Tallinna Ülikool
Informaatika Instituut

Mari-Liis Viet

Scratch-i õpetamisest keskkoolis

Bakalaureusetöö

Autor: Mari-Liis Viet
Juhendaja: Jaagup Kippar

Autor:.....,,.....,,2011
Juhendaja:.....,,.....,,2011
Instituudi direktor:.....,,.....,,2011

Tallinn 2011

Autorideklaratsioon

Deklareerin, et käesolev bakalaureusetöö on minu töö tulemus ja seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

....

(kuupäev) (autor)

Sisukord

Sissejuhatus

Käesoleva bakalaureusetööga teen ma ülevaate *Scratch*-ist, programmeerimise- ning disaini õpetamise meetoditest, eelmainitud programmeerimiskeele õpetamisest nii kesk- kui ka ülikoolis ning koostan *Scratch*-i õppematerjali. Oma töös toetun ma kahele küsitlusele, mille viisin läbi Rapla Vesrioosi arvutiõpetaja ning Tallinna Tehnika Ülikooli dotsendi vastustele tuginedes ning tagasisideküsimustikul, millele vastasid Pelgulinna Gümnaasiumi 10. ning 11. klassi õpilased.

Scratch on visuaalne programmeerimiskeel, mis on mõeldud eeskätt 6-16-aastastele noortele, kuigi tegelikkuses on *Scratch*-i kasutanud igast vanusegrupist inimesed.
Mind ennast huvitab *Scratch* eelkõige õpetaja vaatenurgast. Tegu on väga hea õppevahendiga programmeerimise aluste õpetamiseks ning temast on vägagi kerge aru saada. Samuti näitab *Scratch,* et õppimine ei pea olema igav - sellega saab erinevaid ja huvitavaid asju teha.

Põhjus, miks valisin *Scratch*-i enda bakalaureusetöö teemaks, on järgmine: *Scratch*-ist ei ole seni väga räägitud. Paljud minu ülikooli- ja erialakaaslased teevad teadmatusest suuri silmi ning küsivad, millega on tegu. Tundsin, et *Scratch* on hea teema, millest kirjutada, just nimelt huvitavuse ja kasulikkuse pärast. *Scratch* on materjal, mida kasutades saab selgeks programmeerimise nn. alustalad. Kõik eelnev ongi põhjuseks, miks valisin enda lõputöö teemaks *Scratch*-i.

Minu töö esimene eesmärk oli koostada ülevaade sellest, kuidas kasutavad *Scratch*-i kui õppematerjali praegu Vesiroosi Gümnaasium ning Tallinna Tehnikaülikool. Samuti võrdlen meetodeid, mida kasutatakse programmeerimise ja tarkvara disainimise ning veebirakenduste loomise õpetamiseks, leidmaks, mis on nende kahe õpetamisel ühist, mis erinevat. Põhjus, miks ma seda teen, peitub *Scratch*-i kui programmeerimiskeele visuaalsuses. Kokku on pandud visuaalne ja tekstiline programmeerimine. Eesmärgiks oleks leida kahe erineva teema õpetamise meetotitest nn. kuldne kesktee. Lisaks meetotite uurimisele on minu bakalaureusetöö eesmärgiks välja selgitada, kuidas suhtuvad õpilased *Scratch*-i.

Kuidas ma kõik need eesmärgid saavutasin? Mul avanes võimalus anda 6x90-minutiline tund Pelgulinna Gümnaasiumus: 3x90-minutit 10. klassile ja 3x90-minutit 11. klassile. Koostasin igaks tunniks mingid ülesanded või näidised, mille abil tutvustasin *Scratch*-i erinevaid võimalusi. Samuti andsin ma õpilastele isesesivaid töid. Viimasel tunnil esitasin ma nendel tagasisideküsitluse, mille põhjal sain teha järeldusi sellest, mida nad kõnealusest programmeerimiskeelest arvasid.

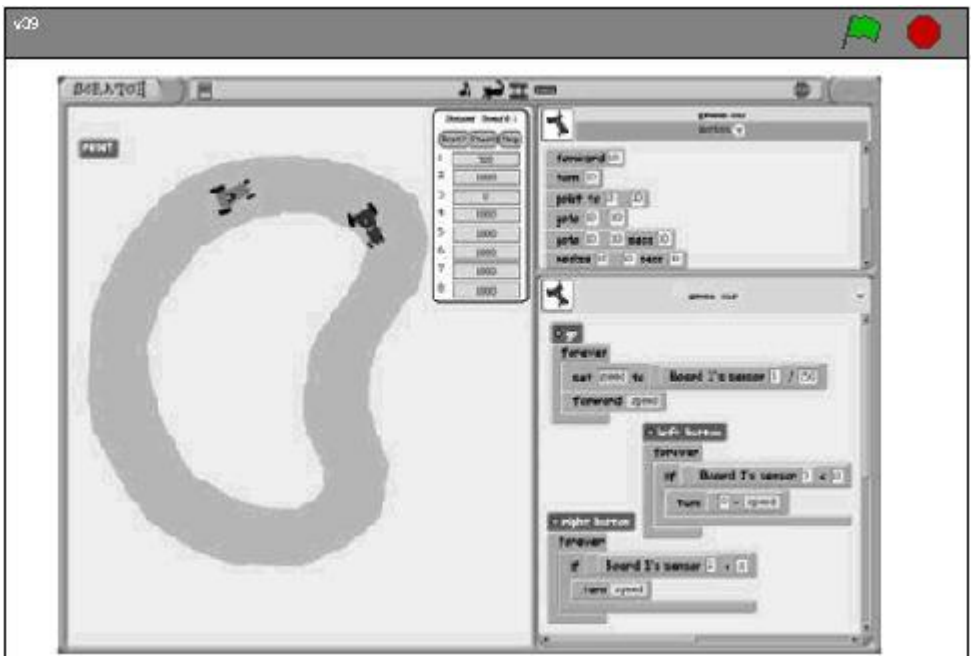
Oma eesmärkide saavutamiseks:

- Tutvun *Scratch*i õppematerjalidega.
- Tutvun programmeerimise ja veebidisaini õpetamise meetotitega.
- Koostan näidisülesandeid
- Annan tunde Pelgulinna Gümnaasiumis.
- Küsitlen TTÜ õppejõudu Jüri Vilipõldu
- Küsitlen Vesiroosi Gümnaasiumi õpetajat Edmund Lavassoni.

1. *Scratch*-ist

Scratch on õppeesmärgiline programmeerimiskeel, mis annab kõigist vanuse- ja kogemusgruppidest inimestele võimaluse katsetada täispaindliku arvutiprogrammeerimise põhiideesid, *vedades* visuaalselt koodijuppe kokku, et kontrollida pilte, muusikat ning heli. (M. Resnick, J. Maloney, A. Monroy-Hernandez, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silverman, Y. Kafai, 2009)

Programmeerimiskeele loojaks on Lifelong Kindergarten Group, mis baseerub Massachusetts'i Tehnoloogia Instituudis (Massachusetts Institute of Technology), tiimi juhiks on Mitchel Resnick, kes avaledas esimese väljalaske 2007. aasta suvel.(vaata pilt 1) (MIT Media Lab, 2007), (BBC, 2007)



Pilt 1. *Scratch*-i esimene versioon

Scratch-i saab paigaldada ning tohib vabalt levitada kõikidel Windowsit, Mac OS X-i või Linuxit kasutatavatel arvutitel. Lähtekood on avalikustatud lepingu all, mis võimaldab kasutajatel seda muuta vaid mittekommertslikel eesmärkidel. (Lifelong Kindergarten group of the MIT Media Lab)

Nimetus „*Scratch*“ tuleneb inglisekeelsest tegusõnast „*to Scratch“*, mille all peetakse silmas plaadi kriipimist. Nime andmisel peeti silmas asjaolu, et plaadi *Scratch*-imine on muusikatöötluse lihtsaim osa – samuti oli võimalik importida ühte koodijuppi koos kõigi sinnakuuluvate muutujatega teise koodi, lubades algajatel saavutada kiiremini paremaid tulemusi ning motiveerides neid edasi katsetama.

Scratch-i kasutatakse ülemaailmselt erinevates keskkondades: koolides, muuseumites ("*Scratch* Day" Minnesota muuseumis, 2011), kogukonnakeskustes ning kodudes. Eeskätt on *Scratch* mõeldud 6-16 aastastele, kuid seda on kasutanud inimesed igast vanusegrupist. Näiteks saavad nooremad lapsed kasutada *Scratch*-i vanemate juhendamisel, kuid kolledžiõpilased saavad seda kasutada esmastel programmeerimiskursustel. (D. J. Malan, H. H. Leitner, 2007)

Loojate põhiprioriteet keele disainimesel oli muuta keel ning selle arenduskeskkond võimalikult intuitiivseks ning lihtsalt õpitavaks laste ning muude väheste programmeerimiskogemustega inimeste jaoks. Keeles on kasutatavad mitmed multimeediafunktsioonid ning programmeerida saab „mitmekihiliselt“, kuid võimaluste arv on teiste keeltega võrreldes siiski üsna piiratud. Kasutajaliides *Scratch*-is jaotab arenduskeskkonna mitmeks osaks: vasakul on koodijupid (e. funktsioonid), keskel käesoleva spraidi kohta info ja skriptiala, ning paremal on etappide- ning spraidinimekiri. Koodijupiala (koodijuppe nimetatakse antud programmerimiskeeles plokkideks e. *blocks*) koosnebki koodijuppidest, mida saab skriptialasse lohistada, et luua programme. Et palett liiga pikaks ei veniks, on plokid jaotatud kaheksasse gruppi: liikumine, välimus, heli, pliiats, juhtimine, andurid, tehted, muutujad.

