Karoliina Sormunen

6.-luokkalaisten ohjelmoinnin osaaminen alakoulun päättyessä

Tietotekniikan pro gradu -tutkielma

21. toukokuuta 2019

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Karoliina Sormunen

Yhteystiedot: karoliina.j.savolainen@student.jyu.fi

Ohjaajat: Antti Ekonoja ja Antti-Jussi Lakanen

Työn nimi: 6.-luokkalaisten ohjelmoinnin osaaminen alakoulun päättyessä

Title in English: Programming skills of 6th graders at the end of Finnish primary school

Työ: Pro gradu -tutkielma

Opintosuunta: Koulutusteknologia

Sivumäärä: 72+17

Tiivistelmä: Tutkimuksessa kartoitettiin kuudennen luokan oppilaiden osaamista ohjelmoinnissa alakoulun päättyessä. Lisäksi tutkimuksessa tutkittiin ohjelmoinnin opetuskäytänteitä. Tutkimukseen valikoitui mukaan kolme eri kuudennetta luokkaa Keski-Suomesta, Pirkanmaalta ja Pohjois-Savosta. Tutkimuksessa opettajia haastateltiin ohjelmoinnin opetuksesta ja oppilaat suorittivat Scratch-ohjelmointiympäristössä kolmea eri konseptia mittaavan testin. Tyypiltään tutkimus oli teoriasidonnainen tutkimus, jonka aineistoa analysoitiin laadullisin ja määrällisin keinoin. Tutkimuksen tulokset osoittivat oppilaiden taitotason yleisesti ottaen heikoksi ohjelmoinnissa. Erityisen heikkoa osaaminen oli ehtolauseiden osaamista mittaavissa tehtävissä. Kansallinen opetussuunnitelma ei linjaa tarkasti ohjelmoinnin osaamisen tavoitteita, joten osaamistavoitteiden selkeyttämiseksi tarkempi määrittely ohjelmoinnin sisältöjen hallitsemisesta valtakunnallisesti olisi tarpeen. Tällöin edellytykset tekstuaalisen ohjelmoinnin pariin siirtymiselle yläkoulun puolelle olisivat paremmat.

Avainsanat: Ohjelmoinnin opetus, OPS 2014, graafinen ohjelmointiympäristö, tieto- ja viestintätekniikka, TVT, ohjelmointi, opetus, opetussuunnitelma, Scratch

Abstract: This research studied programming skills of 6th graders at the end of Finnish primary school. Additionally principles of teaching programming were also inspected. Three 6th grades from Central Finland, Pirkanmaa and Northern Savo were chosen to this research. Primary school teachers were interviewed and their pupils participated exam testing

three conspets of programming in Scratch programming environment. This thesis theory was research based and research material was analysed both qualitatively and quantitatively. The results show programming skills of 6th graders to be generally poor, especially in excercises measuring understanding of conditional statements. National curriculum does not define a spesific objective about programming at the end of primary school. More detailed objectives in national curriculum would make it easier to transfer from primary school and visual programming environment to upper level school and text-based programming.

Keywords: programming teaching, visual programming environment, information and communications technology, programming, national curriculum, teaching, Scratch, computing education research

Termiluettelo

Graafinen ohjelmointiympäristö	Lapsille tarkoitettu ohjelmointiympäristö, jossa koodaaminen tapahtuu koodiblokkeja järjestelemällä ja suorittamalla koodi. Esimerkiksi Scratch.
Java	Ohjelmointikieli.
POPS2014	Opetushallituksen määräämä Perusopetuksen opetussuunnitelman perusteet 2014.
Python	Ohjelmointikieli.
STEM	Science, technology, engineering, mathematics. Kääntyy suomeksi LUMA (luonnontiede-matematiikka).
TVT	Tieto- ja viestintätekniikka.

Kuviot

Kuvio 1. Suomenkielinen Scratch 3.0 aloitusnäkymä. 14
Kuvio 2. Code.orgin silmukkatehtävän aloitusnäkymä. 15
Kuvio 3. Scratchin opettajatunnuksen takaa näkyvä näkymä. 27
Kuvio 4. Vasemmalla Oppilas14:n edistynyt ratkaisu Toisto1-tehtävään. Oikealla Oppilas7:n alkeellinen ratkaisu tehtävään. 33
Kuvio 5. Oppilas27:n sisäkkäisten silmukoiden ratkaisu Toisto2-tehtävässä. 34
Kuvio 6. Oppilas33:n ratkaisu Toisto3-tehtävässä. 35
Kuvio 7. Oppilas10:n ratkaisu Muuttujat2-tehtävään. 36
Kuvio 8. Oppilas34:n ratkaisu Ehtolause1-tehtävään. 37
Kuvio 9. Oppilas19:n ratkaisu Ehtolause2-tehtävään. 38
Kuvio 10. Muuttuja 3-tehtävän aloitusnäkymä 73

Taulukot

Taulukko 1. Oikeiden vastausten suhteellinen osuus osallistujien lukumäärästä toistotehtävissä (n=54). 35
Taulukko 2. Oikeiden vastausten osuus osallistujien lukumäärästä muuttujatehtävissä (n=54). 36
Taulukko 3. Oikeiden vastausten osuus osallistujien lukumäärästä ehtolausetehtävissä (n=54). 38
Taulukko 4. Oikeiden vastausten suhteellinen osuus osallistujien lukumäärästä. 39
Taulukko 5. Vastaukset luokiteltuina 40
Taulukko 6. Oikeiden vastausten suhteellinen osuus palautetuista vastauksista kouluittain. 40
Taulukko 7. Vähintään yhden vastauksen palauttaneiden osallistujien määrä luokittain . . . 41

Sisältö

1 JOHDANTO 1

2 OHJELMOINNIN OPETUS 3
2.1 Ohjelmoinnin opetuksen tarve 3
2.2 Ohjelmoinnin oppimisen ja opetuksen haasteet 5
2.3 Algoritminen ajattelu 6
2.4 Ohjelmoinnin opetuksen pedagogiikkaa 7
2.5 Aikaisempi tutkimus ohjelmoinnista opetuskontekstissa 9
2.6 Pelillinen oppiminen ja opetusrobotit 10
2.7 Graafinen ohjelmointiympäristö 11
2.8 Opettajien valmiudet opettaa ohjelmointia 16

3 OPETUSSUUNNITELMA 2014 JA OHJELMOINTI 17
3.1 Laaja-alainen osaaminen: tieto- ja viestintäteknologinen osaaminen (L5) 18
3.2 Vuosiluokat 1-2 19
3.3 Vuosiluokat 3-6 19
3.4 Vuosiluokat 7-9 20

4 TUTKIMUSASETELMA 22
4.1 Tutkimuskysymykset 22
4.2 Tutkimusstrategia 23
4.3 Tutkimuksen kulku ja tutkimukseen osallistujat 24

5 TUTKIMUSTULOKSET 30
5.1 Opettajien haastattelut 30
5.2 Toistorakenteen osaaminen 32
5.3 Muuttujien osaaminen 35
5.4 Ehtolauseiden osaaminen 37
5.5 Yhteenveto 39

6 JOHTOPÄÄTÖKSET JA POHDINTA 42
6.1 Keskeisimmät johtopäätökset 42
6.2 Taitotasoon vaikuttavat seikat 46
6.3 Tutkimuksen luotettavuus ja rajoitukset 48
6.4 Pohdinta 51
6.5 Jatkotutkimus 53

LÄHTEET 55

LIITTEET 66
A Liite: Tutkimuslupa 66
B Liite: Tietosuojailmoitus 67
C Liite: Haastattelurunko 71
D Liite: Koekysymykset 71

E Liite: Oikeat vastaukset 74

1 Johdanto

Syksyllä 2016 uuteen opetussuunnitelmaan (Opetushallitus 2014) tuli ohjelmointi osaksi opetuksen tavoitteita. Tieto- ja viestintätekniikan (TVT) opetus on myöskin mainittu osana laaja-alaisia osaamistavoitteita (L5, Tieto- ja viestintäteknologinen osaaminen).

Tämän tutkimuksen tavoitteena oli kartoittaa kuudennen luokan oppilaiden kevään 2019 ohjelmoinnin osaamista. Valtakunnallisella tasolla tarkasteltuna keväällä 2019 alakoulun päättävät oppilaat olivat opiskelleet ohjelmointia uuden opetussuunnitelman myötä kolme lukuvuotta. Ohjelmoinnissa siirrytään yläkoulun ohjelmoinnin opetuksessa tekstuaaliseen ohjelmointiin, joten oli mielenkiintoista selvittää, että minkälaisilla edellytyksillä oppilaat siirtyvät yläkoulun puolelle. Lisäksi tutkimuksessa selvitettiin ohjelmoinnin opetustapoja. On tärkeää tutkia oppilaiden taitotasoa ohjelmoinnissa, sillä aiempaa tutkimusta juuri tästä aiheesta ei ole tehty, vaikka ohjelmointi on tällä hetkellä hyvinkin ajankohtainen tutkimusaihe. Valtakunnallinen OPS linjaa ohjelmoinnin osaamisesta hyvin niukasti, *"Oppilas osaa ohjelmoida toimivan ohjelman graafisessa ohjelmointiympäristössä"* (Opetushallitus 2014, 239). Muutamat paikallisen tason opetussuunnitelmat mainitsevat oppilaiden osaamistavoitteiksi ehtolauseiden ja silmukkarakenteiden käyttämisen (Kuopion kaupunki 2019; Jyväskylän kaupunki 2016).

Ohjelmoinnilla tarkoitetaan toimintaohjeiden antamista halutun toiminnan suorittamiseksi (Hyvönen, Lappalainen ja Lakanen 2013, 2). Joissain määritelmissä tietokone on mukana, esimerkiksi Saeli ym. (2011, 79) määrittelevät ohjelmoinnin listaksi käskyjä, jotka ovat kirjoitettu tietokoneen ymmärtämällä kielellä ja jotka tietokone suorittaa siinä järjestyksessä, jossa ohjeet on tietokoneelle annettu. Hyvönen, Lappalainen ja Lakanen (2013, 2) kirjoittavat, että ohjelmoinninkaltaista toimintaa on ihmisten arkielämässä hyvinkin paljon: yksinkertaisimmillaan mikroaaltouunin käyttäminen on alkeellista ohjelmointia, kun uunille annetaan ohjeet kuinka kauan ja millä teholla lämmitystä tulee tehdä.

Usein ohjelmoinnin (engl. *programming*) lisäksi puhutaan koodauksesta (engl. *coding*). Nämä tarkoittavat molemmat tieto- ja viestintäteknologista (TVT) laitetta hyödyntäen tehtyä ohjelmointitoimintaa. Usein ohjelmoinnista puhuttaessa nousee ilmi käsitteet algoritminen

ajattelu, laskennallinen ajattelu ja automatisointiajattelu. Tutkimuksessa puhutaan algoritmisesta ajattelusta viitatessa englanninkieliseen käsitteeseen *computational thinking*. Käsitteiden moninaisuus johtuu siitä, että suomenkielessä ei ole vakiintunutta käsitettä, vaan kontekstista riippuen eri käsitteitä voidaan käyttää kuvaamaan samaa asiaa.

Ohjelmoinnin opetuksen tarvetta on perusteltu eri tavoin. Mannila ym. (2014) mukaan ohjelmoinnin ja tietotekniikan opiskelun on todettu kehittävän nuorten algoritmistä ja kriittistä ajattelukykyä sekä kehittävän nuoria teknologian luojiksi, eikä pelkästään käyttäjiksi. Lisäksi ohjelmoinnin ja tietotekniikan opiskelun taitoja tarvitaan entistä enemmän tulevaisuudessa. (Mannila ym. 2014, 1.) Teknologia on yhä suurempi osa nuorten elämää, ja tämän takia nuorten olisi hyvä oppia jo peruskoulussa algoritmistä ajattelua ja harjaannuttaa ongelmanratkaisutaitoja (Barr ja Stephenson 2011, 49). Ohjelmoinnin opetusta on perusteltu myös lisääntyvällä työvoimatarpeella (EU-komissio 2018).

Tämän pro gradu -tutkielman luvussa 2 käydään läpi ohjelmoinnin opetuksen taustoja, pedagogiikkaa sekä ohjelmoinnin ja graafisten ohjelmointiympäristöjen opetuskäyttöä aiemman tutkimuksen näkökulmasta. Luvussa 3 perehdytään ohjelmoinnin sisältöihin opetussuunnitelman näkökulmasta ja tarkastellaan ohjelmoinnin opetuksen sisältöjä paikallisen tason opetussuunnitelmissa. Luvussa 4 kuvataan tämän tutkimuksen tutkimusasetelma. Luvussa esitellään tutkimuskysymykset, -strategia, tutkimuksen kulku ja tutkimuksessa käytetty koe. Lisäksi luvussa kerrotaan aineiston keräämisestä ja aineiston analysoinnista. Luvussa 5 puolestaan esitellään tutkimustuloksia tehtäväkohtaisesti ja käsitellään vastauksia kokonaisvaltaisesti numeroiden perusteella. Luvussa 6 saatuja tuloksia peilataan aiempaan tutkimukseen ja kootaan yhteen keskeisimmät tutkimustulokset. Lisäksi luvussa esitellään muutamia jatkotutkimusideoita.

2 Ohjelmoinnin opetus

Tässä luvussa esitellään ohjelmoinnin opetuksen historiaa ja ohjelmoinnin oppimisen haasteita. Lisäksi luvussa kerrotaan algoritmisesta ajattelusta, aiemmasta ohjelmoinnin opetuksen tutkimuksesta ja ohjelmoinnin opetuksen opetusympäristöistä.

Ohjelmoinnin opetus on alkanut jo 1960-luvulla Logo-ohjelmointikielellä, joka suunniteltiin tarjoamaan *"käsitteellinen perusta matemaattisten ja loogisten ajattelutapojen opettamiselle ohjelmien ideoiden ja toimintojen kannalta"* (Kalelioğlu 2015, 200). Sen syntaksiherkkyyttä kritisoitiin ja tämän takia Logo turhautti oppilaita (Järvinen 1998).

Myöhemmin on kehitetty esimerkiksi Empirica Control-ohjelmointityökalu, joka pohjautui tekstuaalisen ohjelmoinnin sijasta ikonimuotoiseen ohjelmointiin. Sen kehittäjät uskoivat, että visuaalinen ohjelmointi ikonipohjaisena on helpompaa kuin tekstimuotoinen ohjelmointi. (Lavonen, Meisalo ja Lattu 2001, 23–24.)

2.1 Ohjelmoinnin opetuksen tarve

Ohjelmoinnin opetusta on perusteltu yhteiskunnan lisääntyvällä tiedon määrällä ja oppilaiden ajattelutaitojen harjaannuttamisella. Ohjelmoinnin opettelu on aluksi algoritmisen ajattelutavan vahvistamista. Lapset tällä hetkellä altistuvat entistä enemmän suurelle määrälle tietoa ja dataa, jota ei ole jäsennelty tai muulla tavoin yhdistetty olemassa olevaan tietoon, eivätkä opettajat yleensä joidenkin tutkijoiden mukaan ajanpuutteen vuoksi ohjaa oppilaita yhdistelemään niitä tietoja toisiinsa, joita oppilaat saavat (Flórez ym. 2017, 836; Barr ja Stephenson 2011, 49). Ohjelmoinnin opetus auttaakin Flórez ym. (2017, 836) mukaan oppilasta löytämään piilotettua tietoa siitä materiaalista, jota tämä saa. Ohjelmointia opetellessaan oppilaat eivät ainoastaan törmää ongelmien ratkaisun haasteisiin, vaan he samalla haastavat itseään opetellessaan ohjelmointikielen syntaksia ja semantiikkaa: ratkaisut pitää esittää siinä muodossa, jotta tietokone ymmärtää ratkaisun (Mannila, Peltomäki ja Salakoski 2006, 211).

Ohjelmoinnista puhuttaessa keskusteluun nostetaan hyvin usein tulevaisuuden taidot (eng. *21st century skills*). Binkley ym. (2012) määrittelevät tulevaisuuden taidot useilta eri taitojen

saralta. Yhtenä tulevaisuuden taitona mainitaan työvälineiden hallinta, johon sisältyy informaation lukutaito ja tietotekniikan käyttötaidot. On todettu, että enää tietotekniikka ei ole tärkeää ainoastaan heille, jotka haluavat työskennellä tietotekniikan parissa, vaan tietotekniikan ymmärtäminen on tärkeää kaikille (Mannila ym. 2014, 1). Koulun täten tulee tukea oppilaita enemmän oppimaan ns. tulevaisuuden taitoja. EU-komission (2018) linjauksessa todetaan, että ongelmanratkaisutaidot, tiimityöskentely ja analyyttinen ajattelu ovat 2000-luvun tulevaisuuden taitoja, ja ohjelmoinnin opiskelu auttaa näiden tulevaisuuden taitojen opiskelussa. (EU-komissio, 2018.) Ohjelmointia on luonnehdittu myös uudeksi kirjallisuuden muodoksi, luovuuden kehittäjäksi ja lapsen omien ideoiden kokemusten laajentajaksi (Mannila ym. 2014, 4). Ohjelmointi ja algoritmisen ajattelun taidot nähdään nykyään kaikilla nyky-yhteiskunnan osa-alueilla tärkeinä (Heintz, Mannila ja Färnqvist 2016).

Ohjelmoinnin opetusta on perusteltu myös lisääntyvällä työvoimatarpeella. EU-komissio arvelee linjauksessaan, että vuonna 2020 EU:n alueella löytyy jopa 85 000 avointa IT-alan työpaikkaa ja rohkaiseekin jäsenmaitaan edistämään nuorten oppilaiden ohjelmointimahdollisuuksia (EU-komissio 2018). Myös OPH:n (2009) selvityksessä todetaan, että työpaikkojen määrä kasvaa useamman vuoden ajan IT-alalla (Hanhijoki ym. 2009, 157–159).

Opetussuunnitelma linjaa perusopetuksen tarjoavan laajan yleissivistyksen perustan oppilaalle. Opetussuunnitelmassa todetaankin, että ympäröivän maailman muutos vaikuttaa myös koulun toimintaan (Opetushallitus 2014, 18). Tulevaisuuden taidot OPS:ssa summautuvat yleisesti laaja-alaisissa oppimistavoitteissa (Opetushallitus 2014, 20). OPS:ssa on mainittu tieto- ja viestintäteknologian käyttötaidot omana laaja-alaisena osaamistavoitteena, mutta myös tieto- ja viestintätekniikkaa tulee hyödyntää oppimisympäristössä, jotta oppilaiden yhteisöllisen työskentelyn taidot vahvistuvat (Opetushallitus 2014, 23; 29). Tarkemmin OPS:n sisällöistä kerrotaan luvussa 3.

Ohjelmoinnin opetuksen tavoitteita on pohdittu tutkimuksen kautta. Saeli ym. (2011, 78) pohtivat, että ohjelmoinnin perimmäinen tarkoitus on ratkaista ongelma ja käyttää ohjelmaa ratkaisuna. Heidän mukaansa ohjelmoidessa voidaan erottaa kahta erilaista tietämystä: ohjelman luomista ja ohjelman ymmärtämistä. Ohjelmaa luodessaan ohjelmoija analysoi ongelman, tuottaa algoritmin, joka ratkaisee ongelman ja tuottaa algoritmista ohjelmakoodin. Tämän myötä oppilas pitäisi opettaa ongelmanratkaisun prosessiin, reflektoimaan tätä pro-

sessia ja kehittämään algoritmista ajatteluaan. Ohjelmaa ymmärtäessään ohjelmoija pystyy demonstroimaan, kuinka tehty ohjelma toimii. Saeli ym. (2011, 79) mukaan on tärkeää opettaa yläkoulussa ohjelmakehittelyä ja ohjelmien ymmärtämistä ja Flórez ym. (2017, 837) esittävätkin, että ohjelmoinnin opetus tulisi sisällyttää opetukseen siten, jotta jokainen oppilas oppisi näitä taitoja jo alakoulussa - esimerkiksi osana matematiikan oppiainetta. Lavosen, Meisalon ja Latun (2001, 32) mielestä yleisesti tarvitaan ohjelmointitaitoja, vaikka graafisessa ohjelmointiympäristössä opittuja, jotta ihmiset ymmärtäisivät ideoitaan. Heidän mielestään luovien ajattelutapojen tulisi olla käytettyjä yleisesti, jotta opitaan ajattelemaan positiivisesti, antamaan jäsenneltyä kritiikkiä, kysymään relevantteja kysymyksiä ja auttamaan toisia kehittelemään omia ideoitaan.

2.2 Ohjelmoinnin oppimisen ja opetuksen haasteet

Ohjelmointia opetettaessa on tärkeää tiedostaa yleisiä ohjelmoinnin oppimisen ja opettamisen haasteita. Alkeisopetuksessa yleinen ongelma on orientoituminen, eli sen näkyväksi tekeminen, mihin ohjelmointia voidaan käyttää ja mitkä ovat ohjelmoinnin oppimisen hyödyt. Matala motivaatio vaikuttaa oppimiseen negatiivisesti. (Anderson ym. 2014, 465.) Kun ohjelmointikurssien opetuksessa käytetään tarpeeksi laajasti samaistuttavaa sisältöä ja yhteistyötä vaativia tehtäviä, kuten pariohjelmointia, läpipääsyprosentit ja oppilaiden motivaatio oppimiseen ovat olleet paremmat (Flórez ym. 2017, 841).

Yleisesti oppilailla ohjelmoinnin oppimisessa on ongelmina laajojen kokonaisuuksen ymmärtäminen. Tietyn ongelman ratkaisevan ohjelman suunnittelu, toiminnallisuuden muuttaminen proseduureiksi sekä virheiden löytäminen omasta ohjelmasta ovat useammissa tutkimuksissa nousseet vaikeimmiksi asioiksi oppia (Lahtinen, Ala-Mutka ja Järvinen 2005; Piteira ja Costa 2013). Ylempien vuosiluokkien ohjelmoinnissa on ollut puolestaan erilaisia haasteita. Oppilaat, jotka aloittavat ohjelmointiharjoittelun tekstuaalisista ohjelmointikielistä, kuormittuvat paljon kognitiivisesti ja tutkimukset ovat osoittaneet, että kahdenkin vuoden ohjelmointiharjoittelun jälkeen oppilaat voivat joutua ponnistelemaan kovasti voidakseen olla hyviä ohjelmoinnissa (Flórez ym. 2017, 838). Ohjelmointi vaatiikin useiden taitojen yhteensovittamista ja useiden asioiden muistamista samaan aikaan, esimerkiksi syntaksin yksityiskohtien ja ohjelmointikielen semantiikan pohtimista sekä ratkaisun määrittelyä tietoko-

neen näkökulmasta (Pane ja Myers 1996). Semantiikka voi olla haasteellinen opetuksessa: erilaisilla ohjelmointikielillä eli erilaisella syntaksilla voidaan ratkaista saman semantiikan tehtäviä. (Tan, Ting ja Ling 2009, 44-45; Saeli ym. 2011, 79–80.)

Oppilailla on ollut ohjelmoinnin opiskelun aikana ongelmia sisäistää myös muuttujan, muuttujan alustamisen ja samanaikaisuuden käsitteitä (Meerbaum-Salant, Armoni ja Ben-Ari 2013, 75; Kuittinen ja Sajaniemi 2004; Kohn 2017). Muuttujan käsitteen ymmärtämisessä ongelmana on ollut oppijan tiukka tulkinta matemaattisesti esimerkiksi sijoitusoperaatiosta `x = x+1`. Pane ja Myers (1996, 32) puolestaan toteavat, että silmukat on yleisesti hankala aihealue ohjelmoinnissa aloitteleville ohjelmoijille.

Flórez ym. (2017) nostavat esiin muutamia huomioita ohjelmoinnin opettamisen järjestämisestä: jotta saadaan kehitettyä kriittistä ajattelutaitoja, opetus tulee olla tarkasti määritetty oppimis- ja opetustavoitteiden suhteen. Opetuksen tulee myös sisältää oikeanlaiset kasvatustyökalut ja -strategiat. Lisäksi ohjelmoinnin opetuksessa tulee muistaa ja ymmärtää eritasoisisia oppilaita, koska oppilaat ovat jokapäiväisessä vuorovaikutuksessa teknologian kanssa. Ohjelmoinnin opetuksessa opetusvälineet ja -strategiat tulee valita siten, että ne tukevat nopeaa oppimista ohjelmointikursseilla ja oppilaat kykenevät kehittämään ajattelutaitojaan pienellä vaivalla. (Flórez ym. 2017, 836–837.)

2.3 Algoritminen ajattelu

Algoritmisella ajattelulla tarkoitetaan ongelmanratkaisukykyjä, järjestelmien suunnittelua ja digitaalisten laitteiden toimintaperiaatteiden ymmärtämistä (Wing 2006; Bers ym. 2014, 146). Flórez ym. (2017, 834) määrittelevät algoritmisen ajattelun tavaksi kohdata jokapäiväisiä tilanteita ja ratkaista ongelmia hyödyntäen samoja konsepteja, jotka ovat tuttuja tietotekniikan maailmasta. Sitä on luonnehdittu analyyttiseksi ajatteluksi, ja siinä on samankaltaisuuksia matemaattisen ajattelun kanssa ongelman ratkaisun suunnittelemisessa ja toteuttamisessa. Algoritmisen ajattelun myötä ihmisellä on kykyjä ajatella useilla abstrakteilla tasoilla (Wing 2006, 33-35; Flórez ym. 2017, 836; Barr ja Stephenson 2011, 51).

Flórez ym. (2017, 836–837) uskovat, että ohjelmoinnin opettaminen on paras tapa opettaa algoritmista ajattelua. He toteavat, että oppilaat ohjelmoidessaan kehittävät algoritmisen ajat-

telun lisäksi ongelmanratkaisu- ja logiikkataitojaan sekä virheellisen toiminnan analysointieli debuggaustaitojaan. Debuggaaminen on nostettu yhdeksi tärkeämmäksi taidoksi loogisen ajattelun, ongelman ratkaisun ja sosiaalisen toiminnan kanssa ohjelmoinnissa (Sipitakiat ja Nusen 2012, 98). On huomattavissa, että ohjelmointi opettaa edellä mainittuja taitoja, mutta myös matemaattisia taitoja: Saeli ym. (2011, 78) kirjoittavat ongelmanratkaisukykyjen opettamisesta ohjelmoinnin kautta. Heidän mukaansa ohjelmoinnin avulla oppilaat kehittävät matemaattista käsityksiään, kuten ongelman pilkkomista, muuttujia, funktioita, debuggaamista ja yleistämistä. Tu ja Johnson (1990, 33) toteavat, että ohjelmointikurssin suorittajat kehittivät merkittävästi ongelmanratkaisukykyjään.

