Tartu Ülikool
Sotsiaalteaduste valdkond
Haridusteaduste instituut
Õppekava: haridusteadus (reaalained)

Getriin Kokk

ALGORITMIDE RAKENDAMISE HINDAMINE PROBLEEMIDE LAHENDAMISEL SCRATCH´I PROJEKTIDE NÄITEL

bakalaureusetöö

Juhendajad: lektor Tauno Palts
lektor Mario Mäeots

Tartu 2018

Sisukord

Sissejuhatus

Elame infoühiskonnas, mis on infot tähtsustav ja seda kõigil eluvaldkondades maksimaalselt kasutav (Eesti keele seletav sõnaraamat, 2009). Selleks, et saavutada edu infoühiskonnas, vajame teadmisi, oskusi, hoiakuid ja väärtusi, mis aitaksid kohaneda ja hakkama saada tänapäevaste väljakutsete ja nõudmistega (What is DQ?..., s.a.). Maailma majandusfoorumi uuringu aruanne „Tuleviku töökohad“ kohaselt kaob 2020. aastaks viis miljonit töökohta automatiseerimise tõttu (The Future of Jobs, 2016). Samas loob automatiseerimine ka uusi töökohti ning koos sellega muutuvad oskused, mida tööandjad oma töötajatelt ootavad (The Future of Jobs, 2016). Euroopa Komisjoni raport Digital Agenda Scoreboard juhib tähelepanu sellele, et lähitulevikus on vaja digitehnoloogia rakendamisega seotuid oskuseid ja teadmiseid vähemalt 90% töökohtadest (European Commission, 2013). Näiteks info otsimiseks, selle kasutamiseks ja suhtlemiseks ning probleemide lahendamiseks kasutades digitehnoloogiat (Claro et al., 2012; International ICT Literacy Panel, 2002). Digivahendite kasutamine probleemilahenduses aitab leida loovaid ning alternatiivseid lahenduskäike, selleks et digivahenditega oleks võimalik lahendusi luua, peab mõistma ning arusaama nende kasutamisest. Üheks selliseks võimaluseks on algoritmide rakendamine probleemilahenduses (*computational thinking,* edaspidi kasutatakse lühendit CT) (The Future of Jobs, 2016; What is DQ?..., s.a.).

Algoritmide rakendamise olulisust probleemide lahendamisel rõhutas juba Seymour Papert 1980. aastal ilmunud teoses Mindstorms (Papert, 1980). Hilisem aktiivsem laine CT õpetamisel algas 2006. aastal, kui selle mõiste pakkus taas välja Jeannette R. Wing (Wing, 2006). 2010. aastal defineerivad Cuny, Snyder, Wing (Cuny, Snyder, & Wing, 2010, viidatud Wing, 2010 j) mõiste CT järgnevalt: „CT on mõtlemise protsess, mis sisaldab probleemi sõnastamist ning sellele probleemile lahenduse leidmist nii, et lahendus oleks teostatav arvuti, teise inimese või masina poolt.“ Wing (2006) väidab, et CT-d kasutatakse poes kassa järjekorda valides, koolikotti kokku pannes, kadunud asja leidmiseks sellega läbitud tee meenutamiseks. Samas ei ole tõendeid selle kohta, et CT aitaks igapäevatoimetusi paremini lahendada (Denning, 2017).

Lähtudes 2010. aastal defineeritud CT mõistest (Cuny, Snyder, & Wing, 2010, viidatud Wing, 2010 j) on oluliseks osaks CT juures arvutiteaduse kontekstis programmeerimise põhimõtete õpetamine ning selle vastu huvi äratamine (Basawapatna, Koh, & Repenning, 2010). Tavaliste programmeerimiskeelte sümbolite ja reeglite õppimiseks kulub palju aega ja rõhk võib osutuda põhimõtete asemel süntaksi õppimisele, samas kui

visuaalsete programmeerimiskeeltega on võimalik luua mänge või stimulatsioone vaid mõne tunniga (Basawapatna et al., 2010; Lye & Koh, 2014). Visuaalsete programmeerimiskeelte eesmärk on õpetada programmeerimise põhimõtteid, parandades probleemilahenduse oskusi (Koh, Nickerson, Basawapatna, & Repenning, 2014). Üks enimlevinud visuaalse programmeerimise keskkondi on Scratch. Küll aga puudub ülevaade instrumentidest hindamaks Scratch´i projektides CT olemasolu ja arengut. Monograafia formaadis bakalaureusetöö eesmärk on hinnata Scratch´i projektides CT oskuste olemasolu ja selle arengut.

Töö koosneb neljast osast. Esimeses teoreetilise ülevaate osas kirjeldatakse CT olemust ning aspekte, tuuakse välja võimalused CT hindamiseks. Teises osas kirjeldatakse uurimuse metoodikat. Kolmandas osas esitatakse uurimuse käigus saadud tulemused ning neljandas osa arutletakse saadud tulemuste üle.

Teoreetiline ülevaade

Pärast Wingi (2006) on tegelenud CT defineerimisega mitmed teadlased. Üldiselt nõustutakse, et CT on mõtlemine protsess. Näiteks Guzdial (2008) toob välja, et CT aitab mõista kuidas arvutid töötavad. Samas Aho (2011) määratleb CT-d kui mõtlemise protsessi, mil hõlmab endas laiemalt probleemi sõnastamist nii, et probleemi lahendust saaks esitada algoritmina. Lee jt (2011) leiavad, et CT-l on ühiseid aspekte matemaatilise, tehnilise, loova ja algoritmidel põhineva mõtlemisega. Selby ja Woollard (2013) märgivad, et CT-l on ühiseid aspekte loogilise, tehnilise, matemaatilise ning algoritmidel põhineva mõtlemisega. Resnick (2007) ja Wing (2006) leiavad, et CT üldine tähendus on süsteemide loomine ja inimeste käitumismallide mõistmine toetudes arvutiteaduse põhitõdedele. Töö teoreetiline osa annab ülevaate CT olemusest, CT omandamisest ja toetamisest hariduses ning CT hindamise võimalustest.

Algoritmide rakendamine probleemide lahendamisel ehk CT

Wing (2006) väidab, et CT-d on võimalik kasutada mistahes probleemide lahendamiseks. Ta sõnastab CT olemuse kuue olulise punktiga:

- CT on pigem mõtlemine kui programmeerimine;
- CT on oskus, mida kõik inimesed vajavad infoühiskonnas;
- CT on mõtlemine, mida inimesed kasutavad probleemide lahendamiseks;
- CT sisaldab endas matemaatikat ja inseneriteadust

- CT on mudel, mida kasutada probleemide lahendamiseks
- CT-d kasutavad kõik ja igal pool (Wing, 2006).

Cuny, Snyder ja Wing (Cuny, Snyder, & Wing, 2010, viidatud Wing, 2010 j) defineerivad CT-d: „CT on mõtlemise protsess, mis sisaldab probleemi sõnastamist ning lahenduse leidmist, selliselt et lahendus oleks teostatav arvuti, masina või teise inimese poolt.“ Denning (2017) märgib, et pärast 2006. aastat ei ole ühtegi CT definitsiooni, mis väljendaks algoritmilise mudeli kasutamist. Selleks et CT-d hinnata on oluline defineerida CT aspektid mida hinnatakse. Brennan ja Resnick (2012) ning Moreno-León, Robles, ja Román González (2015) leiavad, et CT oskutse arengut on võimalik toetada visuaalsete programmeerimiskeeltega, hinnates programmides olevaid CT aspekte. Järgmises peatükis antakse ülevaate kirjanduses leiduvatest CT aspektidest.

CT erinevad aspektid

BBC Bitesize (BBC, 2016) on välja toodud, et CT võimaldab keerulist probleemi mõista ning võimaliku lahendust leida. CT abil saame lahendust esitada nii, et arvuti, inimene või mõlemad saaksid probleemist ja selle lahendusest aru (BBC, s.a.; Cuny, Snyder, & Wing, 2010, viidatud Wing, 2010 j).

Repenning, Basawapatna ja Escherle (2016) toetudes Cuny, Snyder ja Wing´i (Cuny, Snyder, & Wing, 2010, viidatud Wing, 2010 j) definitsioonile jagavad CT kolmeks etappiks. Nende etapid põhinevad masinate poolt teostatavatele lahendustele. Etappideks on: probleemi sõnastamine; lahenduse väljendamine, selliselt et arvuti või masin oleks võimeline probleemi lahendama; lahenduse katsetamine ning sellele hinnangu andmine (Repenning et al., 2016).

CT on kirjanduses defineeritud kui mõtlemise ning probleemide lahendamise protsess, mis sisaldab endas järgnevaid aspekte: loogiline mõtlemine (*logical thinking*), algoritmiline mõtlemine (*algorithmic thinking*), tingimuslausete kasutamine, andmete kogumine (*data collection*), andmete süstemaatiline töötlemine, andmete analüüsimine (*data analysis*), andmete esitamine (*data representation*), andmete esitamine mudelite abil, andmete esitamine simulatsioonide abil, keeruliste probleemide väiksemateks osadeks võtmine (*decomposition*), olulisele keskendumine, abstraheerimine (*abstraction*), sarnasuste otsimine, mustrite tuvastamine (*pattern recognition*), erinevate probleemi osade ühisosa leidmine (*parallelization*), analüüsimine (*analysis*), automatiseerimine (*automation*), süsteemide loomine (*system design*), probleemi ja sarnaste probleemide lahendamiseks algoritmi loomine (*algorithm design*), lahendus(t)e analüüsimine, lahendus(t)e modelleerimine (*modeling*), lahendus(t)e simuleerimine (*simulation*), voo haldamine (*flow control*), samm-sammuline

lahendus(t)e esitamine (*algorithms*), lahendus(t)e testimine, lahendus(t)e rakendamine, probleemi(de) lahendamine (*problem solving*), hindamine (*evalution*), probleemi lahendus(t)e üldistamine (*generalization*), süsteemi või lahendus(t)e veaavastused, lahendus(de)st vigade kõrvaldamine (*debugging*) (BBC, s.a.; Csizmadia, Curzon, Dorling, Humphreys, Ng, Selby, & Woollard, 2015; International Society for Technology in Education, 2018; International Society for Technology in Education & Computer Science Teachers Association, 2011; Exploring Computational Thinking, s.a.; Grover & Pea, 2013; Selby & Woollard, 2013; Stephenson & Barr, 2011; Wing, 2008).

CT aspektid probleemilahenduses.

Selby ja Woollard (2013), toetudes CT alase kirjanduse analüüsile ning Wingi (2006) ja CT mõiste (Cuny, Snyder, & Wing, 2010, viidatud Wing, 2010 j) määratlusele, leiavad, et CT definitsioon peaks endas sisaldama kuute komponenti: ideed CT-st kui mõtlemise protsessist (*thought process*); mõistet abstraktsioon; oskust võtta probleeme lihtsamateks osadeks; üldistamist; probleemi lahenduse esitamist üksteisele järgnevate sammudega; hindamist. Eelnimetatud aspektide kirjeldused on toodud tabelis 1. Selby ja Woollard (2013) defineerivad CT-d kui meetodit probleemide lahendamiseks, mis sisaldab eelnimetatud komponente. Csizmadia jt (2015) on CT õpetamiseks välja töötanud juhendi, milles on välja toodud samad CT aspektid, mis Selby ja Woollard (2013), ning lisaks veel loogiline arutluskäik (*logical reasoning*). Aspekti loogiline arutluskäik saab kasutada lahenduse testimisel ja parandamisel (Csizmadia et al., 2015). Selby ja Woollard (2013) ning Csizmadia jt (2015) poolt määratud CT aspekte on kirjeldanud ka teised autorid (BBC, s.a.; Exploring Computational Thinking, s.a.; Grover & Pea, 2013; Selby & Woollard, 2013; Stephenson & Barr, 2011).

Tabel 1. CT määratlustes enim esinevate aspektide kirjeldused Selby ja Woollard järgi (2013)

CT aspekt	Kirjeldus
Mõtlemisprotsess	Mõtlemisprotsess probleemide ja lahenduste sõnastamiseks (Aho, 2011).
Abstraktsioon	Abstraktsioon on oskus olulist eristada ebaolulisest. Abstraktsiooni abil on võimalik peita ebaolulised detailid, selliselt et kõik oluline jääb alles ning selle abil probleem lihtsustub (Csizmadia et al., 2015; Exploring Computational Thinking, s.a.; Grover & Pea, 2013; Stephenson & Barr, 2011).

Osadeks võtmine	Oskus probleemi osadeks liigendada. See lihtsustab osade eraldi hindamist ja funktsiooni uurimist, mis omakorda aitab keerulise probleemi lahenduse leidmist ning selle teostamist (Csizmadia et al., 2015; Exploring Computational Thinking, s.a.; Stephenson & Barr, 2011).
Üldistamine	Üldistamine on oskus märgata mustreid, korduseid, sarnasusi ning seostada neid juba leitud lahendustega, seeläbi lahendada probleem toetudes eelnevatele kogemustele ja lahendustele (Csizmadia et al., 2015; Grover & Pea, 2013; Selby & Woollard, 2013).
Algoritmide kasutamine	Algoritmide kasutamine on oskus esitada lahendust samm-sammuliselt (Csizmadia et al., 2015; Exploring Computational Thinking, s.a.; Selby & Woollard, 2013).
Hindamine	Leitud lahenduse hindamine. Hindamisel peaks keskenduma järgnevale: lahenduse eesmärgipärasus; lahenduse otstarbekus; lahenduse majanduslikkus; lahenduse lihtsus; lahenduse kasutamine (Csizmadia et al., 2015; Selby & Woollard, 2013).

CT mõiste akadeemilised ja pedagoogilised tõlgendused on erinevad (Selby, Dolring, & Woollard, 2014). Selby ja Woollard (2013) poolt välja pakutud CT definitsiooni aspektid kirjeldavad probleemi lahendamise protsessi ning sisaldavad mõtlemisprotsessi, mis lõimivad endas eelnimetatud aspekte. Brennan ja Resnick (2012) on mõiste CT sisu jaganud kolmeks põhiaspektiks: CT aspektid programmeerimises, CT tavad ja CT kasutamine probleemide lahendamisel. Kui Selby ja Woollard (2010) lähenevad CT-le, kui probleemide lahendamisele keskendunud mõtlemisele, siis Brennani ja Resnicku (2012) definitsiooni põhimõiste CT aspektid programmeerimises annab võimaluse toetada CT aspektide õppimist ja arendamist programmeerimisega.

Programmeerimistehnikad CT õppimisel.