Korraldati samuti kogemustepõhiseid uuringuid erinevate võimaluste kohta - need, mis takistasid nn. intuitiivset õppimist, eemaldati. Samal ajal need võimalused, mis säilitasid programmeerimiskeele lihtsuse ning atraktiivsuse, jäeti alles. Sellise arendusmeetodi tulemused olid osati üllatuslikud, muutes *Scratch*-i üsnagi erinevaks teistest õppimiskeeltest (nt. BASIC, Logo või Alice). Näiteks mitmekihiline programmerimine (funktsioon funktsiooni sees) on *Scratch*-is fundamentaalse tähtsusega, ent samas ei leidu *Scratch*-is protseduure nagu faili sisend/väljund (File I/O). Lisaks sellele toetab *Scratch* vaid ühedimensioonilisi massiive (listid).

Ujukomaarvud, skalaarid ning stringid on samuti toetatud alates versioonist 1.4, kuid üsna piiratud manipuleerimisvõimalustega.

Scratch-i võrgukommuuni loosunglause „Kujutle, programmeeri, jaga“ paneb rõhu jagamisele ning loovuse sotsiaalaspektidele, mis mõlemad mängivad olulist rolli *Scratch*-i taga olevas filosoofias. (A. Monroy-Hernandez, M. Resnick, 2008)*Scratch*-i põhiseid projekte ei vaadata kui kinniseid kaste vaid kui koode, mida on võimalik ümber töötada, et luua uusi programme. Projekte saab üles laadida otse arenduskeskkonnast *Scratch*-i veebilehele ning iga kommuuni liige saab laadida alla täieliku lähtekoodi, et seda uurida või edasi arendada. (A, Monroy-Hernandez, M. Resnick, 2008) Liikmed saavad samuti kommenteerida, lisada märksõnu, koostada nimekirju lemmikutest. Kõik projektid laetakse üles Creative Commonsi (San Franciscos baseeruv MTÜ, mis tegutseb eesmärgiga suurendada tasuta jagatavate ja kasutatavate loometeoste hulka) arvele ning neid saab kuvada kõigis veebibrauserites (*Java applet*-i või *Flash Player*-i abil). Veebilehte kuvatakse ühes kuus keskmiselt 10 miljonit korda ning 2010. aasta seisuga oli kommuunil 600 001 registreeritud kasutajat ning ligikaudu 1 500 000 projekti (keskmise kiirusega laetakse üles üks projekt minutis). (MIT Media Lab, 2007) Tihti korraldatakse võistlusi, mille eesmärgiks on innustada programmiloomet ja –disaini, andes võistlejatele ette põhilise disainikontsepti. Lisaks globaalkommuunile omavad eraldi kommuune veel Mehhiko, Iisrael, Portugal ning Araabia Ühendemiraadid. Võrgukommuun on loodud ka haritlastele – *ScratchEd*. (Karen Brennan of Lifelong Kindergarten, OHO Interactive, 2009) Kasutajad saavad teisi kasutajaid oma sõbraks lisatada.

Scratch-ist on loodud mitmeid derivaate, mida nimetatakse *Scratch*-i modifikatsioonideks. Need on loodud, kasutades versioon 1.4 lähtekoodi. Need programmid on põhiliselt sama kontseptiga, mis *Scratch*, kuid sisaldavad ekstraplokke, mida originaalses *Scratch*-is pole, samuti on muudetud kasutajaliidest. Üks sellistest modifikatsioonidest, BYOB (*Build Your Own Blocks*, autoriks Jens Mönig), on kasutuses Berkeley Ülikoolis arvutiteaduste õpetamisel. *Scratch*-i lähtekood on Squeakipõhine, mis omakorda baseerub Smalltalk-80-l. *Scratch* on vaba lähtekoodiga ning seda saab alla laadida lehelt http://info.*Scratch*.mit.edu/Source_Code .

2. Programmeerimise õpetamisest.

Programmeerimise 3 põhilist etappi on disain, arendus ja testimine. Seevastu programmeerimise õpetamise 3 põhilist sammu on keel, testimine ja disain. Nagu näha on õpetamise sammud reaalsusest natuke erinevad. Miks? Põhjuseks võiks olla, et kui koolitundides ei antaks mingit alust keele tundmisele ning kohe minnakse arendamise ja disaini peale, tekiks õpilastel segadus. Puuduks alus, mille peale ehitada kõik järgnev. Ka *Scratch*-i õpetamise puhul sai kasutatud neid kolme etappi: Esmalt sai näidatud, mida teeb üldse *Scratch* – sai kokku pandud programm. Edasi liiguti selleni, et vaadata, kas üldse hakkab midagi toimuma. Ja kui toimub, kas toimub õigesti? Kui kõik oli korras, liiguti edasi ja vaadati üle disain: Kas kõik on nii nagu peab?

Programmeerimise õpetamisel kasutatakse erinevate tasemete puhul erinevaid meetoteid. *Scratch*i puhul sai kõikide meetotitega mingil määral kokku puututud.
7-8. klassi, mida võiks pidada nn algtasemeks, kasutatakse pusle- ja nn käega järje ajamise meetoteid. Puslemeetod koosneb järgnevatest sammudest (Dron, 2003):

- Kirjutada programm, mis koosneb konstruktsioonidest millest õpilased teevad eksimusi.
- Jaotada programm juppideks, mis järgivad järgnevaid reegleid:
    - Ühte sõna ei saa poolitada
    - Jupib tuleb kombineerida unikaalselt
- Aja jupid omavahel sassi.

Puslemeetotit sai mingil määral kasutatud nii, et *Scratch*-i tunnis tegi õpetaja meelega vea sisse juppide kokku kombineerimisel ning õpilaste ülesanne oli leida üles viga. Kui tüüpilisi vigu kõik koo läbi arutada ning näidata, mis ta teeb siis, kui viga esineb, ja mida siis, kui viga on lahendatud, tekib võrdlusmoment ning õpilastel on lihtsam aru saada. Seda enam, kui tuleb ise üles leida viga ning välja pakkuda selle lahenduskäik. See meetod ei vasta küll kõikidele pusle meetoti alampunktidele, kuid mõte on peaaegu sama.

9-11. klassi, kes peaksid kuuluma nn kesktaseme alla, õpetamiseks kasutatavad meetodid oleksid tiimi- ja paaristöö meetod. Tiimitöömeetoti kasutamise reeglid on järgmised (Dron, 2003):

- Õpetaja mängib kliendi ja planeerija osa. Tema roll on kirjutada üles ülesanne ja kirjeldada programmide vahelist sõltuvust.
- Õpilased on jaotatud tiimi. Teiste tiimide tegemistega ollakse vaid kursis tehtud töö põhjal.
- Üks õpilane on valitud tiimiliidriks. Tema ülesanne on tiimi arengu organiseerimine.

Põhineb RUP (*Rational Unified Process*) meetodil (Dron, 2003).

- RUP põhimõtted:
    - Analüüsimine
    - Võimalike riskidepiiritlemine
    - Planeerimine
    - Areng

Tiimitöö meetod on vast kõige levinum meetod üldse õpetamise juures: Klass jagatakse rühmadeks ja igale rühmale antakse õpetaja poolt mingi ülesanne. Tunni lõpus teeb iga rühm ettekande ning näitavad, kui kaugele oma tööga jõuti. Seda meetoti sai proovitud ka *Scratch*i peal ja peaks mainima et see väga ei toiminud. Kõik töötasid ühe arvuti taga ning kippus siiski nii minema, et üks tegi ja teised mulisesid kõrval. Kuid see meetod on kindlasti hea õpetamisemeetod tarkvara arenduse vms põhjal. Samuti ka juhul, kui rühmal on kasutada mitu arvutit ning iga rühmaliige saab reaalselt täita ning lahendada oma ülesandeid. Kuid miks see meetod kasulik võiks olla? Õpilasele on ta kindlasti kasulik sellesmõttes, et probleemi lahendus nign lahenduse käik tuleb ise välja mõelda. Tiimiliider peab arvestama kaasõpilastega oma rühmas, kuid samas jagama kõigile võrdselt mingeid ülesandeid. Lisaks annab see hea koostöökogemuse.

Lisaks tiimitöö meetodile kasutatakse ka paaristöö meetodit, mille kasutamise reeglid on järgmised (Dron, 2003):

- Õpetaja mängib kliendi osa. Tema roll on kirjutada ülesandeid ja varustab õpilaste nn kliendilugudega iga iteratsiooni järel
- Õpilased on jaotatud paarideks. Igal paaril on oma ülesanne.
- Iga iteratsiooni lõpus pärast testimist, paar esitleb oma programmi.

Paaristöö meetod põhineb XP (*eXtreme Programming*) ja AM (*Agile Method*)põhjal.
XP kasutamise reeglid:

- Planeerimine:
- Kasutajalood on üles kirjutatud
- Tehakse tihti väiksemahulisi väljalaskeid
- Projekt on jaotatud iteratsioonideks

Disainimine.

- Lihtsus
- Süsteemile on valitut nn metafoor
- Esialgu ei ole vajadust funktsionaalsust arendada
- Programmile lisatakse tegureid millaliganes ja kusiganes

Koodikirjutamine

- Klient on alati saadaval
- Kood kirjutatakse üksikosade testimiseks
- Kogu tootekood on paarisprogrammeeritud
- Üks paar saab koodi üheaegselt ühendada.

AM'i kasutamise põhimõtted:

- Püsi minimalistlik
- Aktsepteeri muutusi
- Arvesta muutuste kuhjumisega
- Modelleeri eesmärgiga
- Koosta arvukaid mudeleid
- Anna kiiresti tagasisidet

Paaristöö meetod on vast teine üks levinumaid õpetamise meetoteid arvuti – ja programmeerimise õpetamise juures. Scartchi peal sai kasutatud paaritöö meetotit, kuid sellele oli külge poogitud nn ping-pong programmeerimine. Õpilased ei suhtunud sellesse väga positiivselt, kuna nendele tundus raske, et iga 5 minuti tagant saabub 5 minutline tööaeg, kus tuleb programmi täita. Kaasõpilane mõtles küll kaasa, kuid oma mõtteid ta jagada ei tohtinud vaid sai neid realiseerida alles siis, kui tema 5 minutline tööaeg oli käes. Miks on see kasulik? Kasulik on ta sellepärast, et vältida seda, et üks rühmaliige niisama pealt vaataks. Mõlemal paarilisel on korraga aega 5 minutit, et programmi täiustada. Kui paari peaksid juhtuma õpilased, kellest üks saab hästi aru, kuid teine mitte nii hästi, on see kasulik, sest õpilane, kes ei tunne asja nii hästi, saab samuti oma panuse anda.

