Visualisointi ohjelmoinnin oppimisessa

Erkki Kaila, Teemu Rajala & Mikko-Jussi Laakso
Turun Yliopisto, Informaatioteknologian laitos
{ertaka, temira, milaak}@utu.fi

1 Johdanto

Ohjelmointitaitoa voidaan pitää tietojenkäsittelytieteen opintojen tärkeänä yksittäisenä päämääränä. Useimpien tutkimusten mukaan ohjelmointi on kuitenkin aloittelijoille vaikeaa: tarvittavien skeemojen rakentuminen on hidasta ja motivaatio usein pelkästään ulkoista. McCrackenin ym. (2001) ja DuBoulayn (1989) esittämät oppimispäämäärät ohjelmoinnin peruskursseille ovat melko vaativia, mikäli oppijat eivät ole aikaisemmin tutustuneet ohjelmointiin. Koska kurssien sisältöön ei tutkintovaatimusten takia juurikaan voi vaikuttaa, pitää erityistä huomiota kiinnittää opetusmenetelmien valintaan. Visualisointi – tarkoittaen ohjelman tai algoritmin esittämistä graafisten elementtien avulla – on eräs tällaisista menetelmistä.

2 Ohjelmoinnin opetus ja visualisointi

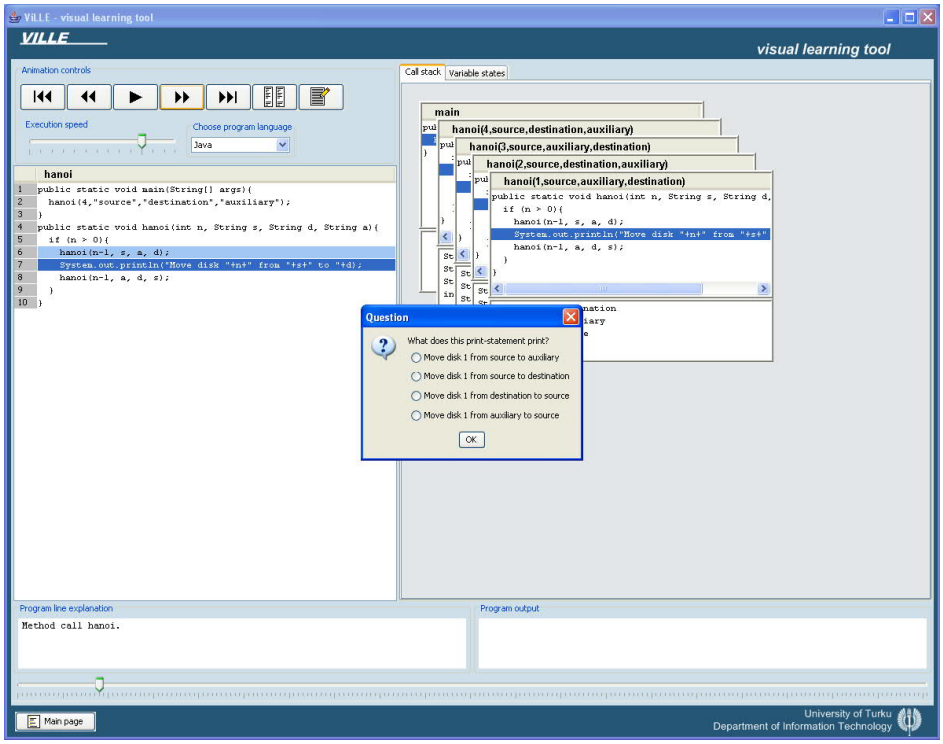
Ohjelmoinnin oppiminen on kaksijakoista: on ensinnäkin hallittava kielen syntaktiset piirteet ja järjestelmän ja ympäristön käyttö, ja toisaalta opittava ratkaisemaan ongelmia ohjelmoinnin avulla. Solowayn (1986) mekanismit ja selitysten muodostaminen on esimerkki kaksijakoisuudesta, ja Bloomin taksonomia (Scott, 2003) edelleen keino jakaa oppimisen päämäärät useampaan kategoriaan. Ohjelmoinnin opettamista voidaan myös tarkastella kahdesta eri näkökulmasta: miten ohjelmointia on *perinteisesti* opetettu, ja toisaalta millaisia *uusia menetelmiä* (joista visualisointi on tämän artikkelin puitteissa tietysti tärkein) on kehitetty avustamaan oppimista. Reekin (1995) mukaan oppimisen pitäisi olla ongelmalähtöistä, ja syntaksin oppimisen toissijaista. Boyle ym. (2003) taas lähtivät uudistamaan opetusmenetelmiä: perinteisten menetelmien rinnalle kehitettiin kokonaan uusia opetusmetodeja, tärkeimpänä erilaiset verkko-opetusympäristöt; tutkimuksen perusteella vaikuttaa siltä, että niin oppimistulokset kuin opiskelumotivaatiokin parantuivat huomattavasti.