CT aspekte saab arendada programmeerimisega, selleks ei pea kasutama traditsioonilise programmeerimiskeeli, nagu Java, C++ või Python, vaid võib kasutada visuaalseid programmeerimiskeeli, mis aitavad õppijatel omandada ja arendada CT aspekte, ilma et peaks õppima programmeerimiskeelele omaseid süntakseid (Basawapatna et al., 2010; Lye & Koh, 2014). Kirjanduses leiduvad CT aspektid, mida on võimalik õppida ja arendada programmeerimisega on järgnevad: algoritmiline mõtlemine, loogiline mõtlemine,

abstraheerimine, modelleerimine, sünkroniseerimine (*synchronization*), lähtestamine, kasutaja kontroll (*user control*), tekke protsess (*generation*), absorbeerimine (*absorption*), kokku põrkamine (*collision*), liigutamine (*transportation*), lükkamine (*push*), tõmbamine (*pull*), segunemine (*diffusion*), liikumine vastavalt algoritmile (*Hill Climbing*), edastus ja vastuvõtmine (*broadcast and recive*), animatsioon (*complex animation*), parallelism (*parallelism*), voo haldamine (*flow control*), kasutaja interaktiivsus (*user interactivity*), andmete esitamine, lahenduse esitamine sammudena (*sequences*), tüklite kasutamine (*loops*), sündmus (*events*), tingimuslaused (*conditionals*), tehtemärgid (*operators*) (Brennan & Resnick, 2012; Lye & Koh, 2014; Repenning, Webb, & Ioannidou, 2010; Moreno-León & Robles, 2015a, 2015b). Brennan ja Resnick (2012) toovad välja seitse CT aspekti, mida saab arendada programmeerimiskeelte kontekstis, nendeks on: lahenduse esitamine sammudena, tsüklite kasutamine, parallelism, sündmus ehk programmi poolt avastatav tegevus või toiming, tingimuslaused, tehtemärk, andmed (*data*). Ka Moreno-León ja Robles (2015a, 2015b) toovad välja seitse CT aspekti, mida on võimalik arendada programmeerimisega ning hinnata programmides. Nimetatud autorite (Brennan & Resnick, 2012; Moreno-León & Robles, 2015a, 2015b) poolt välja toodud CT aspektide kirjeldused ja võrdlus on tabelis 2. Brennani ja Resnicku (2012) ning Moreno-Leóni ja Roblesi (2015a, 2015b) CT aspektide jaotus hõlmab ka teiste autorite (Basawapatna et al., 2010; Csizmadia et al., 2015; Exploring Computational Thinking, s.a.; Ioannidou, Bennett, Repenning, Koh, & Basawapatna, 2011; Koh et al., 2014; Selby & Woollard, 2013; Werner et al., 2012) leitud CT aspekte.

Tabel 2. CT määratlustes programmeerimisele toetuvad aspektid (Brennan & Resnick, 2012; Moreno-León & Robles, 2015a, 2015b)

CT aspekt Brennani ja Resnicku järgi (2012)	CT aspekt Moreno-Leóni ja Roblesi (2015a, 2015b) järgi	Kirjeldus
Lahenduse esitamine sammudena	Abstraheerimine	Lahendus on esitatud üksikute sammude või juhiste järgi nii, et seda on võimalik sooritada arvutil või masinal (Basawapatna et al., 2010; Brennan & Resnick, 2012; Csizmadia et al., 2015; Exploring Computational Thinking, s.a.; Ioannidou, Bennett, Repenning, Koh, & Basawapatna, 2011; Koh et al., 2014; Moreno-León & Robles, 2015a; Moreno-León & Robles, 2015b; Selby & Woollard, 2013;

		Werner et al., 2012).
Tsükkel	Voo haldamine	Tsükkel on korduvale täitmisele kuuluv käsujada (Brennan & Resnick, 2012; Moreno-León & Robles, 2015a, 2015b).
Parallelism	Parallelism	Parallelism ehk samaaegne tegevus programmis, tähendab mitme käsujada samaaegset täitmist (Boe et al. , 2013; Brennan & Resnick, 2012; Dr. Scratch, s.a.; Moreno-León & Robles, 2015a, 2015b).
Sündmus ehk programmi poolt avastatav tegevus või toiming	Kasutaja interaktiivsus Sünkroniseerimine	Sündmus on programmi poolt avastatav tegevus või toiming, näiteks hiireklahvi klõpsamine (Brennan & Resnick, 2012; Werner et al., 2012; Dr. Scratch, s.a.; Moreno-León & Robles, 2015a, 2015b).
Tingimuslaused	Loogiline mõtlemine	Tingimuslausete kasutamine, näiteks „kui…“, „kui …, siis …“ (Basawapatna et al., 2010; Brennan & Resnick, 2012; Ioannidou et al., 2011; Koh et al., 2014; Moreno-León & Robles, 2015a, 2015b).
Tehtemärk	Loogiline mõtlemine	Tehtemärk ehk programmeerimisele ja matemaatilisele loogikale omaste sümbolite kasutamine (Brennan & Resnick, 2012; Moreno-León & Robles, 2015a, 2015b).
Andmed	Andmete esitamine	Andmed ehk kindlal viisil esitatud informatsiooniühikud, nendeks võivad olla näiteks muutuja (*variable*) ja loend (*list*) (Basawapatna et al., 2010; Brennan & Resnick, 2012; Dr. Scratch, 2016; Ioannidou et al., 2011; Koh et al., 2014; Moreno-León & Robles, 2015a, 2015b; Werner et al., 2012).

Brennani ja Resnicku (2012) ning Moreno-Leóni ja Roblesi (2015a, 2015b) leitud seitse CT aspekti, mida on võimalik toetada visuaalsete programmeerimiskeeltega, on sarnased, kuid erineva jaotusega. Kui Brennan ja Resnick (2012) hindavad aspekte tingimuslause ja tehte kasutamine eraldi, siis Moreno-León ja Robles (2015a, 2015b) leiavad,

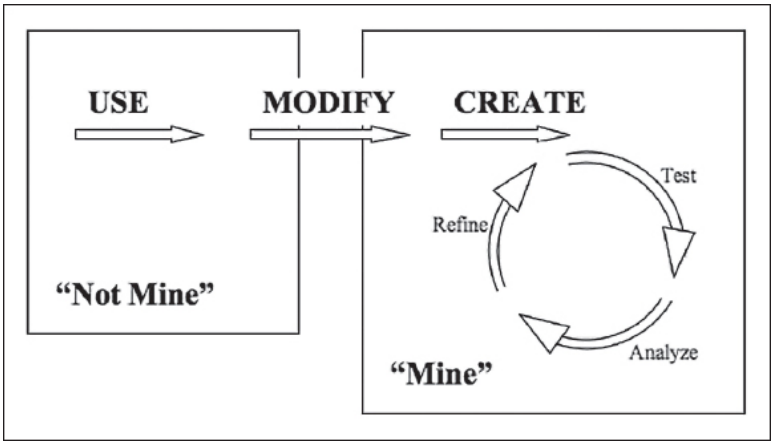
et neid kahte aspekti võib koonda ühe aspekti alla loogiline mõtlemine. Vastupidi Moreno-Leóni ja Roblesi (2015a, 2015b) hindavad Brennani ja Resnicku (2012) välja pakutud aspekti sündmuste toimumine kahe aspektiga, kasutaja interaktiivsus ning sünkroniseerimine. Toetudes CT aspektide loetelule ning erinevate autorite CT määratlusele, annab järgmine peatükk ülevaate CT-st hariduses ning pakub välja võimalikud hariduslikud programmeerimiskeeled, mida CT oskuste arendamiseks kasutada.

CT hariduses

Wing (2006, 2008, 2010) rõhutab, et CT-d kasutatakse igal pool, mis tähendab, et CT puudutab iga inimese elu. Sellest tulenevalt esitab Wing (2008) küsimused, millal peaks CT-d õppima ja mida peaks õppima ning millal peaks kool CT-d õpetama ja mida peaks kool CT-st õpetama. Barr ja Stephenson (2011) toovad välja üheksa CT aspekti ning nende aspektide võimalikud kasutused keskhariduses. Välja toodud üheksa aspekti on järgnevad: andmete kogumine, andmete analüüsimine, andmete esitamine, probleemi väiksemateks osadeks võtmine, abstraheerimine, algoritmide kasutamine ja õppimine, tarkvara kasutamine, erinevate probleemide ühisosa leidmine, lahenduste simuleerimine. Näiteks CT aspekt andmete kogumine võimalik kasutamine matemaatikas on statistika õppimisel täringu veeretamisel või mündi viskamisel saadavad andmed, loodusteaduste valdkonnas on võimalik andmeid koguda katsete tegemisega, sotsiaalteaduste alal on võimalik koguda elanikkonna kohta andmeid, keeleteadustes on võimalik koguda andmeid teostades lingvistilist uurimust, arvutiteaduses on võimalik leida probleemsete valdkondade arvandmeid (Barr ja Stephenson, 2011). Barr ja Stephenson (2011) seovad eelnimetatud üheksa aspekti arvutiteaduse, matemaatika, loodusteaduse, sotsiaalteaduse ja keeleteaduse valdkondadega. Kuigi CT seotus erinevate ainevaldkondadega on olemas ning erinevaid CT aspekte on võimalik koolis õpetada, puudub kontseptsioon CT õpetamiseks koolis ning õpetajate ettevalmistus selleks. Seega vajavad õpetajad tööriistu, mille abil õpilaste CT oskust arendada ning CT-d õpetada. CT õpetamiseks on vajalik keskkond, materjalid ning hindamisvahendid (Grover & Pea, 2013; Lee et al., 2011; Resnick, 2007).

CT arendamist ja omandamist koolis soodustab ning tutvustab kolmeastmelise kasuta-kohanda-loo (*Use-Modify-Create*) mudeli kasutamine, mudeli kasutamine on toodud joonisel 1 (Lee et al., 2011). Mudeli eesmärk on muuta kellegi teise loodu enda omaks. Uued teadmised ja oskused ajendavad vastu võtma suuremaid väljakutsed. Kasuta-kohanda-loo mudeli esimeseks osaks on kogemuste omandamine läbi olemasolevate programmide, selliselt saadakse teadmised programmeerimiskeelest ning tutvutakse loodud programmidega, näiteks

mõne arvutimängu mängimine. Teiseks osaks on olemasolevate programmide täiendamine ja muutmine, näiteks tegelaskuju käitumise või tausta värvi muutmine. Kolmandaks osaks on enda mängu või animatsiooni loomine, selles osas kasutatakse varem saadud teadmisi eelnevatest osadest. Mängu või animatsiooni loomisel katsetatakse mängu, analüüsitakse ning vajadusel täiendatakse (Grover & Pea, 2013; Lee et al., 2011, Resnick, 2007).



Joonis 1. Kasuta-kohanda-loo mudel CT aspektide õppimiseks programmeerimise keskkondades (Lee et al., 2011)

Lähtudes Cuny, Snyder, Wing (Cuny, Snyder, & Wing, 2010, viidatud Wing, 2010 j) CT määratlusest, et probleemi lahendus peaks olema teostatav arvuti, masina või teise inimese poolt, on vaja vahendit või tööriista, mis võimaldaks probleemilahendusega tegeleda. CT arengut koolis aitab toetada visuaalsete programmeerimiskeelte kasutamine.

Visuaalsed programmeerimiskeeled on sellised keskkonnad, kus tavakasutajast saab looja (Grover & Pea, 2013; Lee et al., 2011; Repenning et al., 2010). Algoritmilise probleemilahenduse arendamise vahendid on näiteks StarLogo TNG, StarLogo Nova, Gameblox, Alice, Agentsheets, Scalable Game Design, Agentcubes, Scratch (Grover & Pea, 2013; MIT Scheller Teacher Education Program education arcade (s.a.); Resnick, 2007;). Nimetatud keskkonnad kasutavad kolmeastmelist kasuta-kohana-loo mudelit (Grover & Pea, 2013; Lee et al., 2011; Repenning et al., 2010). Visuaalsed programmeerimiskeeled StarLogo TNG, StarLogo Nova ja Gameblox on arendatud Massachusetts Tehnoloogia Instituudi (Massachusetts Institute of Technology - MIT) Meedia Laboratooriumi ja MIT Teacher Education Program poolt ning võimaldab 3D mudelite ning stimulatsioonide loomist. Gameblox võimaldab olemasolevaid mänge mängida ning uusi luua. Selles uurimuses kasutatakse CT hindamiseks hariduslikku programmeerimiskeelt Scratch. Vabavaraline visuaalne programmeerimiskeel Scratch, mille ametlik versioon ilmus 2007. aastal, on loodud

MIT Meedia Laboratooriumi Lifelong Kindergarten grupi poolt (Scratch, s.a., Scratch Wiki, 2018). Scratch´i eelis on see, et see tõlgitud eesti keelde ning platvormi on võimalik kasutada Mac´i, Windows´i ja Linux´i operatsioonisüsteemidel, lisaks veebipõhisele keskkonnale on võimalik tarkvara alla laadida (Scratch, s.a.; Scratch Wiki, 2018). Järgmine peatükk annab ülevaate Scratch´ist.