Winslow (1996, 19) on artikkelissaan todennut, että ohjelmoitaessa ongelmanratkaisukyky voidaan pilkkoa eri palasiin:

1. Ongelman ymmärtäminen,
2. ongelman ratkaisun määritteleminen,
    (a) jossain muodossa ja
    (b) tietokoneelle sopivassa muodossa,
3. ratkaisun muuttaminen tietokoneen ymmärtämäksi ohjelmointikieleksi ja
4. ohjelman ratkaisu ja debuggaus.

Winslow'n (1996, 20) mukaan oppilailla ei ole ohjelmoidessa ongelmia luoda syntaksisesti oikeita lauseita sen jälkeen, kun oppilaat ovat ymmärtäneet mitä ongelman ratkaisuun tarvitaan. Oppilaat eivät kuitenkaan tiedä missä ja miten käyttää näitä lauseita, jotta he saisivat luotua vaaditun ratkaisun. Lisäksi ohjelmoidessa heiltä unohtuu usein luoda ensin yleiskatsaus valmiista ohjelmasta, jonka takia oppilaiden mielestä on vaikeaa luoda yksityiskohtainen ohjelma. (Winslow 1996, 20.)

2.4 Ohjelmoinnin opetuksen pedagogiikkaa

On esitetty, että ohjelmoinnin opetus pohjautuu konstruktivismiin (Ben-Ari 1998, 261; Boyer, Langevin-Gaspar ja Gaspar 2008, 94; Bers ym. 2014, 146). Konstruktivismissa oppija rakentaa tietouttaan aktiivisella osallistumisella ongelmanratkaisuun, eikä filosofian mukaan

oikeaa totuutta tai valmista tietoa ole maailmassa olemassa (Murphy 1997; Ben-Ari 1998, 257). Jean Piaget'n ja Lev Vygotskyn tutkimukset lapsen kognitiivisesta kehityksesta ovat saaneet tiedeyhteisön puhumaan heistä konstruktivismin luojina. Siinä missä Vygotsky on nähty sosiaalisen konstruktivismin edustajana, Piaget on ollut kognitiivisen konstuktivismin tutkija (Alanen 2000, 98). Piaget'n teorian mukaan lapsen kehitys jakaantuu neljään eri vaiheeseen ikäkausien mukaan: 0-2-vuotiailla sensomotorinen vaihe, 2-6-vuotiailla esioperationaalinen vaihe, 7-12-vuotiailla konkreettisten operaatioiden vaihe ja 12-16-vuotiailla muodollisten operaatioiden vaihe. Alakouluikäisellä ajattelun loogisuus siis kehittyy ja johdonmukaisuus ajattelun taitoina paranee. Yläkouluikäinen taas Piaget'n teorian mukaan kykenee abstraktiin ajatteluun. (Huitt ja Hummel 2003, 2.)

Papert laajensi Piaget'n ajattelua konstruktivismista robotiikan ja tietokoneiden pariin. Papertin mukaan Piaget'n kasvatusfilosofinen ajatus oli, että lapsi rakentaa tietämystään aktiivisesti tekemisen kautta (learning-by-doing), mutta Papert halusi huomioida teoriassa myös ympäröivät rakenteet maailmassa, kuten tietokoneet (Papert 1980).

Konstruktio-opetus antaa lapselle vapauden tutkia mielenkiinnon kohteitaan teknologioiden avulla, samalla kehittäessään ongelmanratkaisu-, päättely- ja metakognitiivisia taitojaan (Bers ym. 2014, 146). Norrena (2013, 150–151) huomauttaa, että oppilaan tulevaisuuden taitoja kehitettäessä opetuksessa tulee käyttää kuitenkin monipuolisia opetuskäytänteitä. Vaikka opettajat kertovat opettavansa konstruktivismiin nojautuen, on opetuksessa kuitenkin paljon perinteisiä, behaviorismiin pohjautuvia piirteitä. Aiempi tutkimus on näyttänyt, että edistyksellinen pedagogiikka ei aina palvele parhaiten tehokasta oppimista tavoitellessa, vaan perinteiset opetusmenetelmät voivat olla tehokkaampia ja konstruktivistista oppimista tuottavia (Rasku-Puttonen ym. 2002).

Koska tietotekniikassa ja ohjelmoinnissa on useita erilaisia abstrakteja käsitteitä (kuten esimerkiksi muuttuja, silmukka tai ehtolause), on tärkeää esitellä lapselle erilaisia käsitteitä siinä tilanteessa, kun lapsi on kielellisesti kehittynyt tarpeeksi. Vygotsky esitti, että tiedon hankkimisen, jäsentämisen ja oivalluksen ketjussa kielen asema on keskeinen. Käsitteitä ei omaksuta selittäessä, vaan tällöin oppija omaksuu sanoja. Opitun tiedon soveltaminen on vaikeaa, koska sanoja omaksuessaan työskentely tapahtuu muistin varassa henkisten työkalujen kehittämisen sijaan. (Nikkola ym. 2013, 148.) Yläkouluikäisille ohjelmoinnin opettamisesta

tehty pelitutkimus selvitti, että pelin avulla ohjelmointikäsitteistöä on vaikea oppia, vaikka peli olisi motivoiva ja oppilaat olisivat innostuneita pelaamaan peliä oppiakseen ohjelmointia (Harju 2015, 54). Flórez ym. (2017, 838) esittääkin yhdeksi ohjelmoinnin opiskelutyökaluksi käsitekarttoja - niiden avulla voi suunnitelma ohjelmia, ne auttavat jäsentämään ajattelua siitä, mitä ohjelma tekee, ja ne kuvaavat sitä, miten ohjelma toimii jotta se suorittaa sen tehtävän, joka sille on annettu. Heidän mukaansa käsitekartat auttavat myös ohjelmoinnin alkeita opiskelevia jäsentelemään ohjelmoinnin perusperiaatteita ja käyttämään niitä oikein ohjelmissaan (Flórez ym. 2017, 838).

Lapsien perehdytys ohjelmointiin aloitetaan hyvin usein käyttämällä imperatiivista ohjelmointia. Siinä suoritettava tehtävä kuvataan tarkasti kohta kohdalta, suoritusjärjestyksessä. Yleinen harjoitus ohjelmoinnin aloituksessa on ihmisrobotin käskyttäminen - toinen oppilaista leikkii robottia, jolle toinen oppilas antaa erilaisia tehtäviä ja ohjeistaa kulkemaan ympäri luokkaa. Näistä harjoituksista voidaan siirtyä Bee-Bot -lattiarobotteihin, ja tällöin opetus tapahtuu ilman tietokoneita, leikin kautta. (T. Hiltunen 2016, 36.) Tietokoneettoman ohjelmoinnin on todettu sopivan nuorille ja nuoret ovat kokeneet tietokoneettoman ohjelmoinnin viihdyttäväksi (Taub, Ben-Ari ja Armoni 2009, 99).

Opetuksen ja oppilaiden taitojen edistyessä voidaan siirtyä graafisen ohjelmointiympäristön käyttöön ohjelmointiharjoituksissa. Monet graafiset ohjelmointiympäristöt ovat suunniteltu siten, että suoritettava ohjelmakoodi haetaan osina eli blokkeina koodikokoelmasta koodialueelle.

2.5 Aikaisempi tutkimus ohjelmoinnista opetuskontekstissa

Suomalaiset tutkimukset ovat hyvin harvakseltaan paneutuneet oppilaiden ohjelmoinnin osaamisen tarkasteluun (esim. Hakkarainen ym. 2000). Ohjelmoinnin osaamisen arvioinnille on nyt kuitenkin alkanut muodostumaan tarvetta ohjelmoinnin tultua opetussuunnitelmaan. Enemmän tällä hetkellä suomalaista tutkimusta on tehty opettajien asenteista ohjelmoinnin opettamisessa (mm. Karvonen ja Laukka 2016; Makkonen ja Pyykönen 2018; Kurkinen 2018). Näissä tutkimuksissa on todettu, että opettajan aiempi ohjelmointiosaaminen vähentää ohjelmoinnin opetuksesta syntyvää henkistä kuormaa ja ohjelmoinnin opettamisen

taidot ovat olleet heikkoja opettajilla (Karvonen ja Laukka 2016, 54; Kenttälä, Kankaanranta ja Neittaanmäki 2016, 45). Ohjelmoinnin opetus on koettu hyödylliseksi aiheeksi opettajien keskuudessa (Karvonen ja Laukka 2016, 55; Kurkinen 2018, 52–53). Myös oppilaiden motivaatiotekijöitä ja käsityksiä ohjelmoinnista on kartoitettu hieman (Neuman 2017; Könönen ja Ruotanen 2018).

Opetushallitus (2014, 30) linjaa OPS:ssa opetuksessa käytettävän monipuolisia työtapoja ja kehottaa käyttämään oppimisen välineenä yhteisöllistä oppimista, jolloin ymmärrystä rakennetaan vuorovaikutuksessa toisten kanssa. OPS:n kohdassa työtavoista kirjataan, että opetuksessa oppilaiden tulisi oppia jakamaan tehtäviä keskenään ja olemaan vastuussa yhteisistä tavoitteista. Tämänkaltainen pari- tai ryhmätyöskentely sopii erityisesti myös ohjelmoinnin oppimiseen: lapset, jotka ovat tehneet pariohjelmointia menestyvät paremmin ohjelmoinnissa kuin he, jotka ohjelmoivat yksin (Werner, Denner ja Campe 2012, 218; McDowell ym. 2002, 38). Pariohjelmointi on myös koettu mielekkäämmäksi tavaksi tehdä ohjelmointia yksinsuorittamisen sijaan (Williams ja Kessler 2000, 109).

Aiemmat tutkimukset (esimerkiksi Atmatzidou ja Demetriadis 2015; Nance 2016) algoritmisen ajattelun osaamisesta on toteutettu erilaisissa laajuuksissa. Tutkimuksen tekijä on voinut pitää alle kymmenen opetuskerran sarjan tai jopa yli vuoden mittaisen ohjelmointikurssin, jonka aikana on harjoiteltu erilaisia algoritmista ajattelua harjoittavia tai ohjelmointitaitoja harjoittavia tehtäviä. Ennen opetuskertoja oppilaat ovat vastanneet alkukyselyyn ja opetuskertojen päätteeksi kehitystä on mitattu uudella testillä tai kyselyllä (Atmatzidou ja Demetriadis 2015; Nance 2016).

2.6 Pelillinen oppiminen ja opetusrobotit

Osana ohjelmoinnin opetusta voidaan käyttää erilaisia pelejä tai robotteja ja opettajalla on tärkeä rooli integroidessa opetuspelejä ja pelillisiä ympäristöjä osaksi opetusta. Opettajan roolina on luoda pedagoginen viitekehys oppimistilanteessa määritellen esimerkiksi pelin käyttötarkoituksen opetuksessa opetussuunnitelman kontekstista. Hyvä opetuspeli ylittää oppiainerajoja luodessaan oppimisympäristön taitojen ja tietojen kehittämiselle. Oppiainerajojen ylittäminen tosin voi olla opettajalle vaikeaa. (Koskinen, Kangas ja Krokfors 2014, 33.)

Pelien opetuskäyttöä yleensä perustellaan opetuksen tai oppimisen tehostumisella, esimerkiksi oppilaan motivaation herättämisellä. Kuitenkin täytyy muistaa, että opetuspelien pedagoginen ajatus tulee sopia oppimisympäristöön, jotta opetuspelit pelit tehostavat pitkällä aikavälillä opetuksen tai oppimisen toimintaa. (Lipponen, Rajala ja Hilppö 2014, 147.)

Tiusanen (2014) määrittelee robotiksi koneen tai laitteen, joka osaa jollakin tavalla toimia itsekseen ympäröivässä maailmassa, mutta jolla täytyy olla jotain ihmismäistä, kuten käsivarsi. Robotiikasta puhuttaessa tarkoitetaan yleensä tekniiikkaa, johon on yhdisteltynä elektroniikan ja mekaaniikan teknologioita (Lau ym. 1999, 26). Opetusrobotit ovat helppo tapa lähestyä STEM-oppiaineiden opetusta, sillä opetusrobotteja voivat käyttää nuoretkin lapset (Mataric, Koenig ja Feil-Seifer 2007, 99). Helposti opetusrobotteja käytettäessä unohdetaan niiden tavoite kasvattaa lapsen kriittistä ajattelua, aktiivisen, luovan ja merkityksellisen oppimisen kautta (Lau ym. 1999, 26). Lisäksi nuorten, 4-6-vuotiaiden lasten robotiikkanäpertely kehittää lapsen hienomotoriikkataitoja sekä silmän ja käden yhteistyötä samalla, kun lapset harjoittelevat yhteistyö- ja tiimityöskenlytaitojaan (Bers ym. 2014).

Suomalaisissa peruskouluissa on robotiikkaa jonkin verran käytössä, muun muassa Blue-Bot -lattiarobotteja (Marttala 2017, 110), legorobotteja (Yle 2018) ja puhuvia, ihmismäisiä robotteja (Yle 2016).

2.7 Graafinen ohjelmointiympäristö

Graafisella tai visuaalisella ohjelmoinnilla tarkoitetaan visuaalista kaksiuloitteista järjestelmää, jossa ohjelmoija voi määrittää ohjelmansa visuaalisilla elementeillä (Golin ja Reiss 1990, 143; Myers 1990, 98). Kelleher ja Pausch (2005) jakavat taksonomiassaan ohjelmoinnin alkeisopetuksessa käytettävät järjestelmät kahteen eri luokkaan: opetusohjelmoinnin ja mekaanisen ohjelmoinnin järjestelmiin. Opetusohjelmoinnissa Kelleherin ja Pauschin mukaan ensisijainen tavoite on saada esimerkiksi Scratchissä käytetty kissa liikkumaan ja ohjelmoinnin oppiminen tapahtuu tämän sivussa. Visuaaliset ohjelmointiympäristöt ja -kielet kuuluvat opetusjärjestelmiin. Valtioneuvoston (Tanhua-Piiroinen ym. 2019, 28) raportin mukaan vuonna 2018 kyselyyn vastanneista oppilaista 47 % oli käyttänyt oppitunnilla graafista ohjelmointiympäristöä.

Graafisessa ohjelmointiympäristössä ohjelmointi voi olla hyvin innostavaa: Lakasen ja Isomöttösen (2015) mukaan luovuuden korostaminen ohjelmoidessa voi innostaa oppilasta, joka näkee ohjelmoinnin mekaanisena tehtävänä, näkemään ohjelmoinnin eri tavalla. Oppilaan mahdollisuus lisätä esimerkiksi visuaalinen kosketus tehtävän toteutuksessa voi lisätä opiskelijan sitoutuneisuutta tehtävän tekemiseen. Erilaiset tutoriaalit ja yksityiskohtainen ohjaus tehtävien tekemisen yhteydessä hyödyttävät niitä oppilaita, jotka vasta harkitsevat varsinaisen ohjelmakoodin kirjoittamista. (Lakanen ja Isomöttönen 2015, 462.) Lakanen (2010, 31) toteaa, että peliohjelmointi ympäristössä, jossa muutosten tekeminen ja toimintoihin tutustuminen on turvallista, antaa mahdollisuuksia oppia ohjelmointia kokeilemisen ja oivaltamisen kautta. Graafinen oppimisympäristö on helppo tapa lähestyä ohjelmointia jopa pienillekin lapsille, sillä käsitteiden ymmärtämisen tueksi oheen on liitetty kuvalliset symbolit toiminnoille, jolloin lapsi ei tarvitse kieli- tai lukutaitoa ohjelmoidakseen (Niukkanen 2014, 254–255). On todettu, että ohjelmointiympäristöjen tärkeimpiä periaatteita ovat mm. ohjelman ominaisuuksien näkyvyys, konkreettisuus, kopioinnin ja muokkaamisen mahdollisuus sekä objektien suorakäyttö eli koodiblokkien siirtely sanallisten komentojen käytön sijaan (Sola 2002, 185–186).

Kuten aiemmin jo mainittiin, yleisiä tehtyjä virheitä ohjelmoinnin opiskelun alussa ovat syntaksi- ja logiikkavirheet (Serafini 2011, 153; Mannila, Peltomäki ja Salakoski 2006, 211) ja vain muutama ohjelmointikieli on kehitetty pelkästään opetuskäyttöä varten (Mannila, Peltomäki ja Salakoski 2006, 212). Jotta syntaksi- ja logiikkavirheiltä vältyttäisiin, on kehitetty graafisia ohjelmointiympäristöjä, jossa jokaista ohjelmointikielen osasta vastaa blokki tai palanen, joka voidaan raahata ja pudottaa suoritusalueelle. Lisäksi näissä ympäristöissä koodin suorittaminen heti on tehty mahdolliseksi ilman, että koodia tarvitsisi erikseen kääntää, jotta tietokone ymmärtäisi ohjelman. (Ouahbia ym. 2015, 1480; Serafini 2011, 153; Fessakis, Gouli ja Mavroudi 2013, 88.) Graafiset ohjelmointiympäristöt ovat hyvin pelillisiä ulkoasultaan ja toiminnoiltaan. Muun muassa Kohn (2017, 349) toteaa, että visuaalinen kerronta auttaa vähentämään virheitä ohjelmoidessa.

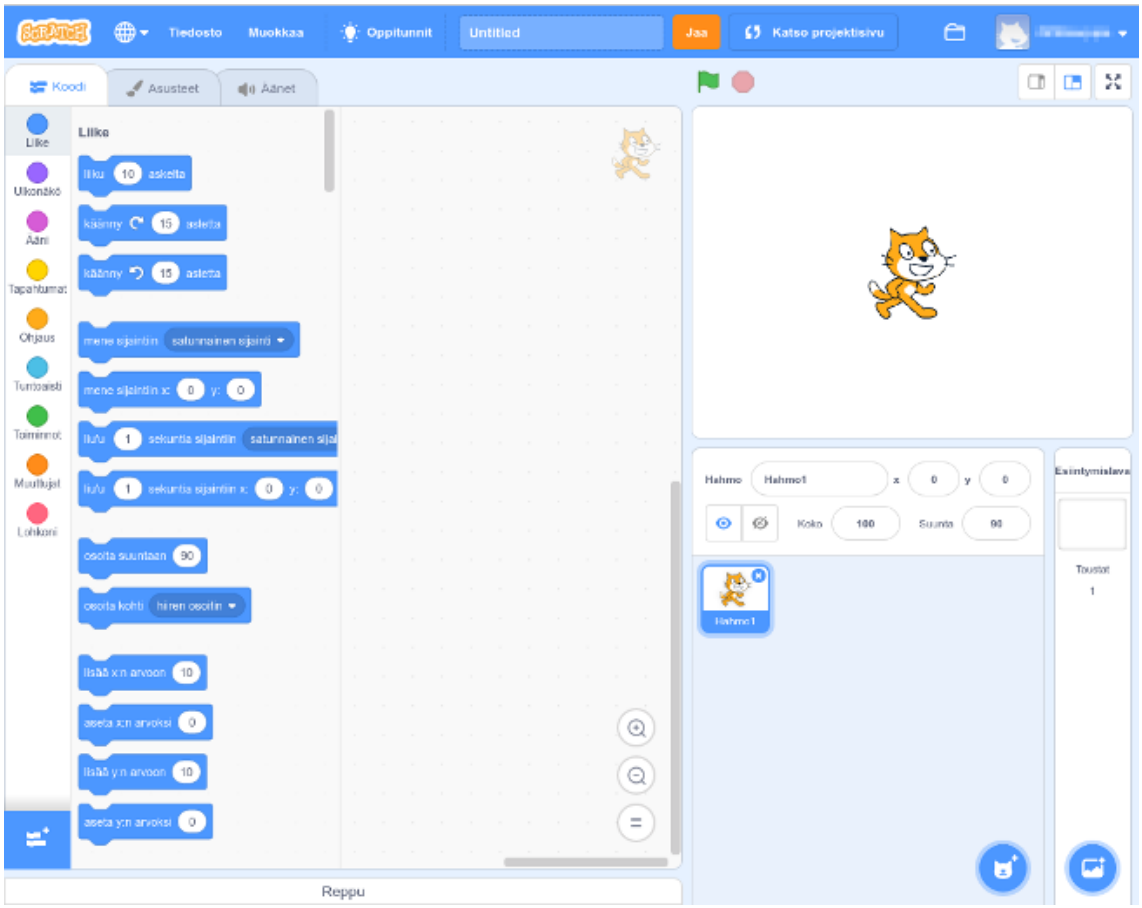
Seuraavaksi esitellään muutama opetuksessa käytetty graafinen ohjelmointiympäristö.

Scratch ja Scratch Jr

Scratch (kts. kuvio 1) on ohjelmointiympäristö, jota voidaan käyttää interaktiivisten satujen, pelien ja animaatioiden ohjelmoitiin ja jonka avulla voidaan myös jakaa työstettyjä teoksia online-yhteisössä (Resnick 2012; Kalelioğlu ja Gülbahar 2014, 34; Ouahbia ym. 2015, 1480). Alunperin Scratch on kehittty Massachusetts Institute of Technology -korkeakoulun MIT Media Lab -tutkimusyksikön projektina (Scratch 2019). Scratch on suunnattu oppilaille, jotka osaavat jo lukea, mutta lukutaidottomille oppilaille, esimerkiksi esikoulu- tai alkuopetusikäisille on tarjolla Scratch Jr. -niminen sovellus, joka toimii ainoastaan tablet-laitteilla. Sovelluksissa näkymä jakaantuu kolmeen osaan: koodi haetaan *blokkialueelta* ja raahataan *koodialueelle*, jossa rakennettu koodikokonaisuus näkyy *näyttämöllä*. Scratch Jr -ohjelmassa koodi on selitetty symbolein, kun taas Scratchissa koodin toiminnallisuus on selitetty tekstillä. Scratch-ohjelmointiympäristön kehittämisessä on painotettu ympäristön personointiin (Adleberg 2013, 19). On myös todettu, että Scratch on sukupuoli-, luokka- ja taitotasoneutraali ohjelmointiympäristö (Adleberg 2013, 38). Nuoret kokevat Scratchin käytön mielekkääksi tavaksi opiskella ohjelmointia (Pampel 2017, 38).

Code.org

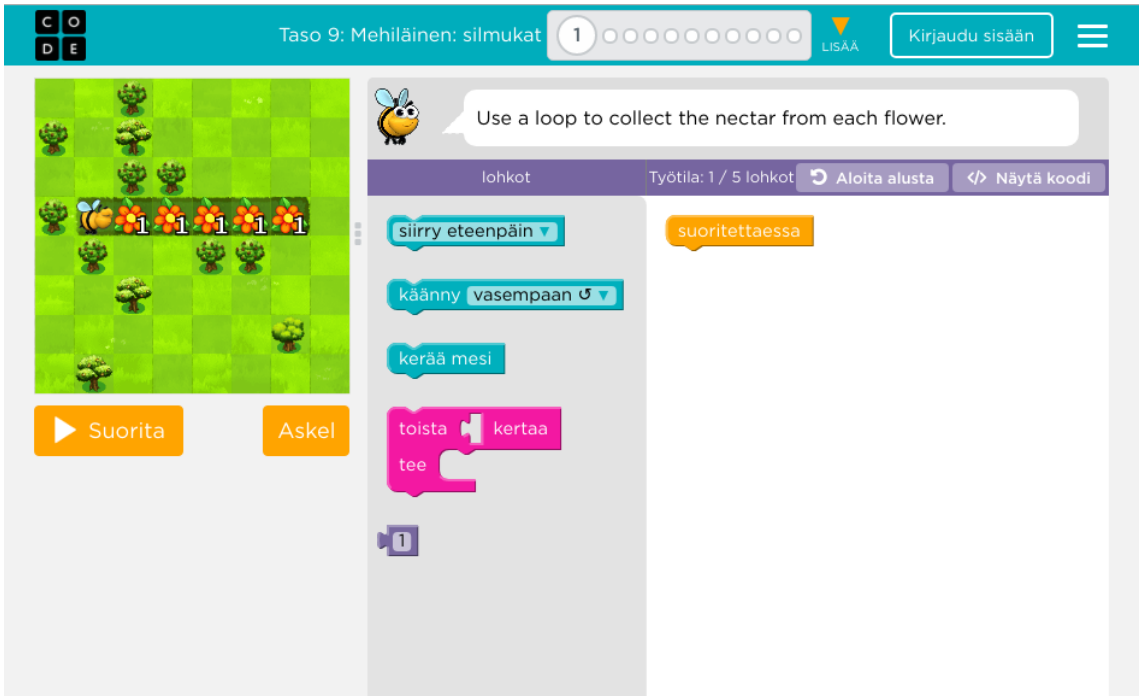
Kuten Scratch, Code.org on myös graafinen ohjelmointiympäristö (kts. kuvio 2), jonka ulkoasu on hyvin pelillinen: tehtävissä seikkailevat mm. Angry Birds -pelihahmot. Ohjelmointi tapahtuu koodiblokkeja raahaamalla ja pudottamalla. Code.org opettaa kattavasti pääasioita ohjelmoinnista: algoritminen ajattelu, muuttujat, silmukat, ehtolauseet, funktiot ja parametrit. Ennen varsinaista ohjelmointia Code.orgin kautta on mahdollista pohtia tietotekniikan maailmaa ilman tietokonetta, kuten että mitä on tietotekniikka ja kuka on tietoteknikko, minkälaisia erilaisia sovelluksia on tietotekniikassa, tutustumista binäärilukuihin, mitä on debuggaaminen ja miten internet toimii. (Kalelioğlu 2015, 202.) Code.orgin suomennostyö on kesken, joten ohjeistukset ovat vaihtelevasti suomeksi ja englanniksi. Siinä missä Scratchissa oppilas saa luoda vapaasti ohjelmansa ja suorittaa sen, Code.orgissa on aina valmiina tehtävänanto ja rajatut koodiblokit tehtävän suorittamiseen. Tämä oli suurin syy, miksi Code.orgia ei valittu tämän tutkimuksen ohjelmointialustaksi.