Visualisointijärjestelmien käyttö on eräs vaihtoehto opetusmenetelmiä uudistettaessa. Visualisoinnilla tarkoitetaan ohjelman suorituksen tai algoritmin (tai niiden osien) esittämistä graafisten elementtien avulla (ks. esim. Baecker, 1988 tai Ben-Ari, 2001). Algoritmien visualisoinnissa esitetään ohjelman tai algoritmin suoritus – esimerkiksi binääripuun järjestäminen – graafisesti (Naps, 1996). Ohjelman suorituksen visualisointi taas keskittyy esittämään ohjelman käyttäytymistä, interaktioita ja tilojen muutoksia suorituksen aikana, yleensä rivi kerrallaan. Tunnettuja algoritmien visualisointijärjestelmiä ovat esimerkiksi TRAKLA2, JHAVE ja Raptor; ohjelmasuoritusta visualisoivia järjestelmiä ovat esimerkiksi Jeliot3 ja JAVAVIS.

3 ViLLE

ViLLE on Turun yliopistossa kehitetty ohjelman suorituksen visualisointijärjestelmä. Se visualisoi ohjelman suoritusta erilaisin graafisin ja tekstimuotoisin elementein, muun muassa näyttämällä aliohjelmat kutsupinossa, muuttujien arvot ja tulostukset omissa ikkunoissaan ja sanalliset selitykset suoritettavasta rivistä (Kuva 1). Järjestelmä tukee joustavasti eri ohjelmointikieliä, ja mahdollistaa uusien kielien ja esimerkkien määrittämisen sisäänrakennetun editorin avulla. Opettaja voi myös muodostaa ponnahdusikkunoin toteutettuja kysymyksiä, jotka

esitetään ohjelman suorituksen aikana. ViLLEn tarkempi kuvaus löytyy Rajalan ym. (2007) artikkelista.



Kuva 1: ViLLEn visualisointinäkymä

3.1 ViLLEn tärkeimmät ominaisuudet

Kieliriippumattomuus ja uusien ohjelmointikielien lisäys. VILLEn avulla opettaja voi itse määritellä ohjelmointikielen opetusmieltymystensä mukaan. VILLEn avulla voidaan suorittaa tämän kielen mukaisia ohjelmia.

Rinnakkaisnäkymä. Vaihtoehtoisesti ohjelman suoritusta voi seurata ns. rinnakkaisnäkymässä, jossa ohjelmakoodi on näkyvissä samanaikaisesti kahdella eri ohjelmointikielellä. Näin käyttäjä voi konkreettisesti havainnoida, miten syntaksin eroista huolimatta ohjelman suoritus eri ohjelmointikielillä etenee samalla tavoin.

Suorituksen visualisointi rivi kerrallaan. VILLE visualisoi ohjelmakoodin suorituksen etenemistä korostamalla ohjelmakoodin rivejä. Suoritettavana olevan rivin lisäksi korostetaan myös edellinen suoritettu rivi. Näin pyritään helpottamaan ohjelman suorituksen etenemisen seurantaa.