Scratch

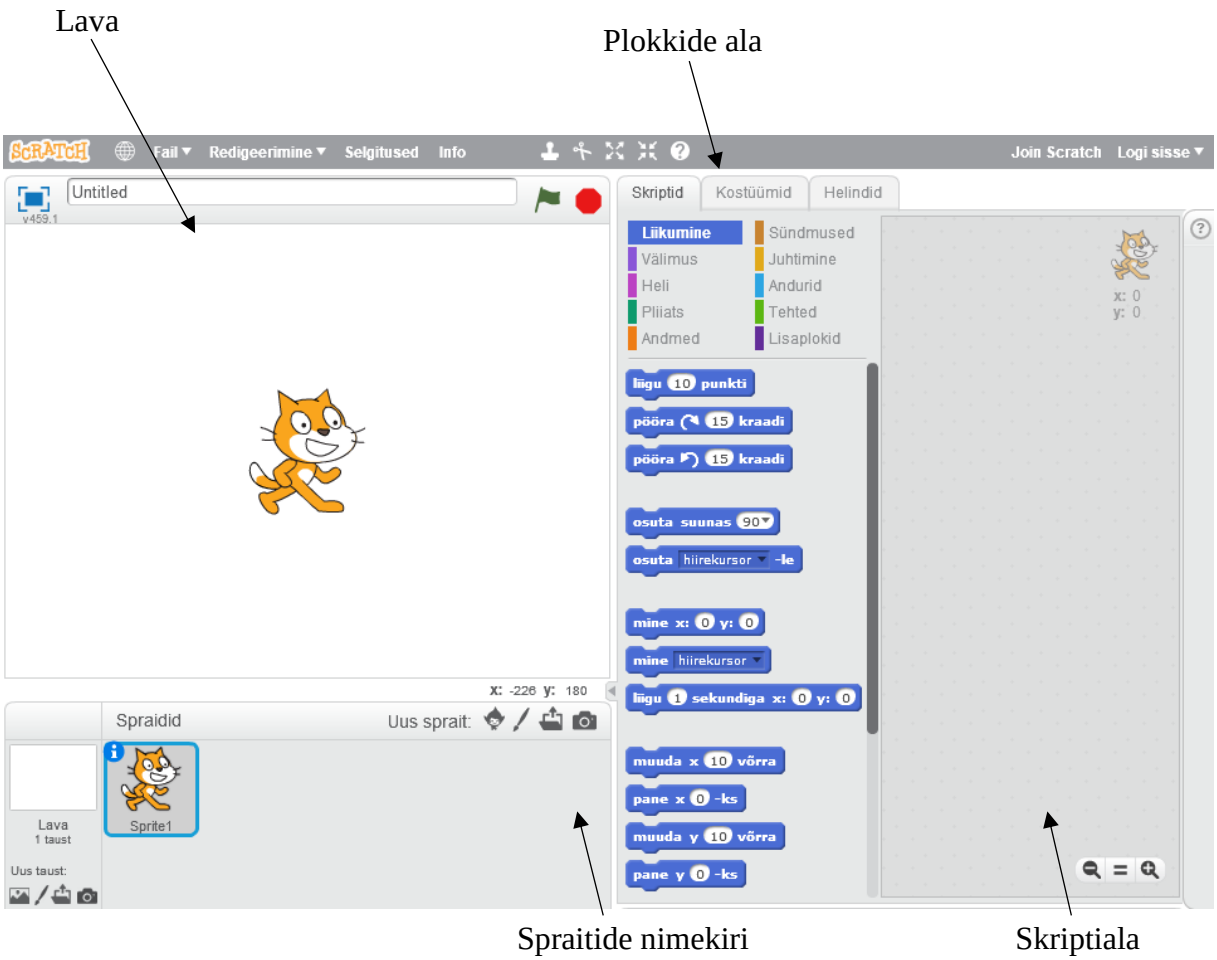
MIT Meedia Laboratooriumi Lifelong Kindergarten grupp on loonud haridusliku programmeerimiskeele Scratch. Scratch´iga saab luua projekte, mis kujutavad endast näiteks mänge, animatsioone, muusikapalu ja palju muud (Resnick, 2007; Scratch, s.a.; Scratch Wiki, 2018). Scratch´i loojate gruppi kuuluv Mitchel Resnick leiab, et Scratch´i eesmärk on ühendada endas mängimine ning õppimine. Mängimine ja õppimine peaksid olema tihedalt seotud, nii nagu need on lasteaias teiste vahenditega. (Resnick, 2007)

Scratch on mõeldud vanusele 8-16. Scratch´is on üle 14 miljoni registreeritud kasutaja ning üle 17 miljoni jagatud projekti. Scratch on saadaval rohkem kui 40 keeles, ka eesti keeles (Scratch Wiki, 2018)

Scratchimine on inglise keeles plaadi kriipimine. Scratchimist kasutatakse uute muusikapalade loomiseks. Programmeerimiskeel Scratch võimaldab erinevaid pilte, helisid, skripte omavahel ühendada nii, et tekiksid uued projektid. Scratch on loodud selleks, et kasutajad õpiksid mõtlema matemaatilisel ja süstemaatiliselt, kasutades ära oma loovust ning ideid. (Scratch, 2016)

Scratch 1.0 versioon ilmus 2007 aastal ning viimane versioon Scratch 2.0 ilmus 2013 aastal. Scratch 2.0 versiooni on võimalik alla laadida või kasutada veebipõhises keskkonnas. (Scratch Wiki, 2018)

Scratch´i kasutajaliides koosneb lavast, spraitide nimekirjast, skriptialast ja plokkide alast (Joonis 2). Lava on koht, kus programmid töötavad. Skripte saab koostada skriptialas tõmmates plokkide alast erinevaid plokke kokku. Spraite saab hallata spraitide nimekirjas. Ühes projektis võib olla mitu erinevat tausta ja spraiti- nii taustal kui ka spraitidel on omaenda skriptid. Ühel spraidil saab olla mitu skripti. Taustasid saab vahetada, ise joonistada, lisada ja kustutada. Taust on pilt. Plokkide ala on jaotatud kaheksasse eri värvi gruppi. Grupid on jaotatud funktsioonide alusel. Iga gruppi iseloomustab üks sõna. Skripti loomiseks peab lohistama plokid plokkide alast skriptialasse ning plokid ühendama skriptiks. Skript on plokkide kogum (Scratch 2.0. Kasutamisjuhend; s.a.; Skriptid ja käsud, s.a.; Scratch Wiki, 2018).



Joonis 2. Scratch´i kasutajaliides

Scratch´i kasutamise muudab lihtsaks programmikoodi plokid . Selleks, et Scratch´is programme luua tuleb omavahel kokku panna graafilised plokid (Joonis 3). Plokid on selliselt loodud, et nad haakuksid üksteisega. Plokkide omavahel haakumisel tekib skript. Skript on käskude kogum. Skript on seotud mingi kindla spraidiga. Sprait on tegelane või objekt. Skriptid täidavad ainult seotud spraidi või lava tegevusi. Salvestatud Scratch´i programmi nimetatakse projektiks (Scratch 2.0. Kasutamisjuhend; s.a.; Skriptid ja käsud, s.a.; Scratch Wiki, 2018).

Joonis 3. Näide keskkonnas Scratch plokkidest loodud skriptist

Töös analüüsitakse CT oskuste taset Scratch´i projektide näitel. Selleks antakse järgnevas peatükis ülevaade erinevatest võimalustest CT oskuste taseme hindamiseks ning millised CT aspekte on võimalik hinnata visuaalsete programmeerimiskeelte kasutamisel CT oskuse arendamiseks.

CT hindamine

CT oskuste hindamiseks on vaja kindlat struktuuri, milliseid CT aspekte hinnatakse, kuidas hinnatakse ja mis infot analüüs annab. Visuaalsed programmeerimiskeeled toetuvad programmeerimisele, järelikult vajatakse ka selle põhist hindamist. Adams ja Webster (2012) toovad oma uurimuses välja, et läbi visuaalsete programmeerimiskeelte mängu luues õpivad õpilased kasutama programmeerimisele omaseid põhimõtteid, näiteks mängu luues kasutatakse projektides muutujaid, tingimuslauset ja tsüklit.

CT kasutamist on võimalik hinnata mitmeti. CT-d on võimalik hinnata intervjuu või vestluse tulemusel, veebitööriista abil, portfoolio või projekti analüüsimise tulemusel (Brennan & Resnick, 2012; Koh et al., 2014; Werner, 2012). Edasises antakse ülevaade võimalikest hindamismeetoditest ning keskkondadest, millega on võimalik hinnata CT oskuste taset.

Intervjuu.

CT oskuseid on võimalik hinnata intervjuu abil. Brennan ja Resnick (2012) viisid läbi intervjuusid, mille kestus oli 60-120 minutit ning selle käigus küsiti küsimusi neljast kategooriast: taust, projektide loomine, kogemuse jagamine ning edasine soov õppida. Näiteid küsimustest: kuidas õpilane on leidnud keskkonna Scratch, mis on Scratch, milleks sa seda keskkonda kasutad, kuidas ja kust leiad ideid projektideks, kas sa jagad ka teistega oma projekte, millised on aspektid sulle Scratch´i juures meeldivad ning millised mitte, milliseid teisi keskkondi kasutad. Kui eelnevalt toodud projekti analüüsi käigus selgub vaid plokkide kasutamine, siis intervjuu käigus selgub, kas ja kuidas on õpilane saanud aru plokkidest. See võib viia ka selleni, et mõned õpilased ei mõtle ega teegi programme ise vaid kopeerivad plokid mõnest varem nähtud programmist. Intervjuu käigus on võimalik näha, kas õpilane mõistab ning saab ka aru teiste programmidest ning oskab neid lugeda. Samas on intervjuu ajamahukas ning grupi õpilaste tagasisidestamiseks on see õpetajale liiga koormav.

Ülesannete lahendamine.

Üheks võimaluseks CT aspektide olemasolu hindamiseks Scratch´i projektides on näidata õppijale kellegi teise koostatud projekti ning seejärel paluda õppijal kirjeldada, mida projekt teeb, kuidas oleks võimalik veel projekti täiendada, leida vead projektis ning parandada need, lisada projekti uus funktsioon. AERA 2012, on loodud kolm plokki ülesandeid, mis igaüks sisaldab kahte konkreetset ülesannet (Brennan ja Resnick, 2012).

Plokk 1 sisaldab kahte ülesannet, mille eesmärk on olemasolevas programmis leida viga, parandada see ning lisada üks osa juurde. Esimeses ülesandes „Nimi“ on ülesandeks kirjutada programm, mis kirjutab nime selliselt, et järgmine täht ilmuks alles siis, kui eelnev on välja joonistatud. Üks täht võiks teha midagi huvitavat, aga ainult siis, kui tähele klõpsatakse. Esimene ploki teine ülesanne „Esinemine“ annab õpilastele ette programmi, milles laulja laulab peale liikumist, ülesandeks on muuta programmi selliselt, et laulja laulaks ja liiguks samal ajal. Lisaks on programmis trummid. Õpilasele on ülesandeks muuta programmi selliselt, et trummid teeks heli vaid siis, kui neil klõpsatakse. Plokkide 2 ja 3 ülesanded on üles ehitatud analoogselt. Plokid sisaldavad kahte ülesannet, mille eesmärk on olemasolevas programmis leida viga, parandada see ning lisada üks osa juurde. Ülesannete lahendamine näitab, kas õpilased on mõistnud programmi sisu, kuidas programmi muuta ning millised võimalused on programmi täiendamiseks. Selline CT aspektide hindamine on töö- ning ajamahukas ning õpilastel puudub võimalus enda CT oskuste arengut hinnata (Brennan ja Resnick, 2012).

Fairy assessment.

Fairy Assesssment on hindamisahend, mis aitab hinnata CT osasid algoritmiline mõtlemine, abstraktsioon ja modelleerimine (Werner et al., 2012). Programm on loodud 3D programmeerimise keskkonnale Alice (Werner et al., 2012). Alice on 3D programmeerimise keskkond, mille abil saab luua mänge ja animatsioone ning neid jagada (Alice, s.a.). Fairy Assessment esitab kasutajale ülesande ning valmis programmi, mida kasutajal tuleb vastavalt ülesandele muuta. Programm on loodud, selliselt et mängu ja paarisprogrammeerimise kaudu toetatakse õpilaste CT oskuste arenemist. Fairy Assessment põhineb kasuta-kohanda-loo mudelil (Lee et al., 2011; Werner et al., 2012), mille põhjal iga ülesanne sisaldab vähemalt kahte osa mudelist: probleemi leidmine programmist, vea märkamine ning programmi muutmine. Fairy Assessment meetodi kasutamisel CT oskuste arendamiseks on kasutajale esitatud kolm üksteisest sõltumatut ning erinevate raskusastmetega ülesannet. Esimese ülesande eesmärk on, et õpilane mõistab programmi, leiab vea, mõtleb algoritmiliselt ning

parandab vea (Joonis 4). See sisaldab endas kasuta-kohana-loo mudeli esimest ja teist etappi - õpilane kasutab kellegi teise poolt loodud programmi ning kohandab seda vastavalt ülesandele. Teine ülesanne nõuab õpilaselt koodi mõistmist, vea tuvastamist ja koodi muutmist vea eemaldamiseks. Kolmas ülesanne vajab algoritmilise mõtlemise oskust, mõistmist, et sündmused muudavad programmi täitmise järjekorda ning uue sündmuse loomist (Lee et al., 2011; Werner et al., 2012). Näiteks Werner jt (2012) andsid õpilastele programmi, milles oli kaks haldjat *HaloSilver* ja *LeafFlame*. Õpilased nägid mängides, kuidas tegelased vestlevad ning *LeafFlame* jalutab võlumetsa, mille tulemusel kahaneb ta poole võrra. Esimeseks ülesandeks oli muuta programmi nii, et *HaloSilver* keeraks ennast vaatama, kui *LeafFlame* võlumetsa kõnnib. Ülesande täitmise eest oli võimalik saada kuni 10 punkti, kui õpilane märkas samaaegsete tegevuste täitmist ning juhiste täitmiste pikkust oli muudetud vastavalt ülesandele. Esimese ülesande käigus kasutati kasuta-kohana-loo mudeli esimest ja teist etappi. Teiseks palub *HaloSilver* õpilastel parandada programm, mis ülesnoole vajutamisel muudaks *LeafFlame´*i suuruse esialgseks. Õpilased peavad leidma olemasolevas programmis vea, mõistma, miks viga on tekkinud ning seejärel selle parandama või uue looma. Ülesande täitmise eest on võimalik saada kuni 10 punkti. Kolmandaks palub *HaloSilver* õpilastel aidata *LeafFlame* metsast välja lennata tema juurde. Ülesande keerukus seisneb selles, et lisaks *LeafFlame* liikumisele peavad lendamisel liikuma ka tema tiivad. Ülesande täitmise eest oli võimalik saada kuni 10 punkti.

Joonis 4. Näide Fairy Assessmenti esimesest ülesandest (Werner et al., 2012)

Fairy Assessment meetodit on võimalik kasutada CT oskuste arendamiseks, kuid meetod ei võimalda programmi koostajal saada kohest tagasisidet. Seega vajab nii õpetaja kui ka õpilane tööriista, mis annab kiire tagasiside ning selge ülevaate, mis ja millised CT

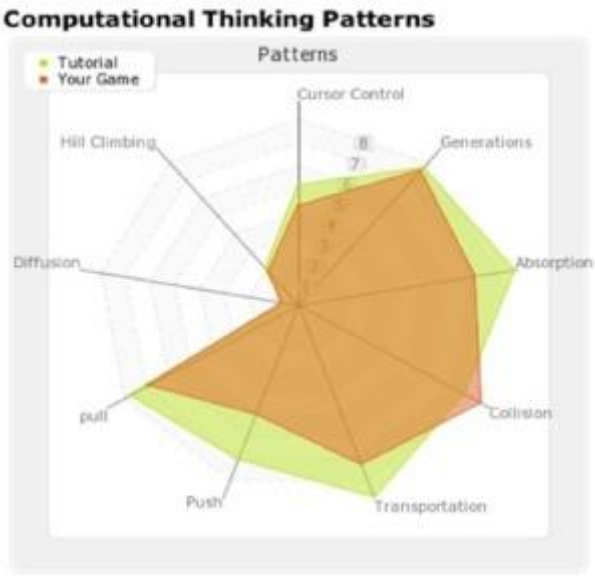
aspektid on koostatud töös olemas.

Computational Thinking Pattern Analysis (CTPA).

