
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       5.1.2005
* KUVAUS:           Antaa käyttäjän asettaa haluamansa asetukset
* HUOMIOITAVAA:     Lomakkeen moduuli.
*****************************************************************************)

unit Asetukset;

interface

uses
  SysUtils, Forms, Controls, StdCtrls, Classes, Dialogs,
  FileCtrl, { Kansiolistaus }
  Moottori, { Moottorin hallinta }
  Kamera,   { Kameran hallinta }
  Main;     { Mittaus }

type
  TfrmAsetukset = class(TForm)
    { Kehykset }
    pnlProjekti     : TGroupBox;
    pnlMittaus      : TGroupBox;
    pnlKommunikointi: TGroupBox;
    pnlEtaisyys     : TGroupBox;
    { Painonapit }
    btnSelaa        : TButton;
    btnCCW          : TButton;
    btnCW           : TButton;
    btnYlos         : TButton;
    btnAlas         : TButton;
    btnAloita       : TButton;
    btnLataa        : TButton;
    btnExit         : TButton;
    { Tekstikehykset }
    txtNimi         : TEdit;
    txtPolku        : TEdit;
    txtAlfa         : TEdit;
    txtBeeta        : TEdit;
    txtVakio_a      : TEdit;
    { Tekstit }
    lblAlfa         : TLabel;
    lblBeeta        : TLabel;
    lblVakio_a      : TLabel;
    lblSarjap       : TLabel;
    lblRinnakkaisp  : TLabel;
    lblKohta        : TLabel;
    lblPysty        : TLabel;
    lblVaaka        : TLabel;
    lblNimi         : TLabel;
    lblPolku        : TLabel;
    { Pudotusvalikot }
    cbVaaka         : TComboBox;
    cbPysty         : TComboBox;
    cbSarjap        : TComboBox;
    cbRinnakkaisp   : TComboBox;

    { Ohjelman aloitus }
    procedure FormCreate(Sender: TObject);
    { Poistu ohjelmasta }
    procedure btnExitClick(Sender: TObject);
    { Aloita mittaus }
    procedure btnAloitaClick(Sender: TObject);
    { Lataa asetukset }
```

```
    procedure btnLataaClick(Sender: TObject);
    { Selaa kansioita }
    procedure btnSelaaClick(Sender: TObject);
    { Laitteen ohjaus oikeaan kohtaan }
    procedure btnYlosClick(Sender: TObject);
    { Rinnakkaisportin vaihto }
    procedure cbRinnakkaispChange(Sender: TObject);
    { Sarjaportin vaihto }
    procedure cbSarjapChange(Sender: TObject);
  end;
var
  { Lomake }
  frmAsetukset: TfrmAsetukset;

implementation

{$R *.dfm}

(*****************************************************************************
* FUNKTIO:          Maarita 1.0
* TARKOITUS:        Ladattujen arvojen antaminen oikeille muuttujille
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       5.1.2005
* KUVAUS:           Antaa arvon keskeisille muuttujille
* PARAMETRIT:       strInput : Tiedostosta ladattu rivi
*                   intSuure : Muuttujaan viittaava luku
* PALUUARVO:        Tosi, jos arvo annettiin
*                   Epätosi, jos todettiin kommentiksi
* HUOMIOITAVAA:     Tarpeen ainoastaan tiedoston alkuosan avaamisessa.
*****************************************************************************)

function Maarita(strInput: string; intSuure: byte): boolean;
begin
  { Poistetaan ylimääräiset välimerkit ym. }
  strInput := Trim(strInput);

  { Tunnistetaan, jos rivi on kommentti }
  if strInput = '' then result := false
  else if strInput[1] = '#' then result := false
  else
  begin
    result := true;
    { Annetaan arvo oikealle muuttujalle }
    case intSuure of
    { 0: frmAsetukset.txtNimi.Text := strInput; }
      1: frmAsetukset.txtAlfa.Text := strInput;
      2: frmAsetukset.txtBeeta.Text := strInput;
      3: frmAsetukset.txtVakio_a.Text := strInput;
      4: frmAsetukset.cbSarjap.ItemIndex := strToInt(strInput)-1;
      5: frmAsetukset.cbRinnakkaisp.ItemIndex := strToInt(strInput);
      6: frmAsetukset.cbPysty.Text := strInput;
      7: frmAsetukset.cbVaaka.Text := strInput;
    end;
  end;
end;

(*****************************************************************************
* FUNKTIO:          Avaa_I 1.0
* TARKOITUS:        Avata esitallennetut arvot käyttäjälle
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       5.1.2005
* KUVAUS:           Antaa arvon keskeisille muuttujille
* PARAMETRIT:       strFilename : Tiedoston nimi
* HUOMIOITAVAA:     Käsittelee ainoastaan tiedoston alkuosaa.
*****************************************************************************)

procedure Avaa_I(strFilename: string);
var
  Tiedosto : TextFile;
```

```
  strTmp   : string;
  intSuure : integer;
begin

  { Avataan tiedosto }
  AssignFile(Tiedosto, strFilename);
  Reset(Tiedosto);
  intSuure := 0;

  { Toistetaan, kunnes ollaan ladattu tiedoston alkuosa }
  while not Eof(Tiedosto) do
  begin
    { Luetaan rivi tiedostosta }
    Readln(Tiedosto, strTmp);

    { Asetetaan arvo oikealle muuttujalle }
    if Maarita(strTmp, intSuure) then inc(intSuure);

    { Poistutaan silmukasta, kun ollaan valmiita }
    if intSuure = 8 then break;
  end;

  { Päivitetään porttitiedot }
  frmasetukset.cbRinnakkaispChange(frmAsetukset.cbRinnakkaisp);
  frmasetukset.cbSarjapChange(frmAsetukset.cbSarjap);

  { Suljetaan tiedosto }
  CloseFile(Tiedosto);

  { Ilmoitetaan, jos ei ladattu kaikkia arvoja }
  if intSuure <> 8 then ShowMessage('Tiedoston rakenne virheellinen.');
end;

{                   TAPAHTUMAKÄSITTELY (frmAsetukset) }

(*****************************************************************************
* ALIOHJELMA:       btnFormCreate 1.0
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       5.1.2005
* KUVAUS:           Suoritetaan, kun sovellus aloitetaan
*****************************************************************************)

procedure TfrmAsetukset.FormCreate(Sender: TObject);
begin
  { Asetetaan oletusarvot }
  txtNimi.Text :=    'Mittaus_'+ FormatDateTime('yyyy-mm-dd', Date);
  txtPolku.Text :=   GetCurrentDir;
  txtAlfa.Text :=    '24.7273';
  txtBeeta.Text :=   '0.00';
  txtVakio_a.Text := '0.00';

  { Avataan rinnakkaisportti }
  Moottori.AvaaPortti;
end;

(*****************************************************************************
* ALIOHJELMA:       btnExitClick 1.0
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       5.1.2005
* KUVAUS:           Sulkee sovelluksen
* HUOMIOITAVAA:     Varmistaa kuvan lataamisen lopettamisen
*****************************************************************************)

procedure TfrmAsetukset.btnExitClick(Sender: TObject);
begin
  { Suljetaan sovellus }
  Kamera.bToimi := false;
  Application.Terminate;
end;
```

```
(*****************************************************************************
* ALIOHJELMA:       btnAloitaClick 1.1
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       6.2.2005
* KUVAUS:           Aloittaa mittauksen; kutsuu toista lomaketta
*****************************************************************************)

procedure TfrmAsetukset.btnAloitaClick(Sender: TObject);
var
  n : integer;
begin
  { Asetetaan määritellyt arvot paikalleen. }
  try
    { Mittauksen muuttujien nollaus ja asetus }
    n := strToInt(cbPysty.Text);
    if (n < 2) or (n > 57) then Abort;
    Main.Pysty := n-1;

    n := strToInt(cbVaaka.Text);
    if (n < 1) or (n > 257) then Abort;
    Main.Vaaka := n;