Visualisoinnin kontrolloiminen sekä eteen- että taaksepäin. Visualisoinnissa voidaan liikkua askel kerrallaan sekä eteen- että taaksepäin käyttäjän haluamassa tahdissa. Taaksepäin liikkuminen ohjelman suorituksessa puuttuu yleensä vastaavanlaisista sovelluksista. Lisäksi suorituksen etenemistä kuvaavan liukusäätimen avulla on mahdollista siirtyä suoraan mihin tahansa ohjelman suorituksen vaiheeseen.

Suorituksen aikaiset kysymykset. Esimerkkien ohjelmariveille voidaan asettaa kysymyksiä, jotka esitetään visualisoinnin yhteydessä. VILLEn kysymyseditorilla opettaja voi luoda monivalintakysymyksiä, jotka ponnahtavat näkyviin opettajan määrittämissä ohjelman vaiheissa.

Koodirivin selitykset. Suoritettavana olevasta ohjelman rivistä on näkyvissä selitysrivi, jossa kerrotaan, mitä rivillä tapahtuu. Lisäksi sovellus näyttää ohjelman tuottamat tulostukset ja suoritushetkellä voimassa olevien muuttujien tilat. Suoritettavan ohjelmarivin selitysteksti puuttuu monista vastaavanlaisista sovelluksista.

Ohjelmien muokkaus suorituksen yhteydessä. Ohjelmia voidaan erillisen opettajille tarkoitetun esimerkkien muokkauseditorin lisäksi muokata myös visualisoinnin yhteydessä. Näin opiskelijatkin pystyvät kokeilemaan millä tavoin heidän tekemänsä muutokset vaikuttavat ohjelman suoritukseen. Visualisoinnin aikana tehdyt muutokset eivät kuitenkaan vaikuta alkuperäiseen esimerkkiin, vaan ne ovat voimassa ainoastaan visualisoinnin aikana.

Kutsupino. Ohjelman suorituksen siirtymistä eri aliohjelmakutsujen välillä havainnollistetaan ns. kutsupinolla. Kun suoritus ohjelmassa siirtyy aliohjelmaan, kutsupinoon avautuu uusi ikkuna, joka säilyy kutsupinossa kunnes aliohjelmasta poistutaan. Aliohjelmakutsusta poistuttaessa kutsupinossa näytetään aliohjelman palautusarvo.

Esimerkkikokoelma. VILLE sisältää valmiin joukon erilaisia ohjelmointiesimerkkejä, jotka on jaettu aihepiireittäin eri kategorioihin. Käyttäjä voi halutessaan lisätä uusia kategorioita ja esimerkkejä tai muokata VILLEen valmiiksi integroituja esimerkkejä. Lisäämällä ja muokkaamalla ohjelmointiesimerkkejä, opettaja voi havainnollistaa luennoilla omasta mielestään ohjelmoinnin oppimisen kannalta tärkeitä asioita.

Esimerkkien julkaisu. VILLEn esimerkit voidaan ”Vie”-toiminnon avulla tallentaa esimerkkikokoelmaksi, joka sisältää esimerkkien lisäksi VILLE-sovelluksen, josta on poistettu esimerkkien lisäys- ja muokkaustoiminnot. Opettaja voikin ”Vie”-toiminnon avulla julkaista ohjelmoinnin kurssiin liittyvät esimerkit verkkoon kurssin opiskelijoiden saataville.