CTPA on veebipõhine programmeerimist hindav tööriist. CTPA analüüsib ja visualiseerib CT komponentide olemasolu mängudes. CTPA võrdleb ette antud mängu üheksa CT aspektiga, selleks kasutab programm varjatud semantika analüüsi (*Latent Semantic Analysis*) (Basawapatna et al., 2010; Ioannidou et al., 2011; Koh et al., 2014). CT aspektid, mida CTPA võrdleb, on:

- kasutaja kontroll;
- tekke protsess – üks tegur mõjutab teist;
- absorbeerima – kokku puutumine/tõmbumine, kuidas objektid ühe objekti poole tõmbud või kokku puutumisel liiguvad;
- kokkupõrge – mängus või simulatsioonis kahe teguri kokkupõrge, mille tagajärjel üks tegur on programmeeritud kokkupõrget visuaalselt näitlikustama, peale kokkupõrget on tegurid programmeeritud haihtuma;
- liigutamine – üks tegur on teise külge kinnitatud ning alumine tegur veab/liigutab ülemist;
- lükkamine – üks tegur lükkab teist tegurit;
- tõmbamine – üks tegur tõmbab teist tegurit;
- segunemine;
- liikumine vastavalt algoritmile – arvutiteaduse algoritmide otsing; liikumine suurema või väiksem väärtuse suunas (Basawapatna et al., 2010; Ioannidou et al., 2011; Koh et al., 2014).

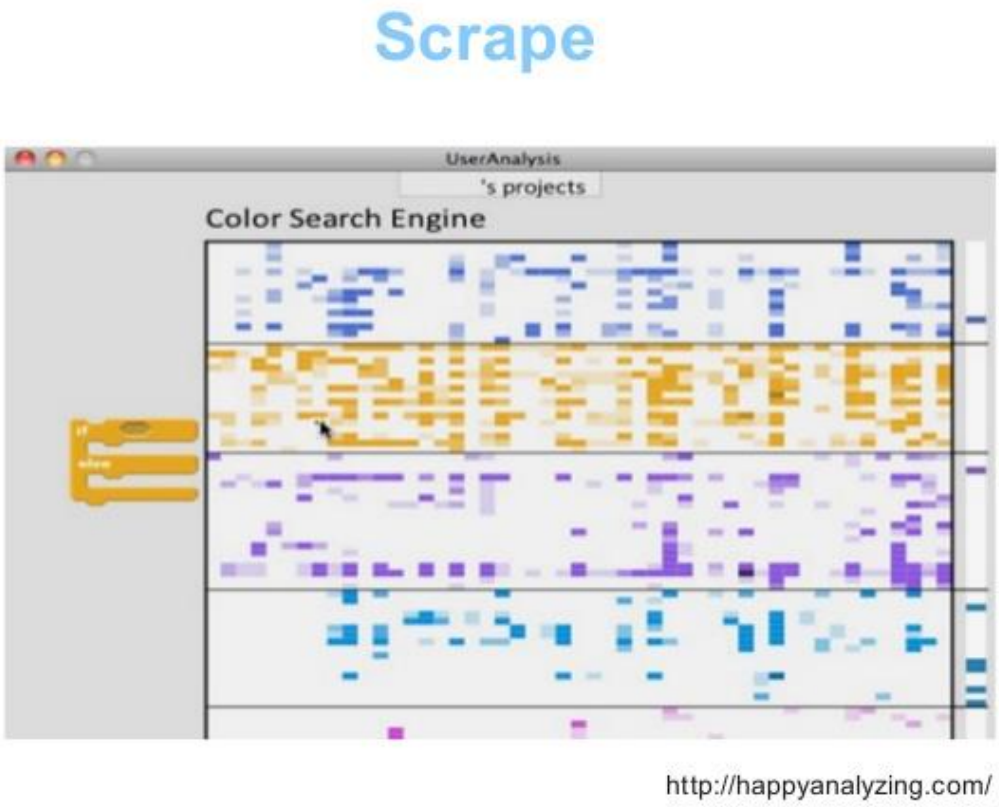
CTPA hindab nimetatud CT aspekte Scalable Game Design Arcade (SGDA) keskkonda laetud mängudes (Basawapatna et al., 2010; Ioannidou et al., 2011; Koh et al., 2014). SGDA keskkond koondab endas kasutaja loodud mänge ning näitena on kasutajatele toodud neli õpetusmängu. Kasutajal on võimalik enda mäng üles laadida ning teiste üles laetud mänge mängida. Lisaks on võimalik hinnata ning anda tagasisidet teiste mängudele. SGDA kasutaja mängu analüüs koosneb neljast osast: mängu ekraanipildist, CTPA graafikust (joonis 5), lähedaste või sarnaste mängude skoorist ning linkidest ja sarnasuse skoorist nelja õpetusmänguga (Basawapatna et al., 2010; Ioannidou et al., 2011; Koh et al., 2014).



Joonis 5. CTPA graafik toob välja CT aspektid kasutaja loodud mängus ning võrdleb mängu eelnevalt ette antud mänguga (Koh, 2015)

Projekti portfoolio analüüs.

Projekti portfoolio analüüs põhineb visuaalses keskkonnas Scratch kasutaja loodud kõikide projektide analüüsist. Brennan ja Resnick (2012) on selleks kasutanud New Jersey kolledži poolt loodud keskkonda Scrape. Projektide analüüs kuvatakse tabelina (joonis 6), milles iga veerg tähistab ühte projekti ning read tähistavad programmis kasutatud plokke, mida tumedam plokk on, seda rohkem on seda selles projektis kasutatud. Viimane veerg kuvab need plokid, mida ei ole mitte kunagi kasutatud. Portfoolio üheks eeliseks on see, et võimalik on näha õpilase kõiki avaldatud töid, millest võib näha, kas õpilase CT oskus on aja jooksul arenenud või mitte. Samas ei oska me sellise analüüsi korral öelda, et kas õpilane on kõik oma programmid ise loonud või mitte ning milline on tema tegelik arusaam programmeerimisest.



Joonis 6. Keskkonnas Scrape Scratch´i projektide analüüs (Happy Analyzing, 2008, viidatud Moreno-León, 2014 j)

Keskkonna Scrape veebipõhine versioon ei tööta, kuid Scrape tööriista on võimalik alla laadida ning sellega on võimalik analüüsida Scratch 1.0 versioonis loodud projekte (RiverSound Media, s.a.).

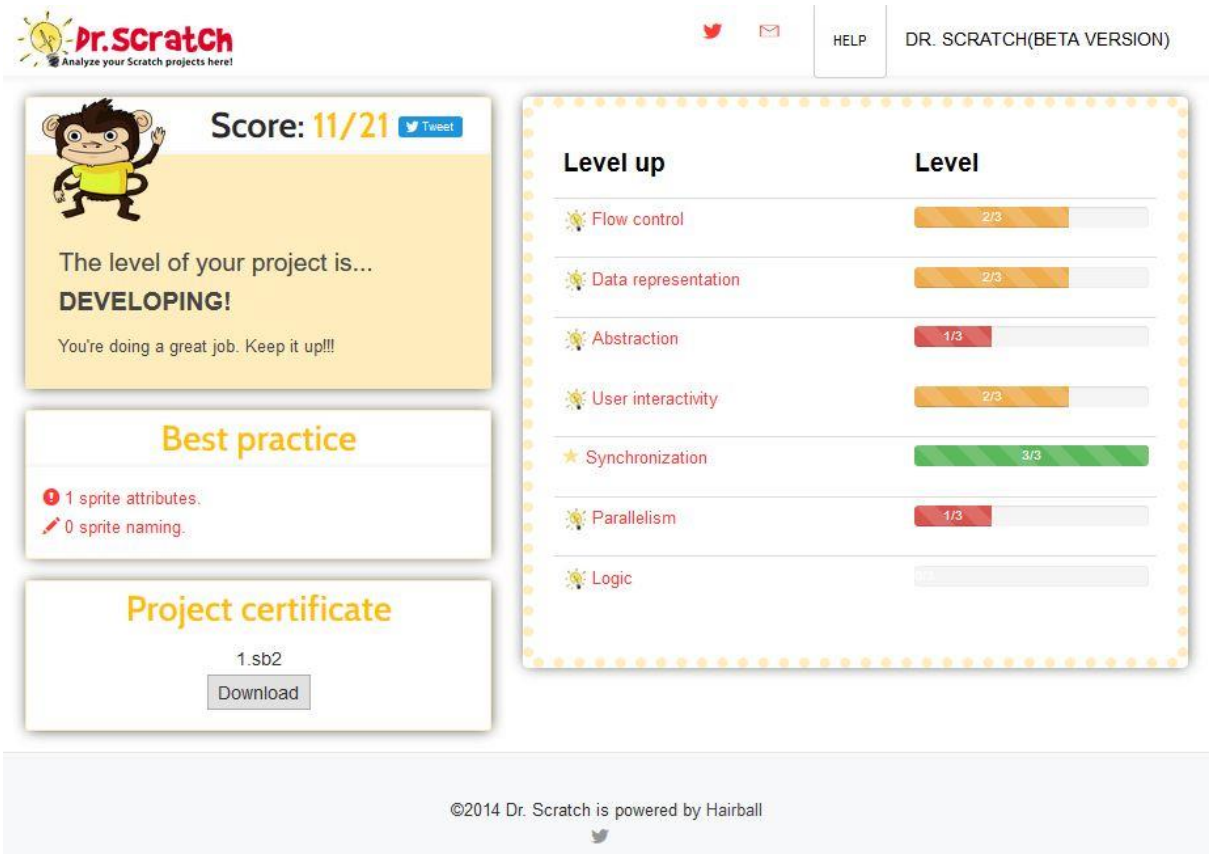
Hairball.

Toetudes arvutiteaduse mõistetele on loodud pistikprogramm Hairball arvutiteadus ja CT aspektide hindamiseks Scratch´i projektides (Boe et al., 2013). Pythoni juurde on loodud lisaklassid kasutades objektorienteeritud programmeerimisele omaseid funktsioone. Nendest klassidest on võimalik tuletada Hairball. Seega on Hairball Pythoni juurde loodud pistikprogramm, mille eesmärk on pakkuda õppijatele võimalust parandada ise enda programmeerimisoskust ning kiirendada Scratch´i projektide analüüsimise protsessi. Hairball analüüsib lähtekoodi ning tuvastab programmeerimisvead skriptis. Pistikprogramm analüüsib Scratch´i projektides nelja arvutiteaduse aspekti, nendeks on: lähtestamine, sünkroniseerimine, edastus ja vastuvõtmine, keerulised animatsioonid. Scratch´i projekti

hinnatakse nimetatud kategooriates selliselt, et Hairball annab kasutajale teada, kas tegemist on korrektse, tähenduslikult vale, vale või pooliku programmiga. Lähtestamise pistikprogramm alustab analüüsimist plokist „kui klõpsatakse rohelist lippu“, seejärel määrab programmi lähteandmed ning annab nende kohta tagasisidet. Sünkroniseerimise pistikprogramm otsib ja tuvastab programmis heli ja faili plokke ning määrab, kas need plokid on programmis õiges järjekorras. Edastus ja vastuvõtu plokk kontrollib edastus ja vastu võtu plokkide olemasolu. Keerulised animatsioonid on selline animatsioon, mis hõlmab kostüüme, liikumist, ajastamist ja tsükleid (Boe et al., 2013). Kuna Hairball on loodud Pythoni juurde pistikprogrammina, siis selle kasutamiseks on vaja omada programmeerimisalaseid teadmisi ja oskuseid, seega kellel oskused puuduvad, ei saa Hairballi kasutada (Moreno-León & Robles, 2015a, 2015b; Moreno-León, Robles, & Román González, 2015). Moreno-León ja Robles (2015a) on loonud Hairballil põhineva kasutajasõbralikuma tarkvara Dr. Scratch, mida tutvustatakse järgmises peatükis.

Dr. Scratch.

Scratch´i projektide hindamiseks on loodud vabavaraline tarkvara Dr. Scratch, mis põhineb Hairball programmil. Dr. Scratch´i eesmärk on analüüsida CT aspekte Scratch´i projektides. Sisestades veebitööriista Scratch´i projekti lingi või laadides projekt üles oma arvutist, on võimalik näha tagasisidet. Tagasiside koosneb neljast komponendist (Joonis 7): kogu punktisumma, hindamise seitsme kategooria punktid, soovitused ning analüüsitud projekt. Veebitööriist annab tagasisidet punktides. Punktide vahemik on 0-24 ning hindamine on seitsmes kategoorias. Igas kategoorias on võimalik saada kolm punkti (Dr. Scratch, s.a.; Moreno-León & Robles, 2015b).



Joonis 7. Ekraanipilt Dr. Scratch´i projekti analüüsi näidisest

Hindamise seitse kategooriat on abstraheerimine, loogiline mõtlemine, sünkroniseerimine, parallelism, voo haldamine, kasutaja interaktiivsus, andmete esitamine:

- abstraheerimine – üldisem ülevaate sellest, kuidas skriptid on koostatud, kas leidub kordusi ning millisel juhul oleks vajalik koostada ühe skripti asemel mitu.
- loogiline mõtlemine – kategooria hindab loogilise mõtlemise kasutamist. Koostatud projekt annab punkte, kui skriptide koostamisel on lähtutud erinevatest olukordadest. Näiteks kui sprait puudutab sinist värvi, siis sprait põrkab tagasi.
- sünkroniseerimine – kategooria hindab, kas loodud projekti skriptide tegevus toimub ajalises järjekorras. Näiteks, üks sprait ütleb teisele midagi, siis teine sprait peab kõige pealt ootama teatud aja, et vastata.
- parallelism – kategooria hindab mitme tegevuse samal ajal toimumist. Projektides on mitu erinevat spraiti ning nende spraitide tegevuste samal ajal toimumist hindabki.
- voo haldamine – kategooria hindab spraitide käitumist ning liikumist. See kategooria annab punkte ka liikumise korduste eest.
- kasutaja interaktiivsus – kategooria hindab interaktiivsete vahendite kasutamist projektis.

- andmete esitamine – kategooria annab ülevaate skripti iseloomustavate andmete olemasolust (Dr. Scratch, s.a.; Moreno-León & Robles, 2015a).

Iga kategooria juures on välja toodud punktisumma 0, 1, 2 või 3 ning selgitus, mida need punktid sisaldavad (Tabel 3). Dr. Scratch toob välja programmeerimise harjumused, mida peaks vältima. Tööriist otsib skripte, mis ei käivitu kunagi, kontrollib spraitide sõnumite vahetamise sünkroonsust, otsib skripte, mis korduvad ning spraite mille nimed ei ole muudetud. Kui programmis leidub halbu programmeerimis harjumusi, siis Dr. Scratch toob need välja ning juhib tähelepanu nende muutmiseks (Dr. Scratch, s.a.; Moreno-León & Robles, 2015b).