    { Luodaan projektille kansio }
    chdir(txtPolku.Text);
    if DirectoryExists(txtNimi.Text) then
      if MessageDlg('Olet kirjoittamassa tietoa jo olemassaolevan projektin '
                    +'päälle. Haluatko jatkaa?',
                    mtWarning,
                    [mbYes,mbNo], 0) = mrNo then Exit;
    { Luodaan kansio }
    mkdir(txtNimi.Text);
    chdir(txtNimi.Text);

    { Muuttujien ja ulkoasun asettaminen }
    Main.H := 0;
    Main.V := 0;
      { Määritellään taulukon koko }
    SetLength(Main.mitat_x, Main.Vaaka*(Main.Pysty+1));
    SetLength(Main.Kuvat,Length(Main.mitat_x));
    Main.bPause := true;

    frmMain.Show;
    frmMain.btnAloita.Caption := 'Aloita';
    frmMain.btnAloita.SetFocus;
    frmMain.Memo1.Text := '';

    { Asetukset piiloon }
    frmAsetukset.Hide;
  except
    MessageDlg('Olet antanut vääränlaisia arvoja tai jättänyt kenttiä tyhjiksi',
               mtWarning, [mbOk], 0);
  end;
end;

(*****************************************************************************
* ALIOHJELMA:       btnLataaClick 1.0
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       5.1.2005
* KUVAUS:           Lataa asetukset vanhasta tiedostosta
*****************************************************************************)

procedure TfrmAsetukset.btnLataaClick(Sender: TObject);
var
  strFilename : string;
begin
  { Pyydetään käyttäjää valitsemaan tiedostonimi. }
  if PromptForFileName(strFilename,
                       'Teksti (*.txt)|*.txt|Kaikki (*.*)|*.*',
```

```
                        '',
                        'Avaa kalibrointitiedosto',
                        GetCurrentDir,
                        false)
  then
    { Jos tiedosto on olemassa, avataan se. }
    if FileExists(strFilename) then
      Avaa_I(strFilename)
    else
      ShowMessage('Valitsemaasi tiedostoa ei ole olemassa.');
end;

(*****************************************************************************
* ALIOHJELMA:       btnSelaaClick 1.0
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       5.1.2005
* KUVAUS:           Antaa käyttäjän valita sopiva tallennuspolku
*****************************************************************************)

procedure TfrmAsetukset.btnSelaaClick(Sender: TObject);
var
  strDirectory : string;
begin
  { Pyydetään käyttäjää valitsemaan hakemisto. }
  if SelectDirectory('Valitse hakemisto projektillesi', '', strDirectory) then
  begin
    txtPolku.Text := strDirectory;
  end;
end;

(*****************************************************************************
* ALIOHJELMA:       btnYlosClick 1.0
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       5.1.2005
* KUVAUS:           Antaa käyttäjän siirtää laitetta manuaalisesti
* HUOMIOITAVAA:     Tätä aliohjelmaa kutsutaan neljän eri napin kautta.
*                   Aliohjelma suorittaa eri toiminnon riippuen kusuvan
*                   napin nimestä.
*****************************************************************************)

procedure TfrmAsetukset.btnYlosClick(Sender: TObject);
begin
  { Vain painonapit voivat kutsua aliohjelmaa }
  if not (Sender is TButton) then exit;

  { Liikutaan yksi askel ylös }
  if (Sender as TButton).Name = 'btnYlos' then
    Moottori.Liiku(1,0);
  { Liikutaan yksi askel alas }
  if (Sender as TButton).Name = 'btnAlas' then
    Moottori.Liiku(-1,0);
  { Liikutaan yksi askel myötäpäivään}
  if (Sender as TButton).Name = 'btnCW' then
    Moottori.Liiku(0,1);
  { Liikutaan yksi askel vastapäivään }
  if (Sender as TButton).Name = 'btnCCW' then
    Moottori.Liiku(0,-1);
end;

(*****************************************************************************
* ALIOHJELMA:       cbRinnakkaispChange 1.0
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       5.1.2005
* KUVAUS:           Muuttaa rinnakkaisportin osoitetta.
* HUOMIOITAVAA:     Portti avataan, mutta sulkemisesta ei huolehdita.
*****************************************************************************)

procedure TfrmAsetukset.cbRinnakkaispChange(Sender: TObject);
begin
```

```
  { Asetetaan rinnakkaisportin osoite }
  case cbRinnakkaisp.ItemIndex of
    0: Moottori.intRPortti := $3BC;
    1: Moottori.intRPortti := $378;
    2: Moottori.intRPortti := $278;
  end;
  { Portti auki }
  Moottori.AvaaPortti;
end;

(*****************************************************************************
* ALIOHJELMA:       cbSarjapChange 1.0
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       5.1.2005
* KUVAUS:           Muuttaa sarjaportin osoitetta.
*****************************************************************************)

procedure TfrmAsetukset.cbSarjapChange(Sender: TObject);
begin
  { Asetetaan sarjaportin osoite }
  Kamera.CommPort1.Open := false;
  case cbSarjap.ItemIndex of
    0: Kamera.CommPort1.ComNumber := 1;
    1: Kamera.CommPort1.ComNumber := 2;
  end;
  Kamera.CommPort1.Open := true;
end;

end.
```

Main

```
(*****************************************************************************
* MODUULI:          Main 1.1 (Mittaohjelma - frmMain)
* KÄÄNTÄJÄ:         Delphi 6
* TARKOITUS:        Mittauksen koordinoiminen
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       27.2.2005
* KUVAUS:           Huolehtii laitteen ohjauksesta
* HUOMIOITAVAA:     Lomakkeen moduuli.
*****************************************************************************)

unit Main;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls,
  Moottori, { Moottorin hallinta }
  Kamera,   { kameran hallinta }
  Etsi;     { Pisteen etsiminen }

type
  TfrmMain = class(TForm)
    imgKuva     : TImage;    { Näyttöruutu }
    btnTakaisin : TButton;   { Takaisin }
    btnAloita   : TButton;   { Aloitus / Keskeytys }
    Memo1       : TMemo;
    procedure btnTakaisinClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btnAloitaClick(Sender: TObject);
  end;

var
  frmMain         : TfrmMain;
  bPause          : boolean;             { Jarru }
  DeltaV, DeltaH,
  H, V,
  Pysty, Vaaka    : integer;             { Mittauksessa käytettäviä arvoja }
  mitat_x         : array of integer;    { Tuloksia }
  Kuvat           : array of string[10]; { Kuvien nimet }
implementation

uses Asetukset, Tarkistus; { Linkitetty ristiin }

{$R *.dfm}

{*****************************************************************************
* ALIOHJELMA:       Piirra 1.1
* TARKOITUS:        Grafinen raportointi
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       27.2.2005
* KUVAUS:           Piirtää käyttäjälle tilanteen näkyviin
* HUOMIOITAVAA:     Aliohjelmalla on omat funktiot astemuunnoksille.
*****************************************************************************}

procedure Piirra;
var
  i, x, y : integer;
  { Askeleet radiaaneiksi }
  function StepsToRad(intSteps : integer): single;
  begin
    result := intSteps / 257 * 2 * 3.141592654;
  end;
  { Askeleet asteiksi }
  function StepsToDeg(intSteps : integer): string;
  begin
    result := intToStr(round(intSteps * 1.4));
```

```
  end;
begin
  with frmMain.imgKuva do
  begin
  { Tyhjennys }
  Canvas.Brush.Color := frmMain.Color;
  Canvas.FillRect(ClientRect);
  { Pystykulma }
    { Vaihtoehtoiset osoitinsuunnat, pysty }
    canvas.Pen.Color := RGB(128,128,255);
    canvas.Pen.Width := 2;
    for i := 0 to Pysty do
    begin
      canvas.MoveTo(300,130);
      x := 300 + round(100* sin(StepsToRad(DeltaV*i)+1.5708));
      y := 130 + round(100* cos(StepsToRad(DeltaV*i)+1.5708));
      canvas.LineTo(X,Y);
    end;

