Ohjelmoinnin opettaminen lapsille

Oulun Yliopisto
Tietojenkäsittelytieteiden laitos
Kandidaatin tutkielma
Arttu Piirainen
24.4.2015

Tiivistelmä

Tämän päivän lapset ovat tietoteknisesti valveutuneita ja heillä on kokemusta erilaisista teknologioista, kuten älypuhelimista, peleistä ja sosiaalisesta mediasta. Silti vain harvalla on kokemusta ohjelmoinnista. Viime vuosina tietoteknisesti kehittyvässä maailmassa on noussut esiin ohjelmoinnin opettaminen jo pienille lapsille peruskoulusta alkaen. Ohjelmointia on jo opetettu lapsille useissa maissa ohjelmointikerhoissa, päiväkodeissa ja peruskouluissa. Suomessa ohjelmointi tulee peruskoulun opetussuunnitelmaan vuonna 2016 matematiikan opettamisen yhteyteen.

Tässä kandidaatin tutkielmassa tarkastellaan aiemman tutkimuksen pohjalta, kuinka ohjelmointia on opetettu lapsille. Tämän työn kahdeksassa empiirisessä tutkimuksessa esitellään tutkimusmenetelmiä ohjelmoinnin opettamisesta eri ikäisille lapsille. Aineiston perusteella ohjelmoinnin opettamisen opetusmenetelmiä voidaan käyttää ihan tavallisissa luokkahuoneissa erilaisia opetuskäytäntöjä soveltaen.

Tässä tutkimuksessa selvitetään myös mitä hyötyä ohjelmoinnista on lapsille ja mitä haasteita ohjelmoinnin opettamisessa on kohdattu. Lisäksi esitellään viisi (Logo, Electronic Blocks, ToonTalk, Scratch ja Alice) lapsille ja noviisikäyttäjille suunnattua ohjelmointiympäristöä.

Asiasanat: ohjelmointi, ohjelmoinnin opettaminen, ohjelmointiympäristöt, K-12

Sisällysluettelo

1. Johdanto

Peruskouluikäiset lapset nauttivat teknologian käytöstä esimerkiksi käyttämällä heidän älypuhelimiaan, surffailemalla internetissä ja käyttämällä sosiaalista mediaa kuten Facebookia. Näistä teknologiakokemuksista huolimatta monella lapsella on vain epämääräisiä käsityksiä siitä, mitä ohjelmointi on. Tämä ei ole sinänsä yllättävää, koska tietojenkäsittelytiedettä ja ohjelmointia ei toistaiseksi opeteta oppiaineena ala –eikä yläkoulussa. (Rodger et al., 2010.)

Ohjelmointia pidetään tärkeänä taitona kehittämään korkeamman asteen ajattelun lisäksi ongelmaratkaisutaitoja (Fessakis, Gouli & Mavroudi, 2013). Ohjelmointi ei ole pelkästään ”koodausta”, vaan se parantaa oppilaiden algoritmista ajattelua (computational thinking). Algoritminen ajattelu on ongelman ratkaisua, jossa käytetään tietojenkäsittelytieteen käsitteitä ja periaatteita. (Lye & Koh, 2014.) Ohjelmoinnin oppimisen avulla oppilaat pääsevät myös tutustumaan sovellusten kehittämiseen ja luomaan uusia ystävyyssuhteita esimerkiksi internetissä. Vain osa meistä tulee ohjelmoimaan ja suunnittelemaan tietojärjestelmiä työkseen, mutta silti useat meistä tarvitsevat ohjelmointia ja sen periaatteita jossain elämän vaiheessa. (Kafai & Burke, 2014.)

Useat opettajat ja IT-alan ammattilaiset ovat yhtä mieltä siitä, että ohjelmoinnin ammattilaisten tarve kasvaa tulevaisuudessa. Samalla akateemisessa maailmassa tietojenkäsittelytieteiden opiskelun suosio on kokenut jyrkkää laskua viimeisen vuosikymmenen aikana. (Lakanen, Isomöttönen & Lappalainen, 2012.) Ohjelmoinnin opettaminen lapsille ja nuorille on myös Suomessa erittäin tärkeää kilpailukyvyn ja tulevaisuuden säilymisen kannalta, sillä hyvistä ohjelmoijista on alalla jo nyt pulaa. Suomen ICT-alalla on 17 000 työpaikan vaje ohjelmoinnin osaajista vuoteen 2019 mennessä. (Liukas & Mykkänen, 2014.)

Ohjelmoinnin opettaminen on tekemässä paluuta kouluihin. 1980-luvulla useat oppilaitokset opettivat esimerkiksi Basic, Logo tai Pascal –ohjelmoinnin perusteita kerran viikossa. 1990-luvun puolivälissä monissa kouluissa ei enää opetettu ohjelmointia. Oppilaitokset unohtivat ohjelmoinnin opettamisen totaalisesti, koska osa oppilaitoksista koki sen täysin tarpeettomana ja osa piti sitä liian vaikeana opettaa ja oppia. (Kafai & Burke, 2014.) Sama ongelma on havaittavissa ympäri maailman, sillä esimerkiksi Taiwanissa ala –ja yläkoulussa opetetaan tyypillisesti Windows –ohjelmien kuten Wordin, PowerPointin, FrontPagen ja Flashin käyttöä unohtaen ohjelmoinnin opettamisen, joka tapahtuu pääsääntöisesti ohjelmointikerhoissa. Ohjelmointia olisikin hyvä opettaa paljon enemmän peruskouluissa. (Lin, Yen, Yang & Chen, 2005.) Suomessa ohjelmoinnin opettaminen peruskouluissa alkaa vuonna 2016, mutta sitä ei opeteta omana oppiaineena vaan matematiikan tuntien yhteydessä (Liukas & Mykkänen, 2014).

On olemassa useita ohjelmointiympäristöjä, jotka on erityisesti tarkoitettu nuorille lapsille. Tällaisia ovat esimerkiksi erilaiset Logo-kielet, ToonTalk, Squeak Etoys, Stagecast Creator, Microworlds JR ja Scratch. Näissä ohjelmointiympäristöissä on erityisiä ominaisuuksia, jotka tekevät niistä kehityksellisesti entisestään sopivampia lapsille. Tällaisia ominaisuuksia ovat: yksinkertaistettu syntaksi, ohjelmointi toiminnallisilla palikoilla, joilla lapsi voi palikoiden paikkaa siirtämällä ja yhdistämällä

luoda ohjelmia, välitön komentojen suorittaminen (ilman kääntämistä) ja ohjelmointitapojen mukauttaminen metaforiksi. (Fessakis et al., 2013.)