Kuvio 1. Suomenkielinen Scratch 3.0 aloitusnäkymä.

Scratch opetuskäytössä

Scratchin opetuskäytöstä on erilaisia tutkimustuloksia. Nancen (2016) mukaan kuuden viikon tietokoneohjelmoinnin opiskelu ei merkittävästi vaikuttanut oppilaiden ongelmanratkaisukykyihin. Oppilaat aloittivat ohjelmoinnin opiskelun tutustumalla tietokoneen peruskomponentteihin, eli he olivat varsin kokemattomia tietokoneen käyttäjinä. (Nance 2016, 61;171.) Kalelioğlu ja Gülbahar (2014) totesivat myös tutkimuksessaan, että Scratchin käyttö ohjelmointiympäristönä ei juuri vaikuttanut 5. luokan oppilaiden ongelmanratkaisukykyjen muutoksiin, mutta arvelivat, että Scratchin käytöllä on vaikutusta oppilaan itsevarmuuteen ongelmanratkaisukyvyistään. Ouahbia ym. (2015, 1482) sekä Malan ja Leitner (2007, 223) ovat todeneet tutkimuksissaan, että Scratch antaa oppilaan tutustua ohjelmoinnin rakenteeseen ja logiikkaongelmiin ennen syntaksin opettelua. Lavonen, Meisalo ja Lattu (2001, 24) huomauttavat, että graafista ohjelmointiympäristöä käytettäessä ohjelmoinnin opetukseen tu-



Kuvio 2. Code.orgin silmukkatehtävän aloitusnäkymä.

lee olla varovainen, ettei opetuksen fokus siirry ongelmanratkaisutaitojen opettelusta työkalun käyttämisen opetteluun.

Oppilaat ovat olleet motivoituneita tehdessään pelejä tai ratkaistessaan algoritmisia ongelmia käyttäen Scratchia. Peliohjelmoinnillisten tehtävien teon aikana oppilaat kokivat iloa saavuttaessaan tavoitteensa ja he kokivat ohjelmoinnin jännittäväksi odottaessaan pääsevänsä kokeilemaan tekemäänsä peliä (Ipek ja Turgut 2018, 186). Pampel (2017, 42) toteaa, että oppilaat ovat pitäneet ohjelmoinnin opiskelusta sen vuoksi, että oppitunteihin tulee vaihtelua ja he saavat työskennellä elektroniikan kanssa. Pampelin mukaan oppilaat ovat kokeneet, että ohjelmoinnin opiskelussa työskentelyä eivät rajoita säännöt ja se kannustaa luovuuteen. Lisäksi ohjelmoidessa myöskin konkreettisesti näkee sen, mitä tekee tietokoneen avulla. (Pampel 2017, 42.) Onkin todettu, että minäpystyvyyden tunne lisää motivaatiota (Deci ja Ryan 2000).

2.8 Opettajien valmiudet opettaa ohjelmointia

Ohjelmointitaito ja kyky opettaa ohjelmointia ovat eri asioita. Ennen opetusta opettajan tulisi miettiä, miksi opettaa, mitä opettaa, mitä oppimisvaikeuksia voi olla ja miten opettaa opetettavaa aihetta. (Saeli ym. 2011, 18–21, 36.) Koska ohjelmoinnin opettaminen voi olla vaikeaa, opettajien perehdyttäminen ohjelmoinnin opettamiseen on tärkeää ja se lisää heidän varmuuttaan opettaa asiaa, joka heille ei välttämättä ole tuttu (Flórez ym. 2017, 837). Esimerkiksi graafisen ohjelmointiympäristön käyttö ja omien tehtävien tekeminen opittiin nopeasti muutaman workshopin perusteella ja opettajat kokivat tämän jälkeen itsevarmuutta ohjelmointitehtävien tekemisessä (Chipman, Rodríguez ja Boyer 2019). Myös Harris (2018, 83) toteaa, että opettajat kasvattivat itsevarmuuttaan opettaa ohjelmointia lapsille lyhyen perehdytyksen jälkeen.

Vuonna 2016 julkaistussa pro gradu -tutkielmassa puolet tutkimukseen osallistuneista uskoi osaavansa ratkaista ohjelmoinnin opetuksen ongelmia ilman ohjeistusta (Karvonen ja Laukka 2016, 55). Myöhemmin on kuitenkin todettu, että opettajat ovat kokeneet kykynsä opettaa ohjelmointia heikoiksi tai alkeisopetuksen tavoitteet saavuttaen (Kurkinen 2018, 43). Opettajat ovat kokeneet 5-6. vuosiluokkien ohjelmointitehtävien keksimisen olevan vaikeampaa kuin 1.-4. vuosiluokan tehtävien. Lisäksi on todettu, että ohjelmoinnin opetuksen monipuolisuus riippuu opettajan asenteesta ja kiinnostuksesta aihetta kohtaan (Makkonen ja Pyykönen 2018, 68).

OAJ:n mukaan opettajien saama pedagoginen TVT-koulutus on ollut riittämätöntä (OAJ 2016, 38). Myös valtioneuvoston rahoittamassa Turun yliopiston Digiajan peruskoulu -hankkeessa havaittiin, että opettajien taitotaso tehdä itse yksinkertaisia ohjelmointitehtäviä tai opettaa ohjelmointia on ollut heikkoa, tosin kylläkin parantunut parin vuoden takaisesta tilannekatsauksesta (Kaarakainen ym. 2017, 42). Lopullisessa, vuoden 2019 raportissa opettajien ICT-taitotestissä heikoin osaaminen on ollut ohjelmoinnissa (Tanhua-Piiroinen ym. 2019, 21). Graafisista ohjelmointiympäristöistä suomalaisille opettajille tutuimmat ovat Scratch, Scratch Jr. ja Code.org (Karvonen ja Laukka 2016, 38).

3 Opetussuunnitelma 2014 ja ohjelmointi

Ohjelmoinnin opetus on otettu käyttöön ympäri Eurooppaa alakoulujen opetuksessa (Mannila ym. 2014, 1). European Schoolnetin (European Schoolnet 2018, 9) tekemässä raportista käy ilmi, että vuonna 2015 Euroopan alueen 16 maata integroivat ohjelmointia opetussuunnitelmissaan joko kansallisella, alueellisella tai paikallisella tasolla. Joissain maissa, kuten Englannissa, Norjassa ja Puolassa, ohjelmointi on oma opetettava aineensa (Heintz, Mannila ja Färnqvist 2016).

Tässä luvussa esitellään Suomen opetussuunnitelmaa ja ohjelmoinnin opetusta opetussuunnitelman näkökulmasta. Vuoden 2014 opetussuunnitelma (Opetushallitus 2014) listaa ohjelmoinnin osaksi opetuksen tavoitteita matematiikassa ja käsityössä.

Nykyisen (vuoden 2014) opetussuunnitelman tekeminen alkoi keväällä 2009, kun perusopetuksen tavoitteita ja tuntijakoa uudistamaan asetettiin työryhmä Opetusministeriön toimesta. Uutta opetussuunnitelmaa tehtiin useamman vuoden ajan ja se viivästyikin alkuperäisestä aikataulustaan. Aikaisemmin eri työryhmien asiakirjoissa ohjelmointia ei ollut mainittu ollenkaan tulevaksi opetussuunnitelmaan, mutta vuonna 2014 TVT-työryhmän asiakirjoissa ohjelmointi oli opetusaiheena mukana. Ilmeisesti Jyrki Kataisen hallituksen hallitusohjelmaan nostettu työelämätarpeiden ja yritysyhteistyön suuri rooli innoitti kansanedustajia ja järjestöjä, ja opetusministerille lobattiin ohjelmoinnin ottaminen osaksi opetussuunnitelmaa. Lopulta uusi opetussuunnitelma hyväksyttiin vuoden 2014 lopulla, kun uusi opetussuunnitelma oli käynyt läpi neljä eri hallitusta ja kolme opetusministeriä. (kts. Marttala 2017, luku 2.2.1.) Opetusministeri Krista Kiuru totesi vuonna 2014, että Suomen vahvaa ICT-osaamista tulee turvata ja tämän takia ohjelmoinnin taito on tarpeellista oppia jo peruskoulussa (Opetus- ja kulttuuriministeriö 2014). Ohjelmoinnin ottamista osaksi opetussuunnitelmaa on perusteltu lasten ohjelmoinnista saatavilla työkaluilla ongelmanratkaisuun ja loogisen ajattelun taitoihin (Koodiaapinen 2018). Lukiossa tapahtuvaa ohjelmoinnin opetusta on aiemmin perusteltu sillä, että oppilas oppii yleisesti algoritmisen ajattelutavan ja pystyy käyttämään sitä apuvälineenä yleisesti elämässä, mutta myös sillä, että oppilaita valmistellaan tulevaisuuden opintoja varten (Mannila, Peltomäki ja Salakoski 2006, 212).

Opetussuunnitelmassa ohjelmoinnin opetus tähtää oppilaan algoritmisen ajattelun kehittämiseen. Englanninkielisessä kirjallisuudessa käytetään muutamaa erilaista käsitettä: *computational thinking* tai *algorithmic thinking*. Kekäläinen (2015) on pohtinut artikkelissaan suomenkielistä käännöstä termille *computational thinking*. Hänen mukaansa *laskenta-ajattelu* ei ole kuvaava käsite, eikä *laskennallinen ajattelu* käsitteenä käänny suoraan siten, että se käsittäisi termiä ensimmäisenä käyttäneen Wingin (2006) ajatusta termin laajuudesta. Kekäläinen (2015, 28) artikkelissaan ehdottaa käytettävän käsitettä *automatisointiajattelu*. Tässä tutkimuksessa kuitenkin puhutaan algoritmisesta ajattelusta viitatessa tähän *computational thinking* -termiin.

Opetussuunnitelmassa ohjelmoinnista puhutaan opetuksen laaja-alaisen osaamisen tavoitteissa, joihin viitataan kirjaimella L. Ohjelmoinnin käsitteet tulevat vastaan myös matematiikan ja käsityön opetuksen tavoitteissa (lyhennyksenä kirjain T) ja sisällöissä (lyhennyksenä kirjain S).

3.1 Laaja-alainen osaaminen: tieto- ja viestintäteknologinen osaaminen (L5)

Opetussuunnitelma listaa laaja-alaisen osaamisen tavoitteita. Laaja-alainen osaaminen on tietojen, taitojen, arvojen, asenteiden ja tahdon muodostama kokonaisuus (Opetushallitus 2014, 20). Laaja-alaisia osaamiskokonaisuuksia on OPS:ssa listattuna seitsemän, ja yksi näistä on Tieto- ja viestintäteknologinen osaaminen, L5. OPS:ssa kirjoitetaan, että TVT-osaaminen on osa monilukutaitoa ja täten tärkeä kansalaistaito. Lisäksi todetaan, että oppilaan opiskelumotivaatioon vaikuttaa yhdessä tekemisen ja oivaltamisen ilo, unohtamatta oppilaiden mahdollisuutta luovuuteen ja itselle parhaimpaan oppimispolkujen ja työskentelytapojen löytämiseen. (Opetushallitus 2014, 23). Opetussuunnitelmassa kirjataan, ettei ohjelmointia opeteta omana aineenaan, vaan integroiden osaksi muiden oppiaineiden opetusta (Opetushallitus 2014, 284).

3.2 Vuosiluokat 1-2

Laaja-alaisen osaamisen Tieto- ja viestintäteknologisen osaamisen (L5) tavoitteissa vuosiluokilla 1-2 on todettu, että oppilaat saavat ja jakavat keskenään kokemuksia digitaalisen median parissa työskentelystä sekä ikäkaudelle sopivasta ohjelmoinnista (Opetushallitus 2014, 101). Osana matematiikan opiskelua vuosiluokilla 1-2 edetään ensin tutustumalla ohjelmoinnin alkeisiin. Oppilaat laativat vaiheittaisia toimintaohjeita ja testaavat näitä (Opetushallitus 2014, 129). Tarvittaessa opetuksessa voi edetä kokonaan ilman laitteita. Alaluokilla oppilaille esitellään myös ohjelmointisovelluksia, jolloin tutustutaan alkeisohjelmointiin yksinkertaisiin ohjeisiin, työskentelytapana esimerkiksi Bee-Bot-lattiarobotit (Makkonen ja Pyykönen 2018, 45).

Kansallinen OPS ei kerro konkreetian tasolla, mitä oppilaan tulisi osata ohjelmoinnissa tiettyjen vuosiluokkien päätteeksi. Tämän takia paneuduttiin myös paikallisen tason opetussuunnitelmiin, jotta tutkimuksen testin suunnittelu oli helpompaa ja jotta testiin tuli valittua aihealueiksi oppilaille todennäköisesti tuttuja teemoja. Kuopiossa paikallisen TVT-tuen mukaan (Kuopion kaupunki 2019) oppilaan tulisi 2. luokan päätteeksi osata leikin avulla jakaa tehtäviä osiin ja antamaan yksikäsitteisiä toimintaohjeita. Oppilas voisi tutustua alaluokilla ehtolauseajatteluun tutkimalla ehdon toteutumista vaatteiden värin myötä johtuvasta toiminnosta. Jyväskylän TVT-opetussuunnitelman mukaan (Jyväskylän kaupunki 2016) oppilas harjoittelee alaluokilla vaiheittaisia toimintaohjeita joko ilman tietokoneita, tai ikätasolle sopivilla roboteilla ja ohjelmointiympäristöillä, kuten Blueboteilla ja Koodaustunti -sivustolla.

3.3 Vuosiluokat 3-6

3-6-vuosiluokkien laaja-alaisen osaamisen tavoitteena on tieto- ja viestintäteknologian myötä, että ohjelmointia kokeillessaan oppilaat huomaisivat, miten teknologian toiminta riippuu ihmisen tekemistä syötteistä, eikä siis tietokone toimi ilman ihmistä (Opetushallitus 2014, 157). Matematiikan opetuksessa vuosiluokilla 3-6 todetaan tavoitteena (T14) olevan innostaa oppilasta laatimaan toimintaohjeita tietokoneohjelmina, työkaluna näillä luokilla käytetään graafisesta ohjelmointiympäristöä. Lisäksi matematiikan opetuksen sisällöissä (S1)

ajattelun taitoina mainitaan ohjelmien suunnittelu ja toteutus graafisessa ohjelmointiympäristössä. Kuudennen vuosiluokan päätteeksi hyvää osaamista kuvaava arviointikriteeri on ohjelmoinnin opetuksen kannalta että *"oppilas osaa ohjelmoida toimivan ohjelman graafisessa ohjelmointiympäristössä"*. (Opetushallitus 2014, 235–239.)

Käsityön opetuksessa vuosiluokilla 3-6 opetuksen keskeisissä sisältöalueissa (S3) mainitaan, että opetuksessa voidaan kokeilemalla ohjelmoida aikaan saatuja toimintoja. Esimerkkinä tällöin voidaan käyttää automatiikkaa ja robotiikkaa käsityön alalla. Näiden kokeilujen pohjalta tuotetta tai teosta kehitetään eteenpäin, eli työskentely olisi opetussuunnitelman mukaan iteratiivista. (Opetushallitus 2014, 271.)

Paikallisella tasolla Jyväskylän TVT-opetussuunnitelmassa (Jyväskylän kaupunki 2016) on kirjattu, että oppilas vuosiluokkien 3-4 aikana harjoittelee ohjelmointia käyttäen kuvakepohjaista ohjelmointiympäristöä, esimerkiksi Scratch Jr:ia. Vuosiluokkien 5-6 puolella voidaan siirtyä Scratch:in käyttöön. Jyväskylän ohjeistuksen mukaan oppilas käyttää myös muuttujia, ehtolauseita ja toistorakenteita ohjelmoinnin harjoittelussaan, ja ohjelmoinnin tuotoksena voi olla esimerkiksi yksinkertainen peli tai seikkaileva satuhahmo. Kuopion TVT-tuen sivut ohjeistavat myös toisto- ja ehtorakenteiden käyttöön (Kuopion kaupunki 2019).

3.4 Vuosiluokat 7-9

Yläkoulun puolella ohjelmoinnin opiskelu painottuu enemmän matematiikan opintoihin. Matematiikan opetuksen sisällöissä (S1) mainitaan algoritmisen ajattelun syventämistä, samalla kun ohjelmoidaan ja harjoitellaan hyviä ohjelmointikäytäntöjä. Opiskelussa voidaan käyttää itse tehtyjen ohjelmien lisäksi valmiita tietokoneohjelmia. Lisäksi OPS toteaa matematiikan päättöarvioinnissa hyvän osaamisen kohdalla, että algoritmisen ajattelun myötä oppilas osaa soveltaa tätä ajattelutapaansa ja ohjelmoida yksinkertaisia ohjelmia. (Opetushallitus 2014, 375–379.)

Paikallisella tasolla on ohjeistettu, että oppilas osaisi yhdeksännen luokan päätteeksi antaa toimintaohjeita laitteelle sen ymmärtämässä muodossa ja olisi täten vahvistanyt ohjelmoidessaan muuttujien ja toisto- sekä ehtorakenteiden käyttämistä (Kuopion kaupunki 2019). Lisäksi ohjelmoinnin toistorakenteita voidaan matematiikan puolella käyttää hyödyksi piir-

tämällä matemaattisia kuvioita. Jyväskylän TVT-opetussuunnitelma (Jyväskylän kaupunki 2016) linjaa, että seitsemännen ja kahdeksannen luokan aikana oppilas tutustuu johonkin tekstuaaliseen ohjelmointikieleen, kuten esimerkiksi Pythoniin. Matematiikan puolella ohjelmointia voidaan käyttää hyväksi ratkaisemalla erilaisia ongelmia. Yhdeksännellä luokalla oppilas osaa ohjelmoida käyttäen jotakin ohjelmointikieltä, jonka tuloksena voi olla yksinkertainen peli, matemaattisen ongelman ratkaiseva ohjelma tai toiminnallinen robotti.

4 Tutkimusasetelma

Tässä luvussa kerrotaan tutkimuskysymyksistä, tutkimusstrategiasta ja tutkimuksen kulusta. Lisäksi tässä luvussa kuvataan empiirisen osuuden läpivienti ja aineiston keruu- ja analysointitavat.

4.1 Tutkimuskysymykset

Uusi opetussuunnitema oli ollut keväällä 2019 käytössä alakoulun puolella kolme lukuvuotta. Oppilaiden siirtyessä yläkouluun he siirtyvät ohjelmoinnissa graafisesta ohjelmointiympäristöstä tekstuaaliseen ohjelmointiin (kts. luku 3.4). Tämän siirtymän takia oli mielenkiintoista selvittää, minkälaisella osaamisella oppilaat siirtyvät harjoittelemaan tekstuaalista ohjelmointia. Tutkimusta oppilaiden taitotasosta ohjelmoinnin suhteen on tutkijan tiedon mukaan Suomessa tehty hyvin vähän, minkä vuoksi tutkimus oli tarpeellinen.

Tutkimuksen tavoitteena oli selvittää 6.-luokkalaisten oppilaiden taitotasoa ohjelmoinnissa. Lisäksi oli tarkoitus selvittää, minkälaisilla edellytyksillä oppilas siirtyy ohjelmoimaan yläkoulun puolelle tekstimuotoisesti. Lisäksi tutkimuksessa tutkittiin opettajia haastattelemalla sitä, että millä työvälineillä ja miten paljon tutkimuksen kouluissa opiskellaan ohjelmointia.

Tämän myötä tutkimuskysymykset ovat seuraavat:

1. Mikä on 6. luokan oppilaiden taitotaso ohjelmoinnissa alakoulun päättyessä?

Ensimmäinen tutkimuskysymys kartoittaa oppilaiden ohjelmoinnin taitoja. Tätä kysymystä tutkittiin oppilaiden suorittamalla portaittain vaikeutuvalla testillä.

2. Millä tavoin ohjelmointia opetetaan alakoulun puolella?

Toisen tutkimuskysymyksen avulla oli tarkoitus saada selville erilaisia käytänteitä ja välineitä, joilla ohjelmointia opetetaan kouluissa. Lisäksi ohjelmoinnin opetuksen integrointi muihin kuin matematiikan opetuksen sisältöihin oli kiinnostavaa selvittää, koska ohjelmointia ei opeteta omana oppiaineenaan. Kysymykseen vastataan haastattelemalla opettajia heidän opetuksestaan.

3. Minkälaisia edellytyksiä 6. luokan oppilailla on siirtyä yläkoulun puolelle tuottamaan tekstimuotoista ohjelmakoodia?

Kolmannella tutkimuskysymyksellä tutkittiin erilaisten ohjelmoinnin rakenteiden osaamista. Tulokset tähän tutkimuskysymykseen muodostuivat oppilaiden tekemän testin vastauksia analysoimalla laadullisin ja määrällisin keinoin.

4.2 Tutkimusstrategia

Koska tutkimuksen tarkoituksena on saada hyvä ymmärrys oppilaiden taitotasosta ohjelmoinnissa, on tutkimus tyypiltään laadullinen tutkimus. Tutkimusta voisi myös ajatella tutkimuksen pienen otannan vuoksi tapaustutkimuksena, sillä otanta käsittää kolme eri kuudennetta luokkaa eri puolilta Suomea. Tuomen ja Sarajärven (2018) mukaan laadullista tutkimusta yleensä tehdään haastattelun, kyselyn tai havainnoinnin keinoin. Näitä tutkimusmenetelmiä voidaan käyttää yhdistellen, toistensa kanssa rinnakkain tai pelkästään vaihtoehtoisesti tutkimusongelman tai tutkimuksen resurssien mukaan. Laadullisessa tutkimuksessa voidaan myös käyttää määrällisen tutkimuksen aineistoja: esimerkiksi formaaliset strukturoimattomat tutkimusasetelmat (kuten avoin haastattelu tai kysely) ovat laadullisen tutkimuksen aineistonkeruumenetelmiä, mutta niitä käytetään myös toisinaan määrällisissä tutkimuksissa. (Tuomi ja Sarajärvi 2018.) Kuitenkin tähän tutkimukseen tulee myös määrällisiä piirteitä, kun oppilaiden taitotasoa tarkastellaan testin suoriutumistason osalta kokonaisuutena ja näitä tuloksia analysoidaan laadullisin keinoin. Koska tutkimusta ohjaa teorian pohjalta aineisto, ei tutkimusta voida lukea täysin teoria- tai aineistopohjaiseksi tutkimukseksi. Aineistosta tehdään tulkintoja, jotka pyritään sitomaan aiempaan teoriaan, jolloin tutkimus on luonteeltaan teoriasidonnainen tutkimus (Saaranen-Kauppinen ja Puusniekka 2019, 15).

Tutkimus jakaantui kahteen osaan: ensimmäisessä osassa luokanopettajaa haastateltiin ohjelmoinnin opetuksesta. Opettajan haastattelukysymykset on kerrottu liitteessä C. Haastattelulla selvitettiin, mitä oppimateriaaleja ja välineitä käyttäen luokan oppilaat ovat opiskelleet ohjelmointia edellisen kolmen lukuvuoden aikana. Toisessa osassa oppilaat suorittivat testin edetessä vaikeutuvan, kolmea konseptia testaavan ohjelmointitehtävien kokonaisuuden, joka mittaa heidän ohjelmoinnin osaamistaan.

4.3 Tutkimuksen kulku ja tutkimukseen osallistujat

Tutkimus alkoi syksyllä 2018 rajaamalla tutkimusaihe ja tutkimuskysymykset. Perehtymällä aiempaan tutkimukseen syksyllä 2018 ja alkuvuodesta 2019 pystyttiin muodostamaan haastattelurunko ja oppilaille suunnattu ohjelmoinnin osaamista mittaava testi. Tutkimukseen osallistuvien koulujen hankkiminen oli haasteellista: tutkimukseen kysyttiin satunnaisesti eri kouluja eri kouluista sähköpostitse, keskittyen erityisesti Keski-Suomen kuntiin lyhyempien välimatkojen toivossa. Kuitenkin opettajien sähköpostien vastausaktiivisuuden ollessa erittäin heikko, tutkimukseen osallistuvien koulujen valikoitumiseen kului runsaasti aikaa. Lopulta tutkimukseen saatiin mukaan kolme eri kuudennetta luokkaa eri puolilta Suomea – tutkimuskouluiksi valikoitui Pirkanmaalta, Keski-Suomesta ja Pohjos-Savosta sijaitsevia keskikokoisia kouluja, joissa oppilaita oli 400-600. Luokkien koot vaihtelivat 19-27 oppilaan välillä. Tutkimukseen osallistui yhteensä 54 oppilasta.

Tutkimusluvat hankittiin aluksi koulujen rehtoreilta ja tämän jälkeen kyseisen kunnan tutkimuksesta vastaavilta viranomaisilta. Koska osa tutkimukseen osallistujista olivat alaikäisiä, ennen tutkimusta oppilaiden vanhemmilta kysyttiin lupa lapsen osallistumisesta tutkimukseen (kts. liite A). Lisäksi tutkimuksesta tuli tehdä tietosuojailmoitus henkilötietojen käsittelystä (kts. liite B). Sama tietosuojailmoitus toimi oppilaiden vanhemmille lisätietona tutkimuksesta ja tutkimusluvan kysymisen yhteydessä korostettiin, että lapsilta ei kerätä henkilötietoja. Näiden vanhemmilta saatujen tutkimuslupien puitteissa keskisuomalaisesta koulusta tutkimukseen osallistui kahdeksan oppilasta kahdestakymmenestä. Pirkanmaalaisen ja pohjoissavolaisten koulujen tutkimusluokkien kaikki oppilaat osallistuivat tutkimukseen: pirkanmaalaisen koulun kuudennella luokalla oli 27 oppilasta ja pohjoissavolaisen koulun kuudennella luokalla 19 oppilasta. Lisäksi tutkimukseen osallistuivat näiden kolmen luokan opettajat.