    { Poistetaan keskiviivat }
    canvas.Pen.Color := canvas.Brush.Color;
    canvas.Ellipse(205,35,395,225);

    canvas.Pen.Color := RGB(128,128,255);
    canvas.Pen.Width := 2;
    { Ala- ja yläviiva }
      canvas.MoveTo(400,130);
      canvas.LineTo(300,130);
      x := 300 + round(100* sin(StepsToRad(DeltaV*Pysty)+1.5708));
      y := 130 + round(100* cos(StepsToRad(DeltaV*Pysty)+1.5708));
      canvas.LineTo(X,Y);

    { Pystykohta, jossa ollaan }
    canvas.Pen.Color := clRed;
      canvas.MoveTo(300,130);
      x := 300 + round(100* sin(StepsToRad(V)+1.5708));
      y := 130 + round(100* cos(StepsToRad(V)+1.5708));
      canvas.LineTo(X,Y);

    { Vaihtoehtoiset osoitinsuunnat, vaaka }
    canvas.Pen.Color := RGB(128,128,255);
    canvas.Pen.Width := 2;
    for i := 0 to Vaaka-1 do
    begin
      canvas.MoveTo(60,80);
      x := 60 + round(55* sin(StepsToRad(DeltaH*i)));
      y := 80 - round(55* cos(StepsToRad(DeltaH*i)));
      canvas.LineTo(X,Y);
    end;

    { Kehä, vaaka }
    canvas.Pen.Width := 2;
    canvas.Ellipse(10,30,110,130);

    { Vaakakohta, jossa ollaan }
    canvas.Pen.Width := 2;
    canvas.Pen.Color := clRed;
      canvas.MoveTo(60,80);
      x := 60 + round(55* sin(StepsToRad(H)));
      y := 80 - round(55* cos(StepsToRad(H)));
      canvas.LineTo(X,Y);

    { Tekstit }
    Canvas.TextOut(30,10,StepsToDeg(H) + #176 + ' (' + intToStr(H) + ')');
    Canvas.TextOut(330,10,StepsToDeg(V) + #176 + ' (' + intToStr(V) + ')');
    try
      Canvas.Draw(130, 25, Etsi.AvaaJPEG(
        Format('%.3d',[H]) +'_'+ Format('%.2d',[V-1]) +'.jpg'));
    except
```

```
      Canvas.TextOut(180, 80, '[Ei kuvaa.]');
    end;
  end;
end;

{*****************************************************************************
* ALIOHJELMA:       Ulos 1.0
* TARKOITUS:        Raportortointi
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       5.1.2005
* KUVAUS:           Kirjoittaa lokitiedon näkyviin käyttäjälle
*****************************************************************************}

procedure Ulos(strUlos: string);
begin
  frmMain.memo1.text := frmMain.memo1.text + #13#10 + strUlos;

  { Siirrytään alaspäin }
  frmMain.Memo1.SelStart := Length(frmMain.Memo1.Text);
  SendMessage(frmMain.Memo1.Handle, EM_SCROLLCARET, 0,0);
end;

{*****************************************************************************
* ALIOHJELMA:       Mittaa 1.1
* TARKOITUS:        Hoitaa mittauksen
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       27.2.2005
* KUVAUS:           Liikuttaa, kuvaa ja laskee
* HUOMIOITAVAA:     Silmukasta voi poistua ja mittaukseen voi palata kutsumalla
*                   tätä aliohjelmaa uudelleen.
*****************************************************************************}

procedure Mittaa;
var
  strFilename : string;
  x, n : integer;
begin
  { Lasketaan delta-arvot }
  DeltaV := 56 div Pysty;
  DeltaH := 257 div Vaaka;

  ulos('Aloitetaan:');

  { Kääntyminen vaakatasossa }
  repeat
    { Keskeytetään, jos käyttäjä haluaa }
    if bPause then Exit;

    { Kääntymisen nopeus (millisekunteja); pystysuunta }
    Moottori.intOdota := 20;

    { Kääntyminen pystytasossa }
    repeat
      { Piirretään tilanne näkyviin }
      Piirra;

      { Laser päälle }
      Moottori.Laser(true);

      { Kuvan nimeksi tulee }
      strFilename := Format('%.3d',[H]) +'_'+ Format('%.2d',[V]) +'.jpg';

      { Otetaan kuva }
      try
        Ulos('Otetaan kuva: '+ strFilename);
        Sleep(500); { Odotetaan hetki }
        Kamera.Lataa(strFilename);

        { Näytetään ladattu kuva }
```

```
        frmMain.imgKuva.Canvas.Draw(130, 25, Etsi.AvaaJPEG(strFilename));
      except
        { Lopetetaan kommunikonti kameran kanssa virhetilanteessa }
        Kamera.laheta($FF);
        MessageDlg(Kamera.strLoki+'[Keskeytetty]', mtInformation, [mbOk], 0);
        Exit;
      end;

      { Laser pois päältä }
      Moottori.Laser(false);

      { Etsitään piste }
      x := Etsi.EtsiPiste(strFilename);

      { Tallennetaan muistiin }
      n := H div deltaH ;
      n := n * (Pysty+1) + V div deltaV;
      mitat_x[n] := x;
      Kuvat[n] := strFilename;
      Ulos(inttostr(n));

      { Ilmoitetaan käyttäjälle }
      Ulos(strFilename);

      Application.ProcessMessages;

      { Lisätään pystymuuttujaan delta-arvo }
      inc(V, DeltaV);

      { Käännetään laitetta ylöspäin, jos tarpeen }
      if V <= (DeltaV * Pysty) then
      begin
        Ulos('Liikutaan ylös kohtaan: '+inttostr(V));
        Moottori.Liiku(DeltaV,0);

        { Keskeytetään, jos käyttäjä haluaa }
        if bPause then Exit;
      end
      else Ulos('Ei liikuta ylös.');

      { Keskeytetään, jos käyttäjä haluaa }
    until V > DeltaV * Pysty;

    { Käännetään laite takaisin alas }
    Ulos('Liikutaan alas: '+inttostr(-deltav*pysty));
    Moottori.Liiku(-DeltaV*Pysty,0);

    { Nollataan  }
    V := 0;

    { Lisätään vaakatason muuttujaan delta-arvo }
    inc(H, DeltaH);

    { Kääntymisen nopeus (millisekunteja); vaakasuunta }
    Moottori.intOdota := 50;

    { Käännetään laitetta myötäpäivään, jos tarpeen }
    if H < DeltaH * Vaaka then
    begin
      Ulos('Liikutaan myötäpäivään: '+inttostr(deltah));
      Moottori.Liiku(0,DeltaH);
    end;

  until H >= DeltaH * Vaaka;

  { Siirrytään takaisin alkuperäiseen asentoon vastapäivään }
  Ulos('Liikutaan takaisin: '+inttostr(deltaH-deltaH*Vaaka));
  Moottori.Liiku(0, deltaH-DeltaH*Vaaka);
```

```
  { Mittaus valmis. Siirrytään varmentamaan tuloksia. }
  SetLength(Etaisyys, Length(mitat_x)); { Muuttujan koko }
  frmMain.Hide;
  frmTarkistus.show;
end;

{                   TAPAHTUMAKÄSITTELY (frmMain) }

(*****************************************************************************
* ALIOHJELMA:       btnTakaisinClick 1.1
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       26.2.2005
* KUVAUS:           Palaaminen takaisin asetuksiin
*****************************************************************************)

procedure TfrmMain.btnTakaisinClick(Sender: TObject);
begin
  { Keskeytetään }
  bPause := true;
  { Lopetetaan keskeneräisen kuvan lataus }
  Kamera.bToimi := false;
  { Laser pois päältä }
  Moottori.Laser(false);
  { Lomake piiloon }
  frmMain.Hide;
  { Näytetään asetukset }
  frmAsetukset.Show;
end;