Paaristöö – ja tiimitöö meetod on kaks põhilisemalt programmeerimise meetotit, millega on töö autor kokku puutunud nii õpilase kui ka õpetajana. Õpetaja seisukohalt on see hea võimalus panna õpilased iseseisvalt tööle ning õppima. Samas õpilase seisukohalt annab see hea kogemuse meeskonnatöö kogemusele.

3. Veebidisaini õpetamisest “Veebiarenduse ja -disaini” kursuse näitel.

Õppekava (Microsofti, BCS Koolitus, 2010) tutvustab üldharidus- ja kutsekoolide õpilastele veebitehnoloogiaid ja –disaini. Erilist rõhuasetust pannakse just nimelt disainile. Tihtipeale keskendutakse selle valdkonna koolitustel just koodi kirjutamisele, kuid antud juhul keskenduti ka küsimustele, miks ja kuidas veebirakendusi disainida, kuidas arvestada tulevase kasutaja e. kliendiga.

Oluline märksõna antud kursusel on kasutajakogemus ehk UX – *User eXperience*. See viitab veebirakenduste loomisele, arvestades kasutajate soove, kogemusi ja harjumusi. Selline õpetamisviis muudab kursuse sobilikuks mitte vaid IT huviga õpilastele vaid ka üldiselt disainihuvilistele.

Kursuse põhieesmärk on anda osalejatele teadmisi ja oskusi interaktiivsete veebilehtede ja –rakenduste kujundamiseks. Samuti tutvustatakse osalejatele uusimat tehnoloogiat interaktiivsete veebirakenduste loomiseks – *Silverlighti*.

IT-valdkonnas teenitakse sageli leiba omal käel omandatud teadmistega, mille abil luuakse erinevaid rakendusi. Need rakendused on tihtipeale aga halvasti kujundatud ning raskesti kasutatavad. Iseseisvalt saab tihti omandada tehnoloogia, kuid mitte disainipõhimõtted. Seega antakse kursusel ülevaade veebi arengust üleüldiselt, selgitatakse kasutajakogemuse olulisust, kirjeldatakse erinevaid rolle veebiarenduses ning veebi loomise tsüklit.

Disain ei saa olla aga võimalik ilma töövahenditeta, seega tutvustatakse inimestele ka neid. Oskuseid omandatakse just nimelt *Microsoft Expression Studio 3* disainitööriistade komplekti kuuluva programmi *Expression Design 3* kasutamiseks. Nimetataud programm on mõeldud just nimelt graafiliseks kujundamiseks ning pakub kõiki põhivahendeid veebirakenduste kasutajaliidese loomiseks. Lisaks sellele on üheks koolituse eesmärgiks tutvustada *Silverlight 3* tehnoloogiat ja anda oskusi selle kasutamiseks. Veebirakenduse kujundamise järel tutvustatakse *Silverlight* rakenduste

arendamiseks mõeldud programmi *Expression Blend 3*, mille abil luuakse interaktiivseid rakendusi ning seotakse need loodud kujundusega. Seega järgib koolitus ka kahe üksteisest eraldataud töövoo põhimõtet – disain ja arendus.

Koolitus on üles ehitatud reaalelulist olukorda imiteerides: on olemas klient ehk tellija, kes soovib kindla funktsionaalsusega veebirakendust, analüüsitakse vajadusi ning võimalusi, koostatakse loodava rakenduse spetsifikatsioon, luuakse kujundus ning seejärel ka töötav rakendus, mis seejärel publitseeritakse. See annab osalejatele reaalse ülevaate veebirakenduse loomise tsüklist ja laseb järele proovida erinevaid rolle veebiarenduses.

Kuidas seda seostada *Scratch*-iga? Põhiliselt saab õppust võtta koolituse ülesehitusest – *Scratch* võib olla osa mingi rakenduse loomise tsüklist. Näiteks: õpetaja, kui klient soovib mingit rakendust - olgu selleks mäng, animatsioon vms. Õpilase ülesanne on analüüsida vajadusi ja võimalusi rakenduse loomiseks, koostata selleks spetsifikatsioon ning kujundus. Kui kõik see on valmis, luuakse mäng või animatsioon ning publitseeritakse see nt ametlikule *Scratch*-i kodulehele.

Antud veebidisainikursust pole otseselt võimalik samastada *Scratchiga* tööriistasid ja –keskkondasid võrreldes. Küll aga saab paralleele tõmmata kogu kursuste metoodikate vahel. Nimelt tuleb nii veebirakenduse kui ka iga muu rakenduse loomiseks läbida samad etapid, mis Veebistuudiumi koolituses – lähtuda kasutaja soovidest, tutvuda kasutajakogemusega läbi aegade, panna rõhku disainile jm.

4. Keskkool vs Ülikool.

Et saada võrdlus *Scratch*-i kasutamisest kesk- ja ülikoolis, sai esitatud küsimused Vesiroosi Gümnaasiumi arvutiõpetajale Edmund Laugassonile ning Tallina Tehnikaülikooli dotsendile Jüri Vilipõllule. Uuritavad said valitud selle järgi, et Vesiroosi Gümnaasium on üks nendest koolidest, kes kasutab *Scratch*-i arvutiõpetuse tunnis, samuti soovitas mulle Tallinna Pelgulinna Gümnaasiumi arvutiõpetaja Birgy Lorenz. Tallinna Tehnikaülikooli dotsent Jüri Vilipõld sai valitud aga selle järgi, et tema poolt on loodud päris mitmed õppematerjalid *Scratch*-i kohta, samuti kasutab ta *Scratch*-i õppetöös. Nimelt kasutab ta *Scratch*-i programmeerimise aluste õpetamisel nii informaatikute kui ka mitteinformaatikute hulgas.

Vesiroosi Gümnaasiumi õpetajale ning TTÜ õppejõule sai esitatud järgmised küsimused:

1. Kui kaua olete oma koolis *Scratch*-i õpetanud?
2. Millisele vanusele olete programmeerimist õpetanud? (5-6-7-9; gümnaasium; ülikooli algajad; ülikooli spetsialistid)
3. Millisele vanusele olete programmeerimisel kasutanud *Scratch* abi? -6;7-9;gümnaasium; ülikool algajad; ülikool spetsialistid (it) ja miks?
4. Mitu tundi olete keskmiselt *Scratch*-i koolitundides käsitlenud?
5. Kuidas hindate *Scratch*-i sobivust gümnaasiumile/ülikoolile?
6. Mitmetunnine pakett sobib Sratchi õpetamiseks? 6/12/24/35, midagi muud), palun (põhjendus)?
7. Milliseid programmeerimisalaseid teadmisi ja oskuseid saab õpetada teie meelest Scrach abil paremini kui teiste prgrammidega?
8. Milliste omaduste arendamiseks sobib teie arust *Scratch*?
9. Kuidas suhtute väitesse, et Scrach on mõeldud 5.-6. klassi õpilastele programeerimise õpetamiseks ning vanemate astmete õpilaste/üliõpilaste puhul tehakse lihtsalt "nalja"?

Edmund Lavassoni arvamus *Scratch*-ist:

Oma koolis on ta õpetanud *Scratch*-i umbes aasta jagu ning seda klassides 5-9, lisaks gümnaasiumis, käsitlenud on ta antud teemat umbes 20 tundi. Õpetaja arvas, et *Scratch* sobib väga hästi nii gümnaasiumis kui ka ülikoolis õpetamiseks ja seda just programmeerimise õpetamise alustamiseks.

Kuna *Scratch*-i on plaanitud hakkama õpetama 35-tunnise kursusena, küsisin ka mõlema õppejõu käest, kas nende arust oleks mõistlik see kursus läbi viia. Keskkooliõpetaja arvas, et see plaan on täiesti teostatav. Põhjendas ta seda järgnevalt: sageli tahetakse kohe kiirelt imeasju teha kuid ühe kursuse raames võib edukalt ka algtõdesid treenida – hiljem on keerulisemad asjad siis lihtsamad. Tema arusts sobib *Scratch* ka tsüklite, paralleelprotisesside, muutujate ja massiivide õpetamiseks, kuna neid on hästi visuaalselt näha. *Scratch*-i skript sarnaneb mõneti plokkskeemile ning see aitab mõista programmi ülesehitust ja selle loogikat.

Vesiroosi Gümnaasiumi õpetaja arvates ei ole oluline kui vanad on need õpilased, kellele tundi antakse. Tema arvates on vanematel õpilastel kergem raskematest süsteemidest aru saada, kui omandatakse teadmised mõne kergema süsteemi (antud juhul *Scratch*) kohta.

Jüri Vilipõllu arvamus *Scratch*-ist:

Scratch-i on ta oma koolis õpetanud algõppefaasil kolm aastat. *Scratch*-i ennast kasutab ta programmeerimise algkursuseks (6-10-tundi) ning seda kõikide erialade tundidel. (TTÜ) Lisaks temale on kasutanud TTÜ õppejõud *Scratch*-i ka kahes gümnaasiumus õpetamiseks. On korraldatud *Scratch*-i töötubasid, mis on kestnud 3-4 tundi ning need on suunatud 8-10. klassi õpilastele. *Scratch*-i õpetamiseks on kasutatud u 6-10 tundi, sõltuvalt erialast. Seda siis just mitteinformaatikutle progarmmeerimise õpetamisel.