Tabel 3. CT oskuste taseme punktide jaotus Dr. Scratch´is (Hoover, Barnes, Fatehi, Moreno-León, Puttick, Tucker-Raymond, Harteveld, 2016; Moreno-León, Robles, Román González, 2015a; Moreno-León, Robles, Román González, 2015b; Moreno-León, Román-González, Harteveld, Robles, 2017)

CT aspekt	CT oskuste tase			
	Puudub 0 punkti	Algaja 1 punkt	Arenev 2 punkti	Ekspert 3 punkti
Abstraheerimine	Ei esine	Kasutab rohkem kui ühte spraiti ja skripti	Defineerib ise ploki	Kasutab kloonimist (*clone*)
Loogiline mõtlemine	Ei esine	Kasutab tingimust „kui…, siis…“ (*if*)	Kasutab tingimust „kui.., siis…, muul juhul“ (*if else*)	Kasutab loogikaoperatsioone
Sünkroniseerimine	Ei esine	Kasutab oote plokki (*wait*)	Kasutab plokki „ütle“ selleks, et peatada kõik, peatada programm või peatada spraidi programm	Kasutab plokke „oota kuni“, „kui taustaks saab …“, „teata ja oota“.
Parallelism	Ei esine	Kasutab plokki „kui klõpsatakse“ kahe skriptiga	Kaks skripti kasutavad klõpsamist	Kaks skripti saadavad sõnumi, video, muusika või vahetavad tausta
Voo haldamine	Ei esine	Plokid on asetatud järjestatult	Kasutab plokki „korda lõputult“	Kasutab plokki „korda kuni“
Kasutaja	Ei esine	Kasutab plokki	Kasutab	Kasutab

interaktiivsus		„kui klõpsatakse“	klõpsamist, spraidile klõpsamist, küsimist ja ootamist, hiirekursori plokke.	veebikaamerat või helisisendit.
Andmete esitamine	Ei esine	Muudab spraitide omadusi	Kasutab muutujaid	Kasutab loendeid

Dr. Scratch soovib anda Scratch´i kasutajatele tagasisidet ning läbi selle julgustada kasutajaid parandama oma programmeerimise oskuseid (Moreno-León & Robles, 2015a, 2015b). Dr. Scratch´i puuduseks on see, et keskkond on ingliskeelne ning õpetajal ei ole võimalik näha kõikide õpilaste projekte koos ehk puudub ülevaade kogu klassist. Samas on võimalik Dr. Scratch´i abil õpilasel märgata, millised võimalused on projekti edasiarendamiseks ning millistele osadele peaks lisatähelepanu suunama või muutma projekti. Scratch´is loodavaid projekte on võimalik arendada silmas pidades vaid kindlaid CT aspekte, nii et Dr. Scratch abil saaks hinnata CT aspekti kasutamist, see võimaldab õpetajal anda õpilastele ülesandeks saavutada teatud aspektis kindel tase.

Selles töös leitud Scratch´i projektide CT hindamise vahenditest võetakse Scratch´i projektide analüüsimiseks kasutusele viimasena tutvustatud Dr. Scratch, sest tarkvara võimaldab nii õppijal kui õpetajal analüüsida CT aspekte, lisaks annab tarkavara informatsiooni töö edasiarendamiseks ning aspektide täiendamiseks. Dr. Scratch hindab projekte seitsmes CT kategoorias, sellest tulenevalt keskendub töö teine osa Dr. Scratch´is hinnatavatele CT aspektide omandamisele ning arendamisele. Lähtudes töö eesmärgist on uurimisküsimused:

- Milline on CT oskuste tase Scratch´i projektide näitel?
- Mil määral toetab keskkonnas Scratch mängu loomine õpilaste CT oskusi?

Metoodika

Peatükis esitatakse uurimuse valimi moodustamise, korraldamise ning analüüsimiseks kasutatud vahendite kirjeldus.

Valim

Töö valim moodustati mugavusvalimi põhimõttel. Mugavusvalimi korral küsitletakse inimesi, kes on uurija jaoks kergesti kättesaadavad. Mugavusvalim moodustub eelkõige

kättesaadavuse abil (Rämmer, 2014). Mugavusvalimi põhimõttel moodustati käesoleva töö valim lähtudes kriteeriumist, et Scratch´i projektide koostajad osalevad 2016. aasta oktoobris Code Weeki nädala raames Scratch´i töötoas. Töötuba oli mõeldud II kooliastme entusiastlikutele noortele, kes pole varasemalt Scratch´iga kokku puutunud ning osalejad olid Tartust või Tartu lähedalt. Uurimuses osales 19 osalejat.

Mõõtevahend

Uurimuse mõõtevahendiks oli hariduslikus programmeerimiskeeles Scratch loodud projektid. Projekte hinnati uurimuses leitud CT aspektide esinemist Dr. Scratch´i abil. Dr. Scratch hindas projekte CT seitsmes aspektis. Projektides hinnati CT kasutamist bakalaureusetöös leitud parameetrite järgi. Hinnatud parameetriteks olid: abstraheerimine, loogiline mõtlemine, sünkroniseerimine, parallelism, voo haldamine, kasutaja interaktiivsus, andmete esitamine. Animatsiooni ja mängu hinnati eraldi. Lisaks võrreldi CT aspektide ilmnemist erinevates osades.

Protseduur

Töö andmed tulenevad Scratch´i projektidest, mis koostati Code Weeki nädala raames 2016. aasta sügisel. Töötuba juhendasid Tauno Palts ja Getriin Kokk. Töötoas loodi projektid töölehtede järgi. Töölehed koostati arvestades CT erinevaid etappe. Töötuba viidi läbi kahes osas. Esimese osa eesmärk oli luua animatsioon ning teise osa eesmärk oli luua mäng. Igale töötoas osalejale jagati tööleht Scratch´i programmi loomiseks. Anonüümsuse säilitamiseks salvestati loodud Scratch´i projektid arvuti töölauale selliselt, et faili pealkiri sisaldas töölehe osa numbrit ning arvuti numbrit. Arvutid olid eelnevalt nummerdatud. Peale töötuba koguti projektid kokku. Töö andmestik koosneb 38st Scratch´i projektist, mille on loonud 19 töötoas osalejat, igaüks koostas kaks projekti.

Andmeanalüüsi teostamiseks kasutati Microsoft Excel 2016 vahendit. Uurimistulemuste kirjeldamiseks kasutati kirjeldavat statistikat, milleks olid projektide miinimum ja maksimum punktisummad, sagedusjaotus, mediaan, mood, aritmeetiline keskmine, standardhälve. Animatsiooni ja mängu projektide võrdlemiseks kasutati sõltuvate valimite t-testi (*t-Test: Paired Teo Sample for Means*).

Töölehed

Töötoas loodi projektid töölehtede järgi (Lisa 1 ja Lisa 2), mis olid koostatud Scratch´i veebilehel (Scratch, s.a.) esiletõstetud projektide, Tartu Ülikooli arvutiteaduse instituudi

Scratch´i materjalide (Scratchi materjalid, s.a.) ning ProgeTiigri Scratch´i videomaterjalide alusel (Scratchi materjalid, s.a.).

Esimese osa eesmärk oli luua animatsioon. Animatsioon sisaldas uue projekti loomist, spraitide liikumist ning kahe spraidi omavahelist rääkimist. Teise osa eesmärk oli luua mäng. Mängu loomine oli jaotatud kaheks osaks. Esimese osa eesmärk oli koostada spraidile 1 nooleklahvidega liikumise programm, samale spraidile koostüümi vahetus liikumisel, spraidile 2 liikumine suvalisse kohta hüppamisega ning punktisüsteem, mille korral sprait 1 kogub sprait 2 puudutamisel punkte.

Teise osa eesmärk oli lisada sprait, kes annab miinuspunkte ning täiustada programmi selliselt, et mängu alguses oleks spraidid alati sama koha peal. Lisaks punktisüsteem nullimine mängu alguses ning mängu peatumine, kui sprait 1 on kogunud ettemääratud koguses punkte. Töölehtedel on toodud juhendid selgitustega ning lõigetega loodavast programmist. Programmilõiked on välja toodud selleks, et plokkide värvi ning teemade järgi saaks töölehe täitja ise vastavad plokid plokkide alast üles leida. Esimeses osas peale igat ploki lisamist skripti ning teises peale mitme skripti koostamist on soovitatud katsetada loodud programmi. Töölehtedel on osa “Proffidele”, kus on iseseisvaks lahendamiseks ülesandeid.

Tulemused

Projekte hinnati Dr. Scratch´i abil seitsmes kategoorias, milleks on abstraheerimine, loogiline mõtlemine, sünkroniseerimine, parallelism, voo haldamine, kasutaja interaktiivsus, andmete esitamine. Dr. Scratch hindas igat kategooriat punktidega 0-3. Peatükkis esitatakse uurimuse tulemused.

Animatsioon

Osalejate poolt loodi 19 animatsiooni projekti. Dr. Scratch´i poolt hinnatud CT aspektide punktisummad animatsiooni projektides on järgnevad: viis projekti hinnati punktisummaga 7, kolmteist projekti hinnati punktisummaga 6 ning üks projekt hinnati punktisummaga 3. Seega kõige sagedamini esinev punktisumma oli 6, mis tähendab, et enim koostati projekte, mis Dr. Scartch hindas punktisummaga 6. Kõigi projektide keskmine punktisumma oli 6.1 (*SD* = 0,88).

Kolmele punktile hinnatud projektis esinesid voo haldamine, andemete esitamine ning kasutaja interaktiivsus. Kuuele punktile hinnatud projektid said kõigis aspektides välja

arvatud loogilises mõtlemises ühe punkti, aspekt loogiline mõtlemine punkti ei andnud. Seitse punkti saanud projektid hinnati sarnaselt kuus punkti saanud projektidele aspektides voo haldamine, andmete esitamine, abstraheerimine, kasutaja interaktiivsus, sünkroniseerimine ja parallelism vähemalt ühe punktiga. Kolmes seitsmele punktile hinnatud projektidest saadi voo haldamise eest võrreldes kuuele punktile hinnatud projektides ühe punkti asemel kaks. Üks seitsmele punktidele hinnatud projekt sai sünkroniseerimise eest kaks punkti ning üks projekt sai loogilise mõtlemise eest ühe punkti, samas kui kõik teised projektid ei saanud selle aspekti eest mitte ühtegi punkti.

Voo haldamise, andmete esitamise, abstraheerimise, kasutaja interaktiivsuse, sünkroonimise ja parallelismi kategooriates on kõige sagedasemalt saadud punktisumma üks. Dr. Scratch leiab, et animatsioonis CT aspekti loogiline mõtlemine ei esine, välja arvatud ühe projekti näitel. Dr. Scratch hindas CT aspekte animatsioonis punktidega null, üks või kaks, seega vähim saadud punktisumma oli null ning suurim kaks. Ühtki projekti ei hinnatud üheski aspektis kolme punktiga. Töölehe täpsel täitmisel hindaks Dr. Scratch animatsiooni projekti kuue punkti kahekümne ühest, kõigis kategooriates peale loogilise mõtlemise ühe punktiga.

Mäng

Osalejate poolt loodi 19 mängu projekti. Dr. Scratch´i poolt hinnatud CT aspektide punktisummad mängu projektides on järgnevad: üks projekt hinnati punktisummaga 6, üks projekt hinnati punktisummaga 8, neli projekti hinnati punktisummaga 10, kaheksa projekti hinnati punktisummaga 11, kolm projekti hinnati punktisummaga 12, üks projekt hinnati punktisummaga 13 ja üks projekt hinnati punktisummaga 14. Seega kõige sagedamini esinev punktisumma oli 11, mis tähendab, et enim koostati projekte, mis Dr. Scartch hindas punktisummaga 11. Kõigi mängu projektide keskmine punktisumma oli 10,7 (*SD* = 1,72).

Kuuele punktile hinnatud projekt sai voo haldamise eest kaks punkti, andmete esitamise, abstraheerimise, kasutaja interaktiivsuse ja parallelismi eest ühe punkti. Kaheksale punktidele hinnatud projekt sai erinevalt kuuele punktile hinnatud projektist kaks punkti andmete esitamises ja kasutaja interaktiivsuses, muudes aspektides saadud punktid olid kuuele punktile hinnatud projektiga samad. Kümme punkti saanud projektidest kahel olid samad punktid kõigis kategooriates, kaks punkti voo haldamises, andmete esitamises, kasutaja interaktiivsuses ja sünkroniseerimises, üks punkt abstraheerimises ja parallelismis. Kolmas projekt erines kahest eelnevast sünkroniseerimises saadud kolme punktiga ning parallelismis null punktiga. Neljas projekt sai sünkroniseerimises, loogilises mõtlemises, abstraheerimises, ja parallelismis ühe punkti ning aspektides voo haldamine, andmete esitamine, kasutaja

interaktiivsus kaks punkti. Kõik üksteist punkti saanud projektid olid Dr. Scratch´i poolt hinnatud kõigis kategooriates samamoodi. Kaks punkti said projektid voo haldamises, andmete esitamises, kasutaja interaktiivsuses, ühe punkti abstraheerimises ning parallelismis ning kolm punkti sünkroniseerimises. Kaksteist punkti saanud projektid said samuti kõikides kategooriates samad punktid, erinevus üksteist punkti saanud projektiga oli parallelismis saadud kaks punkti. Kolmteist punkti saanud projekt erines kaksteist punkti saanud projektidest vaid loogilises mõtlemises saadud punkti võrra ning neliteist punkti saanud projektis oli hinnatud parallelismi kolme punktiga. Töölehe täpsel täitmisel oleksid mängust saadavad punktid järgnevad: abstraheerimine üks punkt , loogiline mõtlemine null punkti, sünkroniseerimine kolm punkti, parallelism üks punkt, voo haldamine kaks punkti, kasutaja interaktiivsus kaks punkti, andmete esitamine kaks punkti.

Dr. Scratch abil Scratch´i projekte hinnates leiti, et mängus CT aspekt loogiline mõtlemine esines kolmes projektis. Aspektid abstraheerimine ning parallelism hindas Dr. Scratch kõige sagedasemalt ühe punktiga ning CT aspekti voo haldamine kõige sagedamini kahe punktiga. Sama punktisumma esines kõige sagedamine ka aspektide andmete esitamine ja kasutaja interaktiivsus korral. Kõige sagedamini saadi võimalikest punktidest kolm punkti aspektis sünkrooniseerimine. Selgus, et Dr. Scratch hindas mängus esinevaid CT aspekte kõigi võimalike punktidega: null, üks, kaks ja kolm. Seega oli projekte, kus osasid CT aspekte ei leidunud, samas leidus projekte, milles esines mingi CT aspekt kolme punkti vääriliselt.

Animatsiooni ja mängu projektide võrdlevad tulemused

Andmeanalüüsi tulemustest võib järeldada, et animatsiooni ja mängu projektid on statistiliselt oluliselt erinevad (t = -13,35; p < 0,001). Mängu projektide tulemused on statistiliselt oluliselt paremad, kui animatsiooni projektide tulemused (t = -13,35; p < 0,001).

