Katsauksia — Översikter

Graafisten tietokoneohjelmien siirrettävyydestä

MARKKU LÖYTÖNEN

Tietokoneet ja niissä käytetyt hyötyohjelmat ovat käyneet läpi suuren muutosprosessin viimeisen viidentoista vuoden kuluessa. Muutoksen käynnisti tekninen innovaatio, mikroprosessori, jonka kehittäminen avasi mahdollisuuden tehokkaan, mutta silti halvan keskusyksikön rakentamiseen. Kun myös oheislaitteet, kuten massamuistit, näytöt sekä kirjoittimet ja piirturit samanaikaisesti kehittyivät paitsi tehokkaammiksi, myös halvemmiksi, oli edellytykset ensimmäisten mikrotietokonelaitteistojen rakentamiseksi luotu. Sellaiset laitevalmistajat kuin Apple Computer ja yrityksen salamanousu autotallin peränurkasta kansainväliseksi suuryritykseksi tai Adam Osborne, joka tuotti ensimmäisen halvan kannettavan mikrotietokoneen, ovat tuttuja lähes kaikille. Samoin mikrotietokoneiden ympärillä omaisuuden ja maailmanmaineen luoneet ohjelmistotalot, kuten esimerkiksi MicroPro tai Microsoft sekä niiden ykköstuotteet WordStar ja MBASIC, tuskin ovat jääneet huomaamatta.

Yliopistoissa ja korkeakouluissa, jotka kaikkialla maailmassa tuntuvat nykyään kärsivän niukoista määrärahoista, on tämä kehitys otettu tyytyväisyydellä vastaan. Pienet määrärahat ovat välttävää vähän paremmin riittäneet laitteistojen hankintaan ja monet hallintotehtävät sekä rutiininomaiset kirjoitustyöt on voitu siirtää koneen hoidettavaksi. Näin on vapautettu enemmän aikaa opetukseen ja tutkimukseen. Myös tutkimusta palvelevan atk-työskentelyn painopiste on nopeasti siirtynyt kömpelöistä, suuria keskuskoneita vaatineista eräajoista kohti vuorovaikutteista, parhaimmillaan integroituja ja grafiikalla tuettuja hyötyohjelmia hyväksikäyttävää työskentelyä. Tietokoneesta on tullut monelle tutkijalle päivittäinen apuväline, jonka käyttö yhä harvemmin edellyttää käyttäjältään erityistä atk-alan koulutusta.

Suomessakin monet maantieteen laitokset ovat jo hankkineet tai parhaillaan ovat hankkimassa omia mikro- tai minitietokonelaitteistoja, joita voidaan käyttää niin opetuksessa, hallinnossa kuin tutkimuksessakin. Samanaikaisesti laitehankintojen kanssa on laitoksissa käynnistynyt opetus- ja tutkimustoiminta, johon liittyy varsin laaja-alaista ohjelmistokehitystyötä mikrotietokoneympäristössä. Kiinnostus näyttää tällä hetkellä kohdistuneen tavallisten tilastollis–matemaattisten ohjelmien kirjoittamisen ohella maantiedettä palvelevien graafisten erityisohjelmien laatimiseen ja siihen liittyvään perustutkimukseen. Kehittelyn tavoitteena on yleensä laatia ohjelmisto, jonka avulla voidaan esimerkiksi automatisoida ja nopeuttaa muutoin niin hitaita ja työläitä kuvanvalmistuksen käsityövaiheita. Ehkä parhaimman esimerkin maantieteellisistä graafisista sovellutuksista tarjoavat kartanpiirustusohjelmat (kts. Yli-Jokipii 1979: 75–80; 1980: 2–7; Pietala 1982: 274–275), joita jo on kehitetty useassa maantieteen laitoksessa.

Jonkin tieteenalan erityistarpeita palveleva graafinen ohjelmistokehittely on poikkeuksetta hyvin työlästä ja vaatii kirjoittajaltaan paitsi asiantuntevaa paneutumista ohjelman aihepiiriin, myös kokemusta grafiikan ohjelmoinnista. Kokemuksesta ja perehtyneisyydestä huolimatta on varsin tavallista, että ohjelmoija ehtii tuskastua monenlaisten ongelmien parissa ennen kuin työ on valmis ja testattu. Tämän vuoksi olisi järkevää, että yhdessä laitoksessa tehdyn graafisen ohjelmistokehittelyn tulokset voitaisiin ottaa käyttöön myös muissa maantieteen laitoksissa. Graafisten ohjelmien siirtäminen muutoksitta valmiina ja toimivana koodina kohtaa kuitenkin monia vaikeuksia.

