TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Helena Talimaa

Süsteem füüsiliste klotsidega programmeerimiseks

Bakalaureusetöö (9 EAP)

Juhendaja: Aivar Annamaa

Tartu 2017

Süsteem füüsiliste klotsidega programmeerimiseks

Lühikokkuvõte:
Käesoleva töö raames valmis füüsiliste klotsidega programmeerimise süsteem, mis on mõeldud programmeerimise tutvustamiseks ning selle õppimise lihtsustamiseks. Sellised süsteemid võimaldavad programme koostada käske kujutavate klotside rittaseadmise teel. Klotsidega programmeerimiskeeled võimaldavad minimeerida või täielikult kaotada programmist süntaksivead ning selle läbi motiveerida inimesi rohkem programmeerimist katsetama ja õppima. Ühtlasi antakse töös ülevaade olemasolevatest lahendustest ning uuritakse, kas on võimalik seniste süsteemidega võrreldes teha klotse kasutajasõbralikumaks. Loodud süsteem koosneb kolmest peamisest osast: klotsidega programmeerimiskeelest, pildianalüüsist ning mängust programmi tulemuse visualiseerimiseks. Loodud süsteemi on võimalik edasi arendada eksponaadiks või Androidi mobiilirakenduseks. Projekt on kirjutatud Javas, klotside loomiseks kasutati rakendust GIMP ning pildianalüüsiks teeke OpenCV ja TopCode.

Võtmesõnad:
klotsidega programmeerimine, programmeerimiskeeled, programmeerimise õppimine

CERCS: P175 - Informaatika, süsteemiteooria, S281 - Arvuti õpiprogrammide kasutamise metoodika ja pedagoogika

System for Tangible Programming

Abstract:
This thesis aims to create a system to generate executable programs constructed from tangible pieces. Its intention is to introduce and teach programming in a simple manner. Tangible programming systems help to minimalize or completely eliminate syntax errors which consequently motivates its users to experiment and further learn programming. Thesis will cover existing solutions and further research is done to improve block design. System constructed in this thesis mainly consists of three parts: tangible programming language, image analysis and a game visualizing the program's output. This system can further be developed to be a showpiece at an exhibition or a standalone Android application. The project has been completely written in Java. OpenCV and TopCode libraries were used for image recognition and analysis. Visual elements were created in GIMP.

Keywords:
tangible programming, programming languages, educational programming

CERCS: P175 - Informatics, systems theory, S281 - Computer-assisted education

Sisukord

1 Sissejuhatus

Füüsiliste klotsidega programmeerimise süsteem võimaldab programme koostada käske kujutavate klotside rittaseadmise teel. Kasutajal on võimalik erinevate käskudega klotside seast valida endale sobivad ning neist koodijada tekitada. Füüsiliste klotsidega programmeerimist võib võrrelda pusle kokkupanekuga, kus on vaja ühendada sobivad klotsid ning selle tulemusena valmib üks tervik.

Tekstipõhistes programmeerimiskeeltes, näiteks Javas või Pythonis, on koodi kirjutamine üsna abstraktne ja virtuaalne. Sellistes keeltes võib ka lihtsate ja lühikeste programmide korral tekkida süntaksivigu, mida algajal on tihti keeruline üles leida. Lihtsad klotsisüsteemid suudavad süntaksivigade tekkimise välistada ja selle läbi motiveerida inimest rohkem programmeerimist katsetama. Samuti on klotside puhul võimalike komponentide loetelu ning võimalikud ühendamisviisid selgemini näha, mis omakorda lihtsustab võhikul programmeerimise õppimist.

Käesoleva töö eesmärk on luua eestikeelne prototüüp füüsiliste klotsidega programmeerimiseks ning selle abil lihtsustada ja julgustada programmeerimisega tutvumist. Loodavate programmide väljundeid demonstreeritakse labürindipõhise mängu abil. Prototüüpi luues on eesmärgiks eraldada üksteisest pildianalüüsi, programmi süntaksipuu genereerimise ning tulemuse visualiseerimise loogikad, et võimaldada vähese vaevaga projekti edasiarendamist eksponaadiks või Androidi mobiilirakenduseks. Ühtlasi uuritakse prototüübi loomise käigus, kas on võimalik seniste süsteemidega võrreldes teha klotse kasutajasõbralikumaks.

Töös antakse esmalt ülevaade olemasolevatest füüsiliste ja virtuaalsete klotsidega programmeerimise lahendustest. Selle käigus uuritakse klotside disainialaseid valikuid ning kasutatavaid käske. Ühtlasi võrreldakse füüsiliste ja virtuaalsete klotsidega süsteeme ning tuuakse välja nende eelised ja puudused. Seejärel kirjeldatakse, milliseid klotse ning käske kasutatakse loodud prototüübis. Töös antakse ülevaade ka pildianalüüsi ja programmi genereerimise loogikast ning programmi väljundi visualiseerimisest mängus. Prototüübi üles seadmise ning rakenduse kasutamise kohta on lisatud töösse juhend. Lõpuks kirjeldatakse prototüübi edasiarendamise võimalusi.

2 Olemasolevad lahendused klotsidega programmeerimiseks

Klotsidega programmeerimise süsteeme on erinevaid. Järgnevas peatükis kirjeldatakse lähemalt kaht füüsiliste klotsidega süsteemi, mis on välja valitud sarnase pildianalüüsi ning populaarsuse tõttu. Samuti kirjeldatakse üht populaarset virtuaalsete klotsidega programmeerimise süsteemi, millega antakse alus virtuaalsete ja füüsiliste klotsidega süsteemide võrdlemiseks. Eelkõige keskendutakse ülevaadetes süsteemides kasutatavatele käskudele ning klotside disainile.

2.1 Tern

Järgnev ülevaade tugineb M. Horni ja R. J. K. Jacobi artiklitele [3] ja [4]. Tern on puust valmistatud klotsidega programmeerimiskeel. Terni eesmärk on lihtsustada programmeerimise õpetamist ning võimaldada seda ka siis, kui arvuteid klassides kasutada pole võimalik või neid on vähe. Terni jaoks on loodud ka püsiv eksponaat Bostoni teadusmuuseumisse. Õpilastele mõeldud variandis juhitakse arvutis tegelasi ruudustikupõhises maailmas, muuseumis kasutatakse programmi väljundi visualiseerimiseks robotit.

Joonis 1. Terni programmeerimiskeeles kasutatavad klotsid [4]

Terni projektis on klotsidega programmeerimine peaaegu täielikult arvutist eraldatud. Klotsid ei ole ühenduses ühegi elektroonikakomponendiga ega oma toiteallikat. Selle asemel kasutatakse arvutiga ühenduses olevat veebikaamerat, mis klotsijadast pilti teeb ja selle arvutile edastab. Seejärel teostatakse pildianalüüs TopCode teegi [18] abil, mis võimaldab sisendpildilt kiiresti tuvastada tasasele pinnale asetatud objekte, kui nen-

de peal on TopCode koodid [17]. Nii tõlgitakse veebikaamerast saadud pilt arvutile arusaadavaks koodiks.

Klassiruumis kasutamiseks mõeldud variandis on võimalik kasutada lihtkäske, tingimuslauseid, tsükleid ning alamprogramme (joonis 1). Programmid algavad START ning lõppevad STOP käsuga. Objekti liigutamiseks on käsk MOVE, paremale ja vasakule pööramiseks vastavalt käsud RIGHT ning LEFT. Tsüklite jaoks on kasutusel JUMP ja LAND käsud, mis ühendatakse omavahel juhtmega, et visualiseerida, kuhu pärast tsüklikordust tagasi hüpatakse. Tingimuslaused on kahe haruga klotsid, millele saab mõlemale harule uusi klotse juurde lisada. Alamprogrammide kujutamiseks on klots kirjaga SKILL, mis tähendab hüpet alamprogrammi. Alamprogramm on üles ehitatud sarnaselt põhiprogrammile, kuid START käsu juures on samuti kirjas SKILL.

Joonis 2. Terni programm [4]

Programm algab alustamiskäsust (joonis 2). Sellele võib järgneda ükskõik mitu lihtkäsku. Tingimuslause klotsi juures jaguneb edasine programm kaheks ning need harud enam omavahel kokku ei saa. See on ka oluline erinevus võrreldes tekstipõhiste struktuursete programmeerimiskeeltega, kus tingimuslausele võivad järgneda käsud, mida täidetakse olenemata sellest, millises harus eelnevalt oldi. Ternis on võimalik sama efekti saavutada ainult siis, kui mõlema haru lõppu lisada täpselt samad käsud täpselt samas järjekorras. Terni klotsidega loodav tsükkel käitub sarnaselt tekstipõhistes programmeerimiskeeltes kasutatava *while*-tsükliga. Esmalt on vaja paika panna maandumisklots. Seejärel võib kasutada ükskõik mitut lihtkäsku, kuid lõpmatu tsükli vältimiseks peab ühel hetkel järgnema tingimuslause klots. Selle üks haru võib jällegi sisaldada lihtkäske, kuid peab lõppema hüppeklotsiga, mille juurest liigutakse programmi täitmisel tagasi juhtmega ühenduses oleva maandumisklotsi juurde. Teine haru jätkab programmi tsüklijärgse loogikaga.

Klotsid on ehitatud viisil, mis ei võimalda süntaksivigade tekkimist. Alustamiskäsule vastav klots on kujundatud nii, et tema ette ei saaks rohkem ühtki käsku panna. Kõigil teistel klotsidel on ees süvend, kuhu saab teisi klotse ühendada. Lõpetamis- ja hüppekäsul puudub seevastu väljuv haru. See näitab, et sealt edasi ei saa enam käske tulla. Tingimuslause klotsist väljub täpselt kaks haru, lihtkäsud nagu liikumine ja pööramised omavad ainult üht väljuvat haru.

Klotside disain koosneb kahest elemendist. Esiteks on igal klotsil TopCode kood, mis võimaldab pildianalüüsil kiiresti tuvastada, mis käsuga tegu on. Selle miinuseks on asjaolu, et enamik klotsi pinnast on kasutajale arusaamatu märgi all. Teiseks on klotsil ka tekst või käsu olemust kirjeldav sümbol. Näiteks tingimuslause klots sisaldab tekste „NO", „YES" ja „BLOCKED?". Tsükli käskudel on tähte ja noolt kujutavad sümbolid. Klotside füüsiline kuju imiteerib noolt, mis näitab programmi kulgemise suunda.