Voo haldamine

Dr. Scratch´i hinnang animatsiooni ja mängu projektides CT aspektile voo haldamine on välja toodud tabelis 4. Andmeanalüüsist selgus, et CT aspekti voo haldamine tulemus on statistiliselt oluline mängu ja animatsiooni projektides (t = -9,80; p < 0,001) ning mängu projektides on CT aspekt voo haldamine statistiliselt oluliselt parem, kui animatsiooni projektides (p < 0,001). Kolmes projektis sai aspekt voo haldamine nii mängus kui ka animatsioonis samad punktid, 16 projekti hinnati samas aspektis animatsioonis ühe punktiga ning mängus kahe punktiga.

Tabel 4. CT aspekti voo haldamine hinnang animatsioonis ja mängus Dr. Scratch´i järgi

	M	*SD*	*Min*	*Max*	*t*	*p*
Animatsioon	1,16	0,37	1	2	-9,80	<0,001
Mäng	2	0	2	2		

Märkus. M = aritmeetiline keskmine; *SD* = standardhälve; *Min* = minimaalne punktisumma; *Max* = maksimaalne punktisumma; *t* = t-test; *p* = olulisustõenäosus. Skaala tähendused: 0 – ei esine; 1 – plokid on järjestatud; 2 – kasutab plokki „korda lõputult“; 3 – kasutab plokki „korda kuni“

Andmeanalüüsist selgub, et aspekt voo haldamine ilmneb nii animatsiooni kui ka mängu projektides.

Andmete esitamine

Dr. Scratch´i hinnang animatsiooni ja mängu projektides CT aspektile andmete esitamine on välja toodud tabelis 5. CT aspekti andmete esitamine tulemus on statistiliselt oluline mängu ja animatsiooni projektides (t = -18; $p < 0,001$) ning mängu projektides on CT aspekt andmete esitamine statistiliselt oluliselt parem, kui animatsiooni projektides ($p < 0,001$). Kui üht projekti aspektis andmete esitamine hinnati nii mängus kui ka animatsioonis samade punktidega, siis 18 projekti selles aspektis hinnati animatsioonis ühe punktiga ning mängus kahe punktiga.

Tabel 5. CT aspekti andmete esitamine hinnang animatsioonis ja mängus Dr. Scratch´i järgi

	M	*SD*	*Min*	*Max*	*t*	*p*
Animatsioon	1	0	1	1	-18	<0,001
Mäng	1,95	0,23	1	2		

Märkus. M = aritmeetiline keskmine; *SD* = standardhälve; *Min* = minimaalne punktisumma; *Max* = maksimaalne punktisumma; *t* = t-test; *p* = olulisustõenäosus. Skaala tähendused: 0 – ei esine; 1 – muudab spraitide omadusi; 2 – kasutab muutujaid; 3 – kasutab loendeid

Aspekti andmete esitamine animatsiooni ja mängu projektide tulemused erinesid ning aspekt leidus kõigis projektides.

Abstraheerimine

Dr. Scratch´i hinnang animatsiooni ja mängu projektides CT aspektile abstraheerimine on välja toodud tabelis 6. Animatsiooni ja mängu CT aspekti abstraheerimine hindamisel olulist erinevust ei leitud ning animatsiooni ja mängu tulemused olid samad (t = -1; $p > 0,05$). Dr. Scratch hindas 18 projekti aspektis abstraheerimine ühe punktiga ning 19 mängu projekti ühe punktiga, ainult üks projekt ei saanud animatsioonis abstraheerimise eest punkti.

Tabel 6. CT aspekti abstraheerimine hinnang animatsioonis ja mängus Dr. Scratch´i järgi

	M	*SD*	*Min*	*Max*	*t*	*p*
Animatsioon	0,95	0,22	0	1	-1	> 0,05
Mäng	1	0	1	1		

Märkus. M = aritmeetiline keskmine; *SD* = standardhälve; *Min* = minimaalne punktisumma; *Max* = maksimaalne punktisumma; *t* = t-test; *p* = olulisustõenäosus. Skaala tähendused: 0 – ei esine; 1 – kasutab rohkem kui ühte spraiti ja skripti; 2 –defineerib ise ploki; 3 – kasutab kloonimist

Mõlemas projektis ilmnes aspekt, kuid animatsiooni ning mängu tulemused olid samad.

Kasutaja interaktiivsus

Dr. Scratch´i hinnang animatsiooni ja mängu projektides CT aspektile kasutaja interaktiivsus on välja toodud tabelis 7. CT aspekti kasutaja interaktiivsus tulemus on statistiliselt oluline mängu ja animatsiooni projektides (t = -18; $p < 0,001$) ning mängu projektides on CT aspekt kasutaja interaktiivsus statistiliselt oluliselt parem, kui animatsiooni projektides ($p < 0,001$). Kui üht projekti aspektis kasutaja interaktiivsus hinnati nii mängus kui ka projektis samade punktidega, siis 18 projekti hinnati animatsioonis ühe punktiga ning mängus kahe punktiga.

Tabel 7. CT aspekti kasutaja interaktiivsus hinnang animatsioonis ja mängus Dr. Scratch´i järgi

	M	*SD*	*Min*	*Max*	*t*	*p*
Animatsioon	1	0	1	1	-18	<0,001
Mäng	1,95	0,23	1	2		

Märkus. M = aritmeetiline keskmine; *SD* = standardhälve; *Min* = minimaalne punktisumma; *Max* = maksimaalne punktisumma; *t* = t-test; *p* = olulisustõenäosus. Skaala tähendused: 0 – ei esine; 1 – kasutab plokki „kui klõpsatakse“; 2 – kasutab klõpsamist, spraidile klõpsamist, küsimist ja ootamist, hiirekursori plokke.; 3 – kasutab veebikaamerat või helisisendit

Animatsiooni ja mängu projektide tulemuste hajuvus on väike (animatsioon *SD* = 0, mäng *SD* = 0,23). See näitab, et kõik animatsiooni projektid hinnati aspektis sünkroniseerimine samade punktidega ning mängu projektides leidusid mõned projektid, mis hinnati teistest erinevalt.

Sünkroniseerimine

Dr. Scratch´i hinnang animatsiooni ja mängu projektides CT aspektile sünkroniseerimine on välja toodud tabelis 8. CT aspekti sünkroniseerimine tulemus on statistiliselt oluline mängu ja animatsiooni projektides (t = -8,61; $p < 0,001$) ning mängu projektides on CT aspekt sünkroniseerimine statistiliselt oluliselt parem, kui animatsiooni projektides ($p < 0,001$). Suurim CT aspektide punktide erinevus animatsiooni ja mängu vahel ilmneb

sünkroniseerimises. Animatsiooni projektidest vähemalt 18 projekti hinnati aspekti sünkroniseerimine tulemusega üks punkt. Mängus hinnati 14 projekti kolme punktiga.

Tabel 8. CT aspekti sünkroniseerimine hinnang animatsioonis ja mängus Dr. Scratch´i järgi

	M	*SD*	*Min*	*Max*	*t*	*p*
Animatsioon	0,95	0,40	0	2	-8,61	<0,001
Mäng	2,47	1,02	0	3		

Märkus. M = aritmeetiline keskmine; *SD* = standardhälve; *Min* = minimaalne punktisumma; *Max* = maksimaalne punktisumma; *t* = t-test; *p* = olulisustõenäosus. Skaala tähendused: 0 – ei esine; 1 – kasutab oote plokki; 2 – kasutab plokki „ütle“ selleks, et peatada kõik, peatada programm või peatada spraidi programm.; 3 – kasutab plokke „oota kuni“, „kui taustaks saab …“, „teata ja oota“

Aspekti sünkroniseerimine mängu projektide punktide hajuvus (*SD* = 1,02) on suurem kui animatsiooni projektides (*SD* = 0,40), seega on mängu projekte hinnatud erinevate võimalike punktidega.

Parallelism

Dr. Scratch´i hinnang animatsiooni ja mängu projektides CT aspektile parallelism on välja toodud tabelis 9. CT aspekti parallelism tulemus ei ole statistiliselt oluline mängu ja animatsiooni projektides (t = -2,05; $p > 0,05$), kuid mängu tulemused on paremad animatsiooni projektide tulemustest ($p < 0,05$). Dr. Scratch hindas 12 projektis aspekti parallelism mõlemas projektis samade punktidega, 11 juhul ühe punktiga ning ühel juhul null punktiga. Viies projektis hinnati aspekt parallelism mängus ühe punkti võrra suuremaks kui animatsioonis. Neljas projektis hinnati nimetatud aspekti animatsioonis ühe punktiga ning mängus kahe punktiga ning ühes projektis animatsioonis null punktiga ning mängus 1 punktiga. Lisaks ühes projektis hinnatakse aspekt parallelism animatsiooni osas ühe punktiga ning mängu osas kolme punktiga. Animatsiooni ja mängu vahel leidub üks projekt, milles animatsioonis hinnati aspekti parallelism ühe punktiga ning mängus null punktiga.

Tabel 9. CT aspekti parallelism hinnang animatsioonis ja mängus Dr. Scratch´i järgi

	M	*SD*	*Min*	*Max*	*t*	*p*
Animatsioon	0,89	0,32	0	1	-2,05	>0,05
Mäng	1,21	0,71	0	3		

Märkus. M = aritmeetiline keskmine; *SD* = standardhälve; *Min* = minimaalne punktisumma; *Max* = maksimaalne punktisumma; *t* = t-test; *p* = olulisustõenäosus. Skaala tähendused: 0 – ei esine; 1 – kasutab plokki „kui klõpsatakse“ kahe skriptiga; 2 – kaks skripti kasutavad klõpsamist.; 3 – kaks skripti saadavad sõnumi, video, muusika või vahetavad tausta

Parallelism ilmneb animatsiooni ja mängu projektides, kuigi tulemused ei erine oluliselt.

Loogiline mõtlemine

Dr. Scratch´i hinnang animatsiooni ja mängu projektides CT aspektile loogiline mõtlemine on välja toodud tabelis 10. CT aspekti loogiline mõtlemine hindamisel olulist erinevust ei leitud ning animatsiooni ja mängu tulemused olid samad (t = 1,46; p > 0,05). Aspekt loogiline mõtlemine hinnati 16 projektis nii animatsiooni osas kui ka mängu osas null punktiga. Kolmes projektis aspekti loogiline mõtlemine punktid muutuvad animatsiooni ja mängu võrdluses. Kaks projekti hinnati aspektis loogiline mõtlemine animatsiooni osas null punktiga ning mängu osas ühe punktiga. Ühe osaleja poolt loodud projektide punktid jäävad samaks, mõlemat projekti hinnatakse ühe punktiga.

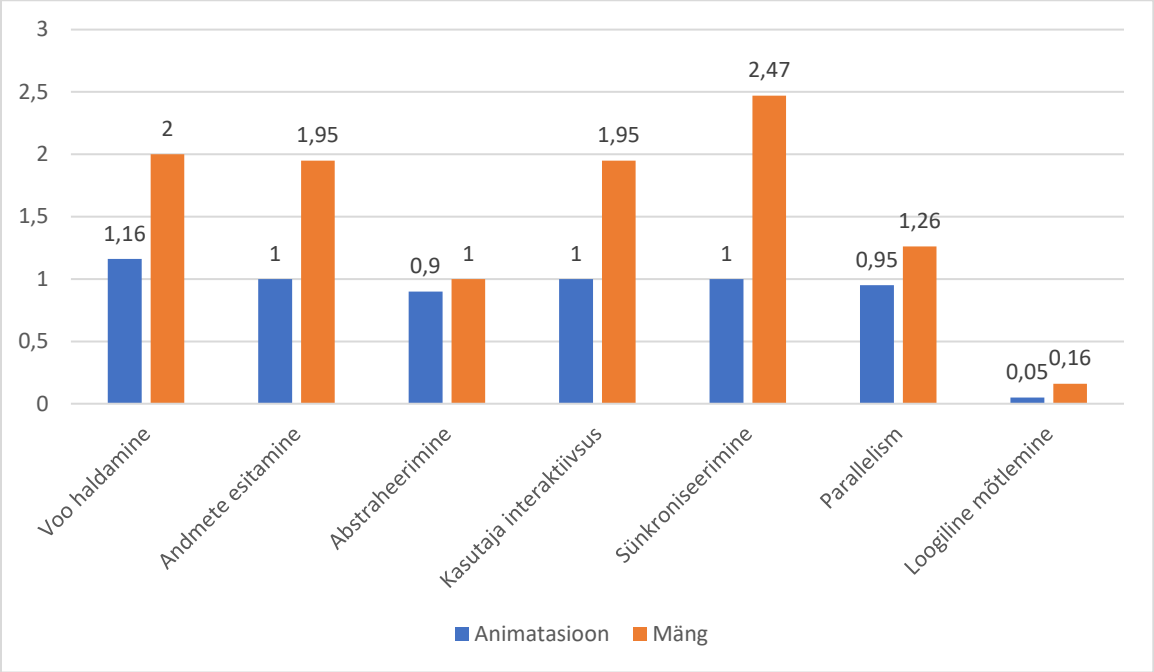
Tabel 10. CT aspekti loogiline mõtlemine hinnang animatsioonis ja mängus Dr. Scratch´i järgi

	M	*SD*	*Min*	*Max*	*t*	*p*
Animatsioon	0,05	0,23	0	1	1,46	>0,05
Mäng	0,16	0,37	0	1		

Märkus. M = aritmeetiline keskmine; *SD* = standardhälve; *Min* = minimaalne punktisumma; *Max* = maksimaalne punktisumma; *t* = t-test; *p* = olulisustõenäosus. Skaala tähendused: 0 – ei esine; 1 – kasutab tingimust „kui…, siis…“; 2 – kasutab tingimust „kui.., siis…, muul juhul“ (*if else*); 3 – kasutab loogikaoperatsioone

Analüüsist selgub, et aspekt loogiline mõtlemine on hinnatud ühe punktiga neljas projektis, millest kolm olid mängu projektid.

Aspektide võrdlus animatsioonis ja mängus



Joonis 8. CT aspektide keskmiste punktide võrdlus animatsiooni ja mängu projektides

Tulemuste analüüsist selgus (joonis 8), et erinevus aspektides voo haldamine, andmete esitamine, kasutaja interaktiivsus ja sünkroniseerimine on märgatav. Suurim erinevus CT aspektide tulemustes animatsiooni ja mängu projektides on aspektis sünkroniseerimine. Aspekti sünkroniseerimine aritmeetiline keskmine animatsiooni projektides on 1 (*SD* = 0,08) ning mängu projektides on 2,47 punkti (*SD* = 1,02). Kui nimetatud nelja aspekti animatsiooni ja mängu projektide tulemuse erinevus on märgatav, siis aspekte abstraheerimine ning parallelism toetati töölehtedega, kuid kahe projekti tulemuste erinevust ei ole. Töölehtedega ei toetatud CT aspekti loogiline mõtlemine.