Siirrettävyyden esteet

Ohjelmien siirron esteet voidaan karkeasti jakaa kolmeen ryhmään, nimittäin laitteistojen välisistä teknisistä eroista johtuviin yhteensopivuusongelmiin, käyttöjärjestelmien erojen aiheuttamiin siirtovaikeuksiin ja eri

Löytönen, Markku (1985). Graafisten tietokoneohjelmien siirrettävyydestä. *Terra* 97:2, 128–130.

ohjelmointikielissä olevien grafiikkakäskyjen välisiin eroihin. Tietokonelaitteistot eroavat toisistaan monella eri tavalla. Esimerkiksi suorittimet ovat toinen toistaan erilaisempia eivätkä edes saman komponenttivalmistajan samaan perheeseen kuuluvat prosessorit välttämättä toimi yhtenevällä tavalla eri laitevalmistajien tietokoneissa. Eri mikrotietokoneiden levykeformaatit poikkeavat toisistaan niin paljon, että levykkeiden lukeminen muussa kuin täsmälleen samanmerkkisessä laitteessa edellyttää erityisesti tätä tehtävää varten kirjoitettua ohjelmaa. Samanlaisia siirrettävyyttä vaikeuttavia laiteperäisiä ongelmia voisi vaivatta luetella vaikka kuinka paljon.

Myös käyttöjärjestelmät poikkeavat toisistaan niin paljon, että toisessa järjestelmässä kirjoitettu ohjelma on jotakuinkin mahdoton siirtää toimivana toiseen järjestelmään. Edes sama suoritin ja sama käyttöjärjestelmä eivät takaa siirron onnistumista, ellei laitevalmistaja ole sama. Eri ohjelmointikielet puolestaan tukevat grafiikkaa hyvin eri tavoin. Grafiikkakäskyjen määrä vaihtelee valtavasti ja eri toimittajien kielissä on suuria eroja. Niinpä esimerkiksi yhden toimittajan BASIC poikkeaa grafiikkakäskyjen osalta toisen toimittajan vastaavasta kielestä yleensä niin paljon, että ohjelman siirtäminen edellyttää melkein poikkeuksetta grafiikan kirjoittamista kokonaan uudestaan.

Ratkaisuvaihtoehtoja

Laitteistojen ja ohjelmointikielien välisten erojen aiheuttamia ongelmia on yritetty poistaa sopimalla erilaisista standardeista, joista grafiikan osalta tunnetuin lienee ISO-standardin mukainen Graphical Kernel System (GKS). Näyttää kuitenkin siltä, että standardit on tehty vain sitä varten, että niistä voidaan poiketa sopivissa kohdissa. Lisäksi monet suuret laite- ja ohjelmatoimittajat, kuten esimerkiksi IBM ja sen PC-sarja, ovat luoneet omilla laitteistoillaan käytännön standardin, joka ei noudata mitään sovittua järjestelmää. Standardoinnista onkin tullut eräänlainen atk-alan iäisyyskysymys, enkä henkilökohtaisesti jaksa uskoa sen johtavan ohjelmien siirrettävyyden helpottumiseen – ainakaan kovin lyhyellä tähtäimellä.

Toinen tapa kiertää siirrettävyysongelmia on hankkia samanlainen laitteisto kaikille niille käyttäjille, jotka haluavat varmistua ohjelmiensa keskinäisestä siirrettävyydestä. Kun ajatellaan sellaista pientä ja erikoista käyttäjäryhmää kuin maantieteen laitokset Suomessa, ei tämä vaihtoehto tunnu järkevältä. Laitosten laitehankinnat palvelevat grafiikan ohella monia muitakin tehtäviä, eikä hankintaa voi ratkaista vain grafiikan yhteensopivuuden perusteella. Laitoksilla on myös hyvin erilaiset laitehankintamäärärahat eikä yhdelle laitokselle sopiva graafinen laitteisto välttämättä vastaa toisen laitoksen kuvankäsittelytarpeita. Tärkeä laitteiston valintaan vaikuttava seikka on myös kysymys siitä, minkälaisessa ympäristössä kukin haluaa työskennellä.

Ehdotus yhteisiksi periaatteiksi

Käytännössä edellä sanottu merkitsee sitä, että yhdessä laitoksessa, tietyssä laiteympäristössä ja tietyllä ohjelmointikielellä kirjoitetun graafisen ohjelman siirto toisen laitoksen käyttöön onnistuu suoraan vain harvoissa tapauksissa. Ohjelmien siirtoa voidaan kuitenkin jossain määrin pyrkiä helpottamaan, jos voimme tieteenalakohtaisesti sopia eräistä yhteisistä menettelytavoista. Mielestäni seuraavat seikat ovat sellaisia, jotka yleisesti hyväksyttyinä periaatteina tarjoaisivat mahdollisuuden graafisten ohjelmien siirtämiseen kohtuullisen vähäisellä vaivalla.