Yleisesti uskotaan, että ohjelmointipohjaiset oppimistapahtumat helpottavat kehittämään algoritmista päättelyä ja ongelmanratkaisukykyä. Tämä edellytys ohjaa kehittämään useita opetuksellisia ohjelmointiympäristöjä myös nuorille lapsille. (Fessakis et al. 2013.) Ohjelmoinnin oppiminen on menossa kohti sosiaalista muutosta (social turn), siirtymässä individualistisista näkemyksistä kohti näkemystä, joka sisältää sosiologisia ja kulttuurisia ulottuvuuksia ja uudelleen käsitteellistää algoritmista ajattelua (computational thinking) enemmän algoritmiseksi osallistumiseksi (computational participation). (Kafai & Burke, 2014.)

Valitsin kandidaatintyön aiheekseni ”ohjelmoinnin opettaminen lapsille”, koska ohjelmoinnin opettaminen aloitetaan Suomen peruskouluissa vuonna 2016, joten aihe oli ihmisten keskuudessa puheenaiheena. Tässä tutkimuksessa selvitetään, miten muualla maailmassa ohjelmoinnin opettamista lapsille on tutkittu.

Hankittuani ja perehdyttyäni lähdeaineistoon asetin tavoitteekseni jäsentää aineiston pohjalta vastaukset seuraaviin kysymyksiin: i) mitä hyötyä ohjelmoinnin opettamisesta on lapsille (luku 3). (ii) mitä haasteita ohjelmoinnin opettamisessa on kohdattu (luku 4) (iii) sekä millaisilla eri ohjelmistoympäristöillä ohjelmointia voisi opettaa lapsille (luku 5). Ennen näitä lukuja, esittelen tutkimusmenetelmäni tarkemmin (luku 2) ja työn lopussa kerron tutkimukseni kokonaistulokset ja pohdin niitä (päätäntä). Tutkielmassa keskitytään alle 16-vuotiaisiin eli peruskouluikäisiin lapsiin.

2. Aineiston hankintaprosessi

Aloitin aineiston keräämisen kandidaatintutkielmaani varten marraskuun loppupuolella 2014. Aineiston hakeminen ja kerääminen kesti noin kuukauden ja tein kirjoitusprosessin aikana lisäksi muutaman täsmentävän ja täydentävän lisähaun.

Tutustuin ensin alani tietokantoihin, joita ovat Scopus, IEEE, ACM ja Web of Science. Lisäksi käytin hakuprosessissa myös Google Scholar -palvelua. Etsin kaikista näistä tietokannoista aineistoa aluksi hakusanalla ”children AND computer programming”, joka tuotti paljon osumia. Järjestin hakutulokset viittausten mukaan, mutta huomasin, että joukossa oli paljon aiheen ulkopuolisia artikkeleita. Kokeilin tämän jälkeen hakusanoiksi ”learning” AND ”programming” AND ”children”, ”learning” AND ”programming” ja ”teaching” AND ”programming”. Hakutuloksia oli jälleen paljon ja joukossa oli edelleen useita täysin aiheen ulkopuolisia artikkeleita. Näillä hakusanoilla sain kosketuksen aiheeseen ja löysin noin kymmenkunta artikkelia työhöni.

Jatkoin edelleen aineiston keruuta erilaisilla hakusanoilla edellä mainituista tietokannoista. Olin rajannut ja valinnut tässä vaiheessa kandidaatintyön aiheekseni ohjelmoinnin opettaminen lapsille, koska ensimmäisen haun aikana mietin aihettani joko opettamisen tai oppimisen näkökulmasta. Löysin edelleen hajanaisesti muutamia artikkeleita hakusanoilla ”coding” AND ”childen”, ”computer science” AND ”children”, ”information technology” AND ”children” ja ”coding” AND ”kids”. Hakutulokset olivat edelleen laajoja ja epätarkkoja, mutta löysin kuitenkin yksittäisiä artikkeleita.

Joulukuun alussa 2014 hakuprosessi oli edelleen käynnissä ja kokeilin hakusanaa ”K-12” AND ”programming”, joka tuotti eniten täsmällisiä osumia ja edisti parhaiten hakuprosessia ja loppuun saattamista. Etsin tällä hakusanalla artikkeleita kaikista tietokannoista. Google Scholarista löysin kymmenkunta käypää artikkelia, Scopuksesta myös kymmenen, IEEE:stä, ACM:stä ja Web of Sciencesta noin viisi jokaisesta. Tämän jälkeen aineisto alkoi olla kokonaisuudessaan kasassa. Materiaalia oli yhteensä 48 artikkelin verran.

Kirjoitustyön alkuvaiheessa tammi-helmikuussa 2015 tein vielä tarkentavan haun jokaiseen tietokantaan käyttäen hakusanoja ”elementary school” AND ”programming”, ”kindergarden” AND ”programming” ja ”primary school” AND ”programming”. Sain täydennettyä tämän prosessin avulla aineistoani, ja yhdessä aiemmin kerätyn aineiston kanssa niistä koostuu tämän työn kokonaisaineisto.

Hakemani aineiston suuruus oli 52 artikkelia. Näistä artikkeleista tähän tutkielmaan käytin tässä työssä 24 artikkelia, sillä näiden avulla oli mahdollista muodostaa vastauksia asettamiini kysymyksiin.

3. Ohjelmoinnin opettamisen hyödyt

Ohjelmoinnin opettaminen auttaa lapsia oppimaan useita hyödyllisiä ja elämässä tarvittavia taitoja, kuten matemaattisia taitoja, yhteisöllisyyttä ja ryhmässä toimimista. Ohjelmointipohjaiset oppimistapahtumat helpottavat kehittämään algoritmista päättelyä ja ongelmanratkaisutaitoja (Fessakis et al., 2013). Ongelmaratkaisutaitojen kehityksen lisäksi ohjelmoinnin opettaminen parantaa lapsien kognitiivista kehitystä. Oppilaat, joilla oli ohjelmointikokemusta, pärjäsivät paremmin erilaisissa kognitiivista päättelykykyä mittaavissa testeissä. (Liao & Bright, 1991.)

Kafai ja Burke (2013) ovat todenneet, että ohjelmoinnin opiskelu kehittää myös algoritmista ajattelua (computational thinking). Algoritminen ajattelu voidaan ymmärtää tietojenkäsittelytieteen periaatteiden soveltamisena muissa tieteenaloissa, esimerkiksi pilkkomalla ongelmat osiin, määrittelemällä niiden suhteet toisiinsa ja osaan suurempaa kokonaisuutta ja kehittää algoritmeja ongelmien ratkaisemiseksi. Algoritminen ajattelu ei rajoitu pelkästään matematiikkaan ja luonnontieteisiin, vaan se koskee myös humanistisia aloja, kuten journalismia ja kirjallisuutta. (Kafai & Burke, 2013.) Algoritminen ajattelu ja ohjelmointi tukevat toisiaan, sillä algoritmisen ajattelun opettamisessa ohjelmoinnin opettaminen ja hyödyntäminen on keskeisessä roolissa. Tietojenkäsittelytieteen opettajien järjestö (Computer Science Teachers Association) on kehittänyt peruskouluille opetusstandardeja ja ohjeita algoritmisen ajattelun puolesta. Ohjelmointi on ollut mukana poikkeuksetta kaikissa ehdotetuissa opetussuunnitelmissa. Yhden tietyn ohjelmointikielen opettaminen ei ole oleellista, sen sijaan ohjelmoinnin peruskäsitteiden opettaminen, välittämättä itse kielestä. (Kafai & Burke, 2014.)