Joonis 3. Valik eksponaadis kasutatavatest klotsidest [5]

Terni jaoks on loodud eksponaat Bostoni teadusmuuseumisse, kus kasutajad saavad luua programme, mis kontrollivad roboti käitumist. Järgnev eksponaadi ülevaade tugineb M. Horni jt artiklile [5]. Eksponaadi jaoks on muudetud klotside disaini ning programmeerimiskeelde on lisatud uusi käske ja võimalusi (joonis 3). Klotsidele on lisatud ka hispaaniakeelsed tõlked. Kuna eksponaadi puhul kontrollivad kasutajad oma programmiga robotit, on lisaks alustamis-, astumis- ning pööramiskäskudele lisatud juurde parameetrid ning tantsuliigutuste ja häälitsuste klotsid. Lisaks saab luua erinevaid tsük-

leid ning nendega koos saab kasutada sensorite klotse. Viimased võimaldavad sensorite abil infot koguda. Ühtlasi saab kasutada loogikaklotse, mida saab jällegi kombineerida sensorite klotsidega.

Eksponaat koosneb arvutiekraanist ning alusest roboti ja programmeerimise jaoks, mille kohale on paigutatud veebikaamera. Kui kasutaja on programmi valmis saanud, saab ta vajutada nuppu, mis alustab pildianalüüsi. Veebikaamerast saadud pildilt leitakse alustamisklots ning tuvastatakse kõik sellele järgnevad klotsid. Saadud info saadetakse robotile, mis käske täitma hakkab. Samal ajal näidatakse kasutajale ekraanil programmipilti, kus iga käsu täitmise korral näidatakse noolega vastava klotsi peale. Lisaks on programmeerimisalusel üleval paremas nurgas eriline süvend. Kasutaja saab sinna paigutada klotsi ja seejärel täidab robot kohe vastavat käsku ning ekraanil kuvatakse käsu kirjeldus.

Eksponaadi põhjal on läbi viidud ka uurimus, mis käsitles kasutajate käitumist ning süsteemi kasutamist. Inimesi vaadeldi kahel korral, millest mõlemal asus umbes 80% eksponaati märganud inimestest seda ka kasutama. Mõlemal korral kasutati süsteemi palju ka gruppides. Esimene kord said ainult pooled grupid süsteemi edukalt tööle, kuid pärast eksponaadi kitsaskohtade parandamist sai juba umbes 69% sellega edukalt hakkama. Uurimusest selgus, et eksponaat on kutsuv, toetab grupitööd ning edasise arendamisega muutub veel lihtsamini kasutatavaks.

2.2 Strawbies

Järgnev ülevaade tugineb F. Hu jt artiklile [7]. Strawbies on lastele mõeldud rakendus, mida saab reaalajas füüsiliste klotside abil mängida. Selle eesmärk on tutvustada arvutuslikku ja programmeerimisalast mõtlemist viisil, mis on lastele lõbus, kaasahaarav ja tuttav. Lapsed saavad iPadi ette tasasele pinnale ehitada puust klotsidest programme ja jälgida mängu iPadi vahendusel. Rakenduseks valiti avatud maailma stiilis mäng, et lapsed saaksid ise selles ringi uudistada. Mängus tuleb juhendada peategelast läbi erinevate ülesannete.

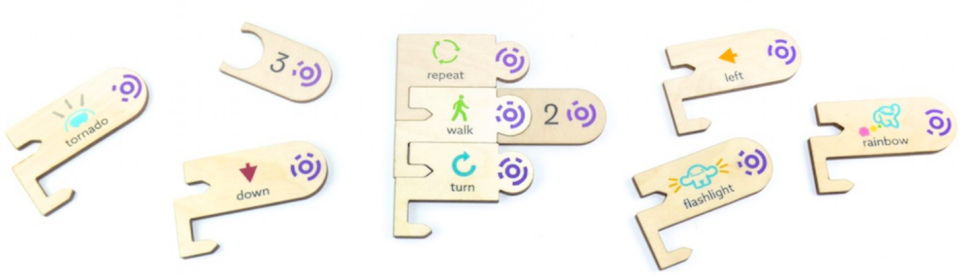
Mängu mängitakse iPadi ja Osmo [13] süsteemi abiga (joonis 4). Tavaliselt nõuaks klotsiseisu reaalajas tuvastamine alust, mille kohale on paigaldatud kaamera, või nutiseadet, millega pidevalt klotsiseisule osutada. Osmo on lisavahend, mis võimaldab klotside tuvastamist iPadi ees ilma seadet ennast liigutamata. See on mängimissüsteem, mis sisaldab endas alust iPadi jaoks ja seadet esikaamera jaoks. Seade sisaldab peeglit, mis võimaldab pidevalt jälgida iPadi ees olevat klotsijada. Eelnimetatud süsteemi ja TopCode teegi abil teostatakse pildianalüüs.

Strawbies kasutab klotse, mis väljendavad parameetreid, tegevusi, tsükleid ning tingimuslauseid (joonis 5). Klotsid jaotati erinevatesse kategooriatesse: tegusõnad, määrsõnad ja numbrid. Programmi on võimalik moodustada tegusõnu üksteise alla ühendades. Nende külge on võimalik lisada numbreid või määrsõnu, mis tegevust kirjeldavad või näitavad, mitu korda seda on vaja korrata. Jada algusesse saab lisada klotsi kirja-

Joonis 4. Strawbies mängu mängitakse iPadi ja Osmo abil [7]

ga „Always” või „When”, millest esimene on tsüklite ning teine tingimuslausete jaoks. Mängus on kolm erilist tegevuse klotsi: vikerkaare, taskulambi ja tornaado klotsid. Vikerkaare käsuga on võimalik mängu tegelast lennutada maailmas suvalisse kohta, tornaado tõmbab teatud raadiuses kõik maasikad tegelase juurde ning taskulamp peletab eemale rotid, kes maasikaid varastada proovivad.

Strawbies klotside kujunduse juures oli kõige tähtsam see, et neid oleks kerge omavahel kokku ja lahti ühendada. Samal ajal oli oluline võimaldada suurte klotsikomplektide mugavat nihutamist ilma neid lõhkumata. Klotsid muudeti pärast esialgset testimist suuremaks ning neile lisati illustratsioonid, et ka lapsed, kes veel lugeda ei oska, suudaksid tegevust klotsiga seostada. Igal klotsil on taaskord peal TopCode kood, mis



Joonis 5. Strawbies mängus kasutatavad klotsid [7]

võimaldab kiiresti pildianalüüsi teha.

Valmis mängu testiti kahes erinevas keskkonnas: esimeses mängisid kaks last korraga, teises oli lapsi mitu ning nad said pidevalt ruumi sisse ja välja käia. Avatud keskkonnas ilmnes, et loodud süsteem soosib laste omavahelist suhtlemist. Nad pidasid pidevalt üksteisega nõu, üritasid jõuda ühiselt lahenduseni ning uute õpilaste lisandumisel õpetati ka neid. Mängude ümber kogunesid viieliikmelised grupid ning igaühel oli midagi teha: valida järgnevaid klotse, täiustada kaamera ees olevat programmi või sealt klotse eemaldada.

2.3 Scratch

Järgnev ülevaade tugineb kirjutisele [14] ning Resnick M. jt artiklile [15]. Scratch [16] on programmeerimiskeel, mis kasutab koodi kirjutamiseks virtuaalseid klotse. Scratchi loomisel peeti silmas, et see oleks lihtsasti õpitav ja visuaalselt atraktiivne keel, mis pakuks huvi nii noortele kui ka vanematele, kes varem programmeerimisega pole kokku puutunud.

Scratch võimaldab luua erineva välimusega tantsivaid, laulvaid või üksteisega suhtlevaid tegelasi. Samuti on võimalik luua animeeritud pilte, interaktiivseid lugusid, sünnipäevakaarte, mänge ning palju muud. Scratchis on võimalik omavahel ühendada graafika, animatsioonid, pildid, muusika ja helid. Keskkonna puhul on oluliseks osaks ka veebileht, kuhu kasutajad saavad oma projekte üles laadida ning teiste liikmete tehtud töid jooksutada ning muuta.

Scratchi arendamiskeskkond koosneb neljast peamisest vaatest (joonis 6). Esimene neist on klotside valiku kuvamiseks. Seal on võimalik kategooriate kaupa vaadata, milliseid klotse kasutada saab. Teine vaade on programmi koostamiseks. Kolmandas aknas näidatakse programmi väljundit ning neljandas saab erinevaid omaloodud objekte valida.

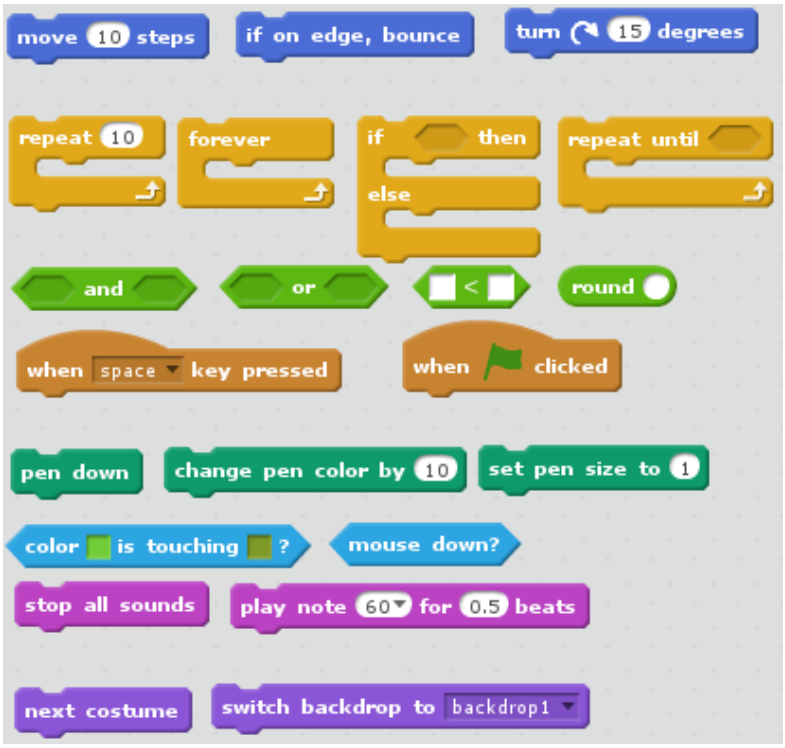
Scratchis kasutatavad klotsid (joonis 7) on disainitud nii, et omavahel saab ühendada ainult selliseid klotse, mille puhul üksteisele järgnevus on loogiline ja mõistlik. Tsükleid

Joonis 6. Scratchi programmeerimiskeskkond [19]

ja tingimusi väljendavad klotsid on C-tähe kujulised, mis näitab selgelt, et koodi on võimalik ka nende sisse paigutada. Klotsid, mis tagastavad väärtusi, on vastavalt väärtuse tüübile disainitud: ovaalid tähistavad numbreid ning kuusnurgad tõeväärtusi. Tingimuslaused ning tingimusega tsükkel vajavad argumendiks kuusnurkset klotsi, mis tähendab seega tõeväärtust.