(*****************************************************************************
* ALIOHJELMA:       FormClose 1.1
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       26.2.2005
* KUVAUS:           Sovelluksen sulkeutumisen varmistaminen
*****************************************************************************)

procedure TfrmMain.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  { Keskeytetään }
  bPause := true;
  { Lopetetaan keskeneräisen kuvan lataus }
  Kamera.bToimi := false;
  { Laser pois päältä }
  Moottori.Laser(false);
  { Varmistetaan sovelluksen sammuminen }
  Application.Terminate;
end;

(*****************************************************************************
* ALIOHJELMA:       btnAloitaClick 1.1
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       26.2.2005
* KUVAUS:           Aloittaa / Keskeyttää mittauksen
*****************************************************************************)

procedure TfrmMain.btnAloitaClick(Sender: TObject);
begin
  { Pausea painettu }
  if bPause then
  begin
    bPause := false;
    btnAloita.caption := 'Keskeytä';
    Mittaa;
    btnAloita.Enabled := true;
  end
  else
  begin
    btnAloita.Enabled := false;
    bPause := true;
```

```
    Kamera.bToimi := false;
    btnAloita.caption := 'Jatka';
  end;
end;

end.
```

Tarkistus

```
(*****************************************************************************
* MODUULI:          Tarkistus 1.0 (Mittaohjelma - frmTarkistus)
* KÄÄNTÄJÄ:         Delphi 6
* TARKOITUS:        Tulosten varmentaminen
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       27.2.2005
* KUVAUS:           Koordinoi varmentamista ja tallentamista
* HUOMIOITAVAA:     Lomakkeen moduuli.
*****************************************************************************)

unit Tarkistus;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls,
  Kamera,    { Kameran hallinta }
  Moottori, { Moottorin hallinta }
  Tiedosto, { Tiedostokäsittely }
  Laske,     { Etäisyyden laskeminen }
  Etsi;      { Pisteen etsiminen }

type
  TfrmTarkistus = class(TForm)
    imgKuva      : TImage;
    pnlPaneli    : TPanel;
    cmdSeuraava  : TButton;
    cmdEdellinen : TButton;
    lblNimi      : TLabel;
    lblEtaisyys  : TLabel;
    procedure cmdSeuraavaClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure imgKuvaMouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure cmdEdellinenClick(Sender: TObject);
    procedure FormActivate(Sender: TObject);
  end;

var
  frmTarkistus : TfrmTarkistus;
  Etaisyys     : array of single; { Taulukoidut etäisyydet }
  kohta        : integer = 0;     { Käsiteltävä kuva }

implementation

uses Asetukset, Main; { Linkitetty muihin ohjelman osiin }

{$R *.dfm}

{*****************************************************************************
* ALIOHJELMA:       Tallenna 1.1
* TARKOITUS:        Lähettää tietoa tallennettavksi
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       26.2.2005
* KUVAUS:           Kokoaa tiedot yhteen ja talletuttaa ne lopuksi
*****************************************************************************}

procedure Tallenna;
var
  strNimi             : string;
  Sarjap, Rinnakkaisp : integer;
begin
  { Muuttujien arvot }
  strNimi := frmAsetukset.txtNimi.Text;
  Sarjap :=  Kamera.CommPort1.ComNumber;
  Rinnakkaisp := frmAsetukset.cbRinnakkaisp.ItemIndex;
```

```
  { Tallennetaan tulokset }
  Tiedosto.Tallenna('tulokset.txt', Etaisyys, Pysty, Vaaka, Sarjap,
                    Rinnakkaisp, strNimi, rad2deg(alfa), rad2deg(beeta), vakio_a)
;

  { Kerrotaan, että ohjelman suoritus on lopuillaan. }
  MessageDlg('Mittaus on päättynyt. '+ #13#10 +
             'Paina OK lopettaaksesi ohjelman suorituksen.',
             mtInformation, [mbOK], 0);

  { Ollaan valmiita. Poistutaan. }
  Application.Terminate;
  Abort; { Lopetetaan oikeasti tähän näin. }
end;

{*****************************************************************************
* ALIOHJELMA:       Siirry 1.1
* TARKOITUS:        Esittää seuraava mittapiste käyttäjälle
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       28.11.2004
* KUVAUS:           Näyttää käyttäjälle seuraavan pisteen
* HUOMIOITAVAA:     Vaatii, että muuttujassa Kuvat on arvoja.
*****************************************************************************}

procedure Siirry;
begin
  { Ollaanko lopussa }
  if kohta = Length(etaisyys) then Tallenna;

  { Laske etaisyys, jos tulos saatu }
  if mitat_x[kohta] < 160 then
    Etaisyys[kohta] := Laske.Etaisyys(mitat_x[kohta]/159)
  else
    Etaisyys[kohta] := 0;

  { Näytetään käyttäjälle }
  frmTarkistus.lblEtaisyys.Caption := Format('Etäisyys: %.f cm',
                                             [Etaisyys[kohta]]);
  { Avaa kuva }
  frmTarkistus.imgKuva.Canvas.Draw(0,0,Etsi.AvaaJPEG(Kuvat[kohta],
                                                     abs(mitat_x[kohta])));
  frmTarkistus.lblNimi.Caption := Kuvat[kohta];
end;

{                   TAPAHTUMAKÄSITTELY (frmTarkistus) }

(*****************************************************************************
* ALIOHJELMA:       cmdSeuraavaClick 1.1
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       26.2.2005
* KUVAUS:           Siirtyy seuraavaan pisteeseen
*****************************************************************************)

procedure TfrmTarkistus.cmdSeuraavaClick(Sender: TObject);
begin
  { Seuraava kohta }
  inc(kohta);
  Siirry;
end;

(*****************************************************************************
* ALIOHJELMA:       cmdEdellinenClick 1.1
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       26.2.2005
* KUVAUS:           Siirtyy edelliseen pisteeseen
*****************************************************************************)

procedure TfrmTarkistus.cmdEdellinenClick(Sender: TObject);
```

```
begin
  if kohta > 0 then dec(kohta);
  Siirry;
end;

(*****************************************************************************
* ALIOHJELMA:       imgKuvaMouseDown 1.1
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       26.2.2005
* KUVAUS:           Laskee uuden tuloksen
*****************************************************************************)

procedure TfrmTarkistus.imgKuvaMouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var
  d : single;
begin
  { Lasketaan etäisyys kohdassa X }
  d := Laske.Etaisyys(X/159);

  { Näytetään käyttäjälle }
  lblEtaisyys.Caption := Format('Etäisyys: %.f cm', [d]);

  { Tallennetaan muistiin }
  Etaisyys[kohta] := d;
end;

(*****************************************************************************
* ALIOHJELMA:       FormClose 1.1
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       26.2.2005
* KUVAUS:           Varoittaa sulkemasta sovellusta
*****************************************************************************)

procedure TfrmTarkistus.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  { Varoitetaan käyttäjää sammuttamasta ohjelmaa }
  if MessageDlg('Olet sulkemassa mittaohjelmaa. '+ #13#10 +
                'Tämä johtaa tulosten tuhoutumiseen.',
                mtInformation, mbOKCancel, 0) = mrCancel then Abort;

  { Varmistetaan sulkeutuminen }
  Application.Terminate;
end;

procedure TfrmTarkistus.FormActivate(Sender: TObject);
begin
  { Asetetaan mittausvakiot paikalleen }
  alfa    := deg2rad(StrToFloat(frmAsetukset.txtAlfa.Text));
  beeta   := deg2rad(StrToFloat(frmAsetukset.txtBeeta.Text));
  vakio_a := StrToFloat(frmAsetukset.txtVakio_a.Text);

  Siirry;
end;

end.
```

Optimointi (Sovellus)