Ohjelmointi ei ole enää yksilökeskeistä toimintaa, jossa lähdekoodi pidetään varjeltuna ja muilta salassa. Koodin jakaminen, yhdistäminen, toisten rohkaiseminen ja muiden haastaminen on ollut kasvussa. Tällainen avoimuus parantaa innovaatioiden mahdollisuutta ja nuoret käyttäjät tukevat jakamista vapaammin kyseenalaistaen vanhat käytännöt. (Kafai & Burke, 2014.) Ohjelmoinnin oppiminen oli ennen toimintaa, jossa tarkkuus ja tehokkuus merkitsivät menestystä. Tänään ei ohjelmoida ohjelmoinnin oppimisen vuoksi vaan opiskelijat luovat oikeita sovelluksia kuten pelejä ja tarinoita osana suurempaa oppimisyhteisöä. Sovellukset takaavat pääsyn koodausyhteisöihin ja edistävät toveruutta. Ohjelmointi on kokemassa sosiaalista muutosta (social turn) ja se on menossa kohti yhteisöllisyyttä. Ohjelmointi ei ole enää eristäytynyttä toimintaa vaan yhteistä sosiaalista toimintaa, jossa hyödynnetään ilmaisia ohjelmointiympäristöjä ja keskinäistä intoa kannustaa osallistumaan. (Kafai & Burke, 2013.)

Ohjelmoinnin osaajista on jatkuva ja kasvava tarve työmarkkinoilla. Lakasen et al. (2012) mukaan useat opettajat ja IT-alan ammattilaiset ovat yhtä mieltä siitä, että ohjelmoinnin ammattilaisten tarve kasvaa tulevaisuudessa. Samalla akateemisessa maailmassa tietojenkäsittelytieteiden opiskelun suosio on kokenut jyrkkää laskua viimeisen vuosikymmenen aikana. (Lakanen et al., 2012.) Puutteellinen johdatus tietotekniikkaan peruskouluissa johtaa siihen, että harvat opiskelijat lähtevät opiskelemaan tietojenkäsittelytieteitä yliopistoon (Rodger et al., 2012). Tietotekniikan alan osaajien tarve on kasvanut useissa maissa. Yhdysvaltojen tilastokeskus (The U.S. Bureau of Labor Statistics) ennusti vuonna 2005, että 65% korkeakoulutettujen työpaikoista vuosina 2004-2014 olisi tietotekniikan alalla (Hecker, 2014).

Yhdysvaltojen tilastokeskus myös ennusti, että Yhdysvallat voi täyttää vain kolmasosan teknologian alan työpaikoista oman maan kansalaisistaan (Rodger et al. 2012).

4. Ohjelmoinnin opettamisen haasteet

Ohjelmoinnin opettamisessa on havaittavissa myös useita haasteita, kuten lasten mielenkiinnon herättäminen ohjelmointia kohtaan, lasten kohtaamat kielelliset ja tekniset ongelmat sekä oppilaitosten kohtaamat haasteet opettamisessa.

Ohjelmoinnin opettamisessa on kohdattu teknisiä haasteita liittyen ohjelmointikielten kielioppiin eli syntaksiin ja monimutkaisiin virheilmoituksiin. Ohjelmoitaessa ei vain yhdistellä satunnaisesti joukkoa merkkejä, kuten ei puhutuissakaan kielissä. Merkkejä tulee käyttää oikeassa muodossa ja järjestyksessä kielioppisääntöjen eli syntaksin mukaisesti. Jotta lapset voisivat rakentaa ohjelmia, heidän täytyy osata käyttää ohjelmointikielien syntaksia. Syntaksin ja semantiikan (syntax & semantic) tietäminen on hyödyllistä, sillä se antaa mahdollisuuden sääntöjen ymmärtämiseen. (Morgado, Cruz & Kahn, 2010). Lapsilla on myös vaikeuksia löytää, korjata ja ymmärtää syntaksivirheitä esimerkiksi loogisista virheistä, kääntövirheistä ja ajonaikaisista virheistä, vaikka ne olisi opetettu lapsille (Lin et al., 2005).

Ohjelmoinnin opettamisessa on kohdattu muitakin, kuin tekniikkaan liittyviä haasteita. Yleinen ongelma on se, että lapsien ja opettajien ohjelmoinnin tekninen tietämys ei ole riittävällä tasolla. Siksi opettajat ja lapset eivät oikein tiedä, mitä kaikkea ohjelmoinnin avulla on mahdollista toteuttaa. Tämä on eritoten huomattavissa lukemaan opettelevien lapsien keskuudessa, koska silloin lapsien toiminta rajoittuu yleensä äärimmäisen yksinkertaiseksi. Ideaalitilanne olisi, jos lastentarhoissa ja esikouluilla olisi apuna ohjelmointiin suuntautunut asiantuntija, jolta lapset ja opettajat saisivat apua. Asiantuntija olisi keskittynyt koulutuksen tarpeisiin ja realiteetteihin, ja hän olisi oppilaiden sekä opettajien tukena. Kun opettajat ja oppilaat kohtaavat tilanteita, joissa heidän ohjelmointitaitonsa eivät ole riittävän hyvät, asiantuntija voisi auttaa ongelman ratkaisussa hyödyntäen eri tekniikoita ja lähestymistavoilla. Näin ohjelmoinnin opettaminen olisi osa suurempaa koulutuksellista kokonaisuutta, eikä pelkästään vain ohjelmoinnin oppimista. (Morgado et al., 2010.) Lisäksi opiskelijan ja taitavan asiantuntijan tai opettajan väliset useat, kasvotusten käydyt opetustuokiot olisivat tehokas tapa oppia ohjelmoimaan (Kahn, 1999).

Oppilaitoksissa opettajien tulisi suunnitella paremmin se, kuinka ohjelmoinnin opettamisesta saisi houkuttelevampaa oppilaiden kannalta. Oppilaiden kiinnostus ohjelmointiin ja tietojenkäsittelytieteeseen on ollut laskussa niin Yhdysvalloissa kuin muuallakin maailmaa. On olemassa useita syitä, miksi ohjelmointi ja tietojenkäsittelytiede eivät ole mielenkiintoisia, mutta yksi erityisen tärkeä syy on se, että tietojenkäsittelytiede ei herätä suurta kiinnostusta oppilaissa. Mikäli yläasteikäisiltä tytöiltä kysyttäisiin, kuinka moni heistä haluaisi oppia ohjelmoimaan, vain harvat nostaisivat todennäköisesti kätensä. Jos vastaavasti kysyttäisiin, että kuinka moni haluaa oppia tekemään animaatioelokuvia kuten Pixar ja Dreamworks tekevät, palaute ja osallistuminen olisi todennäköisesti hyvin erilaista ja paljon myönteisempää, vaikka animaatioelokuvien luominen ja ohjelmoinnin oppiminen voivat olla pohjimmiltaan samankaltaista toimintaa. (Kelleher & Pausch, 2007.)