3.2 ViLLEn tehokkuus

ViLLEn vaikutuksia ohjelmoinnin oppimiseen tutkittiin Turun Yliopistossa ensimmäisellä ohjelmoinnin kurssilla syksyllä 2007. Tutkimuksessa opiskelijat (N = 72) osallistuivat kahden tunnin pituiseen mikroluokkaharjoitukseen, jossa harjoiteltiin ohjelmointikäsitteitä ohjelmointitutoriaalin avulla. Opiskelijat jaettiin satunnaisesti kahteen ryhmään, joista toinen (N = 40) kävi läpi ohjelmointitutoriaalin ja toisella (N = 32) oli lisäksi mahdollisuus tutkia tutoriaalissa esiintyviä ohjelmointiesimerkkejä ViLLEn avulla. Harjoituksen alussa ja lopussa opiskelijat vastasivat testiin, joka sisälsi erilaisia ohjelmointitehtäviä. Tutkimuksen perusteella opiskelijat, joilla ei ollut aikaisempaa ohjelmointikokemusta hyötyivät ViLLEstä eniten, sillä heidän ja aikaisempaa ohjelmointikokemusta omaavien tulokset tasoittuivat lopputestissä tilastollisesti merkitsevästi. Tutkimuksen tulokset on raportoitu tarkemmin Rajalan ym. (2008) artikkelissa.

4 Johtopäätöksiä

Tutkimuksen perusteella näyttää siltä, että visualisointijärjestelmän käyttö on vakavasti otettava menetelmä ohjelmoinnin perusopetuksessa. Lyhyenkin harjoittelutilaisuuden aikana tapahtui merkittävää oppimista. Erityisesti järjestelmästä oli hyötyä niille, joilla ei ollut aikaisempaa ohjelmointikokemusta. Tulevaisuudessa kannattaisi erityisesti panostaa järjestelmän

pitempiaikaisen käytön tutkimiseen, sekä tutkia tarkemmin millaisia eroja eri käyttäjäryhmien välillä on aikaisempi ohjelmointikokemus huomioiden.

5 Lähdeluettelo

Baecker, R. 1988. Enhancing program readability and comprehensibility with tools for program visualization, Proceedings of the 10th international conference on Software engineering, April 11-15, Singapore, s. 356-366.

Ben-Ari, M. 2001. Program visualization in theory and practice. Informatik/Informatique, 2 (April), s. 8-11.

du Boulay, B. 1989. Some difficulties of learning to program. Teoksessa E. Soloway & J. C. Spohrer (toim.). Studying the novice programmer. Hillsdale, NJ: Lawrence Erlbaum, s. 283-299.

Boyle, T., Bradley, C., Chalk, P., Jones, R. & Pickard P. 2003. Using blended learning to improve student success rates in learning to program. Journal of Educational Media, special edition on Blended Learning, 28 (2-3), s. 165-178.

McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y., Laxer, C., Thomas, L., Utting, I. & Wilusz, T. 2001. A Multi-National, Multi-Institutional Study of Assessment of Programming Skills of First-year CS Students. ACM SIGCSE Bulletin, vol. 33, nro. 4, s. 125-140.

Naps, T. L. 1996. Algorithm visualization served off the World Wide Web: why and how. Proceedings of the 1st conference on Integrating technology into computer science education, Barcelona, Spain, s. 66-71.

Rajala, T., Laakso, M.-J., Kaila, E. & Salakoski, T. 2008. Effectiveness of Program Visualization: A Case Study with the ViLLE Tool. To appear in the Journal of Information Technology Education: Innovations in Practice.

Rajala, T., Laakso, M.-J., Kaila, E. & Salakoski, T. 2007. VILLE – A language-independent program visualization tool. Proceedings of the Seventh Baltic Sea Conference on Computing Education Research (Koli Calling 2007), Koli National Park, Finland, November 15-18, 2007. Conferences in Research and Practice in Information Technology, vol. 88, Australian Computer Society. Raymond Lister and Simon, Eds.

Reek, M. 1995. A top-down approach to teaching programming, ACM SIGCSE Bulletin, vol. 27, nro. 1, s. 6-9.

Scott, T. 2003. Bloom's Taxonomy Applied to Testing in Computer Science Classes, in Proceedings of the Twelfth Annual CCSC Rocky Mountain Conference, J.G. Meineke (toim.), Silver City, New Mexico, October 17-18, s. 267-274.

Soloway, W. 1986. Learning to program = learning to construct mechanisms and explanations, Communications of the ACM, vol. 29, nro. 9, s. 850-858.