Oppilaiden vastauksissa ei kerätty minkäänlaisia henkilötietoja eikä täten alaikäisiä ollut tunnistettavissa vastauksista. Mikäli vastauksissa oli jonkinlaisia henkilötietoja (kuten esimerkiksi vastaaja ohjelmoi hahmon tulostamaan oman nimensä näytölle), nämä tiedot anonymisoitiin vastauksia käsitellessä.

Empiirinen toteutus Scratchilla

Empiirisen osuuden tehtävien suunnittelu lähti liikkeelle perehtymällä aiempien tutkimusten tutkimusasetelmiin. Esimerkiksi Werner, Denner ja Campe (2012, 217) tutkivat oppilaiden algoritmista ajattelua 5-8-luokkalaisten keskuudessa kahden vuoden ajan. Tutkimuksessa oppilaat suorittivat kolme eri tehtävää käyttäen Alice-ohjelmointiympäristöä. Nance (2016) tutki kuuden viikon ajan oppilaiden ongelmanratkaisukykyjen kehittymistä ohjelmoinnin avulla. Opettajien rajallisten aikaresurssien ja tutkimusaikataulun haasteellisuuden vuoksi pitkittäistutkimusta tai perehtymistä oppilaan lähtökohtiin ei ollut tässä tilanteessa mahdollista tehdä, vaan arviointi oppilaiden osaamisesta tehtiin yhden testin perusteella, johon pyrittiin sisällyttämään oppilaita innostavia tehtäviä.

Tutkimuksen tasotestissä käytetyt tehtävät suunniteltiin tutkimusta varten itse. Tehtäviä suunnitellessa pohdittiin myös ohjelmointitaitojen mittaustapoja ja vastausten arviointia sekä huomioitiin ensimmäisten tehtävien matala kynnys onnistumiseen, jotta oppilas jatkaisi tai innostuisi enemmän jatkamaan tehtävien tekemistä. Vaikeammista tehtävistä tehtiin sellaisia, että niissä saattoi joutua käyttämään useamman konseptin tietotaitoja, jotta tehtävä tulisi ratkaistua oikein. Brennan ja Resnick (2012) pohtivatkin artikkelissaan, että oppilaan algoritmista ajattelua arvioidessa tehtävien tulisi olla sellaisia, jotka ovat käytännöllisiä oppijoilleen. Tehtävien tulisi sisältää luomista ja kriittisesti tutkivia projekteja. Yksittäiset projektit ovat konkreettisiä ja kontekstualisoituja esimerkkejä siitä, mitä voidaan tutkia ja analysoida monilla eri tavoilla. Useamman projektin kokoelma puolestaan tekee arvioimisesta vielä monipuolisempaa, kun tarjoutuu mahdollisuus nähdä, miten ymmärrys kehittyy ajan myötä. Brennanin ja Resnickin mielestä algoritmisen ajattelun eri osa-alueita ei voida mitata kaksijakoisesti ”osaat” tai ”et osaa”, vaan arvioinnin tulisi kuvata sitä, minkälaisesta lähtöasetelmasta oppilas on lähtenyt, missä hän on tällä hetkellä ja mitä hän vielä voi saavuttaa. He myös muistuttavat arvioinnin monipuolisuudesta: onko oppilas käyttänyt käsitettä tarkoituksellisesti ja oikein, pystyykö hän käymään läpi toimimatonta koodiaan debugaten, kykeneekö oppilas ymmärtämään muiden oppilaiden ratkaisuja ja pystyykö hän analysoimaan ja arvioimaan omaa ja muiden oppilaiden koodia? (Brennan ja Resnick 2012, 23.)

Tutkimusta suunnitellessa eri graafisiin ohjelmointiympäristöihin tutustuttiin ja lopulta tutkimukseen tehtävien tekemisalustaksi valittiin Scratch (kts. luku 2.7). Scratchin käyttöä puol-

sivat aiemmat tutkimukset Scratchin käytöstä ohjelmoinnin kannalta ja tehtävien palauttamisen helppous oppilaalta itseltään Scratchin ympäristössä. Scratch myös antaa vapauden tehdä ns. ”omia tehtäviä”, toisin kuin Code.orgin tai Koodaustunnin ympäristöt, jossa oppilas työstää ympäristön omia valmiita tehtäviä ja tarkistuttaa tehtävävastaukset ympäristössä.

Koetilanteen ohjeistus ja tehtävänannot on kuvattu liitteessä D. Kokeessa oli kolme aihealuetta, joita testattiin: toisto, ehtolauseet ja muuttujat. Oppilaan tuli vastata vähintään yhteen tehtävään jokaisesta aihealueesta. Näihin konsepteihin päädyttiin sen jälkeen, kun eri ohjelmointiympäristöjä oltiin tutkittu – esimerkiksi Code.orgin ympäristössä tehtäviä läpikäymällä oppilas oppii hyvin pian jo ”komentosarjoja”, tämän jälkeen ”silmukoita” ja vielä myös ”ehtolausekkeita” [1]. Myös paikallisten OPS:ien sisällöistä nämä konseptit nousivat esille (kts. luku 3.3).

Toisto-aihealueen tehtävässä pyrittiin testaamaan peräkkäisyyden ja silmukkarakenteen hallintaa. Ensimmäisen ja toisen tehtävän ratkaisussa oppilas voi joko ohjelmoida kissan ilman silmukkaa lisäämällä liikkumis- ja kääntymiskomennot peräjälkeen, tai silmukan kanssa lisäämällä käskyt silmukan sisään. Viimeisessä, vaikeimmassa tehtävässä, pelkän yhden ”liiku” -komennon käyttäminen vaatii ratkaisussa kahden sisäkkäisen silmukan käyttämistä.

Muuttuja-aiheisissa tehtävissä testattiin, ymmärtääkö oppilas muuttujan käsitteen siten, että muuttuja on tietovarasto, johon voi tallentaa tietoa, ja jota voidaan myös myöhemmin muokata. Ensimmäinen tehtävä jo mittaa sitä, osaako oppilas luoda muuttujan ja muuttaa sen arvoa jonkin tapahtuman myötä. Toisen tehtävän kohdalla oppilas saa aikaiseksi vuorovaikutusta ohjelman ja käyttäjän välillä, mutta myöskin tehtävän tehdessään kykenee käyttämään muuttujaa tekstimuotoisen syötteen lisänä, kun kissa tervehtii käyttäjää tämän syöttämällä tekstillä. Kolmas tehtävä vaatii eri konseptin, ehtolauseiden hallitsemista muuttujan käsitteen hallitsemisen lisäksi. Tässä tehtävässä oppilaalta vaaditaan onnistuneeseen suoritukseen myöskin vertailuehdon tekemistä.

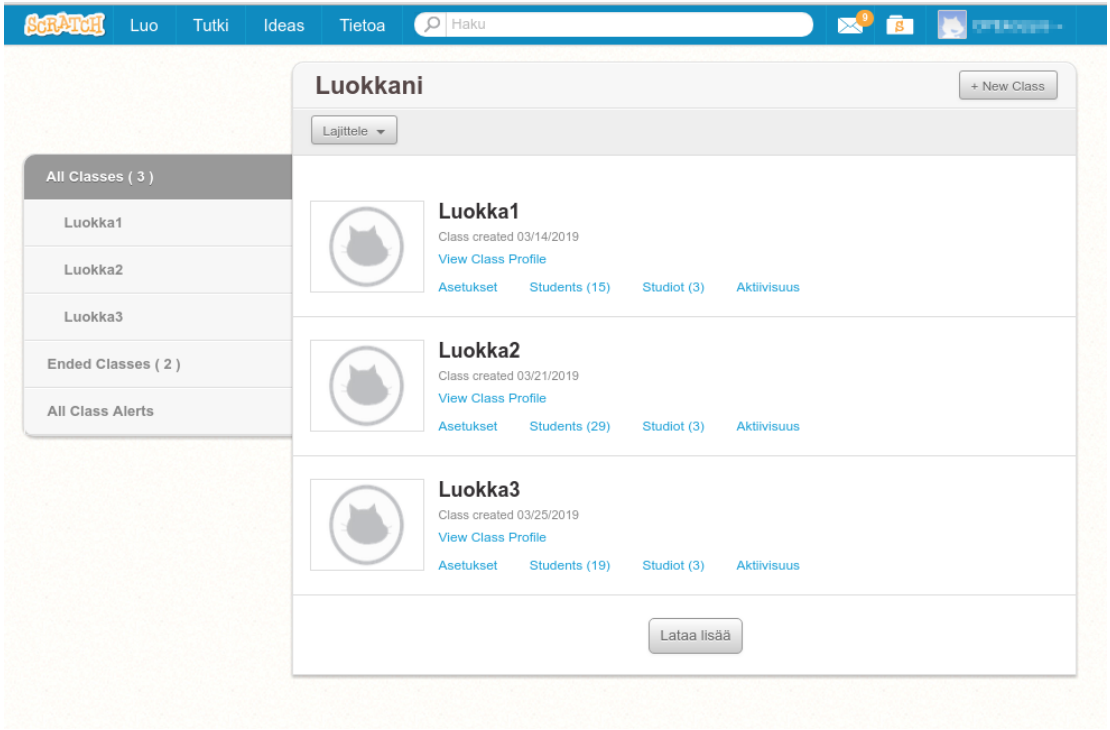
Ehtolause-tehtävissä pyrittiin testaamaan oppilaan kykyä muodostaa yksinkertaisia ehtoja. Kaksi ensimmäistä tehtävää testaa yksinkertaisinta if-ehtolauseen muotoa, viimeinen tehtävä testaa if-else-lauseen osaamista. Lisäksi kolmannen tehtävän suorittamiseksi oppilaan tuli

1. https://studio.code.org/s/course2, luettu 26.4.2019

osata käyttää sujuvasti muuttujia.

Aineiston kerääminen ja analysointi

Tutkimuksen aineisto kerättiin oppilailta sekä opettajilta. Ennen koetilannetta aineiston keruuta varten tuli tehdä Scratchiin oma opettajatunnus, jonka alle tehtiin jokaiselle tutkimukseen osallistuvalle luokalle oma luokka (kuvio 3). Luokan oppilaille tehtiin etukäteen omat tunnukset ja jokaiselle luokalle tehtiin kolme eri studiota. Nämä studiot toimivat tehtävien palautuslaatikkoina. Tutkimuksen jälkeen jokaisesta vastauksesta otettiin kuvakaappaus.



Kuvio 3. Scratchin opettajatunnuksen takaa näkyvä näkymä.

Opettajat haastateltiin ennen oppilaiden tekemää testiä. Haastatteluun kului 20-30 minuuttia ja se toteutettin koulussa paikan päällä (keskisuomalainen ja pohjoisavolainen opettaja) tai Skype-verkkopuhelun välityksellä (pirkanmaalainen opettaja).

Testin tekemiselle oli varattu aikaa yksi oppitunti (45-60min). Tutkimustilanteen aluksi oppilaille kerrottiin tutkimuksesta, sen taustoista, tutkimuksen vapaaehtoisuudesta, tutkimuksen läpiviennistä ja omien vastausten palauttamisesta. Lisäksi ennen tehtävien tekemisen aloit-

tamista oppilaille korostettiin tutkimuksen luottamuksellisuudesta ja tutkimukseen osallistujan anonymiteetistä. Tämän jälkeen oppilaat kirjautuivat Scratch-ohjelmointiympäristöön saamillaan tunnuksilla, tekivät vähintään yhden tehtävän jokaisesta aihealueesta ja palauttivat vastauksensa Scratchissa luotuun tehtäväkohtaiseen studioon. Tutkija oli läsnä tutkimustilanteessa ja vastasi oppilaiden teknisiin kysymyksiin kokeesta, mutta ei neuvonut oppilaita tehtävien tekemisessä. Pirkanmaalainen ja pohjoissavolainen opettaja olivat läsnä koetilanteessa, keskisuomalainen opettaja poistui luokasta muun luokan kanssa toiseen tilaan työskentelemään.

Tutkimuksen aineistoa käsiteltiin laadullisesti ja määrällisesti: oppilaiden ohjelmointitehtävien vastausten analysointi tapahtui tässä tutkimuksessa laadullisen analysoinnin keinoin (tehtävissä ilmenneitä ongelmakohtia käydään läpi, selitetään ja pyritään ymmärtämään syyt vastausten taustalla) sekä määrällisen analysointitapojen avulla (muodostaessa kokonaiskuvaa vastausten tunnusluvuista). Opettajan haastattelua analysoitiin lähinnä laadullisin keinoin, selitettäessä esimerkiksi ohjelmoinnin opetuksen työskentelytapoja ja oppilaiden taitotasoon vaikuttavia asioita. Opettajien haastattelu toteutettiin teemahaastatteluna, jota Hirsjärvi ja Hurme (2008, 48) luonnehtivat laadullisen ja määrällisen tutkimuksen välillä joustavaksi, teeman mukaisesti eteneväksi haastattelumuodoksi. Tutkimuksen haastattelussa kielenkäyttö ei ollut tarkasteltavana, joten suppea litterointitapa oli riittävä tässä tutkimuksessa (Saaranen-Kauppinen ja Puusniekka 2019, 79).

Koska aikaisempaa empiiristä tutkimusta ei ollut juurikaan, analysointi tapahtui aineistolähtöisesti. Tällöin pyrittiin muodostamaan tutkimukselle teoreettinen kokonaisuus tutkimuksen aineiston pohjalta (Tuomi ja Sarajärvi 2018). Laadullisen tutkimuksen analyysi voidaan Tuomen ja Sarajärven mukaan jakaa kahteen eri ryhmään: analyysimuodot, jotka ovat jonkin teorian ohjaamia ja analyysimuodot, joita ei ohjaa teoria, mutta niihin voidaan käyttää teoreettisia lähtökohtia (Tuomi ja Sarajärvi 2018). Eskolan ja Suorannan mukaan taas laadullisen tutkimuksen analyysitapoja on useita. He ovat jakaneet analyysimenetelmät kuuteen eri osaan: kvantitatiiviset analyysitekniikat, teemoittelu, tyypittely, sisällönerittely, diskursiiviset analyysitavat ja keskusteluanalyysi. Nämä analyysitavat ovat Eskolan ja Suorannan mukaan toistensa kanssa lomittuvia, eivätkä selvärajaisia. He toteavatkin, että tutkimuksessa harvoin pystyy käyttämään ja soveltamaan vain yhtä analyysitapaa. (Eskola ja Suoranta

1998.)

Opettajien haastattelun vastauksissa pyrittiin löytämään erilaisia teemoja, jotka voitaisiin koota yhteen. Teemoittelussa Tuomen ja Sarajärven (2018) mukaan painottuu aineiston sisällössä se, mitä kustakin teemasta on mainittu. Aineisto tällöin ryhmitellään ja jaotellaan eri aihepiireittäin (esimerkiksi iän mukaan), jonka myötä voidaan vertailla näiden eri teemojen esiintymistä aineistossa.

5 Tutkimustulokset

Tässä luvussa esitellään tutkimuksen tulokset opettajien haastattelun ja oppilaiden vastausten pohjalta. Ensin tuloksissa esitellään tutkimukseen osallistuneiden ohjelmoinnin opettamisen taustoja, kuten oppimisympäristöjä ja tuntimääriä, ja tämän jälkeen tutkimustuloksia käydään läpi tarkastellen konseptikohtaisesti oppilaiden antamia vastauksia.

Kuvioissa, joissa esitetään oppilaiden vastauksia, on anonymisoitu tekijätiedot, sillä niissä esiintyi pääteltäviä tunnistetietoja koulun nimeen liittyen. Vastauksissa kuitenkin viitataan oppilaiden vastauksiin tunnisteella Oppilas1 – Oppilas54 välillä.

5.1 Opettajien haastattelut

Luokkaa opettanut opettaja haastateltiin ennen oppilaiden tekemää testiä (kts. liite C). Haastattelun tarkoituksena oli kartoittaa ne käytänteet ja välineet, joilla ohjelmointia opetettiin tutkittavissa luokissa. Myös ohjelmoinnin opetuksen integrointia pyrittiin selvittämään haastattelun avulla. Näillä taustatiedoilla pyrittiin selittämään tuloksiin vaikuttavia tekijöitä.

Opettajien taustat ja opetusmetodit ohjelmoinnin parissa olivat samankaltaisia keskenään. Keskisuomalaisessa koulussa luokanopettaja oli toiminut luokkansa opettajana neljännestä vuosiluokasta lähtien. Hänen opetuksessaan oli ollut käytössä Scratch Jr -sovellus tabletlaitteilla, ja lisäksi Lego-roboteilla oli harjoiteltu Robolab-ohjelmointia. Oppilaat olivat voineet tässä koulussa kiinnostuksen mukaan valita viidennellä luokalla valinnaisen oppiaineen kurssilla ohjelmoinnin opiskelun. Pirkanmaalaisessa koulussa luokanopettaja oli toiminut luokkansa opettajana viidennestä vuosiluokasta lähtien. Ohjelmoinnin opetuksessa oli ollut käytössä vaihtelevasti Koodaustunti- ja Scratch-sivustot ja Bee-bot -robotit. Lisäksi opetuksessa oli käytetty erilaisia tablet-laitteiden sovelluksia. Opettajalla olisi ollut mahdollisuus käyttää TVT-opetuksessa apuna digitutoria, eli tietotekniikan opetukseen erikoistunutta opettajaavustajaa. Pohjoissavolainen opettaja oli myös opettanut omaa luokkaansa viidenneltä vuosiluokalta lähtien. Tässä kunnassa oli käytössä digitutor-opettajamalli, ja kunnan digitutor-opettaja olikin ollut oppitunneilla luokanopettajan apuna ohjelmoinnin opetuksessa. Kuten pirkanmaalaisessa luokassa, myös pohjoissavolaisessa luokassa oli ollut ohjelmoinnin ope-

tuksessa käytössä Scratch- ja Koodaustunti-sivustot ja BeeBot-robotit.

Aikaa ohjelmointiin oli käytetty vaihtelevia määriä. Keskisuomalaisessa koulussa oppilaat olivat opiskelleet ohjelmointia neljännellä luokalla viidestä kuuteen tuntia, viidennellä seitsemästä kymmeneen tuntia. Kuudennella luokalla ohjelmointia kyseinen luokka ei ollut ohjelmoinut ollenkaan. Pirkanmaalaisessa koulussa ohjelmoinnin opetusta oli ollut viidennellä ja kuudennella luokalla noin kymmenestä viiteentoista tuntia ja sitä oli suurimmaksi osaksi integroitu matematiikan opetukseen, mutta osa oppilaista oli myös ohjelmoineet osana historian opiskelua. Pohjoissavolaisessa koulussa viidennellä luokalla oppilaiden kanssa ohjelmoinnin opiskeluun oli käytetty n. 10 oppituntia, jotka olivat integroitu matematiikan opetukseen.

Kaikki opettajat haastattelun perusteella olivat ohjelmoinnin opettamisesta konseptina tyytyväisiä. Keskisuomalaisen koulun luokanopettajalla oli ohjelmointitaustaa, sillä hän kertoi aloittaneensa ohjelmoinnin opettamisen oppilaille jo 1990-luvun lopulla. Pirkanmaalaisen koulun opettaja kertoi, että kunnan digitutor-opettaja olisi ollut käytettävissä kerran kuussa koululla, mutta luokanopettaja koki omat taitonsa niin hyväksi matematiikan aineenopettajaopintojen ja noin 30 opintopisteen tietotekniikan yliopisto-opintojen vuoksi, ettei ollut kokenut tarpeelliseksi ottaa digitutor-opettajaa opetuksensa avuksi. Myös pohjoissavolaisen koulun luokanopettaja kertoi käyneensä käsityön oppiaineen sivuaineopinnot, jossa oli käyty myös ohjelmointia läpi.

Koulujen laitekannoissa oli suuriakin eroavaisuuksia. Haastatteluissa laitteiksi opettajat laskivat koulusta löytyvät tietokoneet ja tablet-laitteet, joita käytetään oppilaiden kanssa. Keskisuomalaisen koulun tilanne oli parhain laite/oppilas -suhteen ollessa yksi laite kolmesta neljään oppilasta kohden. Suurin ero oli pirkanmaalaisen koulun tilanteeseen, jossa laitekanta oli heikko – vain yksi laite yhdeksää oppilasta kohden. Pohjoissavolaisessa koulussa laitetta kohden oli noin neljä oppilasta.

Ohjelmoinnin opetus nähtiin yleisesti monipuolistavana asiana, joka ei niinkään ole uusi konsepti, vaikka opetussuunnitelmaan ohjelmointi on uutena asiana kirjattu. Opetuksen monipuolistamisen lisäksi ohjelmointi on ollut yhden opettajan mukaan hyvä lisä opetukseen, jolla saa oppilaat tekemään soveltavia ja muita lisätehtäviä helpommin. Pohjoissavolainen

opettaja totesi lisäksi, että ohjelmointi on hyvä lisä opetukseen, jotta oppilaat ymmärtävät miten nykyajan digiyhteiskunta ja laitteet toimivat.

Opettajat kertoivat opetuksen arjen olevan hyvin ”täynnä” perusasioita, joita oppilaille tulee opettaa. Tämän vuoksi yksikään opettaja ei ollut tutustunut yläkoulun opetuksen tavoitteisiin. Pirkanmaalaisen koulun opettaja naurahti, että voisikin tutustua yläkoulun opetuksen tavoitteisiin miettiessään ylöspäin eriyttäviä tehtäviä ja täten ohjelmoinnin opetuksessa perehdyttää oppilaita tekstuaaliseen ohjelmointiin. Keskisuomalaisen koulun luokanopettaja kuvaili puolestaan arkea opetuksessa:

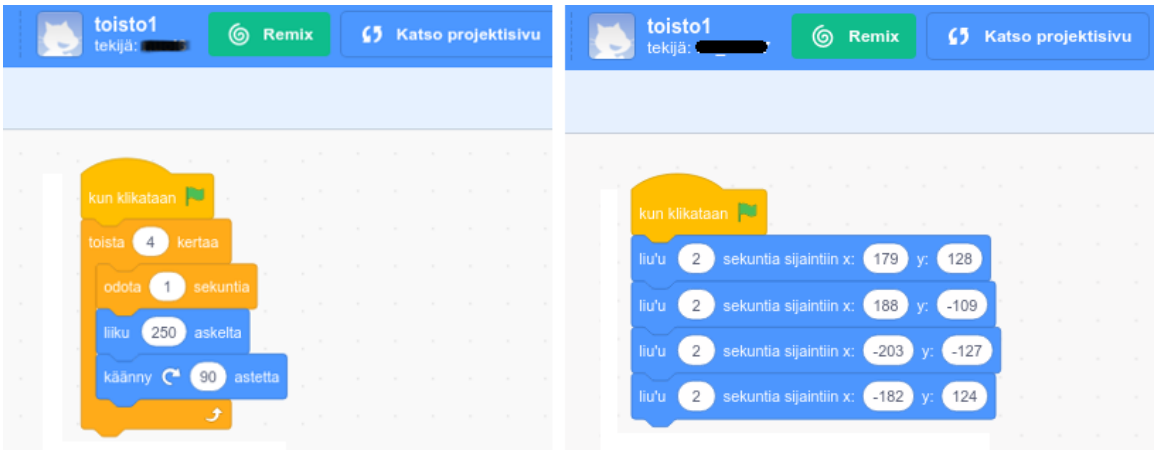
> ”– – täällä [alakoulussa] on [opetuksen sisällöissä] niin paljon tavaraa. – – Meidän [alakoulun opettajien] tehtävänä on yrittää saada nämä asiat, mitkä täällä on, opetettua, ja ne tavoitteet näille [oppilaille], – – jos he saavat sen niinku, sen hyvän taitotason täältä olevista sisällöistä, niin silloin meillä on pohja sinne yläkoulun, hmm, tavoitteiden saavuttamiseen.”

Kysyttäessä opetussuunnitelman arvosanan 8 ”toimivasta ohjelmasta”, ei yksikään opettaja osannut sanoa tarkkaan konsepteja tai muita sisältöjä, mitä tämän arvioinnin saava ohjelma sisältäisi. Pirkanmaalainen opettaja näki ohjelmoinnin vaa’ankieliasemassa olevana asiana matematiikan lopullista arvosanaa päätettäessä: esimerkiksi arvosanojen 9 ja 10 välillä oleva oppilas voi nostaa numeronsa erinomaiseen hyvällä ohjelmointiosaamisella. Keskisuomalaisen koulun opettaja nosti esiin formatiivisen arvioinnin, eli ohjelmoinnin osaamista arvostellessa tarkastellaan koko ohjelmoinnin opetuksen jaksoa. Hän arvioi myös kaikkien oppilaidensa läpäisevän tavoitteen ”toimivan ohjelman” tekemisestä. Pohjoissavolaisen koulun opettaja pohti, että hän arvioinnissaan katsoo luokan yleistä tasoa, ”mihin voi asettaa standardin” ja pohtii, että mikä on ryhmän taso, jonka perusteella arvioi oppilaan ohjelmoinnin osaamista. Hän puolestaan arvioi, että oppilaat menevät ”melko perustaidoilla” tällä hetkellä.

5.2 Toistorakenteen osaaminen

Ensimmäisessä oppilaiden tehtävässä näyttämöllä sijaitsevaa kissaa tuli liikuttaa siten, että kissa liikkuu suorakulmion muotoisen matkan: ensin vasemmalle, tämän jälkeen alas, sitten oikealle ja ylös, takaisin lähtöpaikkaan. Oikeaksi vastaukseksi luettiin kaikki ne vastaukset, jossa tämä liike toteutui (kts. kuvio 4). Alkeellinen oikea vastaus käsitti neljä liukumista eri

x- ja y-koordinaattien avulla tai neljä askellusta ja kääntymistä peräjälkeen. Ensimmäiseen tehtävään pystyi jo vastaamaan edistyneellä tavalla eli käyttäen silmukkaa, kuten kuviossa 4 nähdään.