Muualla, kuin englanninkielisissä maissa ohjelmoinnin opettamisessa on kohdattu haasteita oppilaiden kielitaidon vuoksi. Taiwanin peruskouluissa englannin kielen opettaminen alkaa kolmannelta luokalta lähtien, joten alakoulujen oppilaiden kielitaito

ei ole riittävällä tasolla hallitsemaan englanninkielisiä ohjelmointiympäristöjä, kääntäjien kommentteja ja komentoja (Lin et al. 2005). Voidaankin olettaa, että sama haaste toistuu myös muissa valtioissa, joissa englanti ei ole äidinkielenä.

5. Lapsille suunnattuja ohjelmointiympäristöjä

Lapsille ja noviisikäyttäjille on suunniteltu useita erilaisia ohjelmointiympäristöjä (Fessakis et al. 2013). Ohjelmointiympäristön tulisi olla niin vakuuttava ja mielekäs, että lapset olisivat hyvin motivoituneita käyttämään sitä, eivätkä he luovuttaisi ja lopettaisi sen käyttöä kesken (Wang, Wang & Liu, 2014). Lapsille suunnattu ohjelmointiympäristö motivoi lapsia ohjelmoinnin alkeiden oppimisessa, jos se on 1) työkalu, joka mahdollistaa lapsia tutkimaan ideoita, 2) on itsensä ilmaisun työkalu ja 3) toimii ponnahduslautana ohjelmoinnin maailmaan (Kelleher, Pausch & Kiesler, 2007). Lapsille suunnattujen ohjelmistoympäristöjen tulisi tukea niiden käyttäjiä yhteistyöhön, luovuuteen ja yhteisöllisyyteen. Sisällön pitäisi olla käyttäjiä haastavaa, mutta ei turhauttavaa ja käyttäjätuen pitäisi olla tehokasta ja joustavaa. (Kahn, 2004.) Hyviä teknisiä ominaisuuksia lapsille suunnattuihin ohjelmointiympäristöihin olisi yksinkertaistettu syntaksi, ohjelmoinnin perustuminen vedä ja liitä toimintaan (drag and drop) ja se, että ohjelmiston voisi suorittaa ilman kääntämistä ja linkittämistä (compiling and linking). Lisäksi hyvässä ohjelmointiympäristössä ohjelmointiparadigmat olisi mukautettu metaforiksi, esimerkiksi kuten Logossa, jossa käyttäjät ”puhuttelevat” virtuaalista kilpikonnaa ohjelmoidessaan. (Fessakis et al., 2013.) Tässä tutkimuksessa esittelen tarkemmin Logo, Electronic Blocks, ToonTalk, Scratch ja Alice – ohjelmointiympäristöjen keskeisiä periaatteita, koska ne toistuivat useissa eri tutkimuksissa.

Eräs lapsille suunnattu ohjelmointiympäristö on *Logo*. Logo ja sen useat eri versiot tarjoavat klassisen esimerkin ohjelmointikielestä, jonka päämääränä on saada lapset kiinnostumaan ohjelmoinnista. Lyhyesti todettuna Logo –kielessä käyttäjät piirtävät erilaisia kuviota näytölle ohjaamalla virtuaalista kilpikonnaa. Jopa ensimmäisillä Logo –versioilla voi tehdä näyttäviä piirustuksia, kun taas uudemmat versiot antavat käyttäjille vielä huomattavasti enemmän mahdollisuuksia. (Overmars, 2004.)

Wyeth ja Purchase (2000) ovat esitelleet *Electronic Blocks* –ohjelmointiympäristön, joka on 3-8 -vuotiaille lapsille suunnattu ohjelmointiympäristö. Se sisältää erilaisia palikoita, joissa on eri toimintoja kuten antureita, erilaista toiminnallisuutta ja logiikkaa. Näitä palikoita yhdistelemällä lapset voivat ”ohjelmoida” rakennelmia, jotka ovat vuorovaikutuksessa heitä ympäröivän ympäristön kanssa. Palikat tarjoavat nuorille lapsille ohjelmistoympäristön, joka tekee mahdolliseksi tutustumisen melko monimutkaisiin ohjelmointiperiaatteisiin. Palikoiden yksinkertainen syntaksi tarjoaa myös nuorille lapsille erilaisia mahdollisuuksia. Verrattuna perinteisten ohjelmointikielten käyttöön, palikoiden avulla lapsilla on mahdollista käyttää ja luoda yksinkertaisia ohjelmointirakenteita. Electronic Blocks –ohjelmointiympäristö on kaiken kaikkiaan kehityksellisesti asianmukainen ympäristö ongelmanratkaisuun, jakamaan suunnitelma hallittaviin osiin ja järjestelmälliseen heikkojen ratkaisuiden määrittämiseen. Electronic Block onkin ohjelmoinnin fyysinen olemus, koska siinä on ainutlaatuiset dynaamiset ja ohjelmoitavat ominaisuudet kuten tietokoneissa ilman niiden monimutkaisuutta. (Wyeth & Purchase, 2000.) Electronic Blocksin on tutkittu helpottavan ja parantavan lasten asemaa tulla päteviksi teknologian suunnittelijoiksi ja kehittäjiksi (Wyeth & Purchase, 2002). Electronic Blocksin tarkoitus on myös helpottaa lapsia tutustumaan dynaamisiin ja ohjelmoitaviin järjestelmiin ja oppimaan niiden suunnittelua, kehittämistä ja evaluointia (Wyeth, 2008). Aikaisemmat tutkimukset ovat osoittaneet, että 4-6-vuotiaat lapset pystyivät rakentamaan palikoilla erilaisia robotteja

ja autoja. Lapset olivat erittäin kiinnostuneita käyttämään palikoita, ja palikoiden avulla he oppivat yksinkertaista ohjelmointia ilman aikuisten suurta apua. (Wyeth & Wyeth, 2001.)

ToonTalk on animoitu ja toimintakeskeinen ohjelmointiympäristö (Morgado et al. 2010). Animaation tarkoitus on toimia kommunikointikeinona sekä ihmisille että tietokoneille (Kahn, 1999). ToonTalkissa ohjelmoija on asetettu virtuaalimaailmaan (virtuaalikaupunkiin) ja hän ohjelmoi siellä käyttäen hyväkseen maailman hahmoa. Ohjelmointi tapahtuu liikuttamalla hahmoa, poimimalla esineitä ja käyttämällä virtuaalisia työkaluja kuten pölynimuria ja pyörän pumppua. Lisäksi esimerkiksi pudottamalla numeroita toistensa päälle saa selville niiden summan. (Morgado et al., 2010.) Käyttäjät eivät siis ohjelmoi syöttämällä tekstiä tai järjestelemällä ikoneita vaan toimimalla virtuaalimaailmassa (Kahn, 1999).