```
(*************************************************************************
* OHJELMA:          Optimointi (optimointi 1.0.0)
* KÄÄNTÄJÄ:         Delphi 6
* TARKOITUS:        Mittatulosten käsittely
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       27.1.2005
* KUVAUS:           Muokkaa tuloksia ja tallentaa ne VRML-muotoon
* HUOMIOITAVAA:     Tulosten muokkaus lähinnä nimellistä. Tärkein tarkoitus on
*                   muunnos VRML-muotoon.
*************************************************************************)

program Optimointi;

uses
  Forms,
  Main     in 'Main.pas' {frmMain},
  optimoi  in 'optimoi.pas',
  Type3D   in 'Type3D.pas',
  Tiedosto in 'Tiedosto.pas',
  VRML     in 'VRML.pas';

{$R *.res}

begin
  Application.Initialize;
  Application.Title := 'Optimointi';
  Application.CreateForm(TfrmMain, frmMain);
  Application.Run;
end.
```

Main

```
(*****************************************************************************
* MODUULI:          Main 1.0 (Optimointi - frmMain)
* KÄÄNTÄJÄ:         Delphi 6
* TARKOITUS:        Mittaustulosten muokkaus
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       5.1.2005
* KUVAUS:           Muuttaa tulokset VRML-muotoon ja tarvittaessa muokkaa niitä
* HUOMIOITAVAA:     Lomakkeen moduuli.
*****************************************************************************)

unit Main;

interface

uses
  Windows, SysUtils, Forms, Controls, Classes, Graphics,
  Dialogs, ExtCtrls, StdCtrls,
  Optimoi,  { Tulosten sovitus }
  Type3D,   { Kolmiulotteisten koordinaattien käsittely }
  Tiedosto, { Tiedoston avaaminen }
  VRML;     { Tulosten tallentaminen }

type
  TfrmMain = class(TForm)
    imgKuva     : TImage;  { Kuvakehys }
    btnAvaa     : TButton; { Painonapit }
    btnTallenna : TButton;
    btnSovita   : TButton;
    lbl2        : TLabel;  { Ohjeet }
    lbl3        : TLabel;
    lbl1        : TLabel;
    lbl4        : TLabel;
    procedure imgKuvaMouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure imgKuvaMouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure btnAvaaClick(Sender: TObject);
    procedure btnSovitaClick(Sender: TObject);
    procedure FormMouseWheel(Sender: TObject; Shift: TShiftState;
      WheelDelta: Integer; MousePos: TPoint; var Handled: Boolean);
    procedure btnTallennaClick(Sender: TObject);
  end;
var
  frmMain      : TfrmMain;
  X0,Y0, Pysty : integer;            { Hiiren lähtökoordinaatit }
  Piste        : TKoordinaattiArray; { Käsiteltävät pisteet }
  Polygoni     : TPolygoniArray;     { Käsiteltävät polygonit }
  suhde        : single = 1;         { Zoomin suuruus (oletus 1 = 1 px/cm) }

implementation

{$R *.dfm}

{*****************************************************************************
* FUNKTIO:          LuoPolygoni 1.1
* TARKOITUS:        Muuttujatyypin muuttaminen
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       27.2.2005
* KUVAUS:           Luo annetuista arvoista polygonin
* PARAMETRIT:       a,b,c,d : kulmapisteiden arvot
* PALUUARVO:        Palauttaa arvon muotoa TPolygoni.
*****************************************************************************}

function LuoPolygoni(a, b, c, d: integer): TPolygoni;
begin
  result.a := a; result.b := b;
  result.c := c; result.d := d;
```

```
end;

{*****************************************************************************
* ALIOHJELMA:       Piirra 1.1 (overload)
* TARKOITUS:        Piirtää halutun kuvion näkyviin
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       28.11.2004
* KUVAUS:           Luo annetuista arvoista ympyrän
* PARAMETRIT:       X, Y : koordinaatti
*                   r    : säde
* HUOMIOITAVAA:     Parametriarvoista riippuen kutsutaan eri aliohjelmaa
*****************************************************************************}

procedure Piirra(X,Y,r: single); overload;
begin
  frmMain.imgKuva.Canvas.Ellipse(200+trunc(X*suhde-r),
    200-trunc(Y*suhde-r),200+trunc(X*suhde+r),200-trunc(Y*suhde+r));
end;

{*****************************************************************************
* ALIOHJELMA:       Piirra 1.1 (overload)
* TARKOITUS:        Piirtää halutun kuvion näkyviin
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       28.11.2004
* KUVAUS:           Luo annetuista arvoista janan
* PARAMETRIT:       X1, Y1 : alkupiste
*                   X2, Y2 : loppupiste
* HUOMIOITAVAA:     Parametriarvoista riippuen kutsutaan eri aliohjelmaa
*****************************************************************************}

procedure Piirra(X1,Y1,X2,Y2: single); overload;
begin
  frmMain.imgKuva.Canvas.MoveTo(200+trunc(X1*suhde),200-trunc(Y1*suhde));
  frmMain.imgKuva.Canvas.LineTo(200+trunc(X2*suhde),200-trunc(Y2*suhde));
end;

{*****************************************************************************
* ALIOHJELMA:       Paivita 1.1
* TARKOITUS:        Piirtää pisteet näkyviin
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       28.11.2004
* KUVAUS:           Päivittää kuvan ajan tasalle
*****************************************************************************}

procedure Paivita;
var
  i : integer;
begin
  if Length(piste) = 0 then Exit;

  with frmMain.imgKuva.Canvas do
  begin
    { Vanhat pois }
    FillRect(frmMain.imgKuva.ClientRect);
    Brush.Color := RGB(0,0,255);
    MoveTo(200+trunc(Piste[0].x*suhde),200-trunc(Piste[0].y*suhde));
    i:=0;
    while i < Length(Piste) do
    begin
      LineTo(200+trunc(Piste[i].x*suhde),200-trunc(Piste[i].y*suhde));
      Piirra(Piste[i].x, Piste[i].y, 4);
      inc(i, Pysty);
    end;
    LineTo(200+trunc(Piste[0].x*suhde),200-trunc(Piste[0].y*suhde));

    { Piirretään mittasuhdetta esittävä sininen palkki }
    TextRect(Rect(10,10,trunc(100*suhde+0.5),23), 10,10,'100 cm');

    { Palautetaan väri entiselleen (valkoinen) }
```

```
    Brush.Color := RGB(255,255,255);
  end;
end;

{                   TAPAHTUMAKÄSITTELY (frmMain) }

(*****************************************************************************
* ALIOHJELMA:       imgKuvaMouseDown 1.0
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       25.2.2005
* KUVAUS:           Lähtökoordinaattien asettaminen
*****************************************************************************)

procedure TfrmMain.imgKuvaMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  X0 := X-200; Y0 := 200-Y;
end;

(*****************************************************************************
* ALIOHJELMA:       imgKuvaMouseUp 1.0
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       25.2.2005
* KUVAUS:           Pisteen siirtäminen
*****************************************************************************)

procedure TfrmMain.imgKuvaMouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
var
  i, etaisyys, pieninE, n : integer;
begin
  if Length(piste) = 0 then Exit;

  { Etsitään siirrettävä piste }
  i := 0; n := 0; pieninE := 1000;
  while i < Length(piste) do
  begin
    etaisyys := Trunc(sqrt(sqr(X0-piste[i].x*suhde)
                         + sqr(Y0-piste[i].y*suhde)));
    if etaisyys < pieninE then
    begin
      pieninE := etaisyys;
      n := i;
    end;
    inc(i,Pysty);
  end;

  { Siirretään pistettä }
  caption := Format('Piste %.d valittu.', [n]);

  Piste[n].x := Piste[n].x + (X-200 -X0)/suhde;
  Piste[n].y := Piste[n].y + (200-Y -Y0)/suhde;
  Paivita;
end;