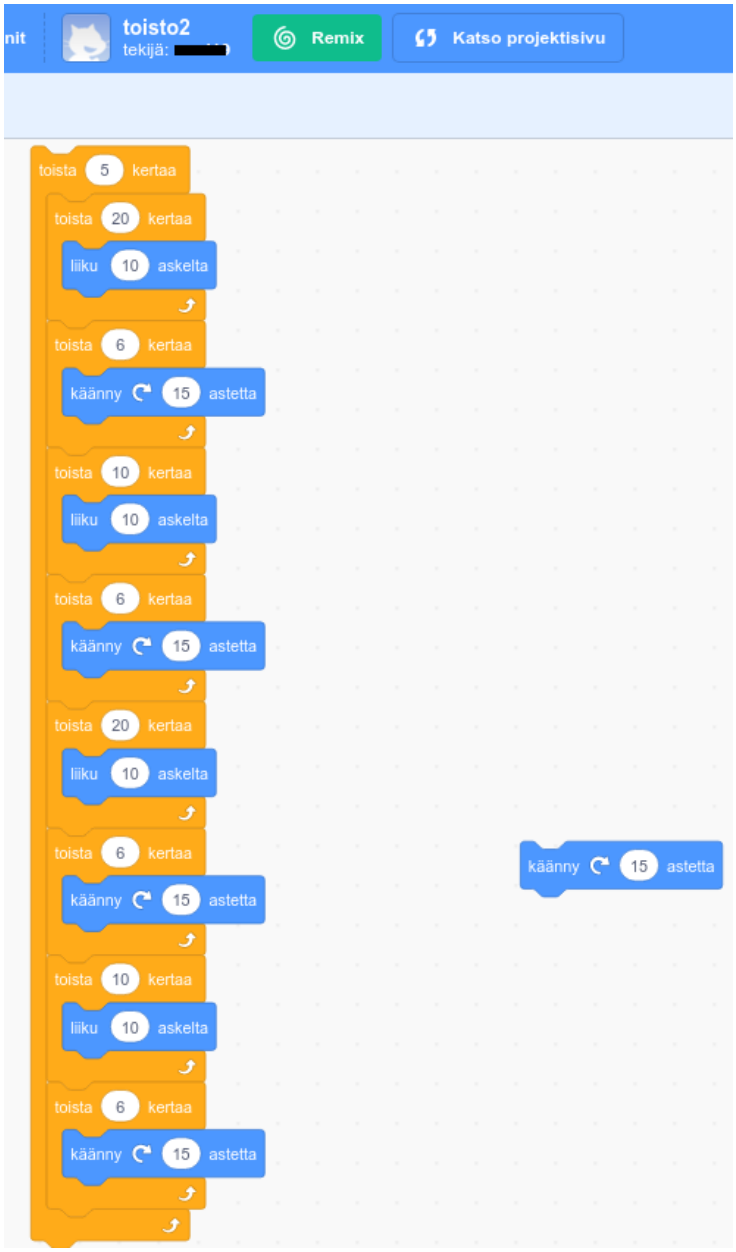


Kuvio 4. Vasemmalla Oppilas14:n edistynyt ratkaisu Toisto1-tehtävään. Oikealla Oppilas7:n alkeellinen ratkaisu tehtävään.

Toisessa tehtävässä kissan liikuttaminen tuli sijoittaa silmukan sisään ja liikuttaa kissaa samalla tavalla kuin ensimmäisessä tehtävässä. Oikeiksi vastauksiksi luettiin ne vastaukset, joissa kissa liikkui halutullta tavalla ja liikuttamiseen oli käytetty silmukkaa.

Viidessä ensimmäisen ja neljässä toisen tason vastauksessa kissan liikuttamisen toistamista oli ratkaistu laittamalla kissa liikkumaan esimerkiksi kymmenen askelta, mutta tämä liikkuminen oli sijoitettu silmukkaan, jossa liikkuminen toistettiin useamman kerran (kts. kuvio 5). Vastaus on esimerkki oikeasta vastauksesta, joka on kuitenkin huonosti ratkaistu. Tämä voisi olla merkki ohjelmointiperiaatteiden tuntemattomuudesta: uskooko oppilas, että kyseistä komentoa ei suoriteta, ellei sitä laiteta ”toistamaan”, vai unohtuuko oppilaalla, että liikkumisen askelmääriä voi kasvattaa ilman silmukan toistomääriä kasvattamalla? Todennäköistä on, että oppilas haluaa kissan liikkeen näkyviin, sillä muuten kissa vain ”ilmaantuisi” niihin paikkoihin, jossa se kääntyy.

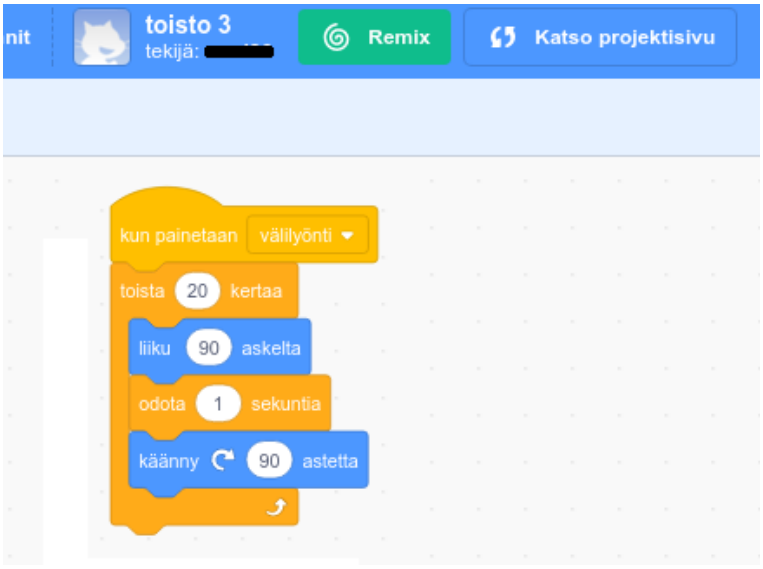
Kolmannen tason tehtävässä haettiin sisäkkäisten silmukoiden käyttämistä. Kuitenkin tehtävänanto antoi ymmärtää, että riittää, kun käyttää yhtä liiku-komentoa, mutta toistojen määrä kasvatetaan kahteenkymmeneen (kuten kuviossa 6). Vastaukset, joissa oli käytetty kahta sisäkkäistä silmukkaa ja yhtä, toistomäärältään kahtakymmentä toistoa, silmukkaa käyttäen



Kuvio 5. Oppilas27:n sisäkkäisten silmukoiden ratkaisu Toisto2-tehtävässä.

luettiin oikeiksi tehtävänannon tulkinnanvaraisuuden vuoksi.

Toistotehtäviin tuli eniten oikeita vastauksia, tosin tämä johtunee siitä, että oppilaita ohjeistettiin aloittamaan ensimmäisestä toistotehtävästä. Toistotehtävien oikeiden vastausten osuus on esitelty taulukossa 1. Tarkempia yksityiskohtia vastausten jakautumisesta esitellään luvussa 5.5.



Kuvio 6. Oppilas33:n ratkaisu Toisto3-tehtävässä.

Taulukko 1. Oikeiden vastausten suhteellinen osuus osallistujien lukumäärästä toistotehtävissä (n=54).

Konsepti	Tehtävä	Oikeat vastaukset
	1	35,2 %
Toisto	2	24,1 %
	3	7,4%

5.3 Muuttujien osaaminen

Muuttujien osaamista mittaavissa tehtävissä ensimmäisenä tehtävänä oli tehdä muuttuja ”elämä” ja kasvattaa sitä yhdellä aina välilyöntiä painamalla. Yksinkertaisin ratkaisu oli tehdä muuttuja, ja komentoalueella yhdistää kaksi blokkia - ”`kun painetaan <välilyönti>`” → ”`lisää muuttujaan <elämä> arvo <1>`” . Tässä vastauksessa tosin ohjelman suorittamisessa muuttujaa ei alustettu. Muuttujan alustaminen tapahtui vain muutamassa vastauksessa: tällöin komennot olivat esimerkiksi ”`kun klikataan <vihreä lippu>`” → ”`aseta <elämä> arvoon <0>`”, ja ”`kun painetaan <välilyönti>`” → ”`lisää muuttujaan <elämä> arvo <1>`”.

Toisessa tehtävässä oli tarkoitus kysyä käyttäjältä nimeä ja tämän jälkeen käsitellä muuttujaa

siten, että se tulee osaksi tervehdystä: ”Moikka, <nimi>”. Onnistuneista vastauksista lähimmäksi tätä pääsivät komentosarjat, jossa kissa laitettiin nimen kysymisen jälkeen sanomaan ensin ”Moikka” ja tämän jälkeen omana blokkinaan ”<vastaus>”, kuten kuviossa 7 näkyy. Muuttujan yhdistäminen tekstin yhteyteen ei ollut siis tuttua oppilaille.

Kuvio 7. Oppilas10:n ratkaisu Muuttujat2-tehtävään.

Kolmannessa tehtävässä oppilaan tuli katsoa ennalta tehtyä koodia ja muokata sitä lisäämällä siihen koodia. Puolivalmiissa ohjelmassa oli taustana valittuna Scratchin oma yksinkertainen piirros maasta ja taivaasta ja ohjelman alkaessa muuttuja ”pisteet” asetettiin arvoon 10. Ohjelman suorittamisen aloittamisen jälkeen koripallo alkoi tippua satunnaisesta kohdasta katosta kohti alareunaa. Omaa koodia tuli lisätä valmiiseen pohjaan siten, että aina pallon osuessa maahan (eli ruskeaan osaan taustasta). Yksikään tutkimukseen osallistunut oppilas ei osannut ratkaista tätä tehtävää. Tehtäviin vastattujen oikeiden vastausten osuudet osallistujiin nähden on esitelty taulukossa 2. Tarkempia yksityiskohtia vastausten jakautumisesta

Taulukko 2. Oikeiden vastausten osuus osallistujien lukumäärästä muuttujatehtävissä (n=54).

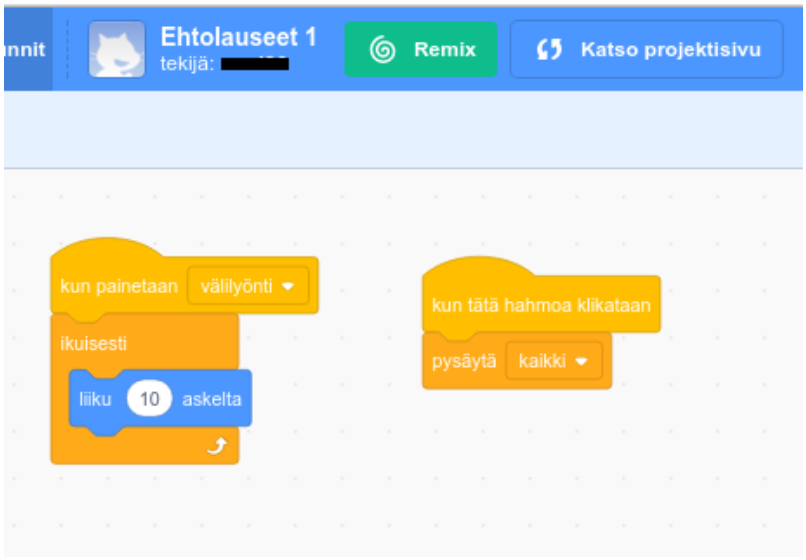
Konsepti	Tehtävä	Oikeat vastaukset
Muuttujat	1	22,2 %
	2	5,6 %
	3	0 %

esitellään luvussa 5.5.

5.4 Ehtolauseiden osaaminen

Ehtolausetehtävissä pyrittiin testaamaan sitä, onko oppilas oppinut ehtolauseen konseptin ja osaako tämä käyttää ehtolausetta yksinkertaisissa ohjelmissa.

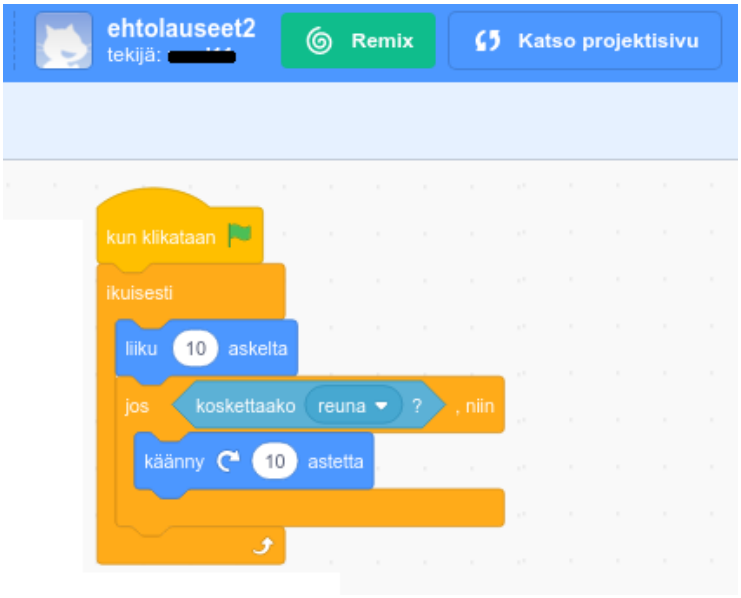
Ensimmäisessä tehtävässä kissan tuli liikkua ikuisesti, kunnes hiiren osoittimen osuessa kissahahmoon sen liike pysähtyy. Muutamassa Ehtolause1-tehtävän vastauksessa esiintyi ratkaisuna `"kun hahmoa klikataan" → "pysäytä kaikki"` -toiminnallisuuden käyttäminen (kuten kuviossa 8), vaikka oikea ratkaisu olisi ollut `"ikuisesti" → "kosketaako hahmo <hiiren osoitin>?" → "pysäytä kaikki"`.



Kuvio 8. Oppilas34:n ratkaisu Ehtolause1-tehtävään.

Ensimmäisen ja toisen kohdan tehtävien ratkaisuissa ei ollut suurta eroa, mutta toisessa tehtävässä oli havaittavissa ns. ”pong”-pelin ominaisuuksista. Ehtolause2-tehtävässä kissan tuli jatkaa liikettä osuessaan seinään, mutta tällöin liike muuttui ennalta määritellyn asteluvun verran eri suuntaan. Tämän kohdan tehtävän oikea ratkaisu näkyy erään oppilaan ratkaisussa kuviossa 9.

Kolmas tehtävä oli testin haasteellisin. Käyttäjältä tuli kysyä kahta lukua, joiden perusteella tehtiin vertailua, kumpi numeroista oli suurempi ja kissan tuli sanoa vertailun lopputulos



Kuvio 9. Oppilas19:n ratkaisu Ehtolause2-tehtävään.

ääneen. Oikea vastaus sisälsi tällöin sisäkkäisiä ehtolauseita, ”jos – tai muuten” -ehtolauseen ja loogisten vertailuoperaattoreiden käyttöä kahden muuttujan käytön välillä. Ehtolause3 -tehtävään ei tullut tutkimukseen osallistuvilta ollenkaan palautuksia.

Ehtolauserakenne ei ollut tuttu tähän kokeeseen osallistuneille oppilaille (kts. taulukko 3). Kun palautettuja vastauksia Ehtolause1-tehtävään oli 21, ehdon muodostaminen tapahtui vain kuudessa vastauksessa, joista kolmella oppilaalla vastaus oli oikein. Tarkempia yksityiskohtia vastausten jakautumisesta esitellään luvussa 5.5.

Taulukko 3. Oikeiden vastausten osuus osallistujien lukumäärästä ehtolausetehtävissä (n=54).

Konsepti	Tehtävä	Oikeat vastaukset
	1	5,56 %
Ehtolause	2	3,7 %
	3	0 %

5.5 Yhteenveto

Taulukossa 4 on esitetty vielä kootusti oikeiden vastausten suhteellinen osuus verrattuna kaikkiin osallistujiin (n=54). Oikeiden vastausten määrä oli pieni suhteutettuna kaikkiin tutkimukseen osallistuneisiin oppilaisiin, sillä ensimmäisessä tehtävässä vain reilu kolmannes tutkimukseen osallistuneista oppilaista vastasi oikein. Muissa konsepteissa ensimmäisen tehtävän oikeat vastausmäärät olivat heikommat, vaikka oppilaita kehoitettiin hyvissä ajoin siirtymään näihin tehtäviin. Vaativimmissa tehtävissä, joissa mitattiin lähinnä oppilaan harrastuneisuutta tai aikaisempaa ohjelmointikokemusta, oikeita vastauksia oli vain muutamia.

Taulukko 4. Oikeiden vastausten suhteellinen osuus osallistujien lukumäärästä.

Konsepti	Tehtävä	Oikeat vastaukset
Toisto	1	35,2 %
	2	24,1 %
	3	7,4%
Muuttujat	1	22,2 %
	2	5,6 %
	3	0 %
Ehtolause	1	5,56 %
	2	3,7 %
	3	0 %

Reilu viidennes kokeeseen osallistuneista osasi vastata oikein Muuttujat1-tehtävään. Kuitenkin koetilanteessa osallistujien esittämät kysymykset tehtävänannosta ja siinä esiintyneestä käsitteestä ”muuttuja” viittaisi siihen, että muuttujan käsitettä ei ollut sisäistetty siten, että muuttujaa osattaisiin käyttää hyödyksi ohjelmoinnissa. Tätä tukevat myös taulukon 5 lukemat, sillä Muuttujat1-tehtävään palautetuista vastauksista alle puolet (40 %) oli oikein.

Taulukkossa 5 tarkastellaan kokeeseen osallistuneiden oikeiden vastausten määrää tehtäväkohtaisesti palautetuista tehtävistä. Taulukossa esitetään myös osuudet vastausten määrästä suhteutettuna kokeeseen osallistuneisiin, eli tämä kertoo siitä, kuinka moni osallistuja on yrittänyt ratkaista tehtävää. Taulukossa myös on jaoteltu tehtävien ratkaisuja siten, että pa-

lautettu vastaus on voinut olla oikein hyvällä ratkaisulla, oikein huonolla ratkaisulla tai sitten oppilas on palauttanut väärän tai tyhjän vastauksen. Mielenkiintoista olisi ollut tehdä jaottelu tyhjien ja toisaalta epätyhjien, mutta väärien vastauten välillä, mutta ns. väärät vastaukset olivat poikkeuksetta vastauksia, joista ilmeni ettei oppilas ollut tehtävänantoa ymmärtänyt lainkaan tai hänellä ei ole ollut ymmärrystä ratkaista tehtävää. Suluissa olevat luvut ovat suhteellisia osuuksia palautettujen vastausten kokonaismäärästä.

Taulukko 5. Vastaukset luokiteltuina

Konsepti	Tehtävä	Vastauksia yhteensä	Vastaajien osuus	Vastaus oikein järkevällä koodilla	Vastaus oikein, mutta ei-järkevä koodi	Tyhjä/väärä vastaus
Toisto	1	41	75,9 %	11 (26,8 %)	8 (19,5 %)	22 (53,7 %)
	2	17	31,4 %	8 (47,0 %)	5 (29,4 %)	4 (23,5 %)
	3	5	9,3 %	1 (20 %)	3 (60 %)	1 (20 %)
Muuttujat	1	30	55,6 %	8 (26,7 %)	4 (13,3 %)	18 (60 %)
	2	9	16,7 %	1 (11,1 %)	2 (22,2 %)	6 (66,7 %)
	3	2	3,7 %	0	0	2 (100 %)
Ehtolause	1	21	38,9 %	1 (4,8 %)	2 (9,5 %)	18 (85,7 %)
	2	4	7,4 %	2 (50 %)	0	2 (50 %)
	3	2	3,7 %	0	0	2 (100 %)

Tarkastellessa kouluittain palautettuja tehtävämääriä ja niiden joukosta oikeiden vastausten lukumäärää (kts. taulukko 6), voidaan huomata keskisuomalaisen ja pirkanmaalaisen koulun oppilaiden onnistuneen testin teossa suhteellisen samalla tasolla: reilu puolet palautetuista vastauksista oli oikein. Pohjoissavolaisessa koulussa palautettiin vain 18 vastausta, joista yksi oli oikein. Tällöin oikeiden vastausten suhteellinen osuus jää reiluun viiteen prosenttiin. Tämä osittain vaikuttaa laskevasti myös oikeiden vastausten suhteelliseen määrään ko-

Taulukko 6. Oikeiden vastausten suhteellinen osuus palautetuista vastauksista kouluittain.

Koulu	Palautetut vastaukset	Oikeat vastaukset	Oikeiden vastausten osuus palautetuista vastauksista
Keski-Suomi	32	17	53,1 %
Pirkanmaa	74	40	54,0 %
Pohjois-Savo	18	1	5,5 %

ko osallistujamäärästä. Kyseisen luokan opettaja haastattelussa totesi luokan työskentelyssä olleen haasteita, joten vastausten vähyys ja erot muihin luokkiin johtunevat tästä.

Oli luokasta kiinni, minkälainen työskentelykulttuuri luokassa vallitsi ja täten myös se, miten hyvin oppilaat kykenivät orientoitumaan testin tekemiseen. Orientoitumisen myötä käytettävissä oleva aika testin tekemiseen saattoi olla hieman erilainen eri luokille, mikäli oppilaita täytyi erikseen kehottaa keskittymään testin tekemiseen tai ohjeistuksen kuuntelemiseen. Jokainen oppilas ohjeistettiin aloittamaan tehtävien tekeminen Toisto1-tehtävästä, ja tämän jälkeen oppilas sai edetä omaan tahtiin tehtävissä. Kuitenkin testin tekemistä rytmitettiin siten, että oppilaita kehotettiin tarvittaessa siirtymään seuraavan konseptin tehtäviin, jotta kaikkien konseptien ensimmäisiin tehtäviin ehdittäisiin tutustua ja vastaamaan. Tämä toteutui tutkimukseen osallistuneiden pirkanmaalaisen ja keskisuomalaisen koulun tutkimusluokissa hyvin – palautuksia tehtäviin tuli usealta tutkimukseen osallistuneelta oppilaalta. Tutkimukseen osallistuneen pohjoissavolaisen koulun luokan työskentely taas puolestaan oli heikkoa, sillä koeasetelman muodostaminen ei onnistunut: oppilaat eivät mieltäneet oppituntia koetilanteeksi vaan tavalliseksi oppitunniksi, jolla pääsi käyttämään tietokonetta. Tämä johti siihen, että oppilaat keskustelivat keskenään läpi koetilanteen, eivätkä keskittyneet koetehtävien tekemiseen. Kuitenkin jokaisesta luokasta suurin osa oppilaista palautti edes yhden tehtävän (taulukko 7):

Taulukko 7. Vähintään yhden vastauksen palauttaneiden osallistujien määrä luokittain

Koulu	Vähintään yhden vastauksen palauttaneiden osuus osallistujista	Osallistujia
Keski-Suomi	87,5 %	8
Pirkanmaa	81,4 %	27
Pohjois-Savo	78,9 %	19

6 Johtopäätökset ja pohdinta

Tutkimuksen tavoitteena oli selvittää 6.-luokkalaisten oppilaiden taitotasoa ohjelmoinnissa alakoulun päättyessä ja samalla kartoittaa ohjelmoinnin opetuksen tapoja. Taitotasoa tutkittiin kolmen konseptin taitotasotestillä, jossa tehtävät vaikeutuivat konseptin sisällä. Lisäksi tähän taitotasoon vaikuttavia asioita pyrittiin selvittämään luokanopettajan haastattelulla, jolla selvitettiin mm. laitteistoa, käytettyjä ohjelmointiympäristöjä ja ohjelmointiin käytettyjä tuntimääriä. Yksi keskeinen tavoite tutkimuksessa oli kartoittaa oppilaiden edellytyksiä siirtyä yläkoulun puolelle tekstuaalisen ohjelmoinnin pariin. Tutkimuksessa pyrittiin tarkastelemaan niitä ohjelmoinnin aiheita, joita oppilaat olivat saavuttaneet parin vuoden aikana ja kartoittamaan sitä, oliko ohjelmoinnin aiheista jokin tietty alue erityisen vaikea oppilaille.

Tutkijan tiedon mukaan vastaavaa aiempaa empiiristä tutkimusta ei olla vielä tehty Suomen kouluissa. Tämä vaikutti koeasetelman muodostamiseen, joka rakentui teorian pohjalta. Tutkimustulosten tulkinta tehtiin aineiston pohjalta teoriaan peilaten.

6.1 Keskeisimmät johtopäätökset

Seuraavaksi käydään läpi tutkimuksen keskeisimpiä johtopäätöksiä tutkimuskysymyksittäin.

6. luokan oppilaiden osaaminen ohjelmoinnissa alakoulun päättyessä

Oppilaiden osaamista ohjelmoinnissa tutkittiin kolmen eri konseptin testillä, jossa tehtävät vaikeutuivat. Suurin osa osallistujista vastasi vähintään yhteen kysymykseen yhdeksästä, mutta silti oikeiden vastausten suhteelliset osuudet vastauksista olivat helpoimmissakin tehtävissä pieniä, alle 50 %.

Toisto- ja muuttujat-tehtäväkonseptien ensimmäiset tehtävät olivat hieman hankalia tutkimukseen osallistuneille. Toisto1-tehtävän palautetuista vastauksista hieman alle puolet oli oikein. Muuttujat1-tehtävän vastauksista vain kaksi viidestä oli oikein. Kuitenkin näistä tehtävissä saatiin suurin piirtein saman verran järkevällä tavalla ratkaistuja vastauksia (26,8 % ja 26,7 % vastauksista). Ehtolause1-tehtävään palautetuista vastauksista vain reilu 14 % oli

oikein, joka on toisin sanottuna yksi vastaus seitsemästä.

Ehkä merkittävin väärien vastausten osuus havaittiin Ehtolause1-tehtävässä, johon vastauksia palautti vajaa 40 % osallistuneista oppilaista. Kuitenkin näistä vastauksista tyhjiä tai vääriä vastauksia oli 85,7 %. Tehtävä oli tyypillisesti jokaisessa koetilanteessa viimeinen tehtävä, jota tutkimukseen osallistuva ratkoi. Tämä voi olla yksi syy suureen tyhjä/väärä -prosenttiin, kun oppilas on ollut jo valmis omasta mielestään siirtymään välitunnille tai muihin tehtäviin.