Arutelu

Bakalaureusetöö eesmärgiks oli CT oskuste hindamine Scratch´i projektide näitel. Töös anti ülevaade kirjanduses leiduvatest CT aspektidest, erinevate autorite püüdlusest defineerida mõistet CT ning toodi välja võimalikud instrumendid hindamaks CT aspektide olemasolu Scratch´i projektides ja nende instrumentidega hinnatavad CT aspektid. Eesmärgi täitmiseks koostati kaks töölehte, mis arvestasid CT erinevaid aspekte (Lisa 1., Lisa 2.). Töötoas koostas osaleja töölehtede järgi kaks projekti, animatsiooni ja mängu. CT aspektide olemasolu ja taset hinnati vabavaralise tarkvaraga Dr. Scratch. Arutelus antakse vastused töös püstitatud uurimisküsimustele ja võrreldakse tulemusi teooriaga. Peatüki lõpus tuuakse välja töö

praktiline väärtus ning töö piirangud.

Uurimuses hinnati Scratch´i projektides CT aspekte tarkvaraga Dr. Scratch. Sellest tulenevalt hinnati õpilaste CT oskuste taset tarkvara Dr. Scratch poolt määratud seitsmes aspektis, milleks olid abstraheerimine, loogiline mõtlemine, sünkroniseerimine, parallelism, voo haldamine, kasutaja interaktiivsus, andmete esitamine (Dr.Scratch, s.a.). Dr. Scratch´i hinnatavad seitse aspekti esinevad ka Brennan ja Resnick (2012) loodud CT aspektide kontseptsioonis, mis hõlmab endas aspekte, mida saab arendada programmeerimisega. Aspektide voo haldamine, andmete esitamine, kasutaja interaktiivsus ja sünkroniseerimine tulemused animatsiooni ja mängu projektides olid statistiliselt olulised ning nendes aspektides mängus saadud punktid olid oluliselt paremad animatsioonis saadud punktidest. Samas kui kahes aspektis abstraheerimine ja parallelism animatsiooni ja mängu projektide punktisummad ei erinenud. Kuigi animatsiooni ja mängu projektides aspektis parallelism statistilist olulisust ei leitud, siis olid mängu projektid hinnatud nimetatud aspektis paremaks kui animatsioonis. Tulemusest võib järeldada, et koostatud töölehed arendasid CT aspekte voo haldamine, andmete esitamine, kasutaja interaktiivsus, sünkroniseerimine, abstraheerimine ja parallelism. Nelja aspekti, voo haldamine, andmete esitamine, kasutaja interaktiivsus, sünkroniseerimine, mida lisaks toetamisele ka arendati, on veel võimalik arendada selliselt, et lisada projektidesse plokid „korda kuni“, „oota kuni“, „kui taustaks saab…“, „teata ja oota“ kasutamise, veebikaamera või helisisendi, loendite kasutamise. Näiteks animatsiooni projekti on võimalik lisada plokk „korda kuni puudutab spraiti“, siis tegevus toimub kuni üks sprait puudutab teist ning mängu projekti plokk „korda kuni punktid = 3“, selle juhul tegevus toimub kuni mängus on saadud kolm punkti. Selliselt töölehti täiendades hindaks Dr. Scratch projektides aspekte voo haldamine, andmete esitamine, sünkroniseerimine ja kasutaja interaktiivsus 3 punktiga (Dr.Scratch, s.a; Moreno-León & Robles, 2015a). Aspekti abstraheerimine on võimalik arendada, kui lisada projektidesse defineerimise plokk või kasutada kloonimist. Aspekti parallelism arendamiseks on võimalik lisada projektidesse „kui klõpsatakse“ plokk, kahe skripti kasutamine klõpsamisel või kasutada sõnumite saatmist, muusika ja video mängimist, tausta vahetamist. Näiteks lisada animatsiooni projektidesse plokk „järgmine taust“, mis vahetab olemasoleva tausta järgmise vastu või kasutada kahes skriptis plokki „kui vajutatakse klahvi tühik“, mis tühiku klahvi vajutamisel annab ette, mida sprait teeb. Näiteks tühiku klahvi klõpsamisel ilmub ekraanile kiri. Samad plokid on võimalik lisada ka mängu projektidesse. CT aspekti loogiline mõtlemine ei toetatud ühegi koostatud töölehega. Aspekti loogiline mõtlemine on võimalik toetada ja arendada, kui lisada tingimused „kui…, siis…, muul juhul“ ning kasutada loogikaoperatsioone. Näiteks on

võimalik mängu projekti lisada plokk „kui puudutad ääred ja elud = 0, siis ütle mäng on läbi ja peata kõik“ ehk lisatud plokk lõpetab mängu ning teatab, et mäng on läbi kui spraidil on ei ole enam ühtegi elu ning ta on puudutanud ekraani servasid. Plokkide lisamise järel hindab Dr. Scratch aspekte 2 või 3 punktiga (Dr.Scratch, s.a; Moreno-León & Robles, 2015a).

Uurimuse piirangud seisnevad selles, et analüüsitud projektid arv oli väike, ulatuslikema järelduste tegemiseks oleksid vajalikud põhjalikumad uurimused suurema valimiga. Töös kasutati vaid individuaalselt animatsiooni ja mängu loomise töölehti, kuid võimalusi laienemiseks on mitmeid. Näiteks on võimalik uurida paaris- ja rühmatööd, pikemaajaliste valikuliste projektide koostamist, Scratch´i käed-külge robootikavahenditega sidumist, mõne teise sarnase haridusliku programmeerimiskeelega tegelemist ja võrdlemist.

CT oskuse arendamiseks ja õppimiseks on vajalik luua õppematerjale ning leida taseme hindamiseks vastavaid instrumente. Sellest vajadusest lähtudes on töö raames analüüsitud, mil määral toetab Scratch´i projektide loomine CT oskusi. Koostatud töölehti on võimalik muuta ning täiustada vastavalt aspektidele, mida soovitakse arendada. Töö täiendab CT ning selle hindamise alaseid uurimusi Eestis.

Kokkuvõte

Algoritmide rakendamise hindamine probleemide lahendamisel Scratch´i projektide näitel.

Bakalaureusetöö eesmärgiks on CT oskuste hindamine Scratch´i projektide näitel. Töö uurimisprobleem seisneb selles, et puudub praktiline näide CT oskuste taseme hindamiseks ning vajalikest instrumentidest, millega kaardistada õpilaste CT taset ning määrata, mil määral arendab Scratch´i projektide loomine CT oskusi. Töös anti ülevaade kirjanduses leiduvatest CT aspektidest, erinevate autorite püüdlusest defineerida mõistet CT ning toodi välja võimalikud instrumendid Scratch´i projektide hindamiseks ja nende instrumentidega hinnatavad CT aspektid.

Uurimuse andmed tulenevad Scratch´i projektidest, mis loodi Code Weeki nädala raames Scratch´i töötoas. Eesmärgi täitmiseks koostati kaks töölehte, mis arvestasid CT erinevaid aspekte. Töötoas koostas osaleja töölehtede järgi kaks projekt, animatsiooni ja mängu. Valimisse kuulus 19 osalejat ning nende poolt loodud 38 projekti. CT aspektide olemasolu ja taset hinnati vabavaralise tarkvaraga Dr. Scratch, mis hindas CT seitset aspekti, nendeks olid abstraheerimine, loogiline mõtlemine, sünkroniseerimine, parallelism, voo haldamine, kasutaja interaktiivsus, andmete esitamine. Andmete analüüsimisel kasutati

kvantitatiivset analüüsi.

Uurimuses hinnatud aspektide voo haldamine, andmete esitamine, kasutaja interaktiivsus ja sünkroniseerimine punktide erinevus animatsiooni ja mängu projektides oli märgatav. Nimetatud aspektide tulemused animatsiooni ja mängu projektides olid statistiliselt olulised ning nendes aspektides mängus saadud punktid olid oluliselt paremad animatsioonis saadud punktidest. Aspekt abstraheerimine sai animatsioonis ja mängus sama palju punkte, sama kehtis ka aspekti parallelism kohta. Kui aspekte abstraheerimine ja parallelism toetati töölehtedega, siis aspekti loogiline mõtlemine ei toetanud ühegi koostatud töölehega.

Projektides hinnati seitset aspekti. Animatsiooni ja mängu projektides olid olemas kuus aspekti: voo haldamine, andmete esitamine, kasutaja interaktiivsus, sünkroniseerimine, abstraheerimine ja parallelism. Nendest neli aspekti arenesid, milleks olid voo haldamine, andmete esitamine, kasutaja interaktiivsus ja sünkroniseerimine. Nimetatud aspektide erinevus animatsiooni ja mängu projektides oli märgatav. Mängus saadud punktid olid oluliselt paremad animatsioonis saadud punktidest. Kaks aspekti abstraheerimine ja parallelism olid projektides, aga need ei arenenud. Aspekt loogiline mõtlemine ei ilmnenud projektides. Uurimistulemustest järeldub, et õpilaste CT oskuste taset on võimalik arendada töölehtede täitmise ning Scratch´i projekti loomisega.

Märksõnad: *computational thinking*, CT, Dr. Scratch, CT hindamine.

Abstract

Using Scratch projects to assess computational thinking.

Bachelor thesis focuses on using Scratch project to evaluate knowledge of computational thinking. The work researches a problem on computational thinking and how there is no practical example of how to evaluate the level of knowledge of CT. Also, the proper tools to evaluate a students level in CT are missing as well as there is no info on how well can Scratch usage develop CT. The theses give an overview of different materials found in other articles and professional literature that mention CT. Work has gathered how other authors have tried to define computational thinking and found tools on how to evaluate Scratch projects and how the tools can also evaluate aspects of CT.

Research data is based on Scratch project that was created on “Code Week” at a Scratch workshop. To reach the goal on how to evaluate CT in Scratch projects two separate worksheets were created containing different aspects of CT. The participants in the workshop

used the two worksheets to create two projects, one was animation and the other was a game. There were 19 participants selected for this theses and all together they made 38 Scratch projects. CT was evaluated with freeware “Dr. Scratch” that took 7 aspects into evaluation: abstraction, logical thinking, synchronization, parallelism, flow control, user interactivity, data presentation.

In all of the projects, there are seven aspects in the evaluation. In animation and game projects were six aspects to evaluate: flow control, data presentation, user interactivity, synchronization, abstraction and parallelism. In those six were four that developed: flow control, data presentation, user interactivity, synchronization. In animation and game projects all four differed in point count noticeably. The game project got significantly more points than animation project did. Two aspects abstraction and parallelism were used in the projects but there was no noticeable development. Research shows that CT in students can be developed with using worksheets and making Scratch projects.

Keywords: computational thinking, automated assessment, Dr. Scratch

Tänusõnad

Autor soovib tänada eelkõige töö juhendajaid, Tauno Paltsi ja Mario Mäeotsa, kes suunasid teemavalikul ning andsid edasiviivat tagasisidet. Tänan perekonnaliikmeid ja sõpru abistamise ning emotsionaalse toetuse eest.

Autorsuse kinnitus

Kinnitan, et olen koostanud ise käesoleva lõputöö ning toonud korrektselt välja teiste autorite ja toetajate panuse. Töö on koostatud lähtudes Tartu Ülikooli haridusteaduste instituudi lõputöö nõuetest ning on kooskõlas heade akadeemiliste tavadega.

21.05.2018

Kasutatud kirjandus

Adams, J., C. & Webster, A., R. (2012). *What do students learn about programming from game, music video, and storytelling projects?* SIGCSE '12 Proceedings of the 43rd ACM technical symposium on Computer Science Education. Raleigh, North Carolina, USA. doi:10.1145/2157136.2157319

Aho, V. A. (2011). Ubiquity symposium: Computation and Computational Thinking. *Ubiquity, 1-8.*

Alice (s.a.). *What is Alice?* Külastatud aadressil http://www.alice.org/index.php?page=what_is_alice/what_is_alice

Basawapatna, A. R., Koh, K. H., & Repenning, A. (2010). *Using scalable game design to teach computer science from middlle school to graduate school*. ITiCSE '10 Proceedings of the fifteenth annual conference on Innovation and technology in computer science education. doi:10.1145/1822090.1822154

BBC Bitesize (s.a.). *Introduction to computational thinking.* Külastatud aadressil http://www.bbc.co.uk/education/guides/zp92mp3/revision/1

Boe, B., Hill, C., Len, M., Dreschler, G., Conrad, P., & Franklin, D. (2013). *Hairball: lint-inspired static analysis of scratch projects.* SIGCSE '13 Proceeding of the 44th ACM technical symposium on Computer science education, Denver, Colorado, USA. doi:10.1145/2445196.2445265

Brennan, K. & Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking.* Külastatud aadressil http://web.media.mit.edu/~kbrennan/files/Brennan_Resnick_AERA2012_CT.pdf

Claro, M., Preiss, D. D., San Martín, E., Jara, I., Hinostroza, J. E., Valenzuela, S., Cortes, F., & Nussbaum, M. (2012). Assessment of 21st century ICT skills in Chile: Test design and results from high school level students. *Computers & Education*, *59*(3), 1042-1053.

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C. & Woollard, J. (2015). Computational Thinking A Guide for Teachers. Külastatud aadressil ; http://community.computingatschool.org.uk/files/6695/original.pdf

International ICT Literacy Panel, (2002). *Digital Transformation: A Framework for ICT Literacy.* Külastatud aadressil https://www.ets.org/Media/Research/pdf/ICTREPORT.pdf

International Society for Technology in Education, (2018). *ISTE Standards for Students.*

Külastatud aadressil http://www.iste.org/standards/standards/for-students-2016

International Society for Technology in Education & Computer Science Teachers Association, (2011). *Operational Definition of Computational Thinking.* Külastatud aadressil http://www.iste.org/docs/ct-documents/computational-thinking-operational-definition-flyer.pdf

Denning. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM, 60*, 33-39.

Dr. Scratch. (s.a.). *Analyze your Scratch projects*. Külastatud aadressil http://www.drscratch.org/

DQ Institute. (s.a.). *What is DQ? 8 Digital Skills We Must Teach Our Children.* Külastatud aadressil https://www.dqinstitute.org/what-is-dq/

Eesti keele seletav sõnaraamat, 2009

Eesti Teadusagentuur, TeaMe, Euroopa Liit, Eesti tuleviku heaks, TTÜ informaatikainstituut. (s.a.). *Skriptid ja käsud.* Külastatud aadressil http://data.vk.edu.ee/SCRATCH/Skriptid_Scratchis.pdf

European Commision (s.a). *COMMISSION STAFF WORKING DOCUMENT Digital Agenda Scoreboard 2013.* Külastatud aadressil https://ec.europa.eu/digital-single-market/sites/digital-agenda/files/DAE%20SCOREBOARD%202013%20-%20SWD%202013%20217%20FINAL.pdf

Grover, S., & Pea, R. (2013). Computational Thinking in K–12 A Review of the State of the Field. *Educational Researcher, 42*, 38-43.