Scratch on visuaalinen, erilaisten toiminnallisten osien (block) yhdistelyyn perustuva ohjelmointiympäristö, joka on suunnattu etenkin aloitteleville ohjelmoijille. Scratch – projekti koostuu muokattavasta taustasta (stage) ja useista liikuteltavista hahmoista (sprite). Jokainen objekti sisältää joukon kuvia, ääniä, muuttujia ja skriptejä. Ohjelmointi tapahtuu vetämällä toiminnallisia osia tekstiruutuun (scripting pane). Yksittäinen toiminnallinen osa tai joukko toiminnallisia osia voidaan ajaa kaksoisklikkaamalla. Scratch tehostaa median käsittelyä ja tukee nuorille mieleisiä ja kiinnostavia ohjelmointiaktiviteettejä kuten animoitujen tarinoiden ja pelien tekoa ja vuorovaikutteista työskentelyä. (Maloney, Peppler, Kafai, Resnick & Rusk, 2008.) Scratch on lapsille helposti ymmärrettävä ohjelmointiympäristö juurikin sen visuaalisuuden vuoksi. Lapset näkevät, mitä he ovat saaneet konkreettisesti aikaan ja tämä edesauttaa oppimista. (Lye & Koh, 2014.)

Alice on ohjelmointiympäristö, jonka avulla käyttäjät voivat luoda interaktiivisia 3D-virtuaalihahmoja. Ohjelmointi tapahtuu vetämällä ja pudottamalla (drag and drop) koodielementtejä toisten perään. Tällainen ohjelmonti poistaa syntaksivirheiden mahdollisuuden. Alicella tehtävät ohjelmat ovat animoituja, joten sen käyttäjien on helppo nähdä tekemänsä virheet. Alicen avulla käyttäjät saavat kokemusta useista ohjelmointikäsitteistä kuten silmukoista (loop), ehdoista (conditional), metodeista, parametreistä, muuttujista ja taulukoista. Alicesta on myös olemassa useita erilaisia versiota kuten Storytelling Alice. Storytelling Alicea käyttäneet lapset olivat motivoituneempia ohjelmoimaan kuin tavallista Alicea (Generic Alice) käyttäneet lapset. Storytelling Alice tukee tarinan luomista tarjoamalla 1) joukon korkean tason animaatioita, joiden avulla on mahdollista tehdä sosiaalisia hahmoja, jotka voivat olla vuorovaikutuksessa toisten hahmojen kanssa, 2) 3D-hahmoja ja maisemia auttamaan tarinan ideoinnissa ja 3) opasohjelman (tutorial), joka opastaa käyttäjiä tekemään Alice –ohjelmia käyttäen tarinoihin perustuvia esimerkkejä. (Kelleher et al., 2007.)

Päätäntä

Ohjelmointi tulee Suomen peruskoulujen opetussuunnitelmaan vuonna 2016. Sen johdosta ohjelmoinnin opettaminen lapsille on ollut Suomessa laajasti esillä niin mediassa kuin lehtien kirjoituksissakin. Tässä tutkimuksessa selvitettiin aiemmin raportoitujen tutkimusten pohjalta kuinka ohjelmointia on opetettu lapsille. Raportoinnissa keskityttiin ohjelmoinnin opettamisessa kohdattuihin haasteisiin ja opettamisen tarjoamiin hyötyihin. Tutkielmassa raportoidaan myös aineistossa esille nousseet viisi lapsille suunnattua ohjelmointiympäristöä ja niiden hyviä puolia ohjelmoinnin opettamisessa.

Ohjelmointi on tärkeä taito, joka kehittää korkeamman asteen ajattelun lisäksi ongelmaratkaisutaitoja, kognitiivista kehitystä ja algoritmista ajattelua. Ohjelmointi kehittää nykyään myös yhteisöllisyyttä ja ryhmätyöskentelyn taitoja. Ohjelmoinnin opettaminen on hyödyllistä ja tarpeellista nykypäivänä, koska tietotekniikan alan osaajien tarve on kasvanut useissa maissa.

Ohjelmoinnin opettamisessa eräs kohdattu haaste on lasten motivointi ja heidän mielenkiintonsa herättäminen ohjelmointia kohtaan. Oppilaitoksissa opettajien tulisi suunnitella paremmin ohjelmoinnin opettaminen, että se olisi oppilaiden kannalta mielenkiintoista. Näkemällä opetteluun vaivaa ja motivoimalla itsensä kuka tahansa voi oppia ohjelmoimaan, vaikka vaativamman tason ohjelmointi vaatii matemaattista logiikkaa, halua ja kykyä ongelmanratkaisuun sekä abstraktia ajattelua (Vivian, Falkner & Szabo, 2014). Ohjelmoinnin opettamista vaikeuttavat myös kielelliset haasteet, koska useat ohjelmointiympäristöt ovat englanninkielisiä. Oppilaitokset ovat lisäksi kohdanneet haasteita ohjelmoinnin opetustapahtumissa, koska tietotaito opettamista kohtaan ei ole ollut riittävää (Morgado et al., 2010).

Lapsille suunnattujen ohjelmointiympäristöjen tulisi olla mielekkäitä käyttää sekä lapsen kehitystaso huomioiden sopivan haastavia. Ohjelmistoympäristöjen tulisi motivoida lapsia ohjelmointiin. Niiden olisi hyvä olla teknisiltä ominaisuuksiltaan tarpeeksi yksinkertaisia, jotta pienetkin lapset voisivat käyttää niitä.