Tarkastellessa palautusprosentteja ja oikeiden vastausten määriä (taulukko 5 luvussa 5.5) tulee muistaa, että koetilanteen läpiviennillä on vaikutusta palautusten määrään. Mikäli tehtävien suoritusjärjestystä olisi muutettu konsepteittain esimerkiksi siten, että ehtolauseita käsittelevät tehtävät tulevat ennen muuttujia, lukemat voisivat olla erilaiset näiden osalta. Nyt trendi on laskeva tehtävien ensimmäisten tehtävien kohdalla – muuttujat-aiheisiin tehtäviin vastattiin enemmän kuin ehtolause-aiheisiin tehtäviin. Konseptien sisällä tehtävien vastausmäärät ovat myös laskevia, sillä tehtävät ovat odotetusti oppilaidenkin mukaan vaikeutuneet.

Saman taulukon oikeiden vastausten tehtäväkohtaisista prosenttiosuuksista nähdään, että osallistujista muutamat yrittivät tehdä myös Muuttujat- ja Ehtolause -konseptien vaikeimpia tehtäviä, mutta eivät onnistuneet tekemään niitä oikein. Näiden vastausten osalta syitä epäonnistumiseen voi olla useampi - esimerkiksi koeajan päättyminen, väärinymmärretty tehtävänanto tai tehtäväratkaisun monimutkaisuus.

Pohjoissavolaisen koulun luokan tuloksia tarkastellessa on oltava hieman kriittinen, sillä on vaikeaa sanoa, kuvastikovat nämä saadut tulokset heidän todellista osaamistasoaan. Vedettäessä tuloksista johtopäätöksiä tulee pohtia, johtuiko huono koetilanteeseen orientoituminen yleisesti luokan työskentelykulttuurista vai oliko poikkeava oppitunti oppilaiden mielestä jännittävä, jonka vuoksi oppilaat olisivat reagoineet ryhmänä koetilanteen muodostamiseen rauhattomuutena. Toisaalta on myös pohdinnan arvoinen huomio, että huono orientoituminen tehtävien tekemiseen voi johtua siitä, että ohjelmointi aiheena ei ole oppilaille tuttua.

Se, mitä oppilaat todella osaavat ohjelmoinnista, on haastavaa johtaa näistä tuloksista. Kokonaisuutta katsellessa parhaiten osatuin tehtävä on toisto-konseptin aloitustehtävä, mutta tämä oli myös tehtävä, josta kaikki oppilaat aloittivat tehtävien tekemisen. Muutamat oppi-

laat, jotka olivat mitä ilmeisimmin harrastuneita ohjelmoinnin saralla vapaa-ajallaankin, suoriutuivat myös vaikeammista tehtävistä. Tarkastellessa tuloksia voidaan todeta, että oppilaat osaavat hyvin niukasti ohjelmoinnin peruskonsepteja, kuten muuttujia ja ehtolauseita. Kuitenkin muutamissa paikallisen tason OPS:ssa oli määritelty, että oppilas osaa käyttää muuttujia, ehtolauseita ja toistorakenteita ohjelmoidessaan (Jyväskylän kaupunki 2016; Kuopion kaupunki 2019). Näiden tavoitteiden valossa suurimman osan oppilaiden osaaminen jäi heikommaksi.

Ohjelmoinnin opetus alakoulussa

Toisena tutkimuskysymyksenä kartoitettiin, millä tavoin ohjelmointia opetetaan alakouluissa. Tutkimukseen osallistuneiden opettajien haastattelusta ilmeni, että jokainen opettaja oli jollain tavalla suuntautunut aiemmissa opinnoissaan ohjelmoinnin opettamiseen tai vähintäänkin koki itsensä ohjelmointimyönteiseksi opettajana. Opettajat nostivat haastattelun aikana esille perustellen sen, miksi ohjelmointi on hyvä lisä opetukseen ja olivat sitä mieltä, että ohjelmoinnille on oma paikkansa opetuksessa. Tutkimukseen osallistuneiden opettajista kaksi oli käyttänyt ohjelmointia opetuksessaan oma-aloitteisesti omien taitojensa pohjalta, yksi opettaja oli käyttänyt aktiivisesti kunnan digitutor-opettajaa apuna ohjelmoinnin opetuksessa. Opettajilla oli hyvin erilaiset näkemykset siitä, miten ohjelmointia osana opetettavaa ainetta ja oppilaan ohjelmoinnin osaamista arvioidaan.

Ohjelmoinnin opetusta tapahtui kouluissa pääosin matematiikan tunteihin integroiden, mutta myös esimerkiksi historian opetukseen ohjelmointia oli käytetty oppilaan omasta aloitteesta. Kaikille opettajille ja oppilaille Scratch oli graafisena ohjelmointiympäristönä tuttu. Kaksi kolmesta opettajasta mainitsi lisäksi käyttäneensä opetuksessaan Koodaustuntia eli ohjelmointiympäristöä, jossa oppilas siirtyy itsenäisesti tehtävästä ja opeteltavasta aiheesta toiseen ja ympäristö tarkistaa oppilaan tehtävävastaukset. Yksi opettajista kertoi käyttäneensä myös Lego-robotteja ja Scratch Jr -sovellusta ohjelmoinnin opetukseen.

Ohjelmoinnin opetukseen on varattu opettajasta riippuen vähän tai kohtuullisesti aikaa. Tyypillisesti ohjelmoinnin opiskelu tapahtui yhden teemajakson aikana, johon oli varattu aikaa vuosiluokittain viidestä viiteentoista oppituntia. Aiemmissa tutkimuksissa aikaresurssien vä-

hyys on arvioitu suurimmaksi haasteeksi TVT-opetuksen integrointiin muuhun opetukseen (ks. Şahin-Kizil 2011).

6.-luokkalaisten edellytykset siirtyä yläkoulun puolelle tekstimuotoiseen ohjelmointiin

Viimeisenä tutkimuskysymyksenä selvitettiin oppilaiden edellytyksiä siirtyä yläkoulun puolella tekstuaaliseen ohjelmointiin kuudennen luokan päätyttyä. Tarkastellessa kokonaiskuvaa vastauksista niin määrällisesti kuin laadullisestikin, voidaan todeta, että edellytykset siirtyä tekstuaaliseen ohjelmointiin ovat testin tulosten perusteella heikohkot suurimmalle osalle oppilaista. Suurin osa oppilaista ei kyennyt tekemään helpointa toistorakenteen osaamista mittaavaa tehtävää. Tätä tulosta mukailee myös Valtioneuvoston (Kaarakainen ym. 2017, 26) raportti, jossa todetaan yhdeksän oppilaan kymmenestä jääneen kokonaan vaille pisteitä alkeisohjelmoinnin tehtävissä.

Tekstuaaliseen ohjelmointiin siirryttäessä oppilaan tulee ymmärtää erityisesti tietokoneen muistin osuus ohjelmoinnissa: muuttujien ja erilaisten tietorakenteiden käyttö vaatii muistia ja muistipaikkojen hyödyntämistä. Varsinkin muuttujien konseptia mittaavien tehtävien suoritusprosentit olivat alhaiset. Syitä muuttujien konseptin osaamattomuuteen voi olla useampia. Code.orgin opetussisältö on tehty siten, että muuttujien käsite tulee ensimmäistä kertaa käsiteltäväksi neljännen kurssin kuudennella oppitunnilla. Kursseilla 2-3 käsitellään tätä ennen paljon toistoa, silmukoita (myös sisäkkäisiä), ehtolauseita ja jopa funktioita. Scratchin opetuskäyttö puolestaan voi rakentua pelkästään opettajan omalle ideoinnille ja tällöin opettajasta johtuen oppilas voi opiskella ohjelmoinnista pelkästään muita konsepteja kuin muuttujia.

Muita johtopäätöksiä

Testin tekemiseen käytetty aika, eli oppitunti, koettiin riittäväksi. Tässä ajassa tutkimukseen osallistujat ohjeistettiin testin tekemiseen (tunnusten käyttämiseen ja tehtävien palauttamiseen) ja kykenivät tekemään tehtäviä niin pitkälle kuin kykenivät, eikä koetilanteessa tullut vastaan sellaista tilannetta, jossa tutkimukseen osallistunutta lasta olisi tarvinnut erikseen kehottaa lopettamaan testin tekeminen ajan loppumisen vuoksi. Osa tutkimukseen osallistu-

vista oppilaista kysyi jo ennen oppitunnin päättymistä testin tekemisen lopettamista, koska osallistuja on kokenut olevansa valmis tehtävien kanssa. Näillä osallistujilla tehtävien tekeminen oli ollut suppeaa, eli pelkästään jokaisen konseptin ensimmäisten tehtävien tekemistä, mikä viittaisi siihen, että osallistuja ei ole osannut tehdä tehtäviä, eikä ole halunnut ponnistella tehtäviä tehdessään. Ainoastaan pohjoissavolaisen koulun luokan kanssa olisi voinut koeasetelman luomiseen käyttää enemmän aikaa – yksi oppitunti tietokoneluokaan siirtymiseen sekä koeasetelman alustamiseen, tunnuksilla ympäristöön kirjautumiseen ja tehtävien palauttamisen ohjeistukseen ja toinen oppitunti sitten varsinaisen testin tekemiseen.

6.2 Taitotasoon vaikuttavat seikat

Ohjelmoinnin osaamisen taustalla on todennäköisesti usean asian yhteisvaikutus: esimerkiksi opettajan asenne ja saama tuki opetettavaan aiheeseen, laitekanta koulussa ja kouluyhteisön tuki laitteiden käyttämiseen. Haastatteluissa kävi ilmi, että opettajien asenteet ohjelmoinnin opettamiseen olivat hyviä ja myönteisiä, ja tukea ohjelmoinnin opetukseen oli joko riittävästi saatavilla tai he kokivat itsensä päteviksi aiempien opintojensa myötä. Keskisuomalaisessa koulussa opettaja oli opettanut ohjelmointia oppilaille ensimmäistä kertaa 1990-luvun lopulla. Pirkanmaalaisen koulun luokanopettaja oli opiskellut arviolta 30 opintopistettä tietotekniikkaa ja täten myös ohjelmointia yliopistossa. Myös pohjoissavolaisen koulun opettaja oli aiemmissa opinnoissaan tutustunut erikseen ohjelmoinnin opettamiseen. Tässä tutkimuksessa opettajan asenne ei näkynyt kuitenkaan erityisen hyvänä tuloksena oppilaiden vastauksissa. Tutkimukseen osallistuneet opettajat kertoivat haastetteluissa, että koulun puolesta erityistä ohjelmoinnin opettamiseen keskittynyttä koulutusta ei ole ollut tarjolla, vaan kahdella kouluista oli käytössä kunnan digitutor-malli, jossa tutoropettaja pitää koulutuksia luokanopettajille digityökalujen käytöstä.

Kansallisella tasolla tuen saamisessa on kuitenkin koettu puutteita. Aiempi tutkimus on osoittanut, että opettajat kokevat tarvitsevansa lisätukea ja ohjausta ohjelmoinnin opettamiseen (Kurkinen 2018, 59). Myös OAJ:n (2019, 8) selvityksessä opettajat ja rehtorit olivat sitä mieltä, että digitaalisiin työvälineisiin liittyvässä ohjauksessa ja koulutuksessa on puutteita. Sama puute on todettu jo vuoden 2015 kyselyn tuloksissa (OAJ 2016, 15–18). Opettajien mielestä digitaalisten oppimisympäristöjen ja kaupallisten sekä avoimien opetusma-

teriaalien käyttämisen ohjausta ja koulutusta saadaan liian vähän. Nämä materiaalit koettiin hankalimmiksi ohjelmiksi käyttää opetuksessa. OAJ peräänkuuluttaakin sitä, että opettajalla tulee opetuksessaan olla pedagoginen vapaus valita käytettävä oppimateriaali (OAJ 2019, 11). Tutkimukseen osallistuneiden opettajien haastattelun perusteella voidaankin sanoa, että ohjelmoinnin opettamiseen on tarjolla monia avoimia oppimateriaaleja ja opetusympäristöjä, joten siltä kannalta pedagoginen vapaus on opettajille turvattu.

Vuoden 2019 Valtioneuvoston Digiajan peruskoulu -raportissa todettiin, että oppilailla oli tehdyssä ICT-taitotestissä *"suuria vaikeuksia suoriutua tehtävistä, jotka liittyivät – – ohjelmointiin"* (Tanhua-Piiroinen ym. 2019, 24). Tämän tutkimuksen, mutta myös Valtioneuvoston raportin tulosten valossa on syytä olettaa, että ne muutamat oppilaat, jotka suoriutuivat testin vaikeimmista tehtävistä, olivat harrastuneita tietoteknisesti jo entuudestaan ja tehneensä ohjelmoinnin saralla esimerkiksi animaatioita tai pelejä. Esimerkiksi keskisuomalaisessa koulussa yksi oppilaista kertoi koetilanteen jälkeen, että hän oli aiemmin tehnyt ohjelmia aiemminkin Scratchilla ja olleensa 5. luokalla valinnaisen aineen ohjelmointikerhossa. Pirkanmaalaisen koulun opettaja kertoi erään luokan oppilaan olleen innokas tekemään historian oppitunnin opetettavasta aiheesta animaation Scratchilla. Pelkkä innostus opetettavaan aiheeseen ei kuitenkaan riitä kehittämään oppilaan tietoteknisiä taitoja, vaan oppilaat tarvitsevat ohjausta oppimiseen (Kuuskorpi ja Kuuskorpi 2016, 50). Opettajan rooli esimerkiksi ohjelmoinnin opetuksessa onkin suuri: opettajan tietoiset ja tiedostamattomat opetuskäytänteet vaikuttavat suuresti oppilaan oppimiseen (Norrena 2013, 163). Mannila ym. (2014, 25) muistuttavat, että ensisijaisesti opettajien tulee tuntea ohjelmoinnin opetus hyödylliseksi heille ja oppilailleen. Tähän heidän mukaansa auttaa merkittävästi opettajien mahdollisuus kouluttautua korkealaatuisella aineistolla.

Laitemäärät ja niiden toimivuus vaikuttavat opettajan ja oppilaan innostukseen TVT:n opetuksessa. Pohjoissavolaisen koulun opettaja kertoi, että koulun laitekanta on riittävä, mutta ajan myötä laitekannan päivittäminen on jäänyt, joka puolestaan tuo ongelmia oppitunneille, kun laitteet eivät toimi yhtenäisesti ja halutulla tavalla. Tämä turhauttaa hänen mukaansa oppilaita. Vuoden 2016 julkaisussaan OAJ (2016, 24) kertoo, että keskimäärin peruskoulun oppilaitoksissa on viisi oppilasta yhtä laitetta kohden ja yli 60 % kyselyyn vastanneista peruskoulun opettajista oli sitä mieltä, että laitteita on liian vähän opetuskäyttöön. OECD (2015,

133) kertoo viimeisimmässä raportissaan Suomessa olevan laitetta kohden noin kolme oppilasta. Tutkimukseen osallistuneet opettajat kertoivat oppilas/laite-suhteen olevan kolmesta yhdeksään oppilasta yhtä laitetta kohden. Laitteiston lyhyt käyttöikä on ollut huolena myös aiemmissa tutkimuksissa, kun opettajia on haastateltu ohjelmoinnin opettamisen haasteista (Makkonen ja Pyykönen 2018, 67). Ratkaisu tähän voisi olla resurssien optimointi ja opettajien sekä kouluyhteisön kouluttaminen teknologian käyttöön: Norrena (2013) mainitsee väitöksessään, että koulun laitteiston määrällä ei ollut merkitystä teknologian laadukkaaseen käyttöön, vaan kouluyhteisön tuki tietotekniikan käyttötavoissa oli ratkaisevaa. Norrenan johtopäätöstä mukailee pirkanmaalaisen koulun teknologian tilanne: yhtä laitetta kohden oli jopa yhdeksän oppilasta, mutta tämä laitteiden vähyys ei näkynyt oppilaiden alisuoriutumisena testissä, vaan todennäköisesti laitteiden käyttämisen optimointi on onnistunut ja oppilaiden opetuksessa on suosittu myös laitteetonta ohjelmointia.

6.3 Tutkimuksen luotettavuus ja rajoitukset

Tutkimuksen luotettavuudella tarkoitetaan sitä, että vastaavatko tulokset siihen, mitä tutkimuksessa ollaan pyritty tutkimaan (Eskola 2018), sillä tutkimusta tehdessä tutkimukseen voi liittyä erilaisia haasteita tai rajoitteita. Tutkimuksen luotettavuutta tarkastellaan reliabiliteetin ja validiteetin kautta. Reliabiliteetti tarkoittaa tutkimuksen kykyä antaa ei-sattumanvaraisia tuloksia ja toistettavuutta. Täten se on tutkimuksen luotettavuuden mittari (L. Hiltunen 2009; Valli 2015).

Tutkimuksen reliaabelius on kohtuullinen, sillä tutkimukseen otanta käsitti kolme eri koulua eri puolilta Suomea: Pohjois-Savosta, Keski-Suomesta ja Pirkanmaalta. Lopulliseen tutkimukseen osallistui 54 oppilasta ja kolme opettajaa. Täten tuloksia ei voida yleistää koko maata kattavaksi, eikä se ollut myöskään tutkimuksen tarkoituksena. Kuitenkin voidaan todeta, että otannan laajuus kuvaa tyypillisiä luokkia ympäri Suomen, sillä työskentelyrauhan vaihtelevuutta on suomalaisissa luokissa. Tulokset kielivät myös siitä, että ohjelmoinnin opetusta on kouluissa tapahtunut ja oppilaat osaavat hieman ohjelmointia graafisissa ohjelmointiympäristöissä. Pohjoissavolaisen luokan koetilanne epäonnistui työskentelyrauhan puuttumiseen, joka johti puolestaan siihen, että vastauksia saatiin vain kolmasosa siitä, mitä olisi ollut mahdollista saada kaiken kaikkiaan. Koetilanteen epäonnistuminen ja vastausten

vähyys täten johtaa tutkimuksen reliaabeliuden huononemiseen, mutta otannan suuruus on silti arvioitu tarpeeksi isoksi, jotta tutkimuksen reliabiliteetti ei kärsi täysin tämänkaltaisten tilanteiden osalta.

Opettajat ja luokat valikoituivat tutkimukseen opettajan suostumuksen myötä, eikä esimerkiksi koulun rehtorin määräyksellä. Tuloksia ja tutkimuksen luotettavuutta tarkastellessa tulee tällöin muistaa, että opettajat olivat todennäköisesti opettajia, jotka kokivat ohjelmoinnin oppiaiheena erityisen positiivisena. Tämä positiivinen suhtautuminen ohjelmointiin ja sen opettamiseen näkyi myös opettajien haastatteluissa.

Validiteetti kertoo siitä, miten hyvin valittu tutkimusmenetelmä sopii tutkimaan tai mittaamaan tutkittavaa aihetta tai asiaa. L. Hiltunen (2009) korostaa, että pelkän tutkimusmenetelmän ansiosta tutkimus ei tuota tietoa, vaan tutkimusmenetelmä on valittava sen perusteella, millaista tietoa tutkimuksessa halutaan. Tärkeää on tutkimuksen validiteettiä arvioidessa arvioida tutkimuksen aineiston riittävyyttä, jotta aineistosta voidaan tehdä toivottuja johtopäätöksiä.

Päätutkimuskysymykseen, eli oppilaiden taitotasoon ohjelmoinnissa, olisi ehkä päästy vielä paremmin pureutumaan, mikäli opettajan haastattelussa olisi selvitetty myös enemmän sitä, että minkälaisia tehtäviä ohjelmoinnissa oppilaat ovat tehneet. Nyt haastattelussa selvitettiin pelkästään oppilaiden käyttämiä ohjelmointiympäristöjä, jolloin esimerkiksi Scratchin käyttäminen on voinut olla hyvinkin erilaista eri luokkien välillä opettajasta riippuen. Kaksi opettajaa mainitsi käyttämäkseen ohjelmointiympäristöksi Koodaustunnin, jossa oppilas tehtävissä edetessä tekee vaikeampia tehtäviä ja harjoittelee eri konsepteja, kuten silmukoita, tapahtumankäsittelijöitä ja funktioita. Keskisuomalaisen ja pirkanmaalaisen koulun luokanopettajat arvioivat oppilaidensa osaavan tehdä ”toimivan ohjelman graafisessa ohjelmointiympäristössä”.

Tehtävien analysointi

Tehtäviä laadittaessa oli tärkeää tehdä ensimmäisestä tehtävästä sellainen, että se mahdollisti oppilaan heikonkin suorituksen. Tehtävät pyrittiin tekemään siten, että jokaisesta aihealueesta olisi kolme tehtävää: ensimmäinen tehtävä olisi helppo, toinen keskitasoinen ja kolmas

olisi edistyneemmän oppijan suoritettavissa. Tutkimuksen yksi valinta oli se, mitä eri ohjelmoinnin konsepteja empiirisessä osuudessa tutkittiin: nyt aihealueina olivat toistorakenteet, muuttujat ja ehtolauseet. Muita testattavia aihealueita olisi voinut olla esimerkiksi loogiset operaatiot, debuggaus, taulukot, aliohjelmat tai rekursiot, mikäli vastaajien taustataidot olisivat sen mahdollistaneet.

Tehtävät eivät välttämättä mitanneet puhtaasti ainoastaan sitä aihealuetta, jonka otsikon alle se oli laitettu, vaan esimerkiksi ehtolausetehtävässä tehtävän suorittamiseksi oppilas joutui käyttämään ratkaisussaan muuttujia. Tehtävät pyrittiin kuitenkin tekemään siten, että useamman aihealueen integroivat tehtävät olivat vasta vaikeimpia tehtäviä, eli helpoin taso oli mahdollista saavuttaa pelkästään kyseisen konseptin osaamisella.

Koekysymyksien asetteluun olisi tullut kiinnittää vielä enemmän tarkkuutta ennen empiirisen osuuden toteuttamista. Nyt tehtävänannot olivat tulkittavissa toisin, esimerkiksi toisto1-tehtävässä, jossa oppilaan tuli toteuttaa ohjelma yhdellä liiku-komennolla ja saada kissa liikkumaan neliön muotoinen liike viisi kertaa. Lopputuloksena oli helposti ratkaisu, jossa yhteen silmukkaan laitetaan toistomääräksi ”toista 20 kertaa”, sen sijaan, että vastauksessa olisi käytetty kahta sisäikkäistä silmukkaa, jossa ulommaisessa toistomäärä on viisi ja sisemmässä neljä.

Muuttujat3-tehtävästä on tärkeää tehdä huomio negatiivisten kokonaislukujen opettamisesta alakoulussa. Opettajasta riippuen negatiivisten lukujen esittely on voinut jäädä pelkästään esimerkiksi lämpömittaria tarkastellessa pakkasasteiden havannointiin, jolloin tehtävässä haettu ratkaisu muuttujan arvon vähentäminen laskuoperaatiolla ”`lisää <-1>`” ei ole oppilaille vielä tuttu tai ainakaan rutinoitunut laskutoimenpide. Opetussuunnitelma ohjeistaa matematiikan opetukseen vuosiluokilla 3-6 että oppilaille *”pohjustetaan negatiivisen luvun käsite ja laajennetaan lukualuetta negatiivisilla kokonaisluvuilla”* (Opetushallitus 2014, 236). Varmuutta negatiivisten kokonaislukujen käsittelyyn tulee vasta yläkoulun puolella (Opetushallitus 2014, 375). Kyseisen tehtävän kohdalla keskisuomalaisessa ja pirkanmaalaisessa koulussa pari oppilasta teki koetilanteessa kyseistä tehtävää, mutta tehtävän ratkeaminen rajoittui tähän ongelmaan.

Ehtolauseiden osaamista mittaavien kahden ensimmäisen tehtävän välinen ero oli kovin pie-

ni: tehtävien välillä kissa tuli joko pysäyttää tai laittaa muuttamaan suuntaansa halutun asteluvun verran hahmon osuessa seinään. Kuitenkin näiden tehtävien välillä oli palautusmäärissä eroja, joten todennäköisesti asteluvut ja niiden käyttäminen eivät olleet oppilaille tuttuja. Ehtolause2-tehtävän olisi voinut kuitenkin saada vielä paremmin oppilaan ohjelmointiosaamista mittaavaksi, eikä uuden matematiikan konseptia opettavaksi.

Tehtävistä olisi voinut tehdä vielä houkuttelevampia oppilaiden innokkuuden ja vastausprosentin kasvatamisesksi. Tutkimuksessa käytetyt tehtävät olivat yhtä tehtävää lukuunottamatta pelkästään valkoisella, tyhjällä pohjalla kissan liikuttamista, jolloin muutamat oppilaat keskittyivät helposti taustan tai kissan koristelemiseen. Toisaalta voidaan pohtia, viekö houkuttelevampi näyttämöympäristö liikaa aikaa sisäistää tehtävänanto, kun oppilaalle tarjoutuu helpompi ympäristö luoda vielä enemmän lisää.

6.4 Pohdinta

Tutkimuksen tulokset ovat merkittäviä niiden tuodessa konkreettisesti ilmi oppilaiden osaamista ohjelmoinnista kansalliseen standardiin peilaten. Tämänkaltaista tutkimusta ei ole tutkijan tiedon mukaan tehty, vaikka tarvetta ohjelmoinnin osaamisen mittaamiselle ja opetuksen kehittämiselle onkin ollut ohjelmoinnin oltua kolme kokonaista lukuvuotta virallisesti opetussuunnitelmassa mukana. Oppilaiden heikko osaaminen ohjelmoinnissa tässä tutkimuksessa ja esimerkiksi Valtioneuvoston (Tanhua-Piiroinen ym. 2019) raportissa kielii todennäköisesti joko siitä, että opettajilla ei ole tarpeeksi osaamista opettaa ohjelmointia ja he tarvitsevat tukea ohjelmoinnin opettamiseen tai ajallisesti ohjelmoinnin opettamiseen ei panosteta riittävästi.

Opettajien pedagogisen tuen lisäksi on tärkeää kiinnittää huomiota koulujen laitteiston määrään ja niiden päivitettävyyteen. Jotta ohjelmointia voi oppia sillä tasolla, että siirtyminen tekstuaaliseen ohjelmointiin on helppoa, tarvitaan graafisten ohjelmointiympäristöjen käyttämistä ohjeiden antamisen harjoittelemiseksi. Tämä taas edellyttää kouluilta tarpeeksi hyvää ja laajaa laitekantaa sekä tarpeeksi aikaresursseja tietokoneiden käyttämiseksi, mikä sinällään on ilmiselvää.