(*****************************************************************************
* ALIOHJELMA:       btnAvaaClick 1.0
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       25.2.2005
* KUVAUS:           Tiedoston avausvalikon näyttäminen
*****************************************************************************)

procedure TfrmMain.btnAvaaClick(Sender: TObject);
var
  strFilename : string;
begin
  { Pyydetään käyttäjää valitsemaan avattava tiedosto }
  if PromptForFileName(strFilename,
                       'Mittaustiedosto (*.txt)|*.txt|Kaikki (*.*)|*.*',
```

```
                        '','Avaa tiedosto',
                        GetCurrentDir, false) then
    if FileExists(strFilename) then
    begin
      { Avataan tiedosto ja luetaan pystyarvo }
      Tiedosto.AvaaMittaus(strFilename, Piste, Polygoni);
      Pysty := Tiedosto.Pysty;

      { Piirretään näkyviin }
      Paivita;
    end
    else
      ShowMessage('Valitsemaasi tiedostoa ei ole olemassa.');
end;

(*****************************************************************************
* ALIOHJELMA:       btnTallennaClick 1.0
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       25.2.2005
* KUVAUS:           Tiedoston tallennusvalikon näyttäminen
*****************************************************************************)

procedure TfrmMain.btnTallennaClick(Sender: TObject);
var
  strFilename : string;
begin
  { Pyydetään käyttäjää valitsemaan nimi tiedostolle }
  if PromptForFileName(strFilename,
                        'VRML (*.wrl)|*.wrl|Kaikki (*.*)|*.*',
                        '.wrl','Tallenna tiedosto',
                        GetCurrentDir, true) then
    { Tallennetaan }
    VRML.Tallenna(strFilename, Piste, Polygoni);
end;

(*****************************************************************************
* ALIOHJELMA:       btnTallennaClick 1.0
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       25.2.2005
* KUVAUS:           Sovittaminen ja tiedoston tallennusvalikon näyttäminen
*****************************************************************************)

procedure TfrmMain.btnSovitaClick(Sender: TObject);
var
  i           : integer;
  korkeus     : single;
  PisteTmp    : TKoordinaattiArray;
  PolygoniTmp : TPolygoniArray;
  strFilename : string;
begin
  { Otetaan erilleen lattiatason arvot }
  SetLength(PisteTmp, Length(Piste) div Pysty);
  for i := 0 to Length(PisteTmp)-1 do
    PisteTmp[i] := piste[Pysty*i];

  { Sovitetaan suorakulmioon }
  Optimoi.Suorakulmio(PisteTmp);

  { Piirretään suorakulmio }
  Piirra(PisteTmp[0].X, PisteTmp[0].Y, PisteTmp[1].X, PisteTmp[1].Y);
  Piirra(PisteTmp[1].X, PisteTmp[1].Y, PisteTmp[2].X, PisteTmp[2].Y);
  Piirra(PisteTmp[2].X, PisteTmp[2].Y, PisteTmp[3].X, PisteTmp[3].Y);
  Piirra(PisteTmp[3].X, PisteTmp[3].Y, PisteTmp[0].X, PisteTmp[0].Y);

  { Lasketaan korkeuden keskiarvo }
  korkeus := 0;
  for i := 1 to Length(Piste) div Pysty do
    korkeus := korkeus + Piste[i*Pysty-1].Z;
```

```
  korkeus := korkeus * Pysty / Length(Piste);

  { Luodaan kulmapisteet }
  SetLength(PisteTmp, 8);
  PisteTmp[4] := PisteTmp[0]; PisteTmp[4].Z := korkeus;
  PisteTmp[5] := PisteTmp[1]; PisteTmp[5].Z := korkeus;
  PisteTmp[6] := PisteTmp[2]; PisteTmp[6].Z := korkeus;
  PisteTmp[7] := PisteTmp[3]; PisteTmp[7].Z := korkeus;

  { Luodaan polygonit }
  SetLength(PolygoniTmp, 6);
  PolygoniTmp[0] := LuoPolygoni(0, 1, 2, 3);
  PolygoniTmp[1] := LuoPolygoni(4, 5, 6, 7);
  PolygoniTmp[2] := LuoPolygoni(0, 4, 5, 1);
  PolygoniTmp[3] := LuoPolygoni(1, 5, 6, 2);
  PolygoniTmp[4] := LuoPolygoni(2, 6, 7, 3);
  PolygoniTmp[5] := LuoPolygoni(3, 7, 4, 0);

  { Pyydetään käyttäjää valitsemaan nimi tiedostolle }
  if PromptForFileName(strFilename,
                       'VRML (*.wrl)|*.wrl|Kaikki (*.*)|*.*',
                       '.wrl','Tallenna optimoitu tiedosto',
                       GetCurrentDir, true) then
    { Tallennetaan }
    VRML.Tallenna(strFilename, PisteTmp, PolygoniTmp);
end;

(*****************************************************************************
* ALIOHJELMA:       FormMouseWheel 1.0
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       25.2.2005
* KUVAUS:           Kuvan zoomaus
*****************************************************************************)

procedure TfrmMain.FormMouseWheel(Sender: TObject; Shift: TShiftState;
  WheelDelta: Integer; MousePos: TPoint; var Handled: Boolean);
begin
  suhde := suhde + WheelDelta / 1000;
  Paivita;
end;

end.
```

Ohjelma3D (Sovellus)

```
(*******************************************************************************
* OHJELMA:          Ohjelma3D (ohjelma3d 1.0.0)
* KÄÄNTÄJÄ:         Delphi 6
* TARKOITUS:        Kappaleiden tarkastelu kolmiulotteisesti
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       1.3.2005
* KUVAUS:           Antaa mahdollisuuden tarkastella tallennettua mittatietoa
* HUOMIOITAVAA:     Ohjelma tukee kahdentyyppisiä tiedostoja:
*                     VRML (*.wrl)
*                     Mittaustiedosto (*.txt)
*                   Tiedostojen avaaminen onnistuu ohjelman sisältä tai
*                   komentoriviltä antamalla tiedoston (polku ja) nimi.
*                   Tiedostot voidaan suoraan assosioida avautumaan ohjelmassa.
*******************************************************************************)

program Ohjelma3D;

uses
  Forms,
  Main     in 'Main.pas' {frmMain},
  Type3D   in 'Type3D.pas',
  VRML     in 'VRML.pas',
  Tiedosto in 'Tiedosto.pas';

{$R *.res}

begin
  Application.Initialize;
  Application.Title := '3D-ohjelma';
  Application.CreateForm(TfrmMain, frmMain);
  Application.Run;
end.
```

Main

```
(*****************************************************************************
* MODUULI:          Main 1.1 (Ohjelma3D - frmMain)
* KÄÄNTÄJÄ:         Delphi 6
* TARKOITUS:        Kappaleen esittäminen kolmiulotteisesti
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       1.3.2005
* KUVAUS:           Piirtää mallin ruudulle mitta- tai VRML-tiedostosta
* HUOMIOITAVAA:     Lomakkeen moduuli.
*****************************************************************************)

unit Main;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, StdCtrls,
  Type3D,    { Muuttujatyypit }
  VRML,      { VRML-tiedostojen käsittely }
  Tiedosto; { Mittatiedostojen käsittely }

type
  TfrmMain = class(TForm)
    imgKuva: TImage; { Kuvakehys, johon piirretään }
    procedure imgKuvaMouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure imgKuvaMouseMove(Sender: TObject; Shift: TShiftState; X,
      Y: Integer);
    procedure FormKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure FormResize(Sender: TObject);
    procedure FormMouseWheel(Sender: TObject; Shift: TShiftState;
      WheelDelta: Integer; MousePos: TPoint; var Handled: Boolean);
    procedure FormActivate(Sender: TObject);
  end;

  { 3D-koordinaattien pyhä kolminaisuus kokonaislukumuodossa }
  TintKoordinaatti = record
      X : integer;
      Y : integer;
      Z : integer;
  end;
var
  frmMain: TfrmMain;
  { Data }
  Piste : TKoordinaattiArray;
  Polygoni : TPolygoniArray;