Vilipõllu arvates sobib *Scratch* Ülikooli sissejuhatavaks osaks programmeerimise õpetamisel. Näiteks maailma kõrgeima retinguga ülikoolis Harvardis, alustatakse arvutiõpetuse sussejuhatavat kurstust just *Scratch*-iga, millest edasi minnakse siis nt

C, Javascripti jms peale. Lisaks Harvardile kasutab *Scratch*-i veel sissejuhataval kursusel ka teine globaalsel tasemel tippülikool - Berkley. Oma kogemuse põhjal julgeb Vilipõld väita, et need kolm aastat, mil ta TTÜs on *Scratch*-i õpetanud, on olnud väga suureks abiks programmeerimise algõpetuses ja seda eriti inimeste jaoks, kes ei õpi informaatika erialal. Tema arvates peaks koolides integreerima programmeerimise algõpetust, kus kasutatakse just nimelt *Scratch*-i abi.

Küsimusele, kas *Scratch* sobiks koolis 35 tunniseks kursuseks õpetamiseks, vastas Vilipõld, et *Scratch* ei ole mõeldud õpetamiseks vaid teda saab kasutada algoritmimise, programmeerimise, modelleerimise ja disaini õpetamiseks. Siinkohal toob ta näite uues õppekavas toodud aine "Rakenduste loomise ja programmeerimise alused" , mis koosneb kahest moodulist: põhimoodul(20-35 tundi) ja lisamoodul(15-20 tundi). Põhimoodulis kasutatakse programmeerimiseks *Scratch*-i ning lisamoodulis mõnda tekstipõhist programeerimisekeelt. Põhimooduli maht ja *Scratch*-i kasutamine sõltub õpetamise eesmärgidest, kooli ja ainevaldkonna iseloomust jm. Valiku peaks saama teha kool ja/või õpetaja. Näiteks IT-õppesuuna jaoks võiks olla *Scratch*-il põhinev sissejuhatus 6-10 tundi programmeerimise põhikontseptsioonide ja meetodite kiireks omandamiseks. Kursuses võib lisamoodul ka puududa ja piirdutakse *Scratch*-iga. Siin võib eristada kahte põhivarianti. *Scratch*-i abil saab teha selgeks kõik algoritmimise ja programmeerimise põhkontseptsioonid ja –oskused/-meetodid, eriti arvestades uues versioonis ettenähtud täiendusi: parameetritega protseduurid ja funktsioonid, rekursioon, massiivide massiivid jm. Teine lähenemisviis võiks sobida humanitaarkallakuga koolidele ja õppesuundadele. Ei üritatagi minna väga sügavale programmeerimisega, vaid keskendutakse multimeedia vahendite kasutamisele (sh ka näiteks arvutimuusika ja –kunsti algetele), disainile, esitluste ja veebirakenduste loomisele jmt.

Scratch ise sobiks näiteks multimeediavahenditie kasutamise õpetamiseks rakendustes, objektide ja muutujate olemuse ja kasutamise põhimõtete õpetamiseks, protsesside juhtimiseks programmides (nt hargnevad ja tsüklilised protsessid) ning paralleelsete protsesside käsitlemiseks. Lisaks sobib *Scratch* veel ka loogilise ja algoritmilise mõtlemise ning süsteemse lähenemisviisi "leiutamise" arendamiseks.

Ka Jüri Vilipõld leiab, et ei ole oluline vanusegrupp, millele *Scratch*-i õpetatakse. Tema arvates seisneb *Scratch*-i fenomenaalsus selles, et *Scratch*-i kasutamise õpetamine ei sõltu eriti õppija vanusest. Kasutatavad vahendid, käsitletavad teemad ja lahendatavad ülesanded saab valida, arvestades õppijate vanust, taset, õpetamise eesmärke jm.

5. Ülesanded

Selle bakalaureusetöö üheks eesmärgiks oli luua *Scratch*-i õppematerjal keskkoolile. Selle loomiseks oli vaja nö katsejäneseid. Õnneks avanes võimalus katsetata ülesandeid ja materjale Pelgulinna Gümnaasiumu õpilaste peal. Uurimisalusteks sai võetud 10. ja 11. klassi õpilased. Tundide arvuks oli kuus üheksakümneminutilist tundi. Allpool on kirjeldatud kõik tunnid eraldi ning välja toodud materjalid ning ülesanded. Kuigi on mainitud, et antud tunde oli 6, on allpool on kirjeldatud 5. Seda järgmisel põhjusel: nii 10. kui ka 11. klassi esimene tund oli sissejuhatav ning seal antud ülesanded olid identsed. Seda selleks, et mõlemad saaksid nn ühesugused baasteadmised *Scratch*-ist.

Ülesannete koostamisel lähtusin eelkõige sellest, et õpilastest pole keegi eelnevalt ei programmeerimise ega ka *Scratch*iga kokku puutunud. eelkõige panustasin ma just animatsioonide loomisele, paralleel protsesside tegemisele, tsükli põhimõtte näitamisele ning üldiselt programmi üleshitusele. Algselt oli plaan ka luua õpilastega nn arvamismäng, kus tegelane küsib tehte ja siis kasutaja vastab, ärats mida antakse teada, kas vastus on õige või mitte. Kuid leidsin, et selle selgitamine ja koostamine oleks õpilastele liiga keerulisena tundunud. Miks? *Scratch*il puudub sisendi andmise võimalus. Võimalus oleks olnud luua ekraanil klaviatuuri moodi tegelane, millele hiirega peale klõpsides oleks saanud vastuse trükkida. Kogu selle lahenduskäigu selgitamine ja kujutamine oleks võtnud palju aega.

Mõlema klassiga sai kohtutud 3 korda ning iga tunni teema oli erinev. Esimene tund töötati koos õpetajaga ning enamus tunniteemasid arutleti ja analüüsiti koos. Teine tund töötati rühmadena: 10. klassis jagati klass rühmadesse(igas rühmas 2-3 inimest) ning töö käis tüüpilisel tiimitöö meetodil. Õpetaja jagas igale rühmale oma ülesande ja alles tunni lõpus nägi iga rühm, mis teised tühmaid teinud olid. 11. klassis kasutati nn ping-pong programmeerimise meetodit. See tähendab, et klass jagati paarideks ja igale paarile anti ülesanne. Üks paariline sai korraga programmi täiendada 5 minutit. Kolmanda tunni eesmärk oli mängu koostamine. Pool tunnist töötas õpetaja koos õpilastega ning teise poole tunnist pidi iga õpilane oma mängu iseseisvalt täiendama. Kuigi tunniteemad kohati kattusid, oli eesmärk siiski teha mõlemas klassis erinevaid

asju. Et läbi katsetada võimalikult palju erinevaid teemasid ja et samuti saada võimalikult palju erinevat tagasidet.

Esimene tund.

1. Tutvu esimese näitega. Loo ise lava, kuhu paigutad tausta ja tegelase. Tegelane vali olemasolevates (Vajutada kassi peale, kes paikneb lava all. Keskmisest tulbast valida Kostüümid. Vajutada import->People. Valida meelepärane tegelane. Taust samamoodi, kuid vajutada kassi asemel tausta(eeldatavasti valge kast kassi kõrval) peale.)
2. Kui tegelane ja taust on saadud, pane tegelane liikuma: Näiteks 10 sammu paremale
(Vasakult tulbast Juhtimise ja Liikumise nime alt otsida vajalik.)
3. Kui tegelase liikumine on saadud nii kaugele, et ta liigub 10 sammu, proovi ta liikuma panna pidevalt. Ning kui tegelane põrkub seinaga, muutub tegelase suund.
4. Kui tegelane pidevalt liigub, proovi ta kohale tekitada jutumull, kus ilmub tervitus
5. Kui üks tegelane on olemas, proovi tuua sisse ka teine tegelane ning talle panna samamoodi liikumine juurde. Kuid nüüd selle asemel, et kasutada olemasolevat tegelast, joonista ise uus tegelane (lava all on kiri: Uus sprait. Sealt 3-st nupust valida see, mille kohale liikudes tekib kiri: Joonista uus sprait).
6. Pane joonistatud tegelane samamoodi liikuma kui eelmine.
7. Proovi kahe tegelase liikumise suund panna risti. St et esimene tegelane liigub üles-alla ja teine paremale-vasakule (siin aitab kaasa, kui vajutada lava all tegelase peale. Keskele tekib teda kirjeldav tabel ning üleval on kujutis temast, kus peal on üks sinine joon. Seda sinist joont liigutades saab määrata tegelase suuna).
8. Nüüd proovi teha nii, et kui kaks tegelast peaksid kokku juhtuma, siis nad põrkuvad ning suund muutub. Siin aitab kaasa „kui“ , kuhu sisse saab panna tingimuse „puudutab“, mille leiad Andurite alt.
9. Kui nüüd on mõlemad tegelased saadud liikuma, proovi panna taustaks heli. Igale elemendile on võimalik panna oma heli. Antud juhul on meil siis 2 spraiti ja 1 taust. Paneme taustale heli(vajuta tausta peale ning keskmisest tulbast vajuta helid peale)

Kolm esimest ülesannet sai tehtud õpilastega koos pärast sissejuhatust *Scratch*-i ning programmide tutvustamist näidete abil. Materjaliks sai võetud järgmine veebiaadress: http://elrond.tud.ttu.ee/~vilip/*Scratch*/Juhend/Scr_juhend.html. Kui kolm esimest ülesannet oli lahendatud, said õpilased korralduse järgnevad ülesanded ise lahendada. Probleem tekkis 8. ülesande juures, kus mitmel õpilasel esines probleeme nende põrkuma panemise teemal. Põhjuseks võiks siinkohal kindlasti tuua, et õpilastel puudus eelnevalt kokkupuude programmiga ning samuti ka sissejuhatamisel ei räägitud sellisest olukorrast ning selle lahendamisest. Kuigi 8. ülesandega tekkis probleeme, said enamus õpilased heli lisamisega iseseisvalt edukalt hakkama.

Programmi tutvustamisel sai räägitud sellest, kuida teha uus projekt, kuidas lisada tegelasi ning muuta nende välimust. Lisaks tegelasele, sai räägitud ka taustast ning selle muutmisest.