Scratchi programmi loomise vaade on interaktiivne ja võimaldab klotsijadale klikkimisel seda automaatselt käivitada. Scratch lubab jadas muudatuste tegemist isegi siis, kui koodi parasjagu täidetakse. Samuti ei pea alati kõiki hetkel ebavajalikke klotse ära kustutama. Need saab klotsijadast lahti ühendada ja jätta alles samasse vaatesse hilisemaks kasutamiseks. See võimaldab uute ideedega mugavalt ja kiirelt eksperimenteerida. Scratch võimaldab ka mitut klotsijada paralleelselt täita. Üks keskkonna eesmärkidest on muuta paralleelne koodi täitmine sama intuitiivseks kui ühe klotsijada käivitamine.

Scratchi on testitud mitmes erinevas keskkonnas [10, 19]. Ühes uurimuses testiti keskkonda Harvardi ülikoolis, et tutvustada algajatele programmeerimist enne Java programmeerimiskeelele üleminekut. Pea kõik algajad nõustusid, et Scratchiga tutvumine oli kasulik. Ainsad, kes midagi uut ei õppinud, olid need, kellel juba oli mingisugune programmeerimiskogemus olemas. Teises uurimuses katsetati Scratchi kaheksanda klassi tüdrukute seas. Neile korraldati kolmetunnine koolitus tutvustamaks prog-



Joonis 7. Valik Scratchi klotsidest

rammeerimise põhiteadmisi. Õpilastel oli juba varasem kogemus programmeerimises olemas, kuid tulemused näitasid, et nad õppisid siiski midagi uut ja protsess oli nauditav.

2.4 Füüsiliste ja virtuaalsete klotsidega süsteemide võrdlus

Selles peatükis võrreldakse füüsiliste ja virtuaalsete klotsidega süsteeme. Mõlema puhul tuuakse välja nende eelised ning puudused.

2.4.1 Füüsiliste klotsidega süsteemide eelised

Michael Horni jt kirjutistes [3] ja [4] ning Marjorie Howardi artiklis [6] on välja toodud, et algkoolis on arvutitunnid mõnevõrra problemaatiliselt üles ehitatud. Kui klassis pole piisavalt arvuteid, peavad õpilased virtuaalsete süsteemide puhul ühe masina taha koonduma ja üksteise vahel nii hiirt kui ka klaviatuuri jagama. Seega tekivad kergesti olukorrad, kus üks tegutseb ja teised pealt vaatavad. Füüsiliste klotsidega süsteemid võimaldavad mitmel õpilasel ühe programmi kallal aktiivselt tegutseda. Kõigil on võimalik

samal ajal käsujadasid koostada ning neist pärast kokku üks terviklik programm moodustada. Seega sobivad füüsiliste klotsidega süsteemid paremini kasutamiseks klassides, kus pole kõigile eraldi arvuteid.

Arvutis programmeerimise puhul on probleemiks ka segavate faktorite rohke olemasolu [4, 6]. Arvutis on mitmeid tegevusi, mis õpilase tähelepanu õppetöölt eemale võivad juhtida. Õpetajatel on üsna tülikas igat arvuti taga tegutsevat õpilast kontrollimas käia. Kui õpilased on arvuti taga ühes grupis koos, on õpetajal keeruline näha, millega antud hetkel tegeletakse. Füüsiliste klotsidega süsteemide puhul on segavaid faktoreid tunduvalt vähem ning ühtlasi on kogu programm avalikult laual näha. See muudab õpetajalt tagasiside saamise tunduvalt kiiremaks ja mugavamaks. Ühtlasi säilib õpetajal kontroll õpilaste tööde üle.

Füüsilised klotsid on käegakatsutavad ja reaalsed. See muudab programmi paremini mõistetavaks ning klotside füüsiline järjestamine annab selgema ülevaate programmi loomise põhimõtetest. Sellised süsteemid julgustavad lapsi rohkem programmeerimist uurima ja katsetama. Horn on toonud välja ka katsetuste tulemuse Bostoni teadusmuuseumis, kus ainult 30% tüdrukuid katsetasid programmeerimist hiire ja klaviatuuriga, kuid lausa 90% protsenti katsetasid füüsiliste klotsidega programmeerimist [6].

Füüsiliste klotsidega süsteemid ei vaja nii palju eelteadmisi kui virtuaalsete klotsidega süsteemid. Virtuaalsete puhul on igal õpilasel vaja osata arvutit kasutada, lisaks sellele on vaja tundma õppida ka tervet uut programmi. See tähendab, et lapsed, kes veel väga hästi lugeda ei oska, ei saa süsteemi kasutada. Seevastu füüsiliste klotside puhul piisab klotsi peal olevast märgendist aru saamisest ning oskusest näiteks iPadis rakendus käivitada. Terni looja Michael Horn on välja toonud, et noorematel lastel ei pruugi motoorika veel nii arenenud olla, et hiirt erinevate objektide liigutamiseks ja lohistamiseks kasutada [6]. Näiteks lasteaialastel on seetõttu lihtsam omavahel kokku sobitada just füüsilisi klotse.

Seega sobivad füüsiliste klotsidega süsteemid pigem lasteaiaealistele ning algkooli õpilastele, kelle lugemisoskus ja motoorika ei pruugi veel nii hea olla ja kellel ei ole arvutitega palju kogemusi. Ühtlasi on selliste süsteemidega hea õpilasi julgustada programmeerimist katsetama.

2.4.2 Virtuaalsete klotsidega süsteemide eelised

Esimene eelis virtuaalsete klotsidega süsteemide puhul on salvestamise võimalus. Kasutajatel on pidevalt võimalus töö seis salvestada, programm sulgeda ning hiljem sama seisu juurde naasta. Suuremate programmide puhul on see oluline eelis. Füüsiliste klotside puhul otsest salvestamise võimalust ei ole, vaid klotsijada jaoks on vaja leida koht, kus seda hoida.

Virtuaalsete klotsidega süsteemide puhul on eeliseks ka oma tööde jagamise ja teiste tööde nägemise võimalus. Oma töö jagamine annab võimaluse koguda tagasisidet, tänu millele saab projekti õiges suunas edasi arendada. Samuti annab teiste tööde nägemi-

ne võimaluse erinevaid ideid koguda, mida enda töödes rakendada. Füüsiliste klotside puhul on võimalik tagasisidet saada ainult lähedalolijatelt ning ideid koguda füüsiliselt läheduses olevatelt programmidelt.

Virtuaalsed klotsid näevad selgemad ja arusaadavamad välja. Selliste süsteemide puhul ei pea tegema pildianalüüsi, mis tähendab, et klotsidele pole vaja lisada kasutajale arusaamatuid sümboleid. Nii jääb rohkem ruumi inimesele arusaadava märgistuse kujundamiseks ning lõpptulemus on kasutajale mõistetavam.

Virtuaalsete klotsidega süsteemid võivad olla rohke klotsivalikuga ning kasutada rohkem multimeedia võimalusi. See aitab vältida õpilastel tüdimuse tekkimist, kuna pidevalt on võimalik uusi asju katsetada [19]. Virtuaalseid klotse on võimalik kogu aeg juurde lisada, mis tähendab, et kasutajatele lisandub järjest uusi võimalusi programmide loomiseks.

Seetõttu sobivad virtuaalsete klotsidega süsteemid pigem alates alg- või põhikoolist, kuna selleks ajaks on õpilaste lugemisoskus hea ja üldjuhul on ka arvutite kasutamise kogemus olemas. Sellises vanuses suudavad nad ka erinevate võimaluste rohkuses paremini orienteeruda.

2.4.3 Võrdluse kokkuvõte

Võrdluse tulemusena saab järeldada, et füüsiliste klotsidega süsteemid sobivad pigem lasteaiaealistele ning algkooli õpilastele, kelle lugemisoskus ja motoorika ei pruugi veel nii hea olla ja kellel ei ole arvutitega palju kogemusi. Ühtlasi sobivad sellised süsteemid pigem programmeerimise tutvustamiseks ning võhikutele õppimiseks. Virtuaalsete klotsidega süsteemid seevastu sobivad pigem alates alg- või põhikoolist, kuna selleks ajaks on õpilaste lugemisoskus hea ja üldjuhul on ka arvutite kasutamise kogemus olemas. Sellises vanuses suudavad nad ka erinevate võimaluste rohkuses paremini orienteeruda. Seega on mõlemal süsteemil oma kindel kasutajaskond ning omad eesmärgid.

3 Prototüüp

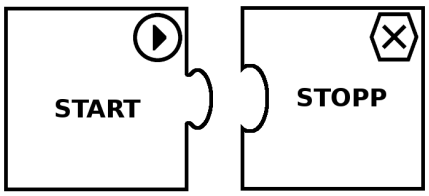
Selles peatükis kirjeldatakse valminud prototüüpi. Selle käigus antakse ülevaade loodud programmeerimiskeeles kasutatavatest käskudest ning klotside disainist. Ühtlasi kirjeldatakse pildianalüüsi ja programmi süntaksipuu loomise protseduure. Seejärel kirjeldatakse rakendust, mida kasutatakse programmi tulemuse visualiseerimiseks. Lõpuks antakse juhend projekti üles seadmise ja rakenduse kasutamise jaoks. Kogu projekt on kirjutatud Java programmeerimiskeeles.

3.1 Klotsid

Järgnevas peatükis antakse ülevaade prototüübis kasutatavatest klotsidest. Sealjuures selgitatakse, millised käsud on loodud programmeerimiskeeles kasutusel, ning põhjendatakse klotside disaini valikuid.

3.1.1 Käsud

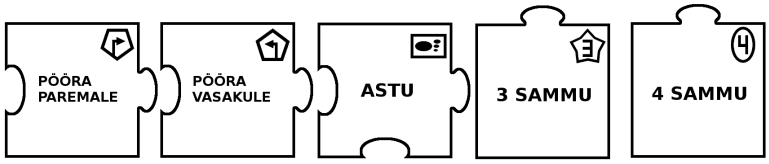
Loodud keel on mõeldud programmeerimise õppimiseks ning lihtsamate käskude ja loogikate tutvustamiseks. Seega toetab loodud keel lihtkäskudest jadade moodustamist ning kahe lihtsama programmeerimise struktuuri - hargnemise ja tsükli - loomist. Lisaks on võimalik kasutada argumente, mis demonstreerivad funktsioonide kasutamist.