  { Koordinaatit }
  X0,Y0          : integer; { Hiirikoordinaatit }
  origoX, origoY : integer; { Origo }
  Projektio      : array of TintKoordinaatti;  { Piirtokoordinaatit }

  { Laskeminen }
  sinA, sinB, cosA, cosB,    { sinit, kosinit }
  A, B,                      { Katselukulmat: b pysty, a vaaka }
  d,z : single;              { Perspektiivi, d=etäisyys, zoomi }

  { Polygonien piirtäminen oikein }
  jarjestys, lista: array of integer;

  { Valittu näyttömoodi }
  nayta : byte = 1;          { Oletusarvona 1 - Wireframe }
implementation

{$R *.dfm}
```

```
{*****************************************************************************
* ALIOHJELMA:       vaihda 1.1
* TARKOITUS:        Muuttujien vaihtaminen
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       1.3.2004
* KUVAUS:           Vaihtaa kahden muuttujan paikkaa
* PARAMETRIT:       a, b : Vaihdettavien solujen kohdat
* HUOMIOITAVAA:     Vaihtaminen toteutettu epäammattimaisesti ilman xor-swappia
*****************************************************************************}

procedure vaihda(a, b: integer);
var
  tmp : integer;
begin
  tmp := jarjestys[a];
  jarjestys[a] := jarjestys[b];
  jarjestys[b] := tmp;

  tmp := lista[a];
  lista[a] := lista[b];
  lista[b] := tmp;
end;

{*****************************************************************************
* ALIOHJELMA:       qsort 1.1
* TARKOITUS:        Taulukon järjesteleminen
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       1.3.2004
* KUVAUS:           Lajittelee taulukon suuruusjärjestykseen
* PARAMETRIT:       alku  : alkukohta
*                   loppu : loppukohta
* HUOMIOITAVAA:     Toimii rekursiivisesti eli kutsuu itseään.
*****************************************************************************}

procedure qsort(alku, loppu: integer);
var
  l,r,pivot : integer;
begin
  if loppu > alku then
  begin
    pivot := lista[loppu];
    l := alku -1;
    r := loppu;
    while r > l do
    begin
      repeat inc(l) until lista[l] >= pivot;   //=
      repeat dec(r) until lista[r] <= pivot;    //=
      vaihda(l,r);
    end;
    vaihda(l,loppu);
    vaihda(loppu,r);

    qsort(alku, l-1);
    qsort(l+1,loppu);
  end;
end;

{*****************************************************************************
* ALIOHJELMA:       Laske 1.1
* TARKOITUS:        Pisteiden uusien koorinaattien selvittäminen
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       1.3.2004
* KUVAUS:           Laskee alkuperäisistä arvoista projisoitavat arvot
* HUOMIOITAVAA:     Arvoa y käytetään polygonien järjestelemisessä
*****************************************************************************}

procedure Laske;
var
  i : integer;
```

```
  xx,yy,zz, k: single; //muunoksen apuna
begin
  for i := 0 to Length(Piste)-1 do
  begin
    xx := Piste[i].X*cosa + Piste[i].Y*sina;
    yy := (Piste[i].Y*cosa - Piste[i].X*sina)*cosb - Piste[i].Z*sinb;
    zz := (Piste[i].Y*cosa - Piste[i].X*sina)*sinb + Piste[i].Z*cosb;
    k := z / (d + yy); //etäisyydestä riippuva projisointierroin
    Projektio[i].X := round(k*xx + origoX);
    Projektio[i].Z := round(-k*zz + origoY);
    Projektio[i].Y := round(yy);
  end;
end;

{*****************************************************************************
* FUNKTIO:          Vari 0.9
* TARKOITUS:        Polygonien väritys (varjostaminen)
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       1.3.2004
* KUVAUS:           Määrää väriarvon pistetulon avulla
* PARAMETRIT:       n : polygonin numero
* PALUUARVO:        Palauttaa väriarvon 0-255 (punainen)
* HUOMIOITAVAA:     Ei toimi täysin halutulla tavalla. Beetaversio.
*****************************************************************************}

function Vari2(n: integer): integer;
var
  X1, Y1, Z1,
  X2, Y2, Z2, tmp: single;
begin
  { Lasketaan vektorien pistetulon avulla kunkin polygonin väri.
    Välinen kulma yhtälöstä a.b = |a||b|cos(a,b) }
  X1 := Projektio[Polygoni[n].a].X - origoX;
  Z1 := Projektio[Polygoni[n].a].Z - origoY;
  Y1 := Projektio[Polygoni[n].a].Y;

  X2 := Projektio[Polygoni[n].c].X - origoX;
  Z2 := Projektio[Polygoni[n].c].Z - origoY;
  Y2 := Projektio[Polygoni[n].c].Y;

  try
    tmp := ((X2-X1)*0+(Y2-Y1)*1+(Z2-Z1)*0)/
           (sqrt(sqr(X2-X1)+sqr(Y2-Y1)
+sqr(Z2-Z1))*sqrt(sqr(0)+sqr(1)+sqr(0)));              { = sqrt(vX^2+vY^2+vZ^2) }

    { Arccos tmp = }
  tmp := ArcTan(-tmp / sqrt(-tmp*tmp+1)) + 1.570796;
    { Väri }
    result := trunc(tmp * 81.169);
    result := result;
  except
    result := 255;
  end;
end;

{*****************************************************************************
* ALIOHJELMA:       Polygon 1.1
* TARKOITUS:        Polygonien piirtäminen
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       1.3.2004
* KUVAUS:           Piirtää värillä täytettyjä monikulmioita näkyviin
* HUOMIOITAVAA:     Ei tarpeen wireframen ollessa käytössä.
*****************************************************************************}

procedure Polygon;
var
  n,i : integer;
begin
  for i := 0 to Length(Polygoni)-1 do
```

```
  begin
    jarjestys[i] := i;
    n := Projektio[Polygoni[i].a].Y + Projektio[Polygoni[i].b].Y
       + Projektio[Polygoni[i].c].Y + Projektio[Polygoni[i].d].Y;
    lista[i] := n;
  end;
  qsort(0, Length(Polygoni)-1);

  if nayta and 1 = 0 then       { Ei ääriviivoja (Wireframe) esillä }
    frmMain.imgKuva.Canvas.Pen.Style := psClear;

  if nayta and 4 = 0 then       { Polgonien väri (punainen) }
    frmMain.imgKuva.Canvas.Brush.Color := RGB(255,0,0);

  for n:= Length(Polygoni)-1 downto 0 do
  begin
    i := jarjestys[n];
    { Polygonin väritys }
    if nayta and 4 = 4 then
      frmMain.imgKuva.Canvas.Brush.Color := vari2(i);

    frmMain.imgKuva.Canvas.Polygon([
      Point(Projektio[Polygoni[i].a].X, Projektio[Polygoni[i].a].Z),
      Point(Projektio[Polygoni[i].b].X, Projektio[Polygoni[i].b].Z),
      Point(Projektio[Polygoni[i].c].X, Projektio[Polygoni[i].c].Z),
      Point(Projektio[Polygoni[i].d].X, Projektio[Polygoni[i].d].Z)
    ]);
  end;
  frmMain.imgKuva.Canvas.Brush.Color := RGB(255,255,255);
  frmMain.imgKuva.Canvas.Pen.Style := psSolid;
end;

{*****************************************************************************
* ALIOHJELMA:       Wireframe 1.1
* TARKOITUS:        Rautalankamallin (Wireframe) piirtäminen
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       1.3.2004
* KUVAUS:           Piirtää monikulmioiden ääriviivat näkyviin
* HUOMIOITAVAA:     Ei käytössä polygonimoodissa.
*****************************************************************************}

procedure Wireframe;
var
  i : integer;
begin
  for i:= 0 to Length(Polygoni)-1 do
    frmMain.imgKuva.Canvas.Polyline([
      Point(Projektio[Polygoni[i].a].X , Projektio[Polygoni[i].a].Z),
      Point(Projektio[Polygoni[i].b].X , Projektio[Polygoni[i].b].Z),
      Point(Projektio[Polygoni[i].c].X , Projektio[Polygoni[i].c].Z),
      Point(Projektio[Polygoni[i].d].X , Projektio[Polygoni[i].d].Z)
    ]);
end;