Ajakava:
25 minutit - sissejuhatus ja *Scratch*-i materjalide ülevaatamine
30 minutit- 3-esimese ülesande tegemine ja ülevaatamine. Järgmiste ülesannete iseseisev tegemine
10 minutit - tehtud ülesannete ülevaatamine. Arutelu, mis läks hästi, mis mitte.
10 minutit - Järgmise tunni ülesannete arutelu.
10 minutit - *Youtube*-st tutorial ning õppematerjali ülevaatamine.

Teine tund

1. Tegelased: Ema, punamütsike, korv
Tegevus: Ema räägib munamütsikesele, et vanaema on haige. Punamütsike võtab korvi ja hakkab vanaema juurde minema. Ema hoiatab metsas olevate ohtude eest.
2. Tegelased: Punamütsike, hunt ja lilled
Tegevus: Räägivad omavahel ning leppivad kokku teekonna vanaema juurde.
3. Tegelased: Vanaema ja hunt. Taustaks vanaema kodu
Hundi ja vanaema vaheline suhtlus.
4. Tegelased: Hunt, punamütsike, jahimees ja vanaema.

Hunt mängib vanaema ning punamütsike esitab talle küsimusi. Lõpeb sellega et vanaema päästetakse ja kõik on õnnelikud.

Ülesanded said arutatud 11. klassiga esimesel tunnil ning läbi viidud 10. klassis. Tunni alguses korrati eelmisel tunnel tehtut ja vaadati üle tunniülesanded. Esimese ülesande lahendamisel räägiti ära uus teema. Esimene ülesanne lahendati poole peale ning siis jagati klass rühmadeks ning jaotati ülesanded ära. Iga grupp sai mingi jupi Punamütsikese muinasjuttust, millest tuli teha nn animatsioon. Kuna tunni eesmärk oli õpilastele seletada ja näidata, kuidas panna teine jupp tööle siis, kui esimene lõpetas, ilmnes järgmine problem iseseisvalt ülesannete lahendamisel(vaata pilt 2).

Pilt 2. "Peata skript" vale asukoht.

Probleem seisneb selles, et jupp "peata skript" peaks asuma kui tsükli sees, mitte väljas. Selle probleemi esinemisel sai õpilastele selgitatud tsüklite põhimõttest, ehk siis põhjus, miks antud juhul element liigub 5 sammu ja peatub.

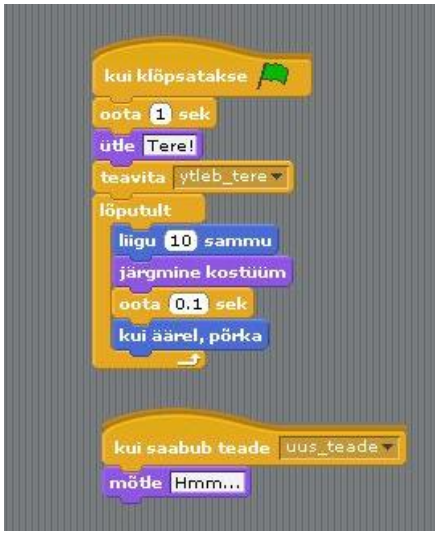
Ajakava:
45 minutit – Eelmise tunni teema kordamine. Uue teema ülevaatamine näidisülesande näitel.
35 minutit – Õpilased tegelevad rühmatööga.
10 minitit – Iga rühm kannab oma töö ette.

Kolmas tund

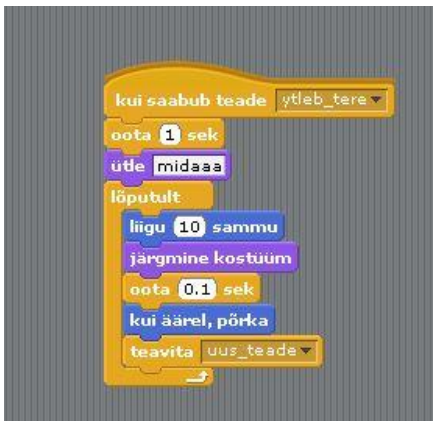
Tunni alguses algusest 15 minutit tutvustati *Scratch*-is dialoogide koostamist ja tausta vahetamist mingi aja tagant. Iseseisvaks tööks oli nn ping-pong programmeerimine. Klass jagati paarideks. Iga paar pidi kujutama oma kodutee, kusjuures igal paaril oli individuaalselt oma ülesanne, mis pidi nende animatsioonis esinema. Ülesanded olid järgemised:

1. ühele paarilisele helistab ema ja palub poodi minna
2. üks paariline unustab vihiku kooli ja peab kooli tagasi minema
3. ühele paarile tuleb vastu hulkuv koer
4. üks paar läheb kinno
5. üks paar on pealtnägijaks autoõnnetusele
6. üks paar läks korvpalli mängima
7. üks paar läheb poodi

Aega oli ülesannete lahendamiseks 35 minutit. Ping-pongimine nägi välja nii, et mõlemal paarilisel oli aega korraga 5 minutit programmide täiendamiseks. Esimesed 5 minutit aega anti selleks, et läbi rääkida, mida ja kuidas teha. Järgmised 5 minutit tegelaste tegemiseks, järgmised tausta valimiseks ja liikuma panemiseks. Edasi liiguti individuaalsete ülesannete juurde. Kuna aega oli vähe, ei jõudnud kõik paarid oma ülesannetega lõpuni, vaid poole peale. Kuna tunni teemaks oli *Scratch*-is dialoogide kooostamine ja tausta vahetamine, oli ülesande täitmise nõudeks see, et animatsioonis esineb vähemalt 15 dialooglauset ja taust vahetub iga 5-sekundi järel. Tunni lõpust kulus 15 minutit animatsioonide esitamisele ning oma töö tutvustamisele. Kuna aega oli vähe ja ülesanne ise liiga mahukas, ei jõudnud ükski paar täielikult valmis. Tausta vahetamise ja dialoogidega saadi edukalt hakkama. Selle tunni probleemiks oli taaskord kui tsükli ülesehitus. Kuna õpilastel puudub igasugune eelnev kogemus programmeerimise osas, ei olda päris kindlad *kui* ja *lõputult* tsükli ülesehitusest. Nimelt kui jätta uue teate saatmine lõputult tsükli sisse, toimub see pidevalt ning teine element saab selle teate koguaeg ning teeb seda tegevus lõputult, mida pannakse ta tegema pärast teate saabumist. (Vaata pilt 4 ja pilt 5)



Pilt 4. Teate saatmine ja uue teate vastuvõtmine.



Pilt 5. Teate vastuvõtmine ja uue saatmine.

Ajakava:

15 minutit – Sissejuhatus tundi. Eelmise tunni teema kordamine

6 x 5 minutit – Ping-pong programmeerimine

20 minutit – Iga paar kannab oma töö ette

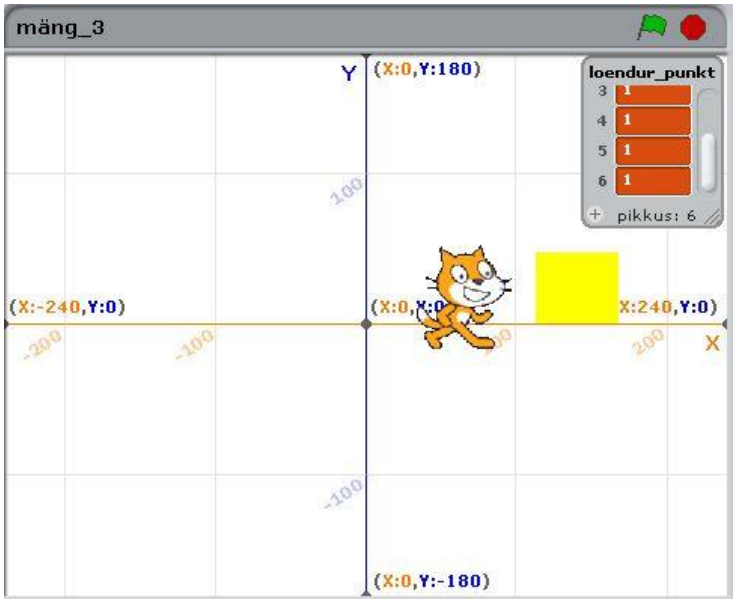
10 minutit – Arutelu, mis läks hästi, mis läks halvasti

10 minutit – Järgmise tunni ülesannete läbi arutlemine

Neljas tund.