Joonis 8. Prototüübis kasutatavad alustamis- ja lõpetamisklotsid

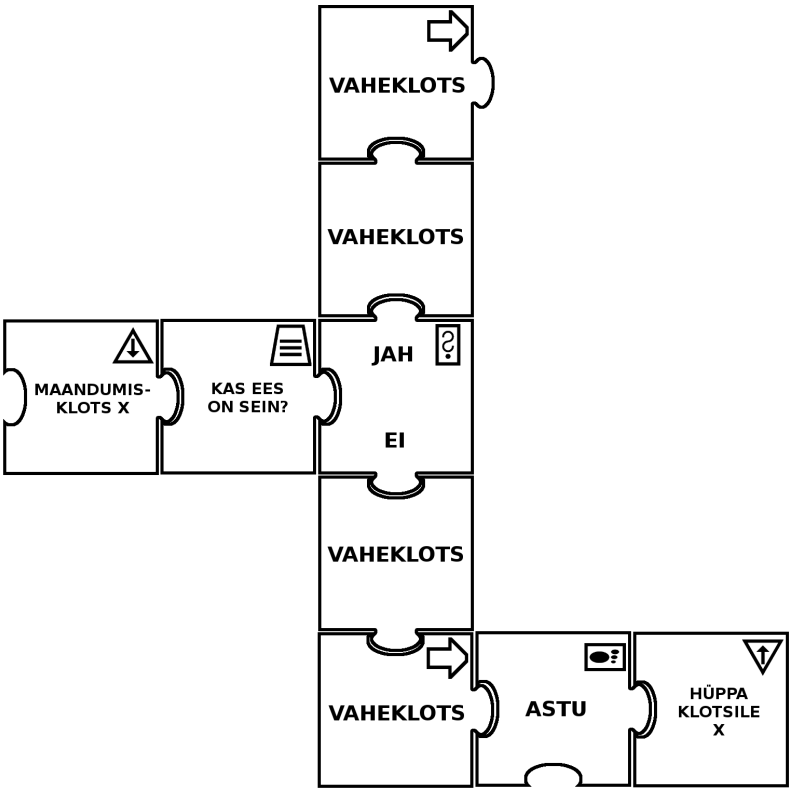
Esmalt on loodud keeles programmi alustamis- ja lõpetamiskäsud (joonis 8). Need klotsid on võetud kasutusele kahel eesmärgil. Esiteks on nende abil võimalik kasutajale näidata, et programm on üks tervik. Igal programmil on kindel algus ning lõpp. Lõpppunkte võib olla ka mitu, kui on kasutatud tingimuslauseid, mille tõttu programm mitmeks hargneb. Teiseks on alustamisklotsi abil võimalik lihtsustada pildianalüüsi ning programmi süntaksipuu ehitamist. Lõpetamiskäsu kasutamine pole kohustuslik, kuid on soovituslik.

Lihtkäskude valimine programmeerimiskeelde sõltub programmi tulemuse visualiseerimiseks mõeldud rakendusest. Käesolevas töös on programmeerimiskeel kasutusel mängus, mille eesmärk on tegelast läbi labürindi juhatada. Seetõttu on lihtkäskudena



Joonis 9. Prototüübis kasutatavad lihtkäsud ning argumendid

kasutusel astumise ning paremale ja vasakule pööramise klotsid (joonis 9). Liikumisklotsile on võimalik juurde lisada ka argument, mis näitab, mitu sammu tegelane astuma peab (joonis 9). See sarnaneb tekstipõhistes programmeerimiskeeltes funktsioonide kasutamisega, kus argumentideks saab anda erinevaid väärtusi. Kui liikumisklotsile argumenti lisatud pole, kasutatakse vaikimisi väärtusena arvu üks.



Joonis 10. Näide tingimuslause ja tsükli struktuurist

Tingimuslause on loodud mitme klotsi kombineerimisega (joonis 10). Esmalt on klots tingimuse väljendamiseks. Käesolevas töös on tingimustena kasutusel seina, lõksu ning porgandi olemasolu kontrollimine tegelase ees. Käsu tulemuseks on tõeväärtus.

Tingimuse käsule järgneb klots, mis visualiseerib tõeväärtuse kaht erinevat tulemust („JAH", „EI") ning suunab programmi täitmise vastavasse harusse. Seejärel saab kasutada ükskõik mitut vaheklotsi ning lõpuks peab olema klots, mis suunab haru paralleelseks eelnevate käsujadadega. Selline vaheklotside kasutamine võimaldab muuta tingimuslause harude kõrgusi ning luua ükskõik kui palju erinevaid hargnemisi, ilma et need omavahel ristuksid. Tingimuslause on jagatud mitmeks erinevaks käsuks kahel eesmärgil. Neist esimene on visualiseerida funktsioonide kasutamist. Tingimuslauset võib vaadata kui funktsiooni, millele saab anda erinevaid argumente ning mis tagastab tõeväärtuse. Prototüübis on saavutatud sama efekt tingimuste vahetamise võimalusega hargnemisklotsi ees. Teine eesmärk on tõsta kasutusmugavust. Kui kasutaja soovib tingimust muuta, on lihtsam asendada klotsi, mis on ühendatud kahe klotsiga, kui klotsi, millega on seotud kolm erinevat käskude jada.

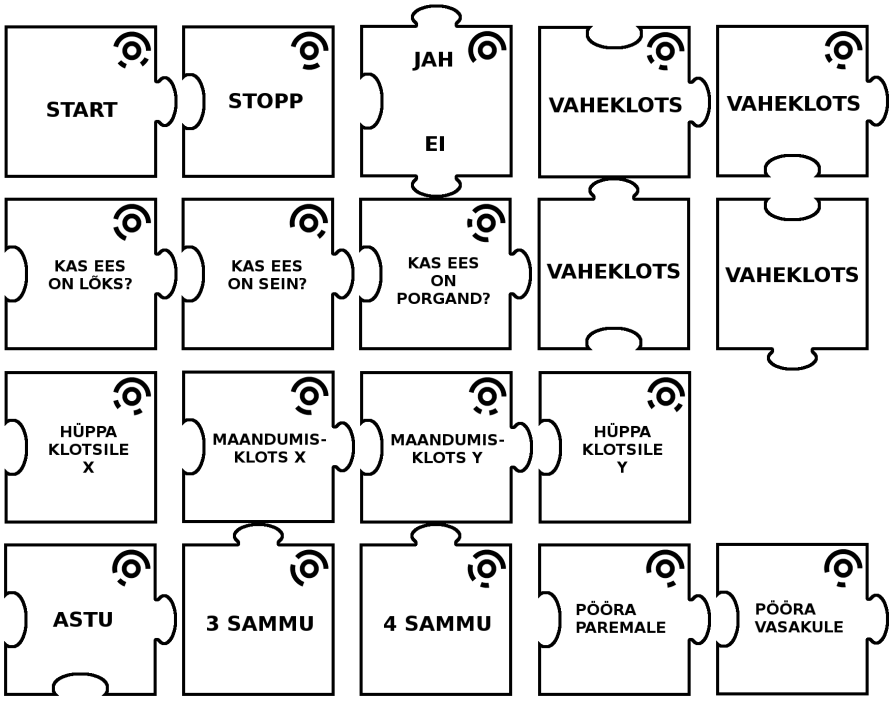
Tsükkel on saavutatud hüppamis- ja maandumiskäskude ning tingimuslause abil (joonis 10). Nende abil konstrueeritud tsükkel meenutab tekstipõhistes programmeerimiskeeltes kasutatavat *while*-tsüklit. Esmalt on vaja programmis paika panna maandumiskäsk. Sellele võib järgneda suvaline arv lihtkäske, kuid lõpmatu tsükli vältimiseks peab mingil hetkel tulema tingimuslause. See määrab ära, mis tingimusel tsüklit jätkatakse. Tingimuslause üks haru peab lõppema hüppamiskäsuga, mille juurest hüpatakse programmi täitmise ajal tagasi maandumisklotsi juurde ning hakatakse sealt taas programmi täitma. Tingimuslause teine haru jätkab tsüklijärgse loogikaga. Keel ei sea tsüklite loomisele erilisi piiranguid. Ühe maandumiskäsuga võib seotud olla mitu hüppamiskäsku ning tsükleid võib konstrueerida ka üksteise sisse. Erinevate tsüklite loomine on saavutatud maandumis- ja hüppamiskäskude märgendamisel tähtedega.

3.1.2 Klotside kuju ja disain

Klotsid on kujundatud programmis GIMP [2], mis võimaldab piksli täpsusega erinevad elemendid paika seada. Klotside loomiseks on tehtud mall, kus kõik erinevad klotsi osad on eraldi kihtidena. See võimaldab ühest mallist vähese vaevaga luua erinevaid klotsitüüpe.

Klotside kujuga saavutatakse kolm peamist eesmärki. Esmalt peeti klotsi kuju disainides silmas, et klotse oleks mugav omavahel ühendada ning programmi alusel liigutades see koost ei laguneks. Seetõttu kasutati klotside jaoks pusletükkidega sarnast kujundust, kus klotsid asetatakse osaliselt üksteise sisse. Nii püsivad klotsid omavahel koos ka neid ringi lohistades. Teine eesmärk oli klotsi kujuga visualiseerida programmi kulgemise suunda. Klotsi väljaulatuv osa moodustab klotsile kujutletava noole. Klots, millel paremal väljaulatuv osa puudub, lõpetab haru ning ka mõttelise noole. Tingimuslause klots lõpetab ühe haru, kuid suunab programmi edasi kahte erinevasse harusse. Seetõttu on klotsil ka kaks väljaulatuvat osa - üleval ja all. Kolmandaks üritatakse klotside kujuga anda kasutajale vihjeid programmi koostamise loogika kohta ning selle abil minimeerida võimalike süntaksivigade arvu. Näiteks alustamis- ja lõpetamiskäskudele

saab klotse juurde lisada ainult vastavalt taha või ette ning see kirjeldab ka nende käskude sisulist poolt. Samuti on liikumiskäsul alumises servas süvend, mis näitab, et antud käsule saab ka alla klotsi juurde lisada. Nii üritatakse kasutajale vihjata, et sellisele käsule on võimalik ka argument lisada.