{*****************************************************************************
* ALIOHJELMA:       Piirrä 1.1
* TARKOITUS:        Koordinoi piirtämistä
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       1.3.2004
* KUVAUS:           Kutsuu haluttuja piirtofunktioita
*****************************************************************************}

procedure Piirra;
begin
  Laske;                    { Lasketaan uudet koordinatit }
  frmMain.imgKuva.Canvas.FillRect(frmMain.imgKuva.ClientRect); { Tyhjennetään }
  if nayta and 2 = 2 then
    Polygon                 { Piirretään polyonit }
  else if nayta and 1 = 1 then
```

```
    Wireframe;              { Pelkkä rautalanka (Wireframe) }
end;

{*****************************************************************************
* ALIOHJELMA:       Ohje 1.1
* TARKOITUS:        Ohjeiden näyttäminen
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       1.3.2004
* KUVAUS:           Näyttää käyttäjälle (käyttö)ohjeet
*****************************************************************************}

procedure Ohje;
begin
  with frmMain.imgKuva.Canvas do
  begin
    Font.Name := 'Courier New';
    TextOut(10,10, 'Arno Solin - päättötyö 2005');
    TextOut(10,25, 'Ohjelma3D 1.0');
    TextOut(10,40, 'Seuraavat komennot ovat mahdollisia:');
    TextOut(20,55, 'F1:     Näytä tämä ohje');
    TextOut(20,70, 'F2:     Rautalankamalli (Wireframe)');
    TextOut(20,85, 'F3:     Polygonit');
    TextOut(20,100,'F4:     Varjostus');
    TextOut(20,115,'Ctrl+O: Avaa');
    TextOut(20,135,'Hiiri (vasen): Käännä');
    TextOut(20,150,'      (oikea): Siirrä');
    TextOut(20,165,'      (rulla): Zoomi');
  end;
end;

{*****************************************************************************
* ALIOHJELMA:       Avaa 1.1
* TARKOITUS:        Tiedoston avaaminen
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       1.3.2004
* KUVAUS:           Kutsuu valitun tiedostotyypin mukaista latausfunktiota
* PARAMETRIT:       strFilename : Tiedoston nimi (vapaaehtoinen)
* HUOMIOITAVAA:     Lataa joko mittaustiedostoja (*.txt) tai
*                   VRML-tiedostoja (*.wrl ).
*****************************************************************************}

procedure Avaa(strFilename: string = '');
begin
  { Pyydetään käyttäjää valitsemaan tiedostonimi. }
  if strFilename = '' then
    if not PromptForFileName(strFilename,
                    'VRML (*.wrl)|*.wrl|Teksti (*.txt)|*.txt|Kaikki (*.*)|*.*',
                    '','Avaa tiedosto',
                    GetCurrentDir, false)
    then Exit;

  { Jos tiedosto on olemassa, avataan se. }
  if FileExists(strFilename) then
  begin
    if ExtractFileExt(LowerCase(strFilename)) = '.wrl' then
      VRML.Avaa(strFilename, Piste, Polygoni)
    else
      Tiedosto.AvaaMittaus(strFilename, Piste, Polygoni);

    frmMain.Caption := '3D - ' + ExtractFilename(strFilename);

    VRML.Suhteuta(frmMain.imgKuva.Width/2, Piste);
    SetLength(jarjestys, Length(Polygoni));
    SetLength(lista, Length(Polygoni));
    Setlength(Projektio, Length(Piste));

    { Piirretään kappale ja näytetään ohje }
    Piirra;
    Ohje;
```

```
  end
  else
    ShowMessage('Valitsemaasi tiedostoa ei ole olemassa.');
end;
{                   TAPAHTUMAKÄSITTELY (frmMain) }

{*****************************************************************************
* ALIOHJELMA:       imgKuvaMouseDown 1.1
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       1.3.2005
* KUVAUS:           Hiiren lähtökoordinaattien asettaminen
*****************************************************************************}

procedure TfrmMain.imgKuvaMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  { Hiiren koordinaatit alkupisteessä }
  X0 := X; Y0 := Y;
end;

{*****************************************************************************
* ALIOHJELMA:       imgKuvaMouseMove 1.1
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       1.3.2005
* KUVAUS:           Kappaleen pyörittely
*****************************************************************************}

procedure TfrmMain.imgKuvaMouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
begin
  if ssLeft in Shift then
  begin
    { Kappaleen kääntyminen }
    a := a - (X-X0)/3/64; b := b + (Y-Y0)/3/64;
    sina := sin(a); cosa := cos(a);
    sinb := sin(b); cosb := cos(b);

    { Piirretään kappale }
    Piirra;

    X0 := X; Y0 := Y;
  end
  { Kappaleen liikuttelu }
  else if ssRight in Shift then
  begin
    origoX := origoX + (X-X0);
    origoY := origoY + (Y-Y0);
    X0 := X; Y0 := Y;

    { Piirretään kappale }
    Piirra;
  end;
end;

{*****************************************************************************
* ALIOHJELMA:       FormKeyDown 1.1
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       1.3.2005
* KUVAUS:           Näppäinpainallukset (Näyttömoodi, Ohje, Avaus)
*****************************************************************************}

procedure TfrmMain.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
var
  n : integer;
begin
  case Key of
    79: if ssCtrl in Shift then Avaa;    { ctrl+O }
    112: Ohje;                           { F1 }
```

```
    113..116:                          { F2 - F4 }
    begin
      n := trunc(Exp((Key-113)*Ln(2))); { eli 2^(x-113) }
      if nayta and n = 0 then
        inc(nayta, n)
      else
        dec(nayta, n);
      Piirra;
    end;
  end;
end;

{*****************************************************************************
* ALIOHJELMA:       FormResize 1.1
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       1.3.2005
* KUVAUS:           Kuvakehyksen piirtoalan muuttaminen
*****************************************************************************}

procedure TfrmMain.FormResize(Sender: TObject);
begin
  imgKuva.Picture.Bitmap.Width  := imgKuva.Width;
  imgKuva.Picture.Bitmap.Height := imgKuva.Height;
end;

{*****************************************************************************
* ALIOHJELMA:       FormMouseWheel 1.1
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       1.3.2005
* KUVAUS:           Hiiren scrollaus (rulla) eli zoomaus
*****************************************************************************}

procedure TfrmMain.FormMouseWheel(Sender: TObject; Shift: TShiftState;
  WheelDelta: Integer; MousePos: TPoint; var Handled: Boolean);
begin
  Z := Z + WheelDelta;
  Piirra;
end;

{*****************************************************************************
* ALIOHJELMA:       FormActivate 1.1
* NIMI:             Arno Solin
* PÄIVÄMÄÄRÄ:       1.3.2005
* KUVAUS:           Arvojen alustus sovellusta avattaessa
*****************************************************************************}

procedure TfrmMain.FormActivate(Sender: TObject);
const
  ra = 57.29578; { Asteet radiaaneiksi; 180 / 3.14159... }
begin
  { Kaksoispuskurointi vähentää välkkymistä }
  frmMain.DoubleBuffered := true;

  d := 1600; z := 1600;
  a := 30/ra; b := 20/ra;
  sinA := sin(a); cosA := cos(a);
  sinB := sin(b); cosB := cos(b);

  { Origo on keskellä }
  origoX := imgKuva.Width div 2;
  origoY := imgKuva.Height div 2;

  { Luetaan komentoriviltä tiedoston nimi ja näyttöasetukset, jos annettu }
  if ParamCount = 2 then
    nayta := StrToInt(ParamStr(2));

  { Avataan tai pyydetään avaamaan }
  if ParamCount > 0 then
    Avaa(ParamStr(1))
```

```
  else
    Avaa;
end;

end.
```