Guzdial, M. (2008). Education: Paving the way for computational thinking. *ACM, 51, 25-27.*

Google for Eduacation (s.a.). *Exploring Computational Thinking*. Külastatud aadressil https://edu.google.com/resources/programs/exploring-computational-thinking/#!home

Hoover, A. K., Barnes, J., Fatehi, B., Moreno-León, J., Puttick, P., Tucker-Raymond, E., Harteveld, C. (2016). *Assessing Computational Thinking in Students' Game Designs.* CHI PLAY Companion '16 Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts (pp. 173-179). Austin, Texas, USA.

Ioannidou, A., Bennett, V., Repenning, A., Koh, K., Basawapatna, A. (2011). Computational Thinking Patterns. *In Proceedings of 2011 Annual Meeting of the American Educational Research Association (AERA) in the symposium "Merging Human Creativity and the Power of Technology: Computational Thinking in the K-12 Classroom".* New Orleans, April 8-12, 2011.

Koh, K. H., (2015). *Computational Thinking Pattern Analysis.* Külastatud aadressil https://www.cs.csustan.edu/~kyuhan/styled-2/styled/

Koh, K. H., Nickerson, H., Basawapatna, A., & Repenning, A. (2014). *Early validation of computational thinking pattern analysis*. ITiCSE '14 Proceedings of the 2014 conference on Innovation & technology in computer science education. doi:10.1145/2591708.2591724

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J., & Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads,* 2(1), 32-37.

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

MIT Scheller Teacher Education Program education arcade (s.a.). *SIMULATION & PROGRAMMING TOOLS: Computing platforms for skill-building and learning complex systems.* Külastatud aadressil http://education.mit.edu/projects/tools/

Moreno-León, J. (2014). *Dr scratch, Automatic analysis of Scratch projects to assess the development of CT* [SlideShare slaidid]. Külastatud aadressil https://www.slideshare.net/jmorenol/dr-scratch

Moreno-León, J., Román-González, M., Harteveld, C., Robles, R. (2017). *On the Automatic Assessment of Computational Thinking Skills: A Comparison with Human Experts.* CHI EA '17 Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems (pp. 2788-2795). Denver, Colorado, USA.

Moreno-León, J. & Robles, R. (2015a). Analyze your Scratch projects with Dr. Scratch and assess your computational thinking skills. Scratch Conference 2015, Amsterdam, The Netherlands, August 12-15, 2015, Proceedings , pages 48–53, 2015.

Moreno-León, J. & Robles, R. (2015b). *Dr. Scratch: a Web Tool to Automatically Evaluate Scratch Projects.* Külastatud aadressil http://delivery.acm.org.ezproxy.utlib.ut.ee/10.1145/2820000/2818338/p132-moreno-leon.pdf

Moreno-León J., Robles R., Román González M. (2015). Dr. Scratch: Automatic Analysis of Scratch Projects to Assess and Foster Computational Thinking. *RED: Revista de Educación a Distancia, 46.*

Repenning, A., Basawapatna, A. & Escherle, N. (2016, 10 November). *Computational thinking tools.* 2016 IEEE Symposium on: Visual Languages and Human-Centric Computing (VL/HCC), Cambridge, UK. doi: 10.1109/VLHCC.2016.7739688

Repenning, A., Webb, D., & Ioannidou, A. (2010). Scalable game design and the development of a checklist for getting computational thinking into public schools. *Proceedings of the 41st ACM Technical Symposium on Computer Science Education (SIGCSE '10),* 265–269. New York, NY: ACM Press.

Resnick M. (2007). *All I Really Need to Know (About Creative Thinking) I Learned (By Studying How Children Learn) in Kindergarten*. Külastatud aadressil http://web.media.mit.edu/~mres/papers/kindergarten-learning-approach.pdf

RiverSound Media. (s.a.). *Thank You For Your Interest.* Külastatud aadressil http://happyanalyzing.com/

Rämmer, A. (2014). Valimi moodustamine. Külastatud aadressil https://sisu.ut.ee/samm/valimid

Scratch Wiki. (2018). *Scratch.* Külastatud aadressil https://wiki.scratch.mit.edu/wiki/Scratch

Selby, C. C., Dorling, M., & Woollard, J. (2014). Evidence of assessing computational thinking. *IFIP TC3 Working Conference: a New Culture of Learning: Computing and Next Generations IFIP TC3 Working Conference: a New Culture of Learning: Computing and Next Generations*, Lithuania.

Selby, C. C., & Woollard, J. (2013). *Computational thinking: the developing definition*. Külastatud aadressil https://eprints.soton.ac.uk/356481/1/Selby_Woollard_bg_soton_eprints.pdf

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

Stephenson, C., & Barr, V. (2011). Defining Computational Thinking for K-12. *CSTA Voice, 7*, 3-4.

TTÜ (s.a.). *Scratch 2.0 Kasutamisejuhend.* Külastatud aadressil http://scratch.ttu.ee/Scratch/Juhend/Scr_juhend14_P.html

Werner, L.,Denner, J., Campe, S., & Kawamoto, D. C., (2012). *The fairy performance assessment: measuring computational thinking in middle school.* SIGCSE '12 Proceedings of the 43rd ACM technical symposium on Computer Science Education. doi: 10.1145/2157136.2157200

Wing J. M. (2006). *Computational thinking.* Külastatud aadressil http://www.cs.cmu.edu/afs/cs/usr/wing/www/publications/Wing06.pdf

Wing J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences, 366,* 3717-3725.

Wing J. M. (2010). *Computational thinking: What and Why?* Külastatud aadressil https://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf

World Economic Forum. (2016). *The Future of Jobs.* World Economic Forum.

Lisad

Lisa 1. Scratchi animatsioon

---

1. Loome animatsiooni, kus on kaks tegelast ning nad vestlevad.

Alustame tegelaste loomisest ning tausta valikust.

- Ava Scratchi lehekülg https://scratch.mit.edu/ .
- Loo uus projekt. (Loo)
- Joonista või vali uus tegelane. (Uus sprait - Spraidi valimine teegist või Uue spraidi joonistamine)
- Muuda tegelase suurust nuppudega  .
- Muuda tegelase suunda nuppudega  .
- Lisa ka taust. (Lava -Taustad - Uus taust)

Oleme valmis, et tegelased liikuma panna. Eesmärk on, et üks tegelane (tegelane_1) liiguks teise (tegelane_2) juurde ning nad vestleksid.

Tegelane_1 (**Spraidid aknas on aktiivne tegelane_1**)

- Lisa tegelasele_1 liikumine.
- Liikumise aeg ja koht vali ise. Selleks, et tegelane_1 liiguks tegelase_2 juurde, mine hiirega tegelase_2 peale ning siis näed paremal all nurgas tema asukohta (x ja y).

. Lisa need koordinaadid.

Katseta! Vajuta nuppu  .

- Nüüd kui tegelane_1 on jõudnud teise tegelaseni, lisa ka see, et tegelane_1 ütleks midagi. Lisa tekst ise.

Katseta!

Tegelane_2 (**Spraidi aknas on aktiivne tegelane_2**)

- Tahame, et teine tegelane vastaks õigel ajal. Seega peab ta ootama üks sekund, kui tegelane_1 liigub ning veel kaks sekundit, kui tegelane_1 ütleb midagi. Lisa tegelase_2 tekst ise.

Katseta!

Tegelane_1 (**Spraidid aknas on aktiivne tegelane_1**)

- Ootab tegelase 2 vastuse ära 2 sekundit ning liigub siis tagasi.

Katseta!

Proffidele:

- Lisa taustale muusika.
- Muuda tegelase_1 liikumist nii, et tagasi liiguks ta suvalisse kohta.

Salvesta fail ka arvutisse (Fail -> Laadi oma arvutisse). Failinimeks võib panna animatsioon ja arvuti number.

Lisa 2. Mängu loomine

Loome mängu, milles Jõehobu püüab Kassi ning kogub punkte.

1.OSA

1. Loome mängu, kus Jõehobu hakkab Kassi taga ajama.

- Loo uus projekt.
- Lisa uus tegelane Jõehobu.
- Muuda Jõehobu suurust.
- Lisa taust.
- Lisa Jõehobule järgnev liikumine.(Spraitide aknas on aktiivne Hippo1)

- Mõtle, kuidas Jõehobu liiguks ka tagurpidi. Lisa tagurpidi liikumine ise. Katseta!
- Kui jõehobu puudutab Kassi, siis Kass ütleb "Kurnjäu". (**Spraitide aknas on aktiivne Sprite1**)

Katseta!

2. Jõehobule liikumise efekti lisamine. (Spraidid aknas on aktiivne Hippo1)

- Jõehobul on kostüümi vaates kaks kostüümi. Kui kostüümid vahetuvad piisavalt kiiresti, siis tekib mulje, et Jõehobu liigub. Lisame Jõehobule kostüümi vahetamise. Kontrolli, et kostüümid oleksid sama suured!

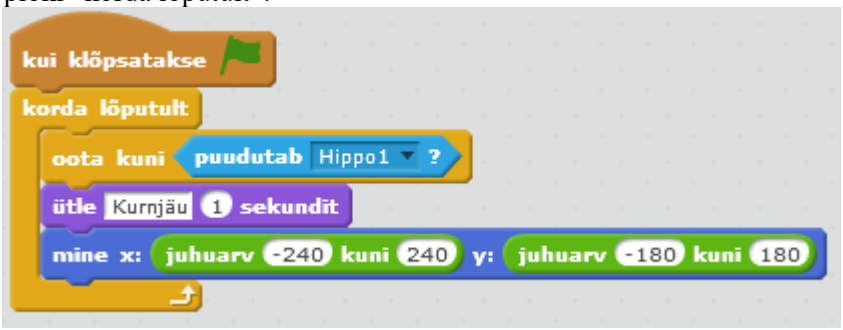
3. Kui Jõehobu saab kassi kätte, siis Kass hüppab eest ära. (Spraidid - Sprite1)

- Lisa Kassile, et Jõehobu puudutuse korral hüppab Kass suvalisse punkti.

Kust tulevad arvud -240,240,-180 ja 180? Vihje: koordinaatteljestik.

Kass hüppab iga kord eest ära, kui Jõehobu on teda puudutanud.

- Selleks, et Kass hüppaks iga kord Jõehobult eest ära peab hüppamine korduma. Selleks lisame ploki “korda lõputult”.



Katseta!

4. Punktisüsteemi loomine (Spraidid - Sprite1)

- Loo muutuja nimega Punktid. (Andmed - Loo muutuja - Punktid - Kõikide spraitide jaoks)
- Punktidele lisatakse +1, kui Jõehobu puudutab Kassi.
- Kui mäng algab, siis punktid lähevad nulli.

Katseta!

2.OSA
Eesmärk luua mängule teine tase ning tegelased kelle puudutamisel saab miinuspunkte.

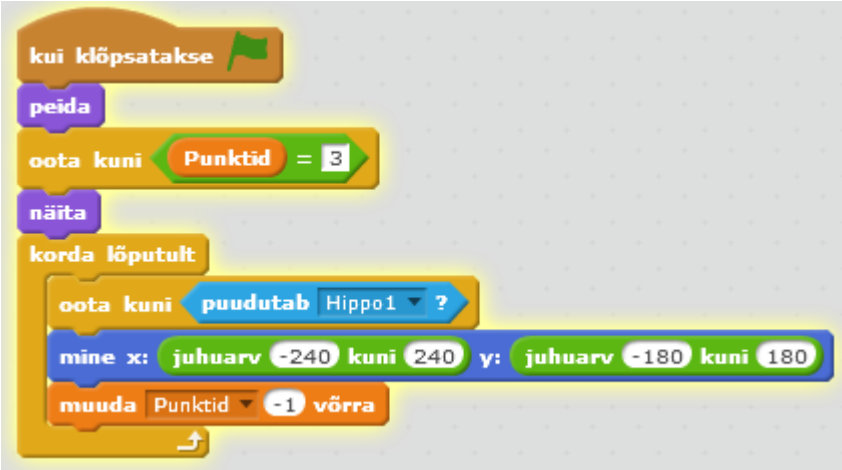
1. Tausta lisamine ning vahetamine. (Spraidid - Lava)

- Lisa uus taust.
- Mängu alguses on taust1 ning kui Jõehobu on saanud Kassi kolm korda kätte, siis taust vahetub.

- Lava - Skriptid

2. Lisa teisele tasemele tegelane, kelle puudutamisel saab miinuspunkte.

- Lisa uueks tegelaseks Dinosaurus.
- Lisa Dinosaurusele samasugune liikumine nagu Kassile.
- Dinosaurus annab puudutamisel -1 punkti.
- Peida Dinosaurust seni, kuni punktid on võrdsed kolmega.
- Näita Dinosaurust siis kui Jõehobu ja Kass on jõudnud teisele tasemele.



3. Mida teha selleks, et mängu alguseks oleksid tegelased alati samas kohas ega liiguks ekraanist välja? Kui on kogutud 6 punkti, siis mäng lõppeb.

- Lisa tegelastele asukoht, kus nad on alati mängu alguses.
- Kui tegelased on ekraani äärel, siis nad põrkavad. Lisa tegelaste põrkamine.
- Tegelased on mängu alguses alati õigetpidi. Lisa .
- Kui punktid on 6, siis mäng peatub. Katseta!

Dinosaurus

Jõehobu

Lava

Proffidele:

- Loo ajavõtusüsteem. Kui aeg saab otsa, siis mäng lõppeb.
  - Kui aeg saab otsa, siis Jõehobu teatab: “Aeg on otsas!”.
  - Vihje: aeg peab muutuma -1 võrra ning siis peab olema paus.
- Pane Jõehobu laulma, kui kogutud on 6 punkti ning peata mäng
  - nii, et aeg jääks seisma
  - ja laul mängiks lõpuni.

Lisa leiad

- http://www.progetiiger.ee/scratchi-materjalid
- http://www.nutilabor.ee/programmeerimine/
- https://scratch.mit.edu/
- https://wiki.scratch.mit.edu/wiki/Scratch_Wiki_Home (ingliskeelne).

Loo oma mäng või animatsioon

Salvesta fail ka arvutisse (Fail -> Laadi oma arvutisse). Failinimeks võib panna minumäng ja arvuti number.

Vaata ka http://www.hitsa.ee/konkursid/hitsa-opilaskonkurss-digistame-ara .

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina Getriin Kokk (sünnikuupäev: 25.03.1995)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose

Algoritmide rakendamise hindamine probleemide lahendamisel Scratchi projektide näitel, mille juhendajad on Tauno Palts ja Mario Mäeots,

1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.
3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, 21.05.2018