Joonis 11. TopCode sümbolitega klotsid

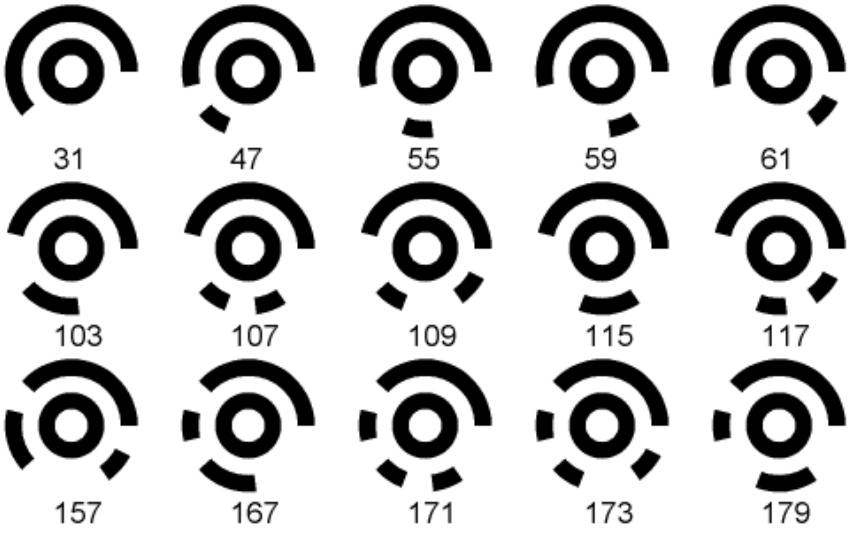
Klotsidele on lisatud vastava käsu kirjeldus. Kirjeldus antakse teksti kujul, et käsk oleks kasutajale üheselt mõistetav. Ainult sümboli või kujundi kasutamine ei pruugi olla kõigile arusaadav ning kaks erinevat inimest võivad tõlgendada üht sümbolit erinevatel viisidel. Lisaks tekstilisele kirjeldusele on klotsi üleval paremas nurgas ka sümbol, mis on vajalik pildianalüüsi jaoks. Käesolevas töös on loodud kaks komplekti klotse, mis erinevad üksteisest kasutatavate sümbolite poolest. Ühel komplektil on sümboliteks TopCode teegi koodid (joonis 11) ning teisel on nendeks autori loodud ikoonid, mis visualiseerivad käskude tähendusi.

3.2 Pildianalüüs

Pildianalüüs on tehtud OpenCV teegiga [11], mis võimaldab pilte töödelda, tuvastada piirjooni, leida pildilt sümboleid ja palju muud. Käesolevas töös on loodud kaks erinevat klotside komplekti, seega on neist mõlemale kirjutatud vastav pildianalüüsi loogika.

3.2.1 TopCode teek

TopCode pildianalüüsi teek [18] võimaldab sisendpildilt kiiresti tuvastada tasasele pinnale asetatud objekte, kui nende peal on TopCode koodid. Koodid koosnevad kahest mustast ringist valgel taustal, millest välimisel on igal koodil erinevad sektorid välja lõigatud (joonis 12). Igale sümbolile vastab unikaalne kahe- või kolmekohaline arv. Erinevaid sümboleid on kokku 99.



Joonis 12. Valik TopCode sümbolitest koos neile vastavate arvudega [17]

TopCode pildianalüüs töötab sõltumatult sisendiks oleva pildi resolutsioonist, kui sümboli suurus on vähemalt 25 x 25 pikslit. Pildianalüüs töötab sõltumatult koodi orientatsioonist ning sümboleid suudetakse tuvastada erinevates valgustingimustes tehtud piltidelt.

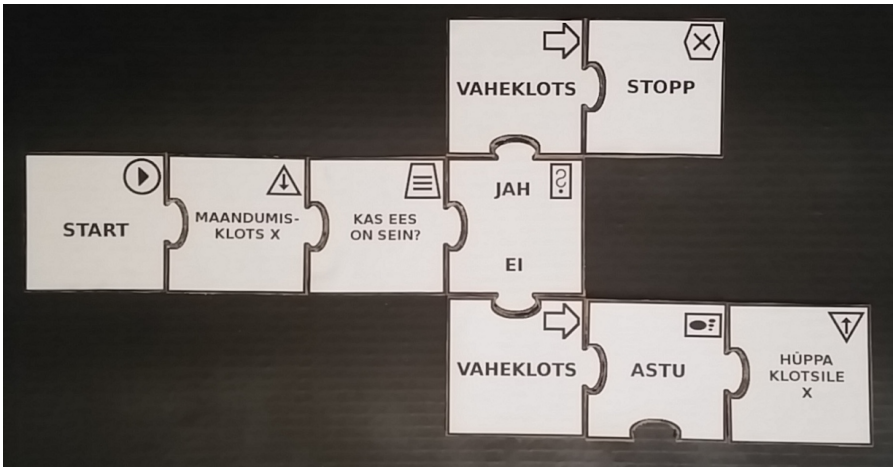
Igale klotsile on pandud unikaalne TopCode kood. Seejärel on programmis kindlaks määratud, missugusele klotsile milline TopCode arv vastab. Kui rakendus käivitatakse ja kasutaja valib sobiva programmipildi, antakse pilt TopCode teegi vastavale meetodile sisendiks. Kasutaja valitud pilt ei vaja eeltöötlemist, vaid on kohe sobival kujul argumendiks andmiseks. Meetod tuvastab pildilt kõik sümbolid ning tagastab järjendi TopCode objektidest. Iga sellise objekti kohta saab küsida koodi arvu, asukohta, orientatsiooni ja läbimõõtu.

TopCode sümbolitega klotside puhul saab toetada pildi erinevaid orientatsioone. Selleks, et kasutada üht programmipuu koostamise loogikat mõlema pildianalüüsi variandi jaoks, on vaja saada klotside asukoht just selliselt pildilt, kus alustamiskäsk on kõige vasakpoolsem ning programmi voog liigub vasakult paremale. Seega on vaja TopCode

objektide asukohti teisendada. Selleks luuakse kõigepealt pildi pööramise jaoks maatriks suurusega 2 x 3 (*rotation matrix*). Pööramismaatriks luuakse vastavalt algpildile ning TopCode sümbolite orientatsioonile. Seejärel käiakse läbi leitud TopCode sümbolid ning korrutatakse nende koordinaadid pööramise maatriksiga. Selle tulemusena saame leitud TopCode objektide koordinaadid horisontaalsihis. Nüüd luuakse järjend paaridest, mis sisaldavad endas klotsitüüpi ning asukohta maatriksis. See järjend antakse edasi programmipuu koostamise protseduurile.

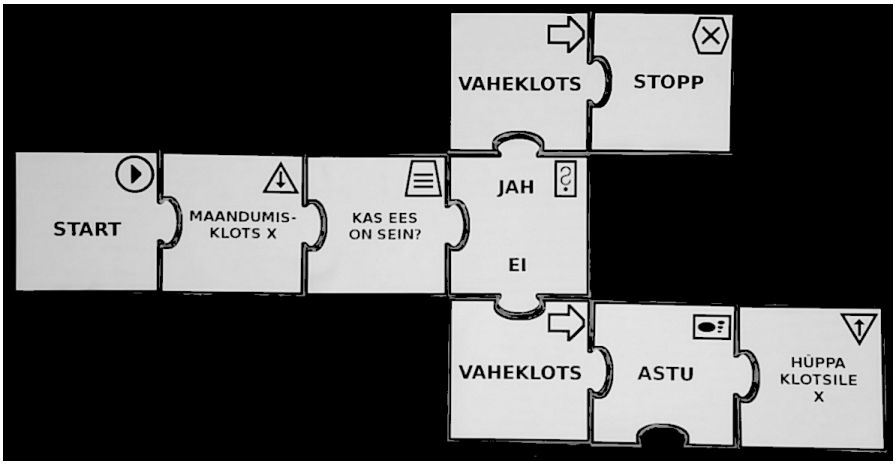
3.2.2 Ikoonipõhise lähenemise pildianalüüs

Ikoonipõhine lähenemine seisneb eelnevalt salvestatud sümbolite otsimisel sisendiks saadud pildilt. Esmalt on vaja luua unikaalsed ikoonid, mis oleksid üksteisest piisavalt erinevad, kuid kirjeldaksid võimalikult hästi klotsile vastavat käsku. Sümbolid koosnevad kinnisest piirjoonest, mis loob igale klotsile unikaalse kujundi. Kujundi sees on enamasti veel täiendav ikoon käsu tähenduse visualiseerimiseks. Esimeses variandis olid kõik sümbolid küll unikaalsed, kuid välimiseks kujundiks oli kõigil ruut. Nii ei suutnud pildianalüüs sümbolitel aga piisavalt vahet teha ning selle tõttu loodi kõigile unikaalne välimine kujund. Igast sümbolist on tehtud pilt ning klotsidele on koodis määratud vastava ikooni pildifaili nimi.



Joonis 13. Sisendiks saadav programmipilt

Sisendiks saadud pilti (joonis 13) on vaja esmalt töödelda. Kuna sümbolite pildid on must-valged, kaotatakse ka programmipildilt värvid ja teisendatakse see halltoonidesse. Seejärel kaotatakse ka halltoonid ning iga piksel muudetakse kas mustaks või valgeks vastavalt tema eelmisele väärtusele (joonis 14). Lisaks sisendpildi töötlemisele muudetakse ka ikoonide piltide suurust vastavalt sisendpildil olevate sümbolite suurusele.



Joonis 14. Töödeldud sisendpilt

Nüüd otsitakse programmipildilt alustamiskäsu sümbolit ja lõigatakse selle lõpust alates pildist välja horisontaalne ristkülik (joonis 15). Väljalõigatud osalt otsitakse kõiki ikoone, välja arvatud alustamis- ja liikumiskäsu argumentide sümboleid. Iga leitud ikooni korral kaetakse pildil sümbolile vastav osa musta ruuduga, et sama ikooni rohkem ei tuvastataks (joonis 15). Kuna leitud ikoonide koordinaadid vastavad välja lõigatud pildile, on vaja need teisendada esialgsele sisendpildile. Kui leitud sümbolite seas oli liikumiskäsu ikoon, lõigatakse tema alt välja ristkülik, kust otsitakse ainult sammude argumentide sümboleid. Koordinaatidele rakendatakse jällegi teisendust. Kui leitud ikoonide seas oli tingimuslause sümbol, lõigatakse esmalt välja vertikaalne ristkülik. Sellelt otsitakse vaheklotsi ikoone. Seejärel lõigatakse nendest alates välja horisontaalsed ristkülikud ning jätkatakse sümbolite leidmist rekursiivselt. Selline väikeste pilditükkide lõikamine ja nendelt ikoonide otsimine vähendas pildianalüüsile kuluvat aega mitu korda, võrreldes sümbolite otsimisega kogu sisendpildilt.