Neljas 90-minutline tund sai pühendatud sellele, et õpilastele sai veelkord ülekorratud tsüklid. Selletunniseks ülesandeks oli panna elukas ekraani peal liikuma nii, et ta

oleks nooltega juhitav. Tunni alguses tegi õpetaja koos õpilastega u 20 minutit koostööd ning näitas ära kuidas on võimalik panna tegelasele juurde funktsioonid, muutmaks tegelase nooltega juhitavaks. Koos räägiti ja töötati läbi ühe nupu näide ning siis oli õpilastel aega 15 minutit, et tegelasele külge pookida ülejäänud 3 nuppu. Kui tegelane sai liikuma pandud, sai õpilastele tutvustada loendureid ja muutujaid. Igal õpilasel paluti lugeda leheküljel http://elrond.tud.ttu.ee/~vilip/*Scratch*/Juhend/Scr_juhend.html olevas materjalis loendurite ja muutujate osa. Seejärel pandi tegelasele juurde loendur, mis luges üles korrad, kui ta mingi elemendiga kokku põrkas. (vaata pilt 6)



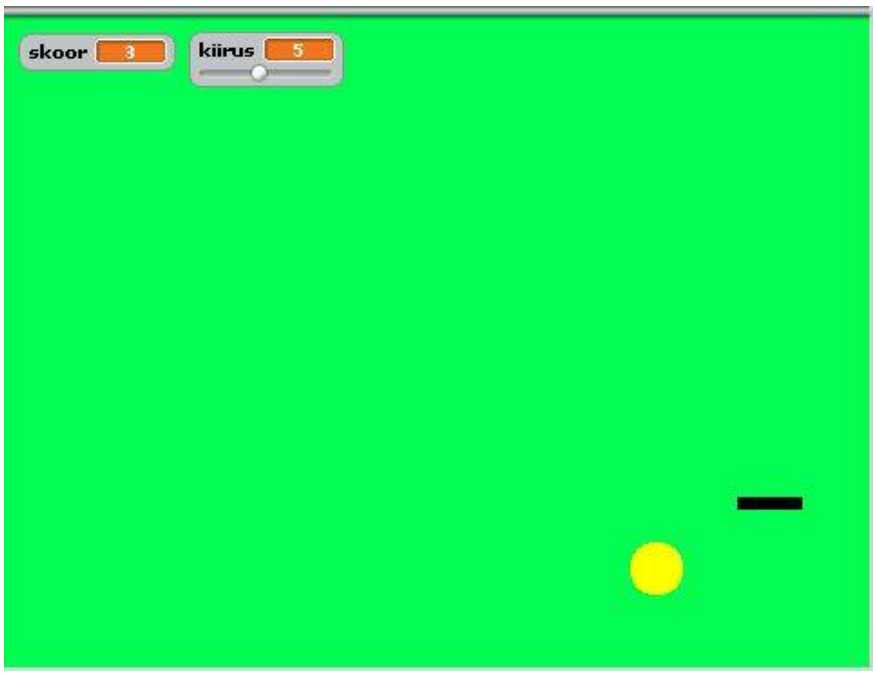
Pilt 6. Loendur

Võrreldes eelmiste tundidega, ei tekkinud siin tunnis enam *kui*-tsükli ülesehitamisega probleem. Enamus õpilasi said kõik ilusti hakkama ning ei esinenud probleemkohti.
Ajakava:
20 minutit – Uue osa tutvustamine(Esimese juhtnuppu tööle panemine. Selgitame kuidas ja miks)
15 minutit – Iseseisev töö(3 ülejäänud juhtnuppu tööle panemine.)
10 minutit – Arutelu ja analüüs.
20 minutit – loendurite ja muutujate tutvustamine

Siin tunnis täitis klass ka tagasiside küsitlus, mistõttu ei tule ajakavas kokku 45 minutit.

Viies tund

Et jätkata eelmises tunis klahvikombinatsioonidega mängimist, oli selle tunni lõpp-produkt ussimäng. Lisaks klahvikombinatsioonide teema jätkamisele, jätkati ka muutuja teemaga. Õpilastega tehti koos algselt nn ühemängija mäng, kus uss roomab mööda lava ringi ning ajab taga toitu. Ekraani nurgas oli 2 muutujat, millest üks oli skoor ja teine kiirus. Kiiruse muutuja oli liuguri kujul ning selle mõtte oli anda mängule nn erinev raskustase ussi liikumiskiiruse näol. Skoori muutuja mõte oli see, et iga kord kui uss sai kätte toidu, muutus skoor suuremaks. Iga kättesaadud toit andis ühe punkti. Et ussimängu mängimist raskemaks teha, sai tehtud koos selline lisa, et kui uss põrkas vastu seina, nulliti skoor ning uss paigutati tagasi oma algpunkti. (vaata pilt 7)



Pilt 7. Muutujad

Siin tunnis esines mitmel õpilasel probleem oskuses loendur ekraanilt ära kaotada. Selle põhjuseks oli see, et loendurikastike tõsteti ekraanilt kogemata ära. Kasti tagasisaamiseks oli vaja teha loenduri ees linnuke, mis toob ta tagasi ekraanile. (Vaata pilt 8)

Pilt 8. Loenduri kuvamine ja mittekuvamine.

55 minutit – Õpilastega koos ussimängu tegemine.
30 minutit – Iga õpilane individuaalselt täiendab mängu.

Ka siin tunnis täitis klass ka tagasiside küsitlus, mistõttu ei tule ajakavas kokku 45 minutit.

6. Õpilaste hinnang

Pelgulinna Gümnaasiumi õpilased olid antud õppematerjali ja ülesannete koostamisel nn katsejänesed. 10 . ja 11. klassile avanes võimalus anda kolm 90 minutilist tundi. Igal tunnil oli oma teema. Esimese tunni teema oli sissejuhatav: kirjeldati programmi ennast ning tema võimalusi. Tunni eesmärgiks oli arendada arutlemis- ning analüüsivõimet. Teise tunni teemaks oli rühmatöö ja animatsiooni koostamine. 10. klassi rühmatöö oli üsna standartne - pandi pead kokku ja arutati üheselt läbi. 11. klassi peal sai proovitud nn ping-pong programmeerimist. Klass jagati paarideks ning mõlemal paarilisel oli korraga aega 5 minutit et animatsiooni edasi arendada. Tunni eesmärgiks oli arendada nende koostöövõimet. Kolmanda tunni teemaks oli mängu loomine- antud tunni eesmärgiks võiks pidada kujutlusvõime ning loovuse arendamist.

Kuid kuidas õpilased ise neid tunde hindasid?

10. klass:

Esimene tund valmistas neile raskusi, kuna teema oli uus ja programm võõras. Minu kui õpetaja hinnangul said õpilased väga edukalt ülesannetega hakkama ning võtsid *Scratch*-i vägagi kiirelt omaks. Teist tundi hinnati küllalti huvitavaks, kuna said ise luua animatsiooni ning teguteda rühmades. 10. klassi õpilastele meeldis kõige rohkem viimane tund, kus oli teemaks mängu tegemine. See tundus nendele kõige arusaadavam ning lihtsam. Paljud kirjutasid, et mängu loomine näitas kui palju erinevaid võimalusi *Scratch*-is on ning et tekkis huvi ka kodus kasutada seda edaspidi.

11. klass:

Tulemused olid üldjoontes samad, mis 10. klassil. Kõige lihtsamaks ja arusaadavamaks hinnati kolmandat tundi, keskmine hinnang anti teisele tunnile ning kõige raskemaks peeti esimest tundi.

Scratch-i ennast hindasid õpilased järgmiselt:

7 õpilast - enam-vähem arusaadav

11 õpilast - kerge

2 õpilast - väga kerge

Scartchi puhul meeldis õpilaste see, et keskkond oli eesti keeles ning et ta oli väga loogiline ja kergesti mõistetav. Lisaks meeldis õpilastele see, et kõik blokid olid arusaadavate nimedega. Kahjuks oli neid õpilasi, kes puudusid mõnest tunnist ning olid järgmistes tundides raskustes, kuna polnud alustalasid all.

Kõige rohkem valmistas raskusi nn tsükliline pool - kuidas anda ühele elemendile teada, et teine element on oma tegevuse lõpetanud ning ootab käsklust edais tegutsemiseks. Samuti tundus nendele alguses raske juppide kokkupanemine – mis süsteemi järgi funktsiooni kokku panna. Lisaks nendele tundus õpilastele raskem ka loendurite ja muutujate teema, mida tutvustati mängu loomisel.

Kokkuvõttes tundus *Scratch* õpilastele väga meeltmööda ning tervelt 18 õpilast leidis, et *Scratch* sobib koolis õpetamiseks. 15 õpilast leidsid et *Scratch* ei sobi 35-tunniseks aineks. Ülejäänud 5 oli vastupidisel arvamusel, leides, et sobib küll.

Kuna *Scratch* on mõeldud lisaks programmeerimise algõpetuse andmiseks ka õpilaste erinevate omaduste arendamiseks, oli küsitluses järgmine: Milliseid oskuseid *Scratch*-i puhl omandasite? Tulemus oli järgmine:

Koostöö	7 õpilast
Info-otsing	2 õpilast
Loomingulisus	15 õpilast
Planeerimisoskus	4 õpilast
Iseseisvate otsuste langetamine	7 õpilast
Esitlusoskus	5 õpilast
Loogika	14 õpilast
Probleemi püstitamine	4 õpilast
Probleemi käsitlemine	7 õpilast
Analüütiline mõtlemine	5 õpilast

Kokkuvõteks võiks öelda, et õpilased võtsid *Scratch*-i vägagi ilusti omaks ning tundub et nendele meeldiks, kui seda veel edasigi õpetatakse. Programm tundus loogiline ning arusaadav ning nii töö autori kui ka õpetajana võin julgelt väita, et õpilased said *Scratch*-iga vägagi edukalt hakkama. Praeguste teadmistega teeksid nad kindlasti valmis erinevaid animatsioone ning kindlasti lihtsama mängu. Programmeerimise aluste suhtes, omavad nad teadmisi üldiselt programmi ülesehituse kohta ning kindlasti ka tsükli töö kohta.

7. Autori hinnang *Scratch*-ile.

Scratch on hea õppematerjal ning sobib kasutamiseks nii kesk- kui ka ülikoolile. Tahaks korrata Jüri Vilipõllu sõnu: Scratchi pole vaja õpetada. Seda saab kasutada algoritmimise, programmeerimise, modelleerimise ja disaini õpetamiseks. Tõepoolest – seda keelt pole tarvis õpetada, temaga saab õpetada. Kuid kas temaga saaks läbi viia 35-tunnist kursust? Pigem on tegu pseudoolukorraga, kuna oleks raske tegeleda nii pikka aega järjest vaid kõnealuse programmeerimiskeelega. Pigem on tegu siiski programmiga, mille abil saab õppida midagi muud. Näiteks siis algoritmimist, programmeerimist, modelleerimist või isegi disaini. Lisaks informaatika ainetele, on *Scratch*-iga võimalik teha ka erinevaid matemaatika või siis võõrkeele teemalisi ülesandeid.

Mida *Scratch*-i puhul silmas pidada? Ei saa eeldada, et õpilased saavad kohe aru, mida *Scratch* teeb ning mida temaga õppida saab. Tihti läheb tarvis mitut näidisprogrammi, et õpilased saaksid aru, mis on antud teema põhieesmärk või mis teadmisi neile üritatakse edasi anda.