Todennäköisesti Suomen kouluista löytyy vielä monia opettajia, jotka ovat teknologiaa ja

uudistuvaa koulua vastaan ohjelmoinnin opetuksen ja yleisen digitalisaation tulemisen myötä. Norrena (2013, 169) toteaa, että opettajankoulutuksen hienouksia onkin se, että opettajilla on vapaus toteuttaa opetuksen tavoitteita, mutta yksintekemisen kulttuuri on taakka koulun muutokselle ja kehittymiselle. Aiemmissa julkaisuissa todetut opettajien heikot koulutusmahdollisuudet (esim. OAJ 2019) voitaisiin tasoittaa esimerkiksi yhteis- tai tiimiopettajuudella: esimerkiksi Oulun ammattikorkeakoulussa tiimiopettajuus, jossa opettajat osallistuvat opetuksen suunnittelusta aina arviointiin asti, on todettu toimivaksi malliksi (Erkkilä ja Perunka 2016). Tällöin opettajat oppivat oppivat toisiltaan tietoja ja taitoja ja saavuttavat enemmän verrattuna siihen, kuin yksin toimiessaan. Tiimiopettajuus nähdään myös ammattillisen taitojen kehitysmahdollisuutena (Erkkilä ja Perunka 2016). Yhteis- tai tiimiopettajuus ei kuitenkaan ole oikotie onneen ohjelmoinninkaan opetuksessa, vaan koulutusta yhteisopettajuuteen ja sähköisten oppimateriaalien käyttämiseen on toivottu. Lisäksi yhteisopettajuus mahdollistaa monipuolisemman oppimisympäristön, joka ottaa huomioon erilaiset oppijat. (Soukkala 2016, 49–50.) Barr ja Stephenson (2011, 53) esittävät artikkelissaan erilaisia keinoja opettajien kannustamiseksi muuttamaan opetustaan muuttuvan koulun myötä. Heidän mukaansa opettajille tulee tarjota resursseja tukemaan muutosta esimerkiksi opettajien vertaisoppimisen kautta.

Kansallinen OPS linjaa varsin leveästi, mitä ohjelmoinnin opetuksen tavoitteet ja oppilaan osaamistavotteet ovat kuudennen luokan päätteeksi: *”Oppilas osaa ohjelmoida toimivan ohjelman graafisessa ohjelmointiympäristössä”* (Opetushallitus 2014, 239). Tämä on luokanopettajasta riippuen sisällöiltään hyvinkin erilainen tavoite: tutkimukseen osallistuvat opettajat eivät haastattelussa osanneet sanoa tarkkaa sisältöä sille, mitä konsepteja tai toiminnallisuutta heidän mukaansa ”toimivassa ohjelmassa” tulisi olla. Kansalliseen OPS:aan voitaisiin linjata ohjelmoinnin saralta hieman tarkemmin kuudennen luokan päätteeksi tarvittavat taidot arvosanalle kahdeksan – esimerkiksi geometrian osaamistavoitteissa mainitaan, että *”oppilas osaa käyttää mittakaavaa sekä tunnistaa suoran ja pisteen suhteen symmetrisiä kuvioita”* (Opetushallitus 2014, 238). Kuten alkupuolella tätä tutkimusta todettiin, on esimerkiksi Kuopion ja Jyväskylän paikallisessa opetussuunnitelmassa määritelty, että oppilas käyttää muuttujia, ehtolauseita ja toistorakenteita ohjelmoidessaan (Jyväskylän kaupunki 2016; Kuopion kaupunki 2019). Osaamistavoitteiden tarkempi määrittely valtakunnallisesti olisi tarpeen, jotta oppilaiden edellytykset siirtyä tekstuaaliseen ohjelmointiin yläkoulun

puolelle olisivat paremmat.

6.5 Jatkotutkimus

Tutkimuksen tekeminen oli antoisaa ja herätti monia ideoita jatkotutkimusaiheiksi. Jotta tämä tutkimus voisi antaa tarkemman analyysin kuudesluokkalaisten taitotasoista ohjelmoinnista, se tulisi vähintäänkin uusia samanlaisella tutkimusasetelmalla, mutta testin laajentaminen myös muita ohjelmoinnin konsepteja tarkastelevaksi olisi tarpeen. Tällöin tulisi huomioida muun muassa testin keston pidentäminen ja parempi orientoituminen, jotta tutkimukseen osallistujat ehtisivät paneutumaan vastauksiinsa huolellisemmin.

Yksi tutkimusasetelma voisi tarkastella myös pari- tai ryhmätyöskentelyä ohjelmoinnissa: aiempi tutkimus on osoittanut, että pariohjelmointia tehneet olivat menestyneet paremmin (Werner, Denner ja Campe 2012, 218). Tällöin tutkimusasetelma voitaisiin toteuttaa esimerkiksi pistetyöskentelynä, jossa parit tai ryhmät siirtyisivät eri konseptia mittaavien pisteiden välillä: käytettävä työskentelyaika tutkimuksessa olisi pidempi, mutta vastausten hiominen olisi ryhmätyöskentelyn myötä tarkempaa ja siirtymät eri konseptia mittaavien tehtävien välillä olisivat selkeämpiä tutkimukseen osallistuville.

Jatkotutkimuksena oppilaiden ohjelmoinnin taitotasoa voitaisiin tarkastella suuremmalla otannalla ympäri Suomen ja pureutua tarkemmin kunnallisten opetussuunnitelmien vaikutuksiin opetuksen sisältöä analysoidessa. Tämä antaisi todennäköisesti kattavamman ja yleistettävemmän kuvan kuudesluokkalaisten alakoulun ohjelmoinnin opetuksen tuloksista.

Jatkotutkimusta voitaisiin myös tehdä yläkoulun puolella samasta aiheesta tutkimalla yhdeksäsluokkalaisten taitotasoja ohjelmoinnissa. Tähän tutkimukseen voitaisiin ottaa näkökulmaksi erilaisten tekstuaalisten ohjelmointikielten käyttö ja niiden vertailu opetuskäytössä.

Opetussuunnitelmassa ohjelmointi on liitetty integroiden muihin oppiaineisiin. Eräs jatkotutkimuksen aihe voisi olla, että saavutetaanko parempi taitotaso ohjelmoinnissa lisäämällä ohjelmointia muihinkin oppiaineisiin kuin matematiikkaan. Tässä tutkimuksessa olisi myös mielenkiintoista seurata, saavutetaanko ohjelmoinnin avulla parempia oppimistuloksia ope-

tettavasta aiheesta.

Lähteet

Adleberg, B. M. 2013. “Scratch Programming and Remix Culture: Gender Differences In Interaction and Motivation For Pre-Adolescents”. Pro gradu -tutkielma.

Alanen, R. 2000. “Vygotsky, Van Lier ja kielenoppiminen: sosiokulttuurinen viitekehys kielellisen tietoisuuden ja vieraan kielen oppimisen tutkimuksessa”. Toimittanut P. Kalaja ja L. Nieminen. *Kielikoulussa – kieli koulussa. AFinLAn vuosikirja 2000. Suomen soveltavan kielitieteen yhdistyksen julkaisuja,* numero 58: 97–120.

Anderson, R., M. Ernst, R. Ordóñez, P. Pham ja S. Wolfman. 2014. “Introductory programming meets the real world: using real problems and data in CS1”. Teoksessa *Proceedings of the 45th ACM technical symposium on Computer science education,* 465–466. ACM.

Atmatzidou, S., ja S. Demetriadis. 2015. “Advancing students’ computational thinking skills through educational robotics: A study on age and gender relevant differences”. *Robotics and Autonomous Systems* 75:661–670.

Barr, V., ja C. Stephenson. 2011. “Bringing Computational Thinking to K-12: What is Involved and What is the Role of the Computer Science Education Community?” *ACM Inroads* 2 (1): 48–54.

Ben-Ari, M. 1998. “Constructivism in computer science education”. Teoksessa *ACM Sigcse Bulletin,* 30:257–261. 1. ACM.

Bers, M. U., L. Flannery, E. R. Kazakoff ja A. Sullivan. 2014. “Computational thinking and tinkering: Exploration of an early childhood robotics curriculum”. *Computers & Education* 72:145–157.

Binkley, Marilyn, Ola Erstad, Joan Herman, Senta Raizen, Martin Ripley, May Miller-Ricci ja Mike Rumble. 2012. “Defining twenty-first century skills”. Teoksessa *Assessment and teaching of 21st century skills,* 17–66. Springer.

Boyer, N., S. Langevin-Gaspar ja A. Gaspar. 2008. “Self direction & constructivism in programming education”. Teoksessa *Proceedings of the 9th Conference on Information Technology Education, SIGITE 2008,* 89–94. doi:10.1145/1414558.1414585.

Brennan, K., ja M. Resnick. 2012. “New frameworks for studying and assessing the development of computational thinking”. Teoksessa *Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada,* 1:25.

Chipman, H., F. Rodríguez ja K. Boyer. 2019. “"I Impressed Myself With How Confident I Felt": Reflections on a Computer Science Assessment for K-8 Teachers”. Teoksessa *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE ’19),* To appear. http://cise.ufl.edu/research/learndialogue/pdf/LearnDialogue-Chipman-SIGCSE-2019.pdf.

Deci, E., ja R. Ryan. 2000. “The "what"and "why"of goal pursuits: Human needs and the self-determination of behavior”. *Psychological inquiry* 11 (4): 227–268.

Erkkilä, R., ja S. Perunka. 2016. “Näkökulma tiimiopettajuuteen”. Teoksessa *ePooki. Oulun ammattikorkeakoulun tutkimus- ja kehitystyön julkaisut.* 28. Viitattu 15. huhtikuuta 2019. http://urn.fi/URN:ISSN:1798-2022.

Eskola, J. 2018. “Laadullisen tutkimuksen juhannustaiat. Laadullisen aineiston analyysi vaihe vaiheelta”. Teoksessa *Ikkunoita tutkimusmetodeihin 2 - Näkökulmia aloittelevalle tutkijalle tutkimuksen teoreettisiin lähtökohtiin ja analyysimenetelmiin,* toimittanut R. Valli. PS-kustannus.

EU-komissio. 2018. “Coding - the 21st century skill”. Viitattu 20. marraskuuta. https://ec.europa.eu/digital-single-market/coding-21st-century-skill.

European Schoolnet. 2018. “Computing our future. Computer programming and coding. Priorities, school curricula and initiatives across Europe”. Viitattu 20. marraskuuta. http://fcl.eun.org/documents/10180/14689/Computing+our+future_final.pdf/746e36b1-e1a6-4bf1-8105-ea27c0d2bbe0.

Fessakis, G., E. Gouli ja E. Mavroudi. 2013. “Problem Solving by 5-6 Years Old Kindergarten Children in a Computer Programming Environment: A Case Study”. *Computers & Education* 63:87–97.

Flórez, F. B., R. Casallas, M. Hernández, A. Reyes, S. Restrepo ja G. Danies. 2017. “Changing a Generation’s Way of Thinking: Teaching Computational Thinking Through Programming”. *Review of Educational Research* 87 (4): 834–860. DOI:%2010.3102/0034654317710096.

Golin, E., ja S. Reiss. 1990. “The specification of visual language syntax”. *Journal of Visual Languages & Computing* 1 (2): 141–157.

Hakkarainen, K., L. Ilomäki, L. Lipponen, H. Muukkonen, M. Rahikainen, T. Tuominen, M. Lakkala ja E. Lehtinen. 2000. “Students’ skills and practices of using ICT: Results of a national assessment in Finland”. *Computers & Education* 34 (2): 103–117.

Hanhijoki, I., J. Katajisto, M. Kimari ja H. Savioja. 2009. “Koulutus ja työvoiman kysyntä 2020. Tulevaisuuden työpaikat – osaajia tarvitaan”. *Helsinki: Opetushallitus.*

Harju, J. 2015. “Ohjelmoinnin opetuspeli yläkouluikäisille”. Pro gradu -tutkielma, University of Oulu.

Harris, C. 2018. “Computational Thinking Unplugged: Comparing the Impact on Confidence and Competence from Analog and Digital Resources in Computer Science Professional Development for Elementary Teachers”. Paper 374. Tohtorinväitöskirja, St. John Fisher College. https://fisherpub.sjfc.edu/education_etd/374.

Heintz, F., L. Mannila ja T. Färnqvist. 2016. “A review of models for introducing computational thinking, computer science and computing in K-12 education”. Teoksessa *2016 IEEE Frontiers in Education conference (FIE),* 1–9. IEEE.

Hiltunen, L. 2009. *Validiteetti ja reliabiliteetti.* Graduryhmä 18.2.2009. Jyväskylän yliopisto. Viitattu 8. huhtikuuta 2019. http://www.mit.jyu.fi/ope/kurssit/Graduryhma/PDFt/validius_ja_reliabiliteetti.pdf.

Hiltunen, T. 2016. “Learning and Teaching Programming Skills in Finnish Primary Schools – The Potential of Games”. Pro gradu -tutkielma, University of Oulu.

Hirsjärvi, S., ja H. Hurme. 2008. *Tutkimushaastattelu.* Gaudeamus Helsinki University Press.

Huitt, W., ja J. Hummel. 2003. “Piaget’s theory of cognitive development”. *Educational psychology interactive* 3 (2): 1–5.

Hyvönen, M., V. Lappalainen ja A.-J. Lakanen. 2013. “Ohjelmointi 1: C”. *Luentomoniste/Jyväskylän yliopisto, tietotekniikan laitos,* numero 17.

Ipek, J., ja G. Turgut. 2018. “Coding With Scratch in Primary Education: A Case Study”. Teoksessa *Educational Policy and Research,* toimittanut H. Arslan, Dorczak R. ja D. Alina-Andreea, 179–189.

Jyväskylän kaupunki. 2016. *Jyväskylän perusopetuksen opetussuunnitelma. Tieto- ja viestintäteknologia.* Viitattu 11. tammikuuta 2019. `https://peda.net/opetussuunnitelma/ksops/jyvaskyla`.

Järvinen, E.-M. 1998. “The Lego/Logo Learning Environment in Technology Education: An Experiment in a Finnish Context”. *Journal of Technology Education* 9 (2): 47–59.

Kaarakainen, M.-T., S.-S. Kaarakainen, E. Tanhua-Piironen, J. Viteli, A. Syvänen ja A. Kivinen. 2017. *Digiajan peruskoulu 2017 - Tilannearvio ja toimenpidesuositukset.* Nide 72. Valtioneuvoston selvitys- ja tutkimustoiminnan julkaisusarja. Valtioneuvoston kanslia. ISBN: 978-952-287-478-8.

Kalelioğlu, F. 2015. “A new way of teaching programming skills to K-12 students: Code.org”. *Computers in Human Behavior* 52:200–210.

Kalelioğlu, F., ja Y. Gülbahar. 2014. “The Effects of Teaching Programming via Scratch on Problem Solving Skills: A Discussion from Learners’ Perspective”. *Informatics in Education* 13 (1): 33–50.

Karvonen, V.-P., ja P. Laukka. 2016. “Suomalaisten opettajien asenteita ja valmiuksia ohjelmoinnin opetukseen”. Pro gradu -tutkielma, Oulun yliopisto.

Kekäläinen, O. 2015. “Onko automatisointiajattelu paras suomennos käsitteestä ”computational thinking”?” Teoksessa *Tuovi 13: Interaktiivinen tekniikka koulutuksessa 2015-konferenssin tutkijatapaamisen artikkelit,* toimittanut Jarmo Viteli ja Anneli Östman, 27–29.

Kelleher, C., ja R. Pausch. 2005. “Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers”. *ACM Computing Surveys (CSUR)* 37 (2): 83–137.

Kenttälä, V., M. Kankaanranta ja P. Neittaanmäki. 2016. *Tieto- ja viestintätekniikka Keski-Suomen peruskouluissa vuonna 2016.* Jyväskylän yliopisto.

Kohn, T. 2017. “Variable Evaluation: an Exploration of Novice Programmers’ Understanding and Common Misconceptions”. Teoksessa *SIGCSE ’17. Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education,* 345–350.

Koodiaapinen. 2018. *Koodiaapinen: Koodaus opiskeluvälineenä.* Viitattu 4. joulukuuta. `http://koodiaapinen.fi/koodaus-opiskeluvalineena/`.

Koskinen, A., M. Kangas ja L. Krokfors. 2014. “Oppimispelien tutkimus pedagogisesta näkökulmasta”. Teoksessa *Oppiminen pelissä – pelit, pelillisyys ja leikillisyys opetuksessa,* toimittanut M. Kangas, K. Kopisto ja L. Krokfors. Osuuskunta Vastapaino.

Kuittinen, M., ja J. Sajaniemi. 2004. “Teaching Roles of Variables in Elementary Programming Courses”. *SIGCSE Bulletin* 36:57–61.

Kuopion kaupunki. 2019. *Tvt-tuki.* Viitattu 11. tammikuuta. `https://peda.net/kuopio/tvt-tuki`.

Kurkinen, H. 2018. “”Koodaaminen ei oo sitä tietokonenippeliä pelkästään”: Luokanopettajien käsityksiä ja kokemuksia ohjelmoinnin opettamisesta”. Pro gradu -tutkielma, Jyväskylän yliopisto.

Kuuskorpi, M., ja T. Kuuskorpi. 2016. “Oppimismotivaation muutokset perusopetuksen digitalisaatiohankkeen yhteydessä”. Teoksessa *Opetuksen digitalisaatio, uudet oppimisympäristöt ja uusi pedagogiikka,* toimittanut M. Kuuskorpi ja K. Sipilä, 28–55. Suomen Yliopistopaino Oy, Tampere.

Könönen, M., ja J. Ruotanen. 2018. “Opettajien ja oppilaiden kokemuksia ohjelmoinnista - Tapaustutkimus alakoulusta”. Pro gradu -tutkielma, Filosofinen tiedekunta. Itä-Suomen yliopisto.

Lahtinen, E., K. Ala-Mutka ja H.-M. Järvinen. 2005. “A study of the difficulties of novice programmers”. *Acm Sigcse Bulletin* 37 (3): 14–18.

Lakanen, A-J. 2010. “Nuorten peliohjelmointi”. Pro gradu -tutkielma, Jyväskylän yliopisto.

Lakanen, A-J., ja V. Isomöttönen. 2015. “What Does It Take to Do Computer Programming? Surveying the K-12 Students’ Conceptions”. Teoksessa *SIGCSE ’15. Proceedings of the 46th ACM Technical Symposium on Computer Science Education,* 458–463.

Lau, K., H. Tan, B. Erwin ja P. Petrovic. 1999. “Creative learning in school with LEGO (R) programmable robotics products”. Teoksessa *Frontiers in Education Conference, 1999. FIE’99. 29th Annual,* 2:26–31. IEEE.

Lavonen, J., V. Meisalo ja M. Lattu. 2001. “Problem Solving with an Icon Oriented Programming Tool : A Case Study in Technology Education”. *Journal of Technology Education* 12 (2).

Lipponen, L., A. Rajala ja J. Hilppö. 2014. “Kuka pelaa ja kenen säännöillä? Ajatuksia pelien pedagogisista seurauksista”. Teoksessa *Oppiminen pelissä – pelit, pelillisyys ja leikillisyys opetuksessa,* toimittanut M. Kangas, K. Kopisto ja L. Krokfors. Osuuskunta Vastapaino.

Makkonen, J., ja A. Pyykönen. 2018. “"Se on mun mielest taas yks tapa rikastuttaa sitä opiskelua ja oppimista, itekki oppii sit uusii juttui."– Alakoulun opettajien käsityksiä ohjelmoinnin opettamisesta”. Pro gradu -tutkielma, Jyväskylän yliopisto.

Malan, D. J., ja H. H. Leitner. 2007. “Scratch for Budding Computer Scientists”. *ACM SIGCSE Bulletin* 39 (1): 223–227.

Mannila, L., M. Peltomäki ja T. Salakoski. 2006. “What about a simple language? Analyzing the difficulties in learning to program”. *Computer Science Education* 16 (3): 211–227.

Mannila, L., A. Settle, V. Dagiene, B. Demo, N. Grgurina, C. Mirolo ja L. Rolandsson. 2014. “Computational Thinking in K-9 Education”, 1–29. Kesäkuu. doi:10.1145/2713609.2713610.

Marttala, L. 2017. “Tietotekniikan valtakunnallisten oppisisältöjen toteutuminen Keski-Suomen peruskoulujen opetuskäytänteissä”. Pro gradu -tutkielma, Jyväskylän yliopisto.

Mataric, M. J., N. P. Koenig ja D. Feil-Seifer. 2007. “Materials for Enabling Hands-On Robotics and STEM Education.” Teoksessa *AAAI spring symposium: Semantic scientific knowledge integration,* 99–102.

McDowell, C., L. Werner, H. Bullock ja J. Fernald. 2002. “The effects of pair-programming on performance in an introductory programming course”. Teoksessa *SIGCSE ’02 Proceedings of the 33rd SIGCSE technical symposium on Computer science education,* 38–42.

Meerbaum-Salant, O., M. Armoni ja M. Ben-Ari. 2013. “Learning Computer Science Concepts with Scratch”. *Computer Science Education* 23 (3): 239–264. doi:https://doi.org/10.1145/1839594.1839607.

Murphy, E. 1997. “Constructivist epistemology”. *Constructivism: From Philosophy to Practice.*

Myers, B. 1990. “Taxonomies of visual programming and program visualization”. *Journal of Visual Languages & Computing* 1 (1): 97–123. ISSN: 1045-926X. doi:https://doi.org/10.1016/S1045-926X(05)80036-9.

Nance, S. 2016. “Using computer programming to enhance problem-solving skills of fifth grade students”. Tohtorinväitöskirja, University of Florida.

Neuman, A. 2017. “Motivoiko ohjelmointi? Kuudennen luokan oppilaiden ja opettajan käsityksiä ohjelmoinnin opetuksesta ja oppimisesta”. Pro gradu -tutkielma, Käyttäytymistieteellinen tiedekunta. Helsingin yliopisto.

Nikkola, T., M. Rautiainen, P. Moilanen, P. Räihä ja P. Löppönen. 2013. “Kielen prosessit oppiaineintegraation perustana”, toimittanut M. Rautiainen, T. Nikkoa ja P. Räihä. Osuuskunta Vastapaino.

Niukkanen, T-P. 2014. “Oppilaat pelien ohjelmoijina. Kodu Game Lab koulussa”. Teoksessa *Oppiminen pelissä – pelit, pelillisyys ja leikillisyys opetuksessa,* toimittanut M. Kangas, K. Kopisto ja L. Krokfors. Osuuskunta Vastapaino.

Norrena, J. 2013. “Opettaja tulevaisuuden taitojen edistäjänä: ”jos haluat opettaa noita taitoja, sinun on ensin hallittava ne itse””. *Jyväskylä studies in computing,* numero 169.

OAJ, Opetusalan Ammattijärjestö. 2016. *Askelmerkit digiloikkaan,* tammikuu. Viitattu 7. huhtikuuta 2019. https://www.oaj.fi/globalassets/julkaisut/2016/askelmerkitdigiloikkaan.pdf.

OAJ, Opetusalan Ammattijärjestö. 2019. *Toimivaa digitalisaatiota!,* maaliskuu. Viitattu 21. maaliskuuta 2019. https://www.oaj.fi/globalassets/julkaisut/2019/toimivaa_digitalisaatiota_3_2019.pdf.

OECD. 2015. *Students, computers and learning: Making the connection.* PISA. OECD Publishing. https://doi.org/10.1787/9789264239555-en.

Opetus- ja kulttuuriministeriö. 2014. *Kiuru: Ohjelmointi peruskoulun opetussuunnitelman perusteisiin.* Viitattu 4. joulukuuta 2018. https://minedu.fi/artikkeli/-/asset_publisher/kiuru-ohjelmointi-peruskoulun-opetussuunnitelman-perusteisiin.

Opetushallitus. 2014. *Perusopetuksen opetussuunnitelman perusteet.* Määräykset ja ohjeet 2014:96. ISBN 978-952-13-5998-9.

Ouahbia, I., F. Kaddaria, H. Darhmaouib, A. Elachqara ja S. Lahminea. 2015. "Learning Basic Programming Concepts by Creating Games with Scratch Programming Environment". *Procedia - Social and Behavioral Sciences* 191 (2): 1479–1482. doi:https://doi.org/10.1016/j.sbspro.2015.04.224.

Pampel, Z. 2017. "Getting a Child's Perspective on Learning Apps That Teach Programming". Tutkielma, University of Oulu.

Pane, J., ja B. Myers. 1996. *Usability issues in the design of novice programming systems.* Tekninen raportti. Carnegie-Mellon University, Pittsburgh, PA. Department Of Computer Science.

Papert, S. 1980. *Mindstorms. Children, computers and powerful ideas.* New York: Basic Books.

Piteira, M., ja C. Costa. 2013. "Learning computer programming: study of difficulties in learning programming". Teoksessa *Proceedings of the 2013 International Conference on Information Systems and Design of Communication,* 75–80. ACM.

Rasku-Puttonen, H., A. Eteläpelto, P. Häkkinen ja M. Arvaja. 2002. "Teachers' instructional scaffolding in an innovative information and communication technology-based history learning environment". *Teacher Development* 6 (2): 269–287.

Resnick, M. 2012. “Mother’s Day, Warrior Cats, and Digital Fluency: Stories from the Scratch Online Community”. Teoksessa *Proceedings of the Constructionism 2012 conference.*

Saaranen-Kauppinen, A., ja A. Puusniekka. 2019. “KvaliMOTV - menetelmä-opetuksen tietovaranto”. Viitattu 29. tammikuuta. `https://www.fsd.uta.fi/fi/tietoarkisto/julkaisut/kvalimotv.pdf`.

Saeli, M., J. Perrenet, W. Jochems ja B. Zwaneveld. 2011. “Teaching Programming in Secondary School: A Pedagogical Content Knowledge Perspective”. *Informatics in Education* 10:73–88.

Şahin-Kizil, A. 2011. “EFL teachers attitudes towards information and communication technologies (ICT)”. Teoksessa *Proceedings of the 5th International Computer & Instructional Technologies Symposium, Firat University, Laziğ Turkey.*

Scratch. 2019. “About Scratch”. Viitattu 24. tammikuuta 2019. `https://scratch.mit.edu/about`.