Joonis 15. Pildilt välja lõigatud ristkülik, millelt on kõik ikoonid tuvastatud

Pildianalüüsi on lisatud ka meetodid, mis võimaldavad leida sisendpildilt sümbolite suurust ning orientatsiooni. Mõlema puhul töötab loogika sarnaselt. Kõigepealt üritatakse pildilt leida algussümbol ning jäetakse parim tulemus meelde. Sobiva orientatsiooni otsimise korral pööratakse pilti 10 kraadi võrra ning korratakse ikooni leidmise protse-

duuri. Tegevust jätkatakse, kuni ring on peale tehtud ning parim tulemus ja nurk leitud. Sümbolite suuruse leidmise korral alustatakse mõõdust 500 x 500 pikslit ning iga tsüklikorraga vähendatakse seda 0,05 korda. Küll aga on need meetodid üsna ajakulukad, mistõttu pole nimetatud meetodeid pildianalüüsis praegu kasutatud. Näiteks sümboli suuruse leidmise meetodi kasutamise korral kestab pildianalüüs keskmiselt 10,4 sekundit. Seega töötab pildianalüüs hetkel piltidega, millel sümbolite suurus on 50 x 50 pikslit ning orientatsioon on sama nagu projektis salvestatud ikoonide piltidel.

3.2.3 TopCode teegi ja ikoonipõhise lähenemise võrdlus

TopCode teegi suurimaks plussiks on kiire sümbolite tuvastamine erinevate resolutsioonide ja orientatsioonidega piltidelt. Kui projekti edasi arendada mobiilirakenduseks, on erinevate resolutsioonide ja orientatsioonide toetamine kriitilise tähtsusega. Plussiks on ka pildianalüüsi loogika implementeerimise lihtsus. Pilt ei vaja töötlemist ning kõik vajalikud meetodid on teegis olemas.

Ikoonipõhise lähenemise suurimaks plussiks on kasutusmugavuse tõstmine. TopCode sümbol on kasutajale arusaamatu ning ei anna klotsi käsu mõistmisele midagi juurde. Kui tegu on lastega, kelle lugemisoskus ei pruugi veel väga hea olla, on neil TopCode koodidega klotside tähendusi keeruline mõista ja motivatsioon rakenduse katsetamiseks langeb. Ikoonipõhine lähenemine lahendab selle, kasutades klotside tähendusi visualiseerivaid sümboleid. Need täiendavad klotside tekstilist kirjeldust ning aitavad klotsi tähendust mõista ka juhul, kui teksti lugemine tekitab raskusi. Samuti on plussiks see, et sümbolite arvul puudub piirang, kuna neid saab lihtsa vaevaga järjest juurde luua. TopCode teek võimaldab luua maksimaalselt 99 erinevat klotsi.

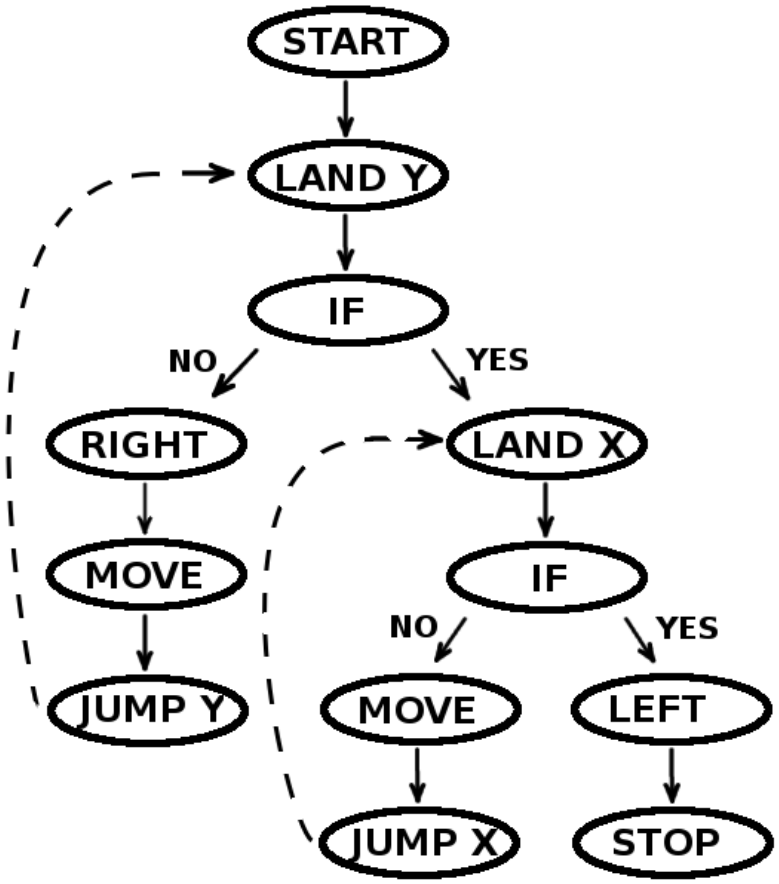
TopCode teegi ja autori loodud sümbolitega pildianalüüsi kestust mõõdeti ühe ja sama 2048 x 1536 resolutsiooniga pildiga samades oludes sada korda. TopCode teegiga pildianalüüs kestab keskmiselt 0,427 sekundit ning ikoonipõhine tuvastamine keskmiselt 0,965 sekundit. Kui projekti edasi arendada eksponaadiks, pole erinevate resolutsioonide toetamine nii oluline, kuna eksponaadis saab kasutada statsionaarset veebikaamerat. Ühtlasi on võimalik alustamisklots kinnitada programmi ladumise aluse peale ning seeläbi kaotada vajadus erinevate orientatsioonide toetamiseks. Sellise eksponaadi ja 2048 x 1536 resolutsiooni puhul on võimalik ikoonipõhise lähenemise puhul 0,5-sekundilise ajakaotusega võita oluliselt kasutusmugavuses.

3.3 Programmi süntaksipuu

Süntaksipuu genereerimise protseduur saab sisendiks erinevate käskude asukohad pildil. Eesmärk on genereerida kahendpuu, mis kirjeldab loodud programmi. Valmis puu antakse edasi rakendusele, mis tegeleb programmi tulemuse visualiseerimisega.

Süntaksipuu tippudeks on käskudele vastavad objektid (joonis 16). Puu juureks on alustamisklotsile vastav isend. Vahetippudeks võivad olla liikumis-, pööramis-, maan-

dumiskäsk või tingimuslause. Lehtedeks on enamasti hüppamis- või lõpetamiskäsud. Igal tipul on viidad vanemale ning alluvatele, kui need eksisteerivad. Tingimuslause tipp on ainus, millel on kaks alluvat: vasak alluv vastab väärale ning parem tõesele tõeväärtusele. Liikumiskäsul hoitakse meeles ka sammude arvu, tingimuslausel tingimust, maandumisklotsil tähte, hüppeklotsil tähte ja viita vastavale maandumisklotsile ning pööramiskäsul suunda.



Joonis 16. Näide süntaksipuust

Puu genereerimise loogika algab sisendiks saadud järjendi sorteerimisega. Järjendis on paarid, mis sisaldavad endas klotsi tüüpi ning selle asukohta pildil. Paarid sorteeritakse asukoha x-koordinaadi järgi. Seejärel võetakse esimene paar, mille klotsitüübiks on alustamisklots, ning leitakse kõik klotsid, mis on sellega samas reas. Ühtlasi kontrollitakse selle käigus, et kahe klotsi vaheline vahemaa poleks liiga suur. Nii leitakse klotsid, mis on kindlasti alustamisklotsist algavas jadas, ja see lubab sisendpildil olla ka klotsidel, mida tegelikult programmis kasutatud pole. Nüüd alustatakse rekursiivselt

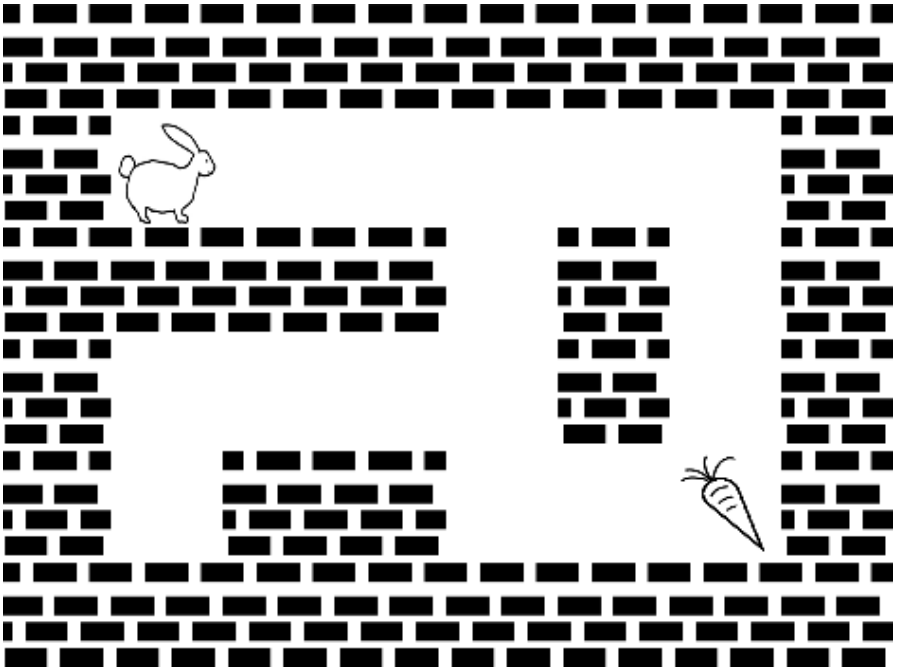
süntaksipuu loomist. Kõik ühel real olevad klotsid käiakse läbi ning vajadusel lisatakse puusse käsule vastav tipp. Tingimust väljendava klotsi korral jäetakse see meelde ning liikumiskäsu puhul üritatakse selle alt leida argumenti. Kui klotsiks on hargnemisklots, leitakse esmalt selle ülemine ja alumine vaheklots. Seejärel leitakse mõlema vaheklotsiga samas reas olevad käsud ning alustatakse tööd rekursiivselt uuesti. Samuti hoitakse jooksvalt meeles maandumiskäsule vastavaid tippe ning hüppamisklotsi juures leitakse mälust talle vastava tähega maandumistipp.

3.4 Programmi tulemuse visualiseerimine rakenduses

Kasutaja programmi tulemust visualiseeritakse mängus, mille eesmärk on juhatada jänes läbi labürindi porgandi juurde. Rakendus on loodud JavaFX teegiga [9]. Rakenduse käivitamisel avaneb kasutajale esmalt avaleht (joonis 17), kus kirjeldatakse mängu ideed ning antakse edasised juhised. Kasutajal on vaja seejärel valida tase. Tasemed lähevad järjest keerulisemaks ning lasevad kasutajal järk-järgult katsetada keerulisemate struktuuride koostamist. Lisaks peab kasutaja täpsustama ka kasutatavate klotside tüübi. Kui vajalikud valikud on tehtud, saab mängu alustada.