Oma õpetamise jooksul, oli näha, et kõige rohkem pakkus õpilastele huvi just mängu tegemine, samuti jäid selle tööülesande käigus omandatud teadmised neile kõige paremini meelde. Seega, kasutades *Scratch*-i, tuleks teda kasutada võimalikult palju mängulisust ja huvitavust, et õpilased paremini aru saaksid. Kindlasti tuleks anda õpilastele ka võimalus võimalikult palju ise mõelda ning improviseerida, lasts neil rakendada maksimaalselt oma fantaasiat. Muidugi ei tähenda see seda, et õpilasele tuleks anda täielikult vabad käed Scratchi kasutamiseks. Pigem seda, et anda tuleks võimalus analüüsida probleeme ning leida nendele ise lahendus.

Lõpetamiseks võib öelda, et S*cratch* on hea abivahend õpetamiseks - olgu õpetatavaks aineks programmeerimine, matemaatika või hoopis mõni võõrkeel.

.

Kokkuvõte

Töö eesmärgiks oli teha ülevaade *Scratch*-ist ja selle õpetamisest keskkoolis. Eesmärgi saavutamiseks sai käidud Pelgulinna Gümnaasiumis, et sealsetele õpilastele tutvustada *Scratch*-i. Lõpuks sai esitatud neile tagasisideküsitlus, mille tulemustest sai aimu sihtrühma suhtumisest kõnealusesse programmeerimiskeelde. Lisaks õpilastele sai küsimusi esitatud ka keskkooliõpetajale ja ka ülikooli õppejõule, et uurida, kuidas nemad *Scratch*-i suhtuvad.

Kuna ajahulk oli suhteliselt piiratud ning töö autoril puudusid igasugused eelnevad kogemused õpetamiseks, ei valminud *Scratch*-i õppematerjal nii suures mahus nagu esialgu planeeritud. Küll sai aga õpilaste peal proovida teatud ülesandeid, mida võiks kindlasti ka tulevikus läbi viia, astudes klassi ette. Ülesannete valimisel sai lähtutud erinevatest programmeerimise õpetamise meetotitest ning uuritud Veebistuudiumi kursust veebirakenduste loomisest.

Scratch on väga hea õppematerjal. Lähemale tuleks seda keelt tuua kindlasti nii ülikoolidele kui ka keskkoolidele. Ta annab vägagi hea aluspõhja programmeerimisele, tutvustades õppuritele enamusi baasfunktsioone ja muutujaid programmeerimises. Samuti on ta väga huvitav ja mitmekülgne ning tundus, et õpilastele üsna lihtsalt omaks võetav. Teda oli huvitav tutvustada ning temaga oli väga huvitav töötada.

Oma tulevas elus, kui töö autor peaks klassi ette astuma, kasutab ta kindlasti *Scratch*-i edasi. Tegu on õppevahendiga, mille abil õpetada programmeerimist igast vanuse- ja oskusgrupist inimesele.

Kasutatud kirjandus

1. A, Monroy-Hernandez, M. Resnick. (2008). Computers can't give credit: How automatic attribution falls short in an online remixing community. Allikas: http://info.Scratch.mit.edu/sites/infoScratch.media.mit.edu/files/file/monroy-hernandez_et_al_chi2011.pdf (Viimati loetud: 25. aprill 2011)
2. "Scratch Day" Minnesota muuseumis. (21. May 2011. a.). Allikas: http://www.smm.org/ltc/Scratchday (Viimati loetud: 25. aprill 2011)
3. A. Monroy-Hernandez, M. Resnick. (2008). Empowering Kids to Create and Share Programmable Media. Allikas: http://info.Scratch.mit.edu/sites/infoScratch.media.mit.edu/files/file/interactions_acm_2007_monroy-hernandez_resnick.pdf (Viimati loetud: 25. aprill 2011)
4. BBC. (2007). Free tool offers "easy" coding. Allikas: http://news.bbc.co.uk/2/hi/technology/6647011.stm (Viimati loetud: 25. aprill 2011)
5. D. J. Malan, H. H. Leitner. (2007). Scratch for Budding Computer Scientists. Allikas: http://portal.acm.org/citation.cfm?doid=1227310.1227388 (Viimati loetud: 25. aprill 2011)
6. Dron, V. (2003). Allikas: http://www14.in.tum.de/konferenzen/Jass04/courses/3/dron.ppt (Viimati loetud: 30. aprill 2011)
7. Karen Brennan of Lifelong Kindergarten, OHO Interactive. (2009). ScratchEd. Allikas: http://Scratched.media.mit.edu/ (Viimati loetud: 25. aprill 2011)
8. Lifelong Kindergarten group of the MIT Media Lab. (kuupäev puudub). Scratchi lähtekood. Allikas: http://info.Scratch.mit.edu/Source_Code (Viimati loetud: 23. aprill 2011)
9. M. Resnick, J. Maloney, A. Monroy-Hernandez, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silverman, Y. Kafai. (2009). Scratch: Programming for All. Allikas: http://cacm.acm.org/magazines/2009/11/48421-Scratch-programming-for-all/fulltext (Viimati loetud: 25. aprill 2011)

10. Microsofti, BCS Koolitus. (2010). Veebirakenduste disain. Allikas: http://www.eneta.ee/SiteCollectionDocuments/vs/Veebistuudium_disain.docx (Viimati loetud: 30. aprill 2011)
11. MIT Media Lab. (2007). Explore up-to-date statistics about the Scratch Online Community. Allikas: http://stats.Scratch.mit.edu/community/ (Viimati loetud: 25. aprill 2011)
12. MIT Media Lab. (2007). *Scratch*: imagine, program, share. Allikas: http://info.*Scratch*.mit.edu/*Scratch*_Credits(Viimati loetud: 25. aprill 2011)

Summary: Teaching Scratch in Secondary School

The purpose of this thesis was to make an overview of Scratch and its teaching methods in High School. In order to achieve my goals, I gave several lectures at Pelgulinna Gymnasium in order to introduce Scratch to the students there. Before the end of the lectures I asked the students to fill out surveys I created for them, so I could see what they thought of Scratch in general. In addition to the students, I also asked for the opinion of a high school teacher and an university lecturer.

Scratch is an educational programming language, suitable to most groups of skill and age (although the target audience ranges from ages 6 to 16). It is used in many universities, colleges and high-schools around the world. It was developed in 2007 by a MIT-based developer team, known as the Lifelong Kindergarten Group, led by Mitchel Resnick. Available for Windows, Mac OS X or Linux based platforms, the source code of Scratch is fully downloadable and editable (for non-commercial purposes). The language is mostly meant to introduce beginners to all the basic functions and variables that they can come across while using more complicated languages.

Before starting giving classes at Pelgulinna High School, I studied different methods of teaching programming. Although the three basic stages in developing are the same on each occasion, the methods varie depending on the target-audience. Some examples include the puzzle-method, the rational unified process method etc.

One cannot be a good teacher just by knowing his or her own art, however. In compliance to that, I felt the need to also study methods of teaching other computer-related sciences. Therefor I compared the teaching methods of programming to the methods of teaching the construction and development of web-based applications. Although the basic outlining of the methods was almost identical, there were some minor differences.

After questioning both a high school teacher and an university lecturer about their viewpoints on Scratch, I also found out that Scratch is a favoured teaching-tool for the both questioned. The minor negativities Scratch may have do not weight up to the possibilities it offers.

The classes I gave went well in the long run. Due to me having no previous experience as a teacher, there arose some problems – sometimes the students could not understand the tasks given to them or could simply not finish them on time. Nevertheless, all the students seemed to like the language and developed an interest in the field of programming. Before the end of the final class, the students were given spreadsheets asking about their viewpoints and opinions on Scratch. Most of the students found it to be rather easy to use and interesting.

Having worked with Scratch for over an year now, I have to agree with the students. Scratch offers many fun possibilities to introduce pupils to the world of programming. All the basic functions and variables you can find in other languages, are also included in Scratch.

All in all, I can definitely say that should I ever stand in front of a programming class again, Scratch would definitely be the language I would use to teach my students.

Lisad

Lisa 1. *Scratch*-i installimisjuhend

Scratch on vabavaraline programm ning tema installfaili saab kätte siit. *Scratch*-il on olemas ka eestikeelne kasutajaliides, mille installimist ma järgnevalt kirjeldan. Minu kasutatavaks operatsiooni süsteemiks oli *Windows 7 Proffesional*.

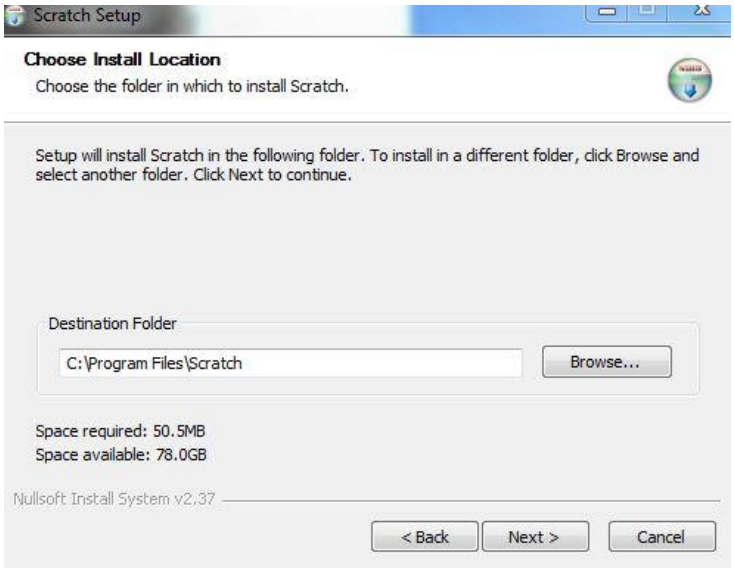
Kõige esimesena tuleks avada fail nimega *ScratchInstaller1.4.exe*, mille leiab kaustast, kuhu ta salvestati. (Vaata pilt 9)