Lasten olisi hyvä oppia ohjelmointia luomalla erityisiä sovelluksia kuten videopelejä tai interaktiivisia tarinoita algoritmisten ohjelmointiharjoitusten ja tietorakenteiden sijaan. Lasten ohjelmoinnin oppimista edistää, että on olemassa lapsille tai noviisikäyttäjille tarkoitettuja ohjelmointiympäristöjä kuten Scratch ja Alice, mutta pelkästään ne eivät riitä. Esimerkiksi Scratch ja Alice -ohjelmointiympäristöillä on laaja, yli miljoonan jäsenen online-yhteisö, jossa lapset voivat ohjelmoida ja jakaa heidän aikaansaannoksiaan. Yhteisössä toteutetuista ja jaetuista ohjelmointiharjoituksista on tullut tehokas apuväline ohjelmoinnin oppimiseen. (Kafai & Burke, 2014.) Ohjelmoinnin opettamisessa lapsille voi olla parempi käyttää ennemmin visuaalisia ohjelmointiympäristöjä kuin perinteisiä kieliä, koska niiden avulla tarpeeton syntaksi vähenee ja komennot ovat lähempänä puhuttua kieltä. Visuaaliset ohjelmointiympäristöt kuten Scratch ja Alice ovat lapsille helposti ymmärrettäviä, koska niiden avulla lapset näkevät visuaalisesti, mitä he ovat tehneet ja niiden avulla lapset voivat luoda vuorovaikutteisia ohjelmia kuten animaatioita ja pelejä. (Lye & Koh, 2014.) Yksi hyvä tapa ohjelmoinnin opettamisen integroimiseksi ala –ja yläakouluikäisille on hyödyntää atk-opettajia, jotka toimivat yhteystyössä muun opetushenkilökunnan kanssa. Atk-

opettajat ja muut opettajat pystyisivät yhdessä hyödyntämään syvempää kokemusta ohjelmoinnin opettamisessa. (Rodger et al. 2012.) Jotta ylipäätään ohjelmointia voisi hyödyntää opettamisessa, tulee osata ohjelmoida. (Morgado et al. 2010.)

Tässä työssä käyttämästäni aineistosta kahdeksan artikkelia oli lapsista tehtyjä empiirisiä tutkimuksia liittyen erilaisiin ohjelmointitehtäviin –ja haasteisiin (Fessakis, et al., 2013; Kelleher et al., 2007; Lakanen et al., 2012; Lin et al., 2005; Maloney et al., 2008; Wang, 2014; Wyeth, 2008; Wyeth & Wyeth, 2001). Kolmessa artikkelissa tutkimuksen kohteena oli oppilaitokset ympäri maailmaa ja eri koulutusasteen opettajat (Rodger, 2010; Rodger, 2012; Vivian, 2014.) Loput artikkelit aineistostani käsittelivät ja esittelivät pääsääntöisesti eri ohjelmointiympäristöjä. (Hecker ,2014; Kafai & Burke, 2014; Kafai & Burke, 2013; Kahn, 1999; Kahn, 2004; Kelleher & Pausch, 2007; Liao & Bright, 1999; Lye & Koh, 2014; Morgado et al., 2010; Liukas & Mykkänen, 2014; Overmars, 2004; Wyeth & Purchase, 2000; Wyeth & Purchase, 2002).

Tutkittavien lapset olivat yhtä tutkimusta lukuun ottamatta (Maloney et al., 2008) peruskouluikäisiä lapsia. Kaksi tutkimusta käsitteli pelkästään päiväkoti-ikäisiä lapsia (Fessakis, et al., 2013; Wyeth & Wyeth, 2001), mutta pääsääntöisesti tutkimuskohteena olivat peruskoulun ala-kouluikäiset lapset. Tutkimuksen kohteena olivat niin pojat kuin tytötkin, paitsi yhdessä tutkimuksessa (Kelleher et al., (2007) käsiteltiin ainoastaan peruskoulun yläkouluikäisten tyttöjen suhtautumista ohjelmointiin.

Aineistossani tutkimukset tehtiin pääsääntöisesti lapsille ja noviisikäyttäjille suunnatuilla ohjelmistoympäristöillä, joita olen esitellyt luvussa 5, mutta poikkeuksiakin löytyi. Kahdessa tutkimuksessa käytettiin perinteisiä ohjelmointiympäristöjä ja ohjelmointikieliä kuten C# -kieltä ja Visual Basic –ohjelmointiympäristöä (Lakanen et al., 2012; Lin et al., 2005). Aineistoni eniten käytetyt ohjelmointiympäristöt olivat Alice ja sen eri versiot (Kelleher et al., 2007; Kelleher & Pausch, 2007; Rodger, 2010; Rodger, 2012 sekä Electronic Blocks (Wyeth, 2008; Wyeth & Wyeth, 2001; Wyeth & Purchase, 2000; Wyeth & Purchase, 2002.

Tutkimusmenetelmät olivat pääsääntöisesti empiirisiä. Lapsille annetut tehtävät olivat erilaisia, osassa tutkimuksissa lapset toimivat sekä ryhmässä (Lin et al., 2005) että yksin (Fessakis et al., 2013; Kelleher et al., 2007; Lakanen et al., 2012; Lin et al., 2005; Maloney et al., 2008; Wang, 2014; Wyeth, 2008; Wyeth & Wyeth, 2001). Tutkimusmenetelmien perusteella lapset saivat perusteet ohjelmointiin (Fessakis, et al., 2013; Kelleher et al., 2007; Lakanen et al., 2012; Lin et al., 2005; Maloney et al., 2008; Wang, 2014; Wyeth, 2008; Wyeth & Wyeth, 2001, tekivät joko ohjatut ja annetut harjoitukset (Fessakis, et al., 2013; Kelleher et al., 2007; Lakanen et al., 2012; Lin et al., 2005; Wang, 2014; Wyeth, 2008; Wyeth & Wyeth, 2001 tai tekivät omia ohjelmia (Kelleher et al., 2007; Maloney et al., 2008). Tutkimusryhmä havainnoi lasten työskentelyä ja he käyttivät muutamassa tutkimuksessa tarkkailuapunaan videokameroita ja äänityslaitteita. Lopuksi lasten, opettajien tai vanhempien kokemuksia kerättiin usein haastattelemalla tai kyselyillä.

Suomessa ohjelmoinnin opettaminen lapsille on melko uusi asia, eikä siitä ole laajasti tutkittu. Tämän tutkimuksen avulla pyrin luomaan käsityksen ohjelmoinnin opettamista lapsille. Ohjelmoinnin opettamista lapsille olisi hyvä tutkia tulevaisuudessa Suomessa enemmän, koska ohjelmointi tulee osaksi opetussuunnitelmaa vuonna 2016. Ohjelmoinnin opetuksessa tulisi keskittyä käytännön haasteisiin ja siihen, kuinka opettamisesta saisi tehokasta ja kiinnostavaa lapsille.

Tämän tutkimuksen aineisto koostuu artikkeleista. Tämän tutkimuksen luotettavuutta voitaisiin parantaa laajentamalla aineistoa entisestään, ottamalla mukaan enemmän tämän vuosikymmenen artikkeleita, sekä lisäämällä tutkimukseen myös empiirisiä osia.