Joonis 17. Rakenduse avaleht

Kasutajale kuvatakse valitud tasemele vastav maailm (joonis 18) ning antakse valik kõigist valitud klotsitüübiga projekti salvestatud programmipiltidest. Seejärel peab kasutaja valima sobiva programmi ning vajutades alustamisnuppu antakse valitud pilt edasi pildianalüüsile. Kasutajal on valida kahe erineva programmi käivitamise võimaluse vahel. Üks variant on lasta programmi automaatselt täita ning teine võimalus on



Joonis 18. Näide tasemele vastavast maailmast

kasutajal ise samm-sammult programmi läbida. Mõlemal juhul antakse kasutajale teada, et programmi hakati koostama ning see võib võtta aega. Programmi täitmine seisneb selle süntaksipuu läbimises. Kui tegu on jänese liigutamise või pööramise käsuga, muudetakse vastavalt tegelase suunda või asukohta maailmas. Iga tipu juures näidatakse programmi pildil ka vastava klotsi juures punast täppi, mis näitab kasutajale jooksvalt, kui kaugel programmi täitmisega ollakse. Kui tegu on automaatse käivitamisega, ootab programm iga tipu vaatlemise vahepeal 0,7 sekundit, et kasutajal oleks aega muudatusi registreerida.

Korrektse programmi süntaksipuu läbimine võib lõppeda kolme erineva tulemusega. Esmalt võis jänes jõuda porgandini ning sellisel juhul kuvatakse kasutajale teade taseme edukast sooritamisest. Porgandist läbi liikumist ei loeta taseme edukaks läbimiseks. Teine võimalik lõpptulemus on jänese liikumine vastu seina või lõksu sattumine. Sellisel juhul teavitatakse kasutajat tekkinud veast ning taset tuleb uuesti alustada. Lõpuks on võimalik, et jänes ei jõudnud porgandini ega sattunud ka lõksu ega vastu seina. Nüüd on kasutajal võimalik valida uus programmi pilt ning jätkata taseme läbimist. Iga lõpptulemuse korral on kasutajal võimalik alati liikuda tagasi avalehele, eelmise või järgmise taseme juurde. Samuti on alati võimalik taset uuesti alustada.

Pildianalüüsi tegemisel ning programmi puu koostamisel võidakse programmist tuvastada viga ainult juhul, kui pildil puudub alustamisklots. Sellisel juhul antakse ka-

sutajale vastav teade ning võimalus valida uus programmipilt. Samuti on võimalik, et kasutaja programm pole korrektne. Programmi täitmisel võib tekkida olukord, kus mõnel hüppeklotsil puudub vastav maandumisklots või hargnemisklotsil puudub eelnev tingimust väljendav klots. Probleemi tekkimisel teavitatakse sellest kasutajat ning antakse võimalus taset uue pildiga uuesti alustada. Kui programmis tekib lõpmatu tsükkel, mis ei lõpe jänese lõksu või vastu seina sattumisega, jäädakse programmi täitma, kuni kasutaja sellele ise reageerib. Nimelt on automaatse programmi täitmise ajal võimalik see protsess alati ise lõpetada. Manuaalse läbimise korral saab programmi täitmise ajal alati taset uuesti alustada.

3.5 Projekti üles seadmine

Järgnevas peatükis kirjeldatakse, kuidas loodud projekti üles seada ning käivitada. Eraldi juhendid antakse Windowsi ning Linuxi keskkonna jaoks. Ühtlasi on kasutajal võimalik rakendust ehitada ka lähtekoodist, mille jaoks antakse samuti õpetus. Juhendites eeldatakse, et kasutajal on olemas Java 8 arenduskeskkond [8]. Tähtis on, et arenduskeskkond sisaldaks ka JavaFX teeki [9]. Lähekoodi saab alla laadida GitHubist:

```
https://github.com/THelena/TangibleProgramming.git
```

Soovi korral on võimalik rakendust käivitada ka virtuaalmasinas. Selleks on loodud Ubuntu 16.04 virutaalmasin, kus on kogu projekt eelnevalt üles seatud ning mille saab alla laadida lingilt:

```
https://www.dropbox.com/s/rdxun207xjynphj/
        TangibleProgramming.ova?dl=0
```

Virtuaalmasina salasõna on `tangible`. Rakenduse lihtsaks käivitamiseks on kodukausta loodud `tangibleProgramming.sh` skript. Selle jooksutamiseks on vaja avada käsurida ning jooksutada seal järgnevat käsku:

```
./tangibleProgramming.sh
```

Kui virtuaalmasina käivitamisel tekib viga, siis on tarvis muuta virtuaalmasina USB kontrolleri sätteid. Selleks tuleb avada loodud virtuaalmasina seaded, navigeerida USB sätete juurde ning valida sealt USB 1.1.

3.5.1 Juhend Windowsi jaoks

Järgnevat juhendit on testitud Windows 8 ning Windows 10 peal. Projekti jooksutamiseks on vajalik OpenCV teek versiooniga 3.2.0, mille saab alla laadida OpenCV koduleheküljelt [12]. Sealt saab alla laadida `.exe` vormingus faili (*Win pack*), mida jooksutades installitakse teek.

Projekti juurkataloogis on eelnevalt kompileeritud rakenduse kokkupakitud `.jar` fail. Jooksutamiseks on vaja avada käsuviip (*command prompt*) ning navigeerida projekti juurkataloogi. Seejärel saab järgmise käsuga rakenduse käivitada:

```
java -Djava.library.path=C:\path\to\opencv\build\java\x64
-jar tangible-programming-1.0-jar-with-dependencies.jar
```

Oluline on, et käsus oleva `java.library.path` parameetri väärtuseks antav tee (*path*) oleks muudetud vastavalt OpenCV teegi installatsiooni kaustale enda keskkonnas. Tee peaks viitama Java liidese jaoks genereeritud `.dll` teeki sisaldavale kaustale, mis 64-bitilise operatsioonisüsteemi puhul on vaikimisi `opencv\build\java\x64`.

3.5.2 Juhend Linuxi jaoks

Järgnevat juhendit on testitud Ubuntu 16.04 peal. Linuxi jaoks ei ole saadaval eelnevalt kompileeritud OpenCV teeki nagu Windowsi jaoks. Seetõttu tuleb OpenCV 3.2.0 lähtekoodist ise ehitada. Selle juhendi leiab OpenCV dokumentatsioonist:

```
http://docs.opencv.org/2.4/doc/tutorials/introduction/
            linux_install/linux_install.html
```

Kindlasti tuleks kontrollida enne juhendis toodud `make install` käsu jooksutamist, et Java oleks installitud ning `JAVA_HOME` keskkonnamuutuja oleks defineeritud vastavalt Java JDK installatsiooni kaustale. Lisaks on vaja veenduda, et Apache Ant [1] teek oleks installitud. Eelnevate puudumisel ei ehitata Java jaoks vajalikku OpenCV liidest.

Pärast edukat `make install` käsu jooksutamist tekivad kompileerimise väljundkausta `lib/` kataloogi vajalikud `.so` laiendiga teegid. Seejärel saab rakendust jooksutada sama käsuga nagu Windowsis. Siin tuleks `java.library.path` parameetri väärtuseks anda eelnevalt tekkinud `lib/` kausta tee.

3.5.3 Rakenduse ehitamine lähtekoodist

Järgnevat juhendit on testitud Ubuntu 16.04 peal. Rakenduse saab edukalt lähtekoodist kokku panna ka oma keskkonnas. Selleks on tarvis, et oleks Maven 3.x eelnevalt installitud ning `PATH` keskkonnamuutujas saadaval. Projekti kompileerimiseks ning kokkupakkimiseks on eelnevalt tarvis lokaalsesse Maveni repositooriumisse installeerida kaks vajalikku sõltuvust, mis on projektiga kaasa pandud. Selleks tuleb projekti juurkataloogis jooksutada järgmisi käske:

```
mvn install:install-file -Dfile=./lib/topcodes.jar
-DgroupId=topcodes -DartifactId=topcodes -Dversion=1.0.0
-Dpackaging=jar
```

```
mvn install:install-file -Dfile=./lib/opencv-320.jar
-DgroupId=org.opencv -DartifactId=opencv -Dversion=3.2.0
-Dpackaging=jar
```

Pärast nende sõltuvuste installimist saab rakenduse kokku ehitada järgmist käsku projekti juurkataloogis jooksutades:

```
mvn clean package
```

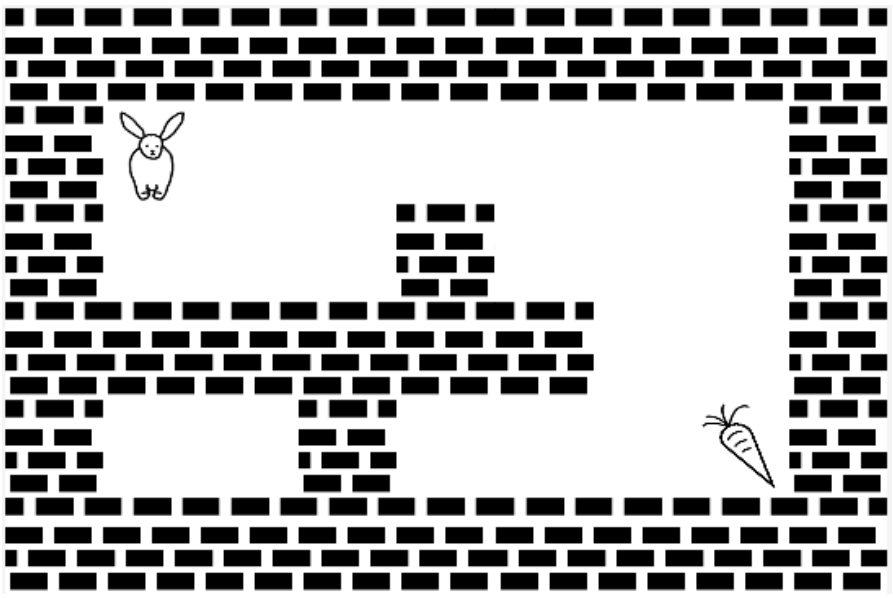
Eelneva tulemusel tekib `target/` kausta kaks `.jar` laiendiga faili, millest

```
tangible-programming-1.0-jar-with-dependencies.jar
```

sisaldab ka vajalikke TopCode ning OpenCV teekide sõltuvusi.