Ensinnäkin olisi voitava sopia siitä, että graafiset ohjelmat kirjoitetaan jollakin grafiikkaa tukevalla korkean tason strukturoidulla ja modulaarisella kielellä. Tämä mahdollistaa selkeän ja havainnollisen ohjelmarakenteen sekä eri toimintojen kokoamisen omiin lohkoihinsa. Kysymys siitä, mitä kieliä käytetään, ei ole ratkaisevan tärkeää, kunhan niistä löytyvät edellä mainitut ominaisuudet. Itse pidän Pascal- ja C-kieliä tähän tehtävään sopivina, joskin monet muutkin kielet tukevat älykästä grafiikkaa. Esimerkkinä mainittakoon APL, joka soveltuu hyvin juuri vektoripohjaiseen kuvankäsittelyyn. Suurta perehtyneisyyttä, erikoislaitteita tai mikrotietokoneita suurempia yksiköitä vaativia kieliä – esimerkiksi jokaiseen edelliseen kohtaan sopii jälleen APL – olisi kuitenkin käytettävä vain erikoistapauksissa. Toiseksi graafiset ohjelmat tulisi pyrkiä kirjoittamaan modulaarisesti siten, että laiteriippuvat piirteet, kuten oheislaitteiden ohjaus, aineiston luku, muunnokset ja skaalaukset, laskenta ja graafinen tulostus olisi selkeästi erotettu toisistaan

omiksi moduleikseen, jotka taas kutsuvat mielekkäitä aliohjelmamoduleita. Ohjelman sovittaminen toiseen ympäristöön on tällöin vaivattomampaa, sillä muutokset on helpompi toteuttaa. Jos aineiston muunnokset ja skaalaukset, laskenta ja oheislaitteiden ohjaus sekä grafiikan tulostus on sotkettu yhteen lohkottomaan ja kommentoimattomaan ohjelmakoodiin, jää hyvä ja kiinnostava ohjelma jotakuinkin varmasti sovittamatta.

Ohjelman modulirakenteen tulisi ylimmällä tasolla käsittää ainakin seuraavat lohkot:

1) Määrittelymoduli, joka kuvaa laiteympäristön erityisesti graafisten tulostuslaitteiden osalta, voimassa olevat oletusarvot ja ohjeet niiden muuttamiseksi sekä selvittää tarkasti kaikki ohjelman laiteriippuvat piirteet.
2) Lukumoduli, joka kuvaa ohjelman hyväksymän syöttötiedoston vaatimukset sekä lukee aineiston.
3) Muunnosmoduli, jossa tehdään tarvittavat muunnokset ja skaalaukset eli sovitetaan aineisto laitteiston kanssa yhteensopivaan koordinaatistoon.
4) Laskentamoduli, jossa tehdään tehtävän vaatimat laskennat ja talletetaan aineisto tulostusta varten.
5) Tulostusmoduli, joka käsittää oheislaitteiden, lähinnä piirturin ja näytön ohjauksen sekä tulostuksen rutiinit.
6) Menumoduli, johon kootaan eri tason valikot sekä mahdollisesti näytölle pyynnöstä tulostettavat opasteet ja kuvaukset.
7) Pääohjelmamoduli, jossa määritellään ohjelman ylimmän tason toimintarakenne ja toimintaehdot.

Kolmanneksi grafiikan osalta tulisi pyrkiä noudattamaan GKS-standardia josta poikkeaminen olisi kuvattava määrittelymodulissa.

Neljänneksi ohjelmakoodi olisi kommentoitava perusteellisesti ja siitä tulisi kirjoittaa selkeä dokumentti. Tiedän omasta kokemuksesta varsin hyvin kuinka helppoa on »unohtaa» oman ohjelman kommentointi ja dokumentointi. Toisaalta kaikki tietävät, kuinka vaikeaa on yrittää ymmärtää toisen kirjoittamaa ohjelmakoodia, jos sitä ei ole kommentoitu.

Edellä sanottu on käytännössä ehdotus eräänlaiseksi hyvin yleisen tason standardiksi. Toteutuessaan sen ei tulisi sitoa tutkijoita, vaan pikemminkin toimia ohjaavana tavoitteena. Jokaisella ohjelmoijalla on oma käsialansa ja tietty työskentelyrutiini, jonka muuttaminen jonkin standardin mukaiseksi on aina työlästä. En kuitenkaan usko, että edellä kuvatun yleisen tason standardin noudattaminen – onhan monin paikoin kysymys vain hyvän ohjelmointitavan mukaisista periaatteista – osoittautuu kovin vaivalloiseksi.

Oheiseen kirjallisuusluetteloon olen viitteiden lisäksi merkinnyt muutaman teoksen, joissa käsitellään aihepiiriin liittyviä kysymyksiä yksityiskohtaisemmin.

Kirjallisuutta

Foley, J.D. & A. van Dam (1982). *Fundamentals of interactive computer graphics.* Addison Wesley, Reading.

Monmonier, Mark S. (1982). *Computer-assisted cartography: principles and prospects.* Prentice-Hall, Englewood Cliffs.

Pietala, Jorma (1982). Tietokonekartografian kurssi Turussa. *Terra* 94(4): 274–275.

Taylor, D.R. Fraser (ed.) (1980). The computer in contemporary cartography. *Progress in Contemporary Cartography* vol I. John Wiley & Sons, Chichester.

Sedgewick, Robert (1983). *Algorithms.* Addison-Wesley, Reading.

Yli-Jokipii, Pentti (1979). Kartografian tutkimus – Suomessa unohdettu maantieteen osalohko. *Terra* 91(2): 75–80.

Yli-Jokipii, Pentti (1980). Kartografian tutkimus ja opetus eräissä Yhdysvaltain yliopistoissa. *Terra* 92(1): 2–7.