Serafini, G. 2011. “Teaching Programming at Primary Schools: Visions, Experiences, and Long-Term Research Prospects”. Teoksessa *Informatics in Schools. Contributing to 21st Century Education. 5th International Conference on Informatics in Schools: Situation, Evolution and Perspectives, ISSEP 2011 (Slovakia, October 26-29, 2011),* toimittanut I. Kalas ja R. T. Mittermeir, 143–154.

Sipitakiat, A., ja N. Nusen. 2012. “Robo-Blocks: designing debugging abilities in a tangible programming system for early primary school children”. Teoksessa *IDC ’12 Proceedings of the 11th International Conference on Interaction Design and Children,* 98–105.

Sola, H. 2002. “Ohjelmointia ilman koodausta”. Teoksessa *Lasten käyttöliittymät,* toimittanut P. Hietala ja S. Ovaska, 184–194. Tampereen yliopisto.

Soukkala, R. 2016. “Jaettu opettajuus: samanaikaisopettajuus opettajien kokemana”. Pro gradu -tutkielma, Jyväskylän yliopisto.

Tan, P.-H., C.-Y. Ting ja S.-W. Ling. 2009. “Learning Difficulties in Programming Courses: Undergraduates’ Perspective and Perception”. Teoksessa *Proceedings of the 2009 International Conference on Computer Technology and Development - Volume 01,* 42–46. ICCTD ’09. Washington, DC, USA: IEEE Computer Society.

Tanhua-Piiroinen, E., S.-S. Kaarakainen, M.-T. Kaarakainen, J. Viteli, A. Syvänen ja A. Kivinen. 2019. “Digiajan peruskoulu”, Valtioneuvoston selvitys- ja tutkimustoiminnan julkaisusarja. http://urn.fi/URN:ISBN:978-952-287-634-8.

Taub, R., M. Ben-Ari ja M. Armoni. 2009. “The Effect of CS Unplugged on Middle School Students’ Views of CS”. Teoksessa *ITiCSE ’09 Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education (Paris, France 06-09 July 2009),* 99–103.

Tiusanen, T. 2014. “Robotit ovat tulevaisuutta”. *Tekninen opettaja.*

Tu, J-J., ja J. R. Johnson. 1990. “Can Computer Programming Improve Problem-Solving Ability?” *SIGCSE BULLETIN* 22 (2): 30–37.

Tuomi, J., ja A. Sarajärvi. 2018. *Laadullinen tutkimus ja sisällönanalyysi.* Kustannusosakeyhtiö Tammi.

Valli, R. 2015. *Johdatus tilastolliseen tutkimukseen.* PS-kustannus.

Werner, L., J. Denner ja S. Campe. 2012. *The Fairy Performance Assessment: Measuring Computational Thinking in Middle School.* SIGCSE ‘12 Proceedings of the 43rd ACM Technical Symposium on Computer Science Education, Raleigh, NC. doi:10.1145/2157136.2157200.

Williams, L. A., ja R. R. Kessler. 2000. “All I Really Need to Know about Pair Programming I Learned In Kindergarten”. *COMMUNICATIONS OF THE ACM* 43 (5): 108–114.

Wing, J. 2006. *Computational Thinking.* http://www.cs.cmu.edu/~./15110-s13/Wing06-ct.pdf.

Winslow, L. E. 1996. “Programming Pedagogy - A Psychological Overview”. *SIGCSE BULLETIN* 28 (3): 17–25.

Yle. 2016. *Ohjailtava koulurobotti opettaa yläastelaisille teknologiaa ja vastuuta.* Viitattu 25. tammikuuta 2019. https://yle.fi/uutiset/3-8717861.

———. 2018. *Teknologia ja robotit tulevat osaksi peruskoulujen opetusta – "Se on uusi tapa opettaa asioita".* Viitattu 25. tammikuuta 2019. https://yle.fi/uutiset/3-10216063.

Liitteet

A Liite: Tutkimuslupa

JYVÄSKYLÄN YLIOPISTO

SUOSTUMUS TIETEELLISEEN TUTKIMUKSEEN

Lastani _____ on pyydetty osallistumaan tutkimukseen "6. luokkalaisten taitotasot ohjelmoinnissa alakoulun päättyessä".

Olen perehtynyt tutkimusta koskevaan tiedotteeseen (tietosuojailmoitus) ja saanut riittävästi tietoa tutkimuksesta ja sen toteuttamisesta. Tutkimuksen sisältö on kerrottu minulle kirjallisesti ja olen saanut riittävän vastauksen kaikkiin tutkimusta koskeviin kysymyksiini.

Ymmärrän, että tähän tutkimukseen osallistuminen on vapaaehtoista. Lapsellani on oikeus, milloin tahansa tutkimuksen aikana ja syytä ilmoittamatta, keskeyttää tutkimukseen osallistuminen tai peruuttaa suostumus tutkimukseen. Tutkimuksen keskeyttämisestä tai suostumuksen peruuttamisesta ei aiheudu lapselle kielteisiä seuraamuksia.

Olen tutustunut tietosuojailmoituksessa kerrottuihin rekisteröidyn oikeusiin ja rajoituksiin.

Allekirjoittamalla suostumuslomakkeen hyväksyn tietojeni käytön tietosuojailmoituksessa kuvattuun tutkimukseen.

Allekirjoituksellani vahvistan ja sallin, että lapseni saa osallistua tutkimukseen sekä annan luvan edellä kerrottuihin asioihin.

_____ _____

Allekirjoitus *Päiväys ja paikka*

_____

Nimen selvennys

Y-tunnus: 02458947
Sähköposti: etunimi.sukunimi@jyu.fi
Puhelin: (014) 260 1211
Faksi: (014) 260 1021
Jyväskylän yliopisto
PL 35
40014 Jyväskylän yliopisto
www.jyu.fi

B Liite: Tietosuojailmoitus

Tietosuoja-asetus (679/2016) 12-14, 30 artikla

TIETOSUOJAILMOITUS TUTKIMUKSESTA TUTKIMUKSEEN OSALLISTUVALLE

Gradut ja kandidaatintutkielmat

4.3.2018

Tutkimukseen osallistuminen on vapaaehtoista, eikä tutkittavan ole pakko toimittaa mitään tietoja, tutkimukseen osallistumisen voi keskeyttää.

1. Tutkimuksen nimi, luonne ja kesto

Tutkimuksen nimi: 6. luokkalaisten taitotasot ohjelmoinnissa alakoulun päättyessä

Tutkimus on kertatutkimus. Tutkimustulokset valmistuvat arviolta toukokuun 2019 loppuun mennessä.

2. Mihin henkilötietojen käsittely perustuu

EU:n yleinen tietosuoja-asetus, artikla 6, kohta 1

☒ **Tutkittavan suostumus** tai

☐ **Tutkittavan nimenomainen suostumus**.

3. Tutkimuksesta vastaavat tahot

Tutkimuksen tekijä(t):

Karoliina Sormunen,
p. [redacted]
jokasavo@student.jyu.fi
[redacted]

Tutkimuksen ohjaajat:

Antti Ekonoja
antti.j.ekonoja@jyu.fi
+358 40 8053257

Agora AgC522.3, Mattilanniemi 2, 40100 Jyväskylä

Antti-Jussi Lakanen
anlakane@jyu.fi
040 805 3276
Agora AgC414.2, Mattilanniemi 2, 40100 Jyväskylä.

4. Tutkimuksen tausta ja tarkoitus

Uuteen opetussuunnitelmaan (POPS2014) tuli ohjelmointi osaksi opetuksen tavoitteita. Tieto- ja viestintätekniikan (TVT) opetus on myöskin mainittu osana laaja-alaisia osaamistavoitteita (L5, Tieto- ja viestintäteknologinen osaaminen). Tämän pro gradu -tutkielman tavoitteena on kartoittaa kuudennen luokan oppilaiden tämän hetkistä taitotasoa ohjelmoinnin osalta. Keväällä 2019 alakoulun päättävät oppilaat ovat opiskelleet ohjelmointia kolme lukuvuotta. Ohjelmoinnissa siirrytään seitsemännellä vuosiluokalla tekstuaaliseen ohjelmointiin, joten on mielenkiintoista selvittää, että minkälaisilla taitotasoilla oppilaat siirtyvät yläkoulun puolelle.

Tämän tutkimuksen tavoitteena on selvittää 6. luokkalaisten taitotasoja ohjelmoinnissa alakoulun päättyessä.

Tutkimukseen osallistuvat henkilöt, jotka ovat 12-13-vuotiaita, kuudennen vuosiluokan päättäviä oppilaita keväällä 2019 ja näiden oppilaiden opettajat. Tutkimukseen osallistuu noin 60 tutkittavaa lasta ja näiden kolme opettajaa.

Tutkimuksessa ei ole välttämätöntä käsitellä mitään henkilötietoja eikä henkilötietoja kerätä tutkimuksen aikana. Tutkimuksen vastauksiin liitettävät tiedot käsitellään sähköisessä muodossa (kuvakaappaukset oppilaiden ohjelmointitehtävistä Scratch-ohjelmointiympäristössä), haastattelulitterointeina (opettajan haastattelu) ja muina muistiinpanoina.

5. Tutkimuksen toteuttaminen käytännössä

Tutkimukseen osallistuminen kestää yhden päivän: oppilas tekee ohjelmointitehtävän yhtenä päivänä ja opettaja haastatellaan yhtenä päivänä.

6. Tutkimuksen mahdolliset hyödyt ja haitat tutkittaville

Oppilas saa tietoa omasta ohjelmointiosaamisestaan.

7. Henkilötietojen suojaaminen

Tutkimuksessa kerättyjä tietoja ja tutkimustuloksia käsitellään luottamuksellisesti tietosuojalainsäädännön edellyttämällä tavalla. Tietoja ei voida tunnistaa tutkimukseen liittyvistä tutkimustuloksista, selvityksistä tai julkaisuista.

Henkilötiedot suojataan tutkimuksen aikana anonymisoimalla tiedot ja niitä säilytetään Jyväskylän yliopiston salatulla verkkolevyllä, johon pääsy on ainoastaan tutkijalla.

Tutkimustuloksissa ja muissa asiakirjoissa tutkittavaan viitataan vain tunnistekoodilla.

Tutkimusaineistoa säilytetään Jyväskylän yliopiston tutkimusaineiston käsittelyä koskevien tietoturvakäytänteiden mukaisesti.

8. Tutkimustulokset

Tutkimuksesta valmistuu opinnäytetyö.

9. Tutkittavan oikeudet ja niistä poikkeaminen

Tutkittavalla on oikeus peruuttaa antamansa suostumus, kun henkilötietojen käsittely perustuu suostumukseen. Jos tutkittava peruuttaa suostumuksensa, hänen tietojaan ei käytetä enää tutkimuksessa.

Tutkittavalla on oikeus tehdä valitus Tietosuojavaltuutetun toimistoon, mikäli tutkittava katsoo, että häntä koskevien henkilötietojen käsittelyssä on rikottu voimassa olevaa tietosuojalainsäädäntöä. (lue lisää: http://www.tietosuoja.fi).

Tutkimuksessa ei poiketa muista tietosuojalainsäädännön mukaisista tutkittavan oikeuksista.

Henkilötietojen säilyttäminen ja arkistointi

Rekisteriä säilytetään yhdessä paikassa ilman tunnistetietoja, kunnes tutkimus on päättynyt ja opinnäytetyö on hyväksytty. Tämän jälkeen aineisto hävitetään.

10. Rekisteröidyn oikeuksien toteuttaminen

Jos sinulla on kysyttävää rekisteröidyn oikeuksista, voit olla yhteydessä tutkimuksen tekijään.

JYVÄSKYLÄN YLIOPISTO
UNIVERSITY OF JYVÄSKYLÄ

Seminaarinkatu 15
PO BOX 35, FI-40014
Jyväskylä,
Finland

Tel +358 14 260 1211
Business ID:
0245894-7
VAT number:
FI02458947

C Liite: Haastattelurunko

1. Miten kauan olet opettanut kyseistä luokkaa? Opetatko tieto- ja viestintätekniikkaa luokallesi vai onko koulussanne TVT-vastaava, joka opettaa?
2. Minkälaisia laitteita teillä on käytettävissä tieto- ja viestintätekniikan opettamiseen?
3. Koetko hallitsevasi TVT-laitteiden opetuskäytön sujuvasti?
4. Mihin aineisiin olet integroinut ohjelmoinnin opetusta, vai pidättekö pelkästään ohjelmointitunteja?
5. Kuinka monta oppituntia arviolta olet käyttänyt ohjelmoinnin opettamiseen vuositasolla? Mitä ohjelmointiympäristöä olette käyttäneet ohjelmoinnin opetukseen?
6. Koetko ohjelmoinnin opetuksen tärkeäksi?
7. Mitä kautta olet hankkinut tietoa ohjelmoinnin opetuksesta? / Oletko saanut perehdytystä ohjelmoinnin opettamiseen?
8. Oletko tutustunut ohjelmoinnin opetuksen tavoitteisiin yläkoulun puolella ja pohtinut, mitä asioita ohjelmoinnin opetuksessa tulee käsitellä alakoulun puolella?
9. Minkälainen on mielestäsi hyvän osaamisen graafisessa ohjelmointiympäristössä tehty ohjelma?

D Liite: Koekysymykset

Aikaa testin tekemiselle on oppitunti. Ohessa on kokeen kysymykset.

1. Kirjaudu saamillasi tunnuksilla Scratch -ohjelmointiympäristöön (https://scratch.mit.edu/).
2. Etusivun kohdasta ”Luo” pääset ohjelmoimaan). Editorin ja ohjelman kielen voi muuttaa editorin ylälaidasta, maapallon kuvakkeesta.
3. Lue kunkin tehtävän kaikki alakohdat läpi ennen kuin aloitat tehtävän tekemisen. Tee jokaisesta tehtävästä vähintään ensimmäinen ohjelma - etene vasta tämän jälkeen vaativimpiin tehtäviin.
4. Ohjelman luotuasi nimeä ohjelmasi tehtävätyypin ja tehtävänumeron mukaan, esim. ”toisto1”, ”toisto2” tai ”toisto3”.
5. Klikkaa ohjelmointialueen ylälaidasta ”Jaa” -nappia, jonka jälkeen lisää se oikeaan Studioon (Studio Toisto/Muuttujat/Ehtolauseet palautettavan tehtävän mukaan).

Tehtävä 1: Toisto

1. Tee ohjelma, jossa liikutat kissaa ympäri näyttämöä siten, että kissa liikkuu suorakulmion muotoisen matkan: ensin oikealle, alaspäin, vasemmalle ja tämän jälkeen ylös.
2. Tee ohjelma, jossa liikutat kissaa viisi kertaa ympäri näyttämöä edellisen ohjeen mukaisesti.
3. Tee edellisen kohdan ohjelma käyttäen vain yhtä ”liiku” -komentoa. Mikäli teit edellisen kohdan käyttäen yhtä liiku -komentoa, voit palauttaa tähän saman vastauksen.

Tehtävä 2: Muuttujat

1. Tee ohjelma, jossa alussa kissalla on aluksi nolla elämää. Välilyöntiä painettaessa kissan elämien määrä kasvaa yhdellä.
2. Tee ohjelma, jossa kissa ensin kysyy käyttäjän nimeä. Tämän jälkeen kissa tervehtii käyttäjää tämän nimellä (”Moikka <nimi>”).
3. Ennen tätä tehtävää, tee vähintään Ehtolause1 -tehtävä.

   Muokkaa ohjelmaa: Studio2:sta löytyy valmiina ohjelma, ota siitä itsellesi remix. Ohjelmassa koripallo tippuu maata kohden ja pelaajalla on pelin alkaessa kymmenen pistettä. Aina hiireen osuttaessa pallo ottaa uuden sijainnin näyttämön ylälaidasta ja jatkaa putoamista alaspäin. Tehtävänäsi on muokata ohjelmaa siten, että maahan (ruskeaan osaan taustasta) osuessaan pistemäärä vähenee (voit tarkastella tätä käyttäen ”onko väri ___” -komentoa). Pistemäärän ollessa nolla, peli päättyy.

Tehtävä 3: Ehtolauseet

1. Tee ohjelma, joka liikuttaa kissaa eteenpäin ikuisesti. Hiiren osoittimen osuessa kissaan kaikki liike pysähtyy.
2. Tee ohjelma, joka liikuttaa kissaa eteenpäin ikuisesti. Seinään törmätessään kissa kääntyy haluamasi asteluvun verran eri suuntaan ja jatkaa matkaa.
3. Tee ohjelma, jossa kissa kysyy käyttäjältä kaksi lukua. Tee näiden numeroiden välillä vertailu ja laita kissa toteamaan "ensimmäinen numero oli isompi"tai "toinen numero oli isompi". Numeroiden ollessa yhtä suuria kissa toteaa "numerot ovat yhtä suuria".

Muuttujat3-tehtävän aloitus

Kuvio 10. Muuttuja 3-tehtävän aloitusnäkymä

E Liite: Oikeat vastaukset

Tehtävä 1: Toisto

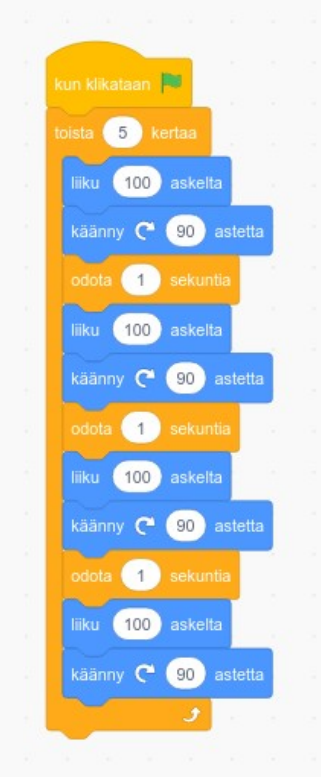
1. Tee ohjelma, jossa liikutat kissaa ympäri näyttämöä siten, että kissa liikkuu suorakulmion muotoisen matkan: ensin oikealle, alaspäin, vasemmalle ja tämän jälkeen ylös.

Ohessa esimerkkivastaus. Tauot on lisätty näyttämään enemmän kissan liikkumista, sillä muuten kissan siirtymistä ei näy ohjelmaa suorittaessa.

***Myös** kissan liu'uttaminen ("liu'u 1 sekuntia sijaintiin xx, yy") käyttäen x- ja y-koordinaatteja on oikein tässä tehtävässä.*

2. Tee ohjelma, jossa liikutat kissaa viisi kertaa ympäri näyttämöä edellisen ohjeen mukaisesti.

Ohessa esimerkkivastaus. Tehtävässä riitti lisätä silmukka edellisen tehtävävastauksen ympärille ja muuttaa toistoluku viideksi.



3. Tee edellisen kohdan ohjelma käyttäen vain yhtä ”liiku” -komentoa. Mikäli teit edellisen kohdan käyttäen yhtä liiku -komentoa, voit palauttaa tähän saman vastauksen.

Ohessa esimerkkivastaus. Edellisen tehtävän tekemisen jälkeen voidaan huomata, että silmukan sisällä toistuu kaava ”liiku – käänny – odota”. Tämä kaava voidaan täten lisätä sisälle silmukkaan. Tehtävän oikein suorittaminen vaati sisäkkäisten silmukoiden käyttöä.

Tehtävä 2: Muuttujat

1. Tee ohjelma, jossa alussa kissalla on aluksi nolla elämää. Välilyöntiä painettaessa kissan elämien määrä kasvaa yhdellä.

Yllä on esimerkkivastaus. Tehtävässä ei välttämättä tarvinnut alustaa muuttujan arvoa nollaksi, eli pelkästään oikean puoleinen kokonaisuus on myös oikein. Muuttujan alustaminen on kuitenkin edistyneemmän vastauksen merkki.

2. Tee ohjelma, jossa kissa ensin kysyy käyttäjän nimeä. Tämän jälkeen kissa tervehtii käyttäjää tämän nimellä (”Moikka <nimi>”).

Tehtävään oikeaksi vastaukseksi hyväksyttiin kahdenlaisia vastauksia. Ensimmäisenä on esitelty yleinen vastaus:

Toinen vastaus oli enemmän sitä, mitä haettiin:

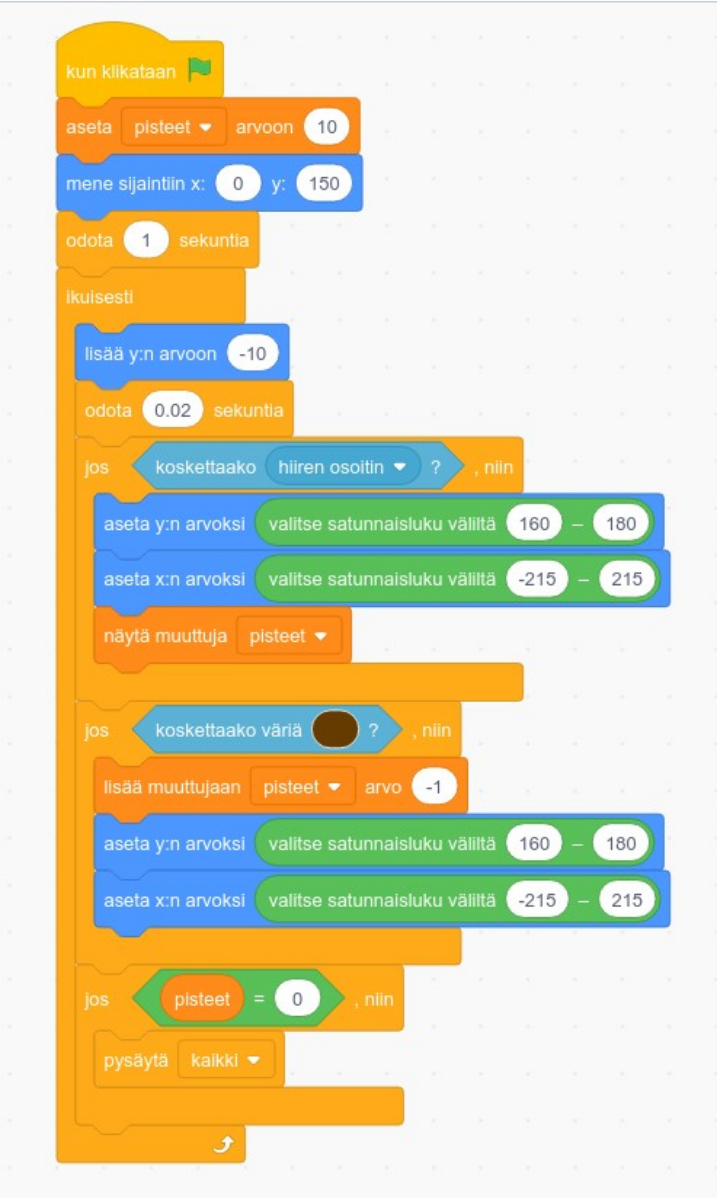
Kyseinen vastaus ilmentää sen, että muuttujan käsite on oppilaalla hallussa: muuttujaa voidaan käyttää hyödyksi osana tekstiä.

3. Ennen tätä tehtävää, tee vähintään Ehtolause1 -tehtävä.

Muokkaa ohjelmaa: Studio2:sta löytyy valmiina ohjelma, ota siitä itsellesi remix. Ohjelmassa koripallo tippuu maata kohden ja pelaajalla on pelin alkaessa kymmenen pistettä. Aina hiireen osuttaessa pallo ottaa uuden sijainnin näyttämön ylälaidasta ja jatkaa putoamista alaspäin. Tehtävänäsi on muokata ohjelmaa siten, että maahan (ruskeaan osaan taustasta) osuessaan pistemäärä vähenee (voit tarkastella tätä käyttäen ”onko väri ___” -komentoa). Pistemäärän ollessa nolla, peli päättyy.

Ohessa esimerkkivastaus.

Tehtävä kysyy oppilaalta jonkinlaista ymmärrystä tai lukutaitoa x- ja y-koordinaatien suhteen, mutta lisäksi loogisten operaatioden hallintaa (”jos pisteet = 0”).



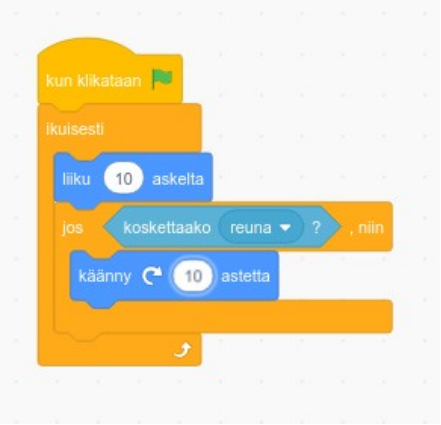
Tehtävä 3: Ehtolauseet

1. Tee ohjelma, joka liikuttaa kissaa eteenpäin ikuisesti. Hiiren osoittimen osuessa kissaan kaikki liike pysähtyy.

Ohessa esimerkkivastaus. Kysymyksenasettelu ei anna suoraan vihjettä ehtolauseen käyttämisestä – siksi myös ”toista kunnes <koskettaako hiiren osoitin>” -vastaus luetaan tähän tehtävään oikeaksi. Kuitenkin haettu ehtolauseen käyttö on kysymyspatteriston otsikosta havaittavissa.

2. Tee ohjelma, joka liikuttaa kissaa eteenpäin ikuisesti. Seinään törmätessään kissa kääntyy haluamasi asteluvun verran eri suuntaan ja jatkaa matkaa.

Ohessa esimerkkivastaus. Tehtävässä on nähtävissä jo pieni kohta oman ”pong-pelin” tekemistä. Edelliseen tehtävään nähden ratkaisu ei poikkea paljoa.



3. Tee ohjelma, jossa kissa kysyy käyttäjältä kaksi lukua. Tee näiden numeroiden välillä vertailu ja laita kissa toteamaan "ensimmäinen numero oli isompi"tai "toinen numero oli isompi". Numeroiden ollessa yhtä suuria kissa toteaa "numerot ovat yhtä suuria".

Ohessa esimerkkivastaus. Oikea ratkaisu vaatii jo paljon taitoja.