Jatkotutkimusaiheita ohjelmoinnin opettamisesta löytyy laajasti, sillä ohjelmoinnin opetusta Suomessa on tutkittu verrattain vähän. Tämä kävi ilmi etsiessäni aineistoa tätä kirjallisuuskatsausta varten. Ohjelmoinnin opetuksen tullessa osaksi peruskoulua, avautuu tutkijoille hyödyllinen tilaisuus tutkia, kuinka ohjelmoinnin opettaminen toteutetaan uuden opetussuunnitelmamuutoksen jälkeen Suomessa. Jatkotutkimusta voidaan tehdä myös siitä, millaisia oppimistuloksia erilaisten ohjelmointiohjelmien käytöstä syntyy. Tämän tutkimuksen luonnollinen jatke olisi tutkia, kuinka edellä esitellyt ohjelmoinnin opetuksen avuksi suunnitellut sovellukset toimivat käytännössä. Ohjelmoinnin opettamista lapsille olisi hyvä tutkia määrällisiä tai laadullisia tutkimusmenetelmiä käyttäen pelkän kirjallisuuskatsauksen sijaan.

Lähteet

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5-6 years old kindergarten children in a computer programming environment: A case study. *Computers and Education, 63*, 87-97.

Hecker, D. E. (2005). Occupational employment projections to 2014. *Monthly Labor Review, 128*(11), 70–81.

Kafai, Y. B., & Burke Q. (2014). Computer programming goes back to school. *Education Week,* 61-65, Lainattu 13.4.2014, saatavilla: http://www.edweek.org/ew/articles/2013/09/0 1/kappan_kafai.html.

Kafai, Y. B., & Burke Q. (2013). The social turn in K-12 programming: Moving from computational thinking to computational participation. *SIGCSE 2013 - Proceedings of the 44th ACM Technical Symposium on Computer Science Education,* 603-608.

Kahn, K. (1999). A computer game to teach programming. *Proceedings of the National Educational Computing Conference*, 127-136.

Kahn, K. (2004). Toontalk – Steps towards ideal computer-based learning environments. In M. Tokoro & L. Steels (Eds.), *A Learning Zone of One's Own: Sharing Representations and Flow in Collaborative Learning Environments*, (pp. 259–270). Amsterdam, The Netherlands: IOS Press Inc.

Kelleher, C., & Pausch, R. (2007). Using Storytelling to motivate programming. *Communications of the ACM, 50*(7) 58-64.

Kelleher, C., Pausch, R., & Kiesler, S. (2007). Storytelling Alice motivates middle school girls to learn computer programming. *Proceedings of the SIGCHI conference on Human factors in computing systems*, 1455-1464, ACM.

Lakanen, A-J., Isomöttönen, V., & Lappalainen V. (2012). Life two years after a game programming course: longitudinal viewpoints on K-12 outreach. *SIGCSE'12 - Proceedings of the 43rd ACM Technical Symposium on Computer Science Education,* 481-486.

Liao, Y. C., & Bright, G. W. (1991). Effects of computer programming on cognitive outcomes: a meta-analysis. *Journal of Educational Computing Research, 7*(3), 251–266.

Lin, J. M., Yen, L., Yang, M., & Chen, C. (2005). Teaching computer programming in elementary schools: A pilot study. In *National Educational Computing Conference,* 1-8.

Liukas, L., & Mykkänen, J. (2014). Koodi2016 - ensiapua ohjelmoinnin opetukseen peruskoulussa. Lainattu 13.4.2014, saatavilla: http://www.koodi2016.fi.

Lye, S. Y., & Koh J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior, 41,* 51-61.

Maloney, J., Peppler, K., Kafai, Y. B., Resnick, M., & Rusk, N. (2008). Programming by choice: urban youth learning programming with Scratch. *SIGCSE'08 - Proceedings of the 39th ACM Technical Symposium on Computer Science Education,* 367-371.

Morgado, L., Cruz, M., & Kahn, K. (2010). Preschool cookbook of computer programming topics. *Australasian Journal of Educational Technology*, *26*(3), 309-326.

Overmars, M. (2004). Teaching computer science through game design. *Computer, 37*(4), 81-83.

Rodger, S., Dalis, M., Gadwal, C., Hayes, J., Li, P., & Liang L. (2012). Integrating computing into middle school disciplines through projects. *SIGCSE'12 - Proceedings of the 43rd ACM Technical Symposium on Computer Science Education,* 421-426.

Rodger, S. H., Bashford, M., Dyck, L., Hayes, J., Liang, L., Nelson D., & Qin H. (2010). Enchanging K-12 education with Alice programming adventures. *ITiCSE'10 - Proceedings of the 2010 ACM SIGCSE Annual Conference on Innovation and Technology in Computer Science Education,* 234-238.

Vivian, R., Falkner, K., & Szabo C. (2014). Can everybody learn to code? Computer science community perceptions about learning the fundamentals of programming. *Proceedings of the 14th Koli Calling International Conference on Computing Education Research,* 41-50.

Wang, D., Wang, T., & Liu Z. (2014). A Tangible programming tool for children to cultivate computational thinking. *The Scientific World Journal,* 1-10.

Wyeth, P. (2008). How young children learn to program with sensor, action and logic blocks. *The Journal of the Learning Science 17*(4), 517-550.

Wyeth, P., & Purchase H.C (2002). Tangible programming elements for young children. In *CHI'02 Extended Abstracts on Human Factors in Computing Systems*, 774-775, ACM.

Wyeth, P., & Purchase H.C (2000). Programming without a computer: A new interface for children under eight. *First Australasian Conference on User Interface, AUIC 2000,* 141-148.

Wyeth, P., & Wyeth, G (2001). Electronic blocks: Tangible programming elements for preschoolers. *IFIP TC. 13 International Conference on Human-Computer Interaction, 1,* 496-503.

Liitteet A. Taulukko

Kirjoittaja	**Käytetty ohjelmointiympäristö**	**Ikä**	**Tutkimusmenetelmät**
Fessakis, et al. (2013)	IWB, LOGO	5-6 –vuotiaista.	Empiirinen: lapset ratkovat ongelmia opettajien ohjaamina ja ryhmässä.
Hecker (2014)	-	-	-
Kafai & Burke (2014)	Scratch	-	-
Kafai & Burke (2013)	Scratch	-	-
Kahn (1999)	ToonTalk	-	-
Kahn (2004)	ToonTalk	-	-
Kelleher et al. (2007)	Storytelling Alice, Generic Alice	Yläkouluikäisistä tytöistä (12-vuotiaista).	88 tyttöä. Empiirinen: yksi noin kahden tunnin työpaja, jossa lapset tekivät opastetun tehtävän ja loivat ohjelman.
Kelleher & Pausch (2007)	Storytelling Alice	-	-
Lakanen et al. (2012)	Jypeli, C#	12-14 –vuotiaista.	Viikon pituinen peliohjelmointikurssi. Ohjatut oppitunnit. Loppuhaastattelu.
Liao & Bright (1999)	-	-	-
Lin et al. (2005)	Stagecast Creator, HANDS, Visual Basic	10-12 –vuotiaista.	Ohjelmointikesäleiri; 81 lasta, kolme tuntia ohjelmoinnin opettamista, viitenä päivänä viikossa kahden viikon ajan. Itsenäistä ja ryhmätyöskentelyä. Luokkahuonehavainnointi ja kyselytutkimus vanhemmille ja lapsille.
Lye & Koh (2014)	-	-	-
Maloney et al. (2008)	Scratch	8-18 –vuotiaista.	Analysoivat lasten tekemiä Scratch -ohjelmia. Haastattelivat ohjelmointikerhon (Computer Clubhouse) jäseniä
Morgado, et al. (2010)	ToonTalk	-	-
Liukas & Mykkänen (2014)	-	-	-
Overmars (2004)	Logo, Lego	-	-
Rodger et al. (2010)	Alice	-	Kesäkurssi opettajille, jossa perehdytettiin opettajia Aliceen ja käyttämään sitä osana opettamista.
Rodger et al. (2012)	Alice	-	Peruskoulujen opettajien perehdytystä ohjelmointiin ja