3.6 Prototüübi kasutamise juhend näite põhjal

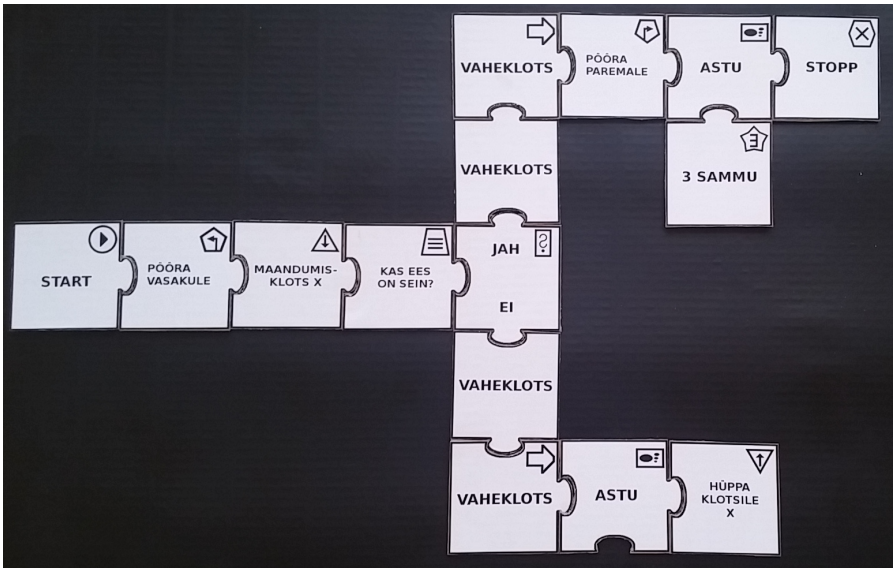
Olgu rakendus käivitatud, klotsitüübiks valitud ikoonipõhine komplekt ning raskuseks määratud seitsmes tase. Pärast alustamisnupu vajutamist avaneb kasutajale vastava taseme vaade (joonis 19). Seejärel on vaja koostada programm, mis juhataks jänese porgandini.



Joonis 19. Seitsmendale tasemele vastav maailm

Esmalt on vaja paika panna alustamiskäsk. Nüüd peab jänes pöörama vasakule, et ta saaks liikuma hakata. Seejärel on vaja otsustada, kuidas tegelane seinani liigutada.

Variante on selleks kolm: kasutada mitut liikumiskäsku järjest, lisada liikumiskäsule argument või kasutada tsüklit. Esimene variant pole mõistlik, kuna see eeldaks kuue sama käsu järjestikust rakendamist. Argumendi kasutamine pole hetkel võimalik, kuna puudub klots numbri 6 jaoks. Seega on hea kasutada tsüklit, mis kestab seni, kuni jänes on jõudnud seinani. Selleks on vaja programmi lisada maandumisklots ning pärast seda seina olemasolu kontrolliv tingimuse klots koos hargnemisklotsiga. Vaheklotse kasutades saab valida harude jaoks sobiva kõrguse. Väärale tõeväärtusele vastav haru tegeleb tsüklisisese loogikaga. Seega on vaja selles jänest ühe sammu võrra edasi liigutada ning seejärel tagasi tsükli algusesse hüpata. Tõesele tõeväärtusele vastav haru tegeleb tsüklijärgse loogikaga. Kui jänes on seinani jõudnud, on vaja pöörata paremale. Seejärel on vaja liikuda otse porgandini ning selle jaoks on jällegi kolm võimalikku varianti. Kuna vaja on astuda kolm sammu, saab siin mugavalt kasutada liikumiskäsku koos argumendiga. Nüüd on jänes jõudnud porgandini ning harusse saab lisada lõpetamiskäsu. Valminud on programm (joonis 20), mis lahendab seitsmenda taseme.



Joonis 20. Seitsmenda taseme lahendusprogramm

Nüüd on vaja valminud programmist teha pilt ja salvestada see projektis vastavasse kausta või kasutada juba olemasolevaid pilte. Kui nende seas täpselt õiget programmi pole, saab vaadata, kas leidub mõni alternatiiv. Kuna ühele tasemele on võimalikke lahendusi palju, võib leiduda pilt, mis annab loodud programmiga sama tulemuse, kuid on erineva süntaksiga. Samuti on võimalik uurida, kas mitu olemasolevat pilti annavad järjestikusel rakendamisel kokku sama tulemuse. Kui tase on edukalt läbitud, saab liikuda tagasi avalehele või järgmise taseme juurde.

4 Edasised arendamisvõimalused

Projekt on üles ehitatud eesmärgiga võimaldada seda edasi arendada kahes suunas: eksponaadiks ning Androidi mobiilirakenduseks. Seetõttu on pildianalüüsi, programmi süntaksipuu ning tulemuse visualiseerimise loogikad üksteisest eraldatud. Samuti võiks ikoonipõhist pildianalüüsi edasi arendada.

Autori disainitud ikoonid on kõik sarnase ülesehitusega - iga sümbol sisaldab kinnist piirjoont, mis loob unikaalse kujundi. Pildianalüüsi võiks proovida edasi arendada just selles suunas, et tuvastada pildilt piirjooni ning nendest moodustatud kujundeid. Selline lähenemine peaks lahendama ka resolutsiooni ja orientatsiooni piirangud praegusel lahendusel.

Loodud süsteem on eksponaadina kasutamiseks suures osas valmis. Vaja on muuta veel programmipildi valimise loogikat. Selleks on vaja kasutada veebikaamerat ning teeki, mis võimaldab pilti teha ning seda salvestada. Selleks sobib ka TopCode teek. Rakendusest on vaja eemaldada eelnevalt salvestatud piltide näitamine ning lisada veebikaamerast pildi saamine. Lisaks on saadud pilti vaja kuvada ka programmi täitmise ajal. Teine võimalus on lisada eksponaadile laser, mis osutab klotsile, mida parasjagu täidetakse.

Mobiilirakenduse jaoks on esmalt vaja lahendada pildi resolutsiooni ja orientatsiooni piirangu probleem. Seejärel on vaja luua rakendus, milles programmi tulemust visualiseeritakse. Kuna pildianalüüsi ja süntaksipuu tegemise loogikad on kirjutatud Javas ja eraldatud praegusest tulemuse visualiseerimise rakendusest, saab neid kasutada ka Androidi rakenduses.

5 Kokkuvõte

Käesoleva töö raames valmis eestikeelne prototüüp füüsiliste klotsidega programmeerimiseks. Sellised süsteemid soosivad programmeerimisega tutvumist ning julgustavad inimesi seda proovima.

Prototüübis loodud programmeerimiskeel võimaldab luua lihtkäskudest jadasid, tingimuslauseid ja tsükleid ning kasutada argumente. Lihtkäsud kujutavad endast klotse astumise ning pööramise jaoks ja tingimuslause on saavutatud tingimust väljendava klotsi ning hargnemisklotsi kombineerimisel. Tsüklit kujutavad hüppamis- ning maandumiskäsud.

Pildianalüüsi jaoks loodi kaks erinevat klotside komplekti, mis erinevad klotsituvastuseks kasutatava sümboli poolest. TopCode teegi suurimaks plussiks on kiire koodide leidmine erinevate pildi resolutsioonide ja orientatsioonide puhul. Suurimaks miinuseks on sellise lähenemise puhul kasutaja jaoks arusaamatu kood klotsil, mis käsu kirjeldust edasi ei anna. Ikoonipõhine lähenemine parandab selle probleemi ehk klotsidel olevad sümbolid kirjeldavad vastavat käsku. Selle lähenemise miinuseks võib lugeda, et see ei toeta erinevaid resolutsioone ja orientatsioone või teeb seda suure ajakuluga.

Tööd on võimalik mitmes suunas edasi arendada. Esimene variant on klotside tuvastamist arendada ning proovida ikoonipõhises pildianalüüsis kontuuride abil leida igal klotsil olevad unikaalsed kujundid. Lisaks on võimalik prototüüpi edasi arendada eksponaadiks või mobiilirakenduseks. Eksponaadi puhul on vaja toetada veebikaamerast pildi saamist ning mobiilirakenduse korral tuleb tulemuse visualiseerimiseks luua eraldi Androidi rakendus, mis saab kasutada olemasolevat pildianalüüsi ja programmi genereerimise loogikaid.

Viidatud kirjandus

[1] Apache Ant. http://ant.apache.org/ (08.05.2017)

[2] GNU Image Manipulation Program (GIMP) https://www.gimp.org/ (01.05.2017)

[3] Horn M., Jacob R. J. K. Designing Tangible Programming Languages for Classroom Use. http://hci.cs.tufts.edu/tern/horn-jacob-tei07.pdf (30.04.2017)

[4] Horn M., Jacob R. J. K. Tangible Programming in the Classroom with Tern. 2007. http://www.cs.tufts.edu/~jacob/papers/chi07.horn.pdf (01.04.2017)

[5] Horn M., Solovey E. T., Jacob R. J. K. Tangible Programming and Informal Science Learning: Making TUIs Work for Museums. http://tidal.northwestern.edu/media/files/pubs/idc08.pdf (07.05.2017)

[6] Howard M. I Robot, You Programmer. 2008. http://tuftsjournal.tufts.edu/2008/10/features/01/ (07.05.2017)

[7] Hu F., Zekelman A., Horn M. S., Judd F. Strawbies: Explorations in Tangible Programming. 2015. http://strawbies.com/idc2015.pdf (01.04.2017)

[8] Java SE Development Kit 8 Downloads. http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html (08.05.2017)

[9] JavaFX 8. Overview. http://docs.oracle.com/javase/8/javafx/api/toc.htm (10.05.2017)

[10] Malan D. J., Leitner H. H. Scratch for Budding Computer Scientists http://cs.harvard.edu/malan/publications/fp079-malan.pdf (01.04.2017)

[11] OpenCV. http://opencv.org/ (01.05.2017)

[12] OpenCV. Releases. http://opencv.org/releases.html (08.05.2017)

[13] Osmo. https://www.playosmo.com/en/ (01.05.2017)

[14] Reference Guide. https://download.scratch.mit.edu/ScratchReferenceGuide14.pdf (01.04.2017)

[15] Resnick M., Maloney J., Monroy-Hernández A., Rusk N., Eastmond E. jt. 2009. Scratch: Programming for all http://web.media.mit.edu/~mres/papers/Scratch-CACM-final.pdf (01.04.2017)

[16] Scratch. https://scratch.mit.edu/ (01.05.2017)

[17] TopCodes. http://users.eecs.northwestern.edu/~mhorn/topcodes/topcodes.pdf (01.04.2017)

[18] TopCodes. Tangible Object Placement Codes. http://users.eecs.northwestern.edu/~mhorn/topcodes/ (01.04.2017)

[19] Wilson A., Moffat D. C. Evaluating Scratch to introduce younger schoolchildren to programming. http://scratched.gse.harvard.edu/sites/default/files/wilson-moffat-ppig2010-final.pdf (03.04.2017)

Lisad

I. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, **Helena Talimaa**,

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

   Süsteem füüsiliste klotsidega programmeerimiseks

   mille juhendaja on Aivar Annamaa

   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. 3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, 11.05.2017