			Alice –oppaiden teko opettajien kanssa.
Vivian (2014)	-	-	Kysely eri maiden ja eri oppiasteiden opettajille ohjelmoinnista ja sen oppimisesta.
Wang et al. (2014)	T-Maze	5-9 –vuotiaista.	Empiirinen: ohjattujen harjoitusten teko ja haastattelu
Wyeth (2008)	Electronic Blocks	7-8 – vuotiaista.	Empiirinen: 12 lasta osallistui harjoitukseen. Tehtävät, käyttäjien tarkkailu videokameralla ja äänittämällä.
Wyeth & Purchase (2000)	Electronic Blocks	-	-
Wyeth & Purchase (2002)	Electronic Blocks	-	-
Wyeth & Wyeth (2001)	Electronic Blocks	4-6 –vuotiaista.	Empiirinen: 21 lasta osallistui kuuteen kokeelliseen harjoitukseen. Käyttäjien tarkkailu videokameralla ja äänittämällä.

Huomio. Viiva (-) tarkoittaa ettei aineistosta käynyt ilmi kyseistä kohtaa.

Liitteet B. Käyttämätön aineisto

Baldwin, L. P., & Kuljis, J. (2001). Learning programming using program visualization techniques. *System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference on,* 8 pp.

Bancroft, P., & Roe, P. (2006). Program annotations: Feedback for students learning to program. *Proceedings of the 8th Australasian Conference on Computing Education-Volume 52,* 19-23.

Bell, S., Frey, T., & Vasserman, E. (2014). Spreading the word: Introducing pre-service teachers to programming in the K12 classroom. *Proceedings of the 45th ACM Technical Symposium on Computer Science Education,* 187-192.

Burke, Q., & Kafai, Y. B. (2010). Programming & storytelling: Opportunities for learning about coding & composition. *Proceedings of the 9th International Conference on Interaction Design and Children,* 348-351.

Carter, E., Blank, G., & Walz, J. (2012). Bringing the breadth of computer science to middle schools. *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education,* 203-208.

Druin, A. (2002). The role of children in the design of new technology. *Behaviour and Information Technology, 21*(1), 1-25.

Esteves, M., Fonseca, B., Morgado, L., & Martins, P. (2011). Improving teaching and learning of computer programming through the use of the second life virtual world. *British Journal of Educational Technology, 42*(4), 624-637.

Gibson, J. P. (2003). A noughts and crosses java applet to teach programming to primary school children. *Proceedings of the 2nd International Conference on Principles and Practice of Programming in Java,* 85-88.

Goldberg, D., Grunwald, D., Lewis, C., Feld, J., Donley, K., & Edbrooke, O. (2013). Addressing 21st century skills by embedding computer science in k-12 classes. *Proceeding of the 44th ACM Technical Symposium on Computer Science Education,* 637-638.

Guzdial, M. (2004). Programming environments for novices. *Computer Science Education Research, 2004*, 127-154.

Hood, C. S., & Hood, D. J. (2005). Teaching programming and language concepts using LEGOs®. *ACM SIGCSE Bulletin, 37*(3), 19-23.

Isomöttönen, V., Lakanen, A., & Lappalainen, V. (2011). K-12 game programming course concept using textual programming. *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education,* 459-464.

Kafai, Y. B. (1996). Software by kids for kids. *Communications of the ACM, 39*(4), 38-39.

Kahn, K. (1996). Toontalk TM—an animated programming environment for children. *Journal of Visual Languages & Computing, 7*(2), 197-217.

Kahn, K. (2004). The child-engineering of arithmetic in ToonTalk. *Proceedings of the 2004 Conference on Interaction Design and Children: Building a Community,* 141-142.

Kay, J. S., & Moss, J. G. (2012). Using robots to teach programming to K-12 teachers. *Frontiers in Education Conference (FIE), 2012,* 1-6.

Kuljis, J., & P Baldwin, L. (2000). Visualisation techniques for learning and teaching programming. *CIT.Journal of Computing and Information Technology, 8*(4), 285-291.

Lakanen, A., Isomöttönen, V., & Lappalainen, V. (2014). Understanding differences among coding club students. *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education,* 159-164.

Lau, W. W., & Yuen, A. H. (2009). Exploring the effects of gender and learning styles on computer programming performance: Implications for programming pedagogy. *British Journal of Educational Technology, 40*(4), 696-712.

Mayer, R. E. (2004). Should there be a three-strikes rule against pure discovery learning? *American Psychologist, 59*(1), 14.

Morgado, L., Cristóvao-Morgado, R., Cruz, M. G. B., & Kahn, K. (2005). Embedding computer activities into the context of preschools. *CHALLENGES 2005-IV Conferência Internacional Em Tecnologias De Informação e Comunicação Na Educação, 11, 12 e 13 De Maio De 2005,* 8 pages.

Pane, J. F. (1998). Designing a programming system for children with a focus on usability. *CHI 98 Cconference Summary on Human Factors in Computing Systems,* 62-63.

Rader, C., Brand, C., & Lewis, C. (1997). Degrees of comprehension: Children's understanding of a visual programming environment. *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems,* 351-358.

Roschelle, J. M., Pea, R. D., Hoadley, C. M., Gordin, D. N., & Means, B. M. (2000). Changing how and what children learn in school with computer-based technologies. *The Future of Children*, 76-101.

Sánchez-Ruíz, A. J., & Jamba, L. A. (2008). FunFonts: Introducing 4 th and 5 th graders to programming using squeak. *Proceedings of the 46th Annual Southeast Regional Conference on XX,* 24-29.

Smith, D. C., Cypher, A., & Tesler, L. (2000). Programming by example: Novice programming comes of age. *Communications of the ACM, 43*(3), 75-81.

Wyeth, P. (2008). How young children learn to program with sensor, action, and logic blocks. *The Journal of the Learning Sciences, 17*(4), 517-550.

Yeum, Y., Kwon, D. Y., Yoo, S., Lee, W., Kanemune, S., & Kuno, Y. (2007). Multilingual programming language environments for intercultural collaboration of

programming education in K-12. *Convergence Information Technology, 2007. International Conference on,* 1708-1713.