Pilt 9. ***ScratchInstaller1.4.exe***

Vajutades failile peale, küsib ta luba installeerimiseks ja kindlasti tuleks vajutada „*Yes*“. (Vaata pilt 10)

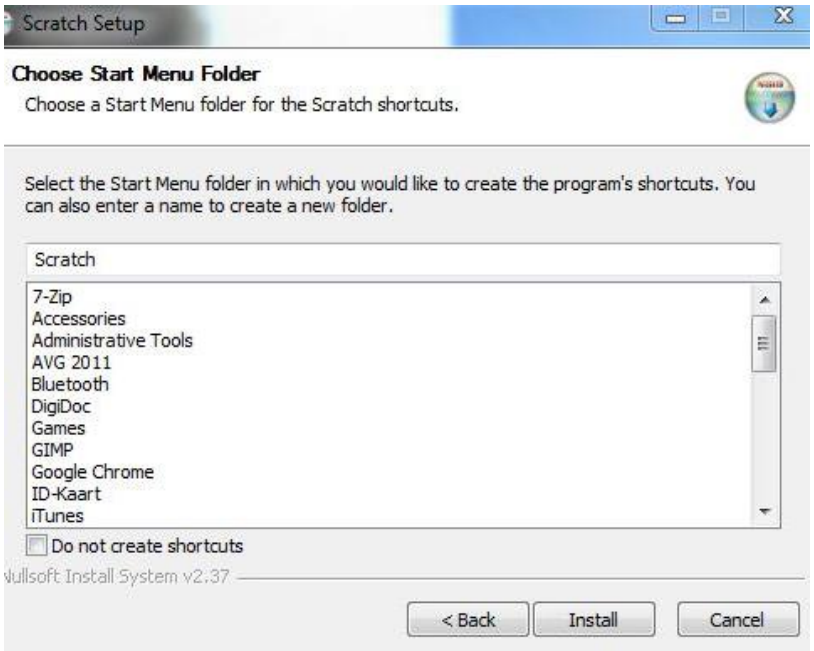
Pilt 10. ***Welcome to Scratch 1.4***

Vajutama peaks siinkohal kindlasti *Next*, et alustada installeerimist. Järgmisena küsib ta asukohta, kuhu soovita ta installida. Ehk siis kuhu läheb programmi alamkaust jne. (Vaata pilt 11)



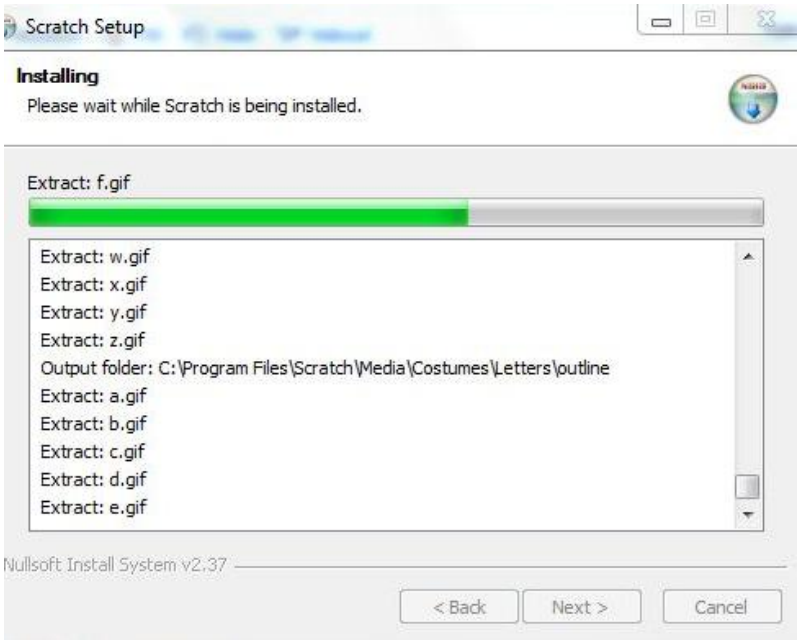
Pilt 11. Asukoha küsimine

Kui asukoht valitud, tuleks vajutada „*Next*“. Edasi küsib ta Start menüü kohta. *Windows*-i masinas on *Start* menüü see koht, kus on programmide ja kataloogide loetelu. Antud juhul küsitakse meie käest, kuhu alla *Scratch* nn lühitee paigutatakse.(Vaata pilt 12)



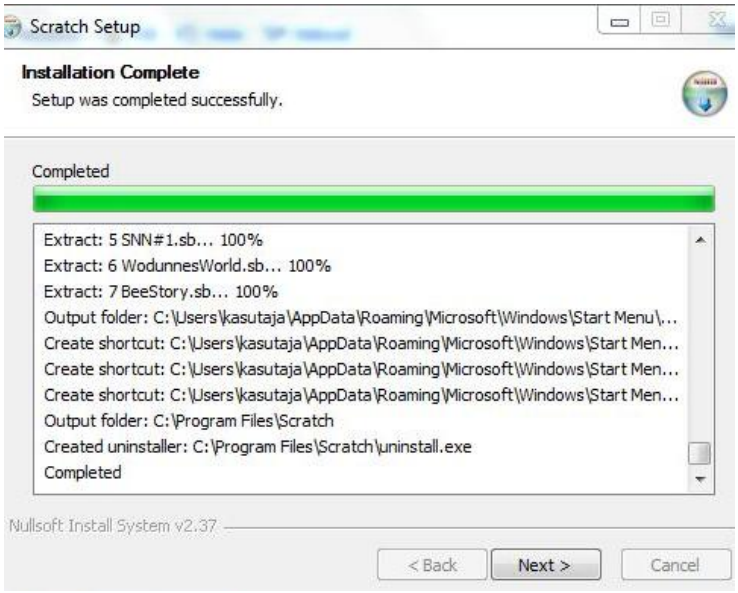
Pilt 12. Lühitee küsimine

Muidugi on ka võimalus, et lühiteed ei looda. Selleks oleks vaja panna linnuke teksti „*Do not create shortcuts"* ette. Pildl on näha, et lahtris ilusteb sõna *Scratch*. See tähendab et Start menüüse luuaksegi lühitee nimega „*Scratch*" ning teda ei paigutada kuskile alla. Kui see on tehtud, asutakse *Scartchi* installima. (Vaata pilt 13)



Pilt 13. Installimise protsess.

Siin on näha, et installimine on töös. Installimise lõpetamisel on ainuke võimalus vajutada nuppu *"Next"*.(Vaata pilt 14)

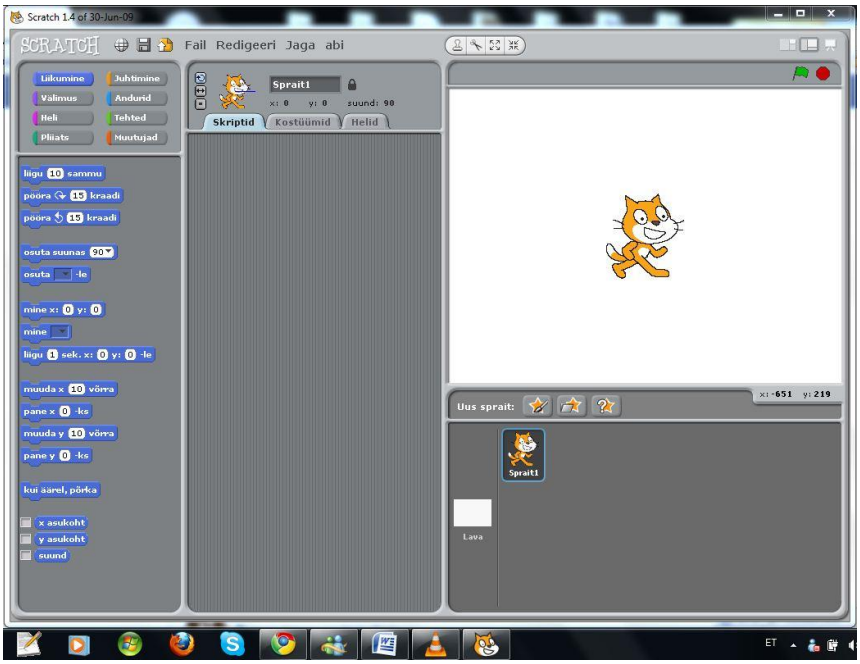


Pilt 14. Installimise lõpetamine.

Nuppu *"Next"* vajutades küsitakse, kas luua töölauale lühitee ja kas avada programm. (Vaata pilt 15)

Pilt 15. Completing the *Scratch* Setup.

Jättes linnukesed ette ja vajutades „*Finish"*, avatakse *Scratch*-i programm. (Vaata pilt 16)



Pilt 16. *Scratch*i programmi põhiaken.

Nagu näha koosneb programm kolmest tulbast: vasakul tulbas on toodud erinevate värvide ja teema all skriptid, mis näevad välja kui pusle jupid. Keskmises tulbas on see koht, kus tuleks kõik pusle jupid kokku lohistada – st et pannakse kokku programm. Paremas tulbas asub lava, kus kuvatakse mida teeb antud programm, mis on skriptidest kokku lohistatud. Siin kohal pisike näide nn kokku lohistatud pusle juppidest(Vaata pilt 17).

Pilt 17. *Scratch*-i programmi skripti jupp.

Lisa 2. Tunni ülesanded ja materjalid.

Ülesanded

1. Tutvu esimese näitega. Loo ise lava, kuhu paigutad tausta ja tegelase. Tegelane vali olemasolevates (Vajutada kassi peale, kes paikneb lava all. Keskmisest tulbast valida Kostüümid. Vajutada import->People. Valida meelepärane tegelane. Taust samamoodi, kuid vajutada kassi asemel tausta(eeldatavasti valge kast kassi kõrval) peale.)
2. Kui tegelane ja taust on saadud, pane tegelane liikuma: Näiteks 10 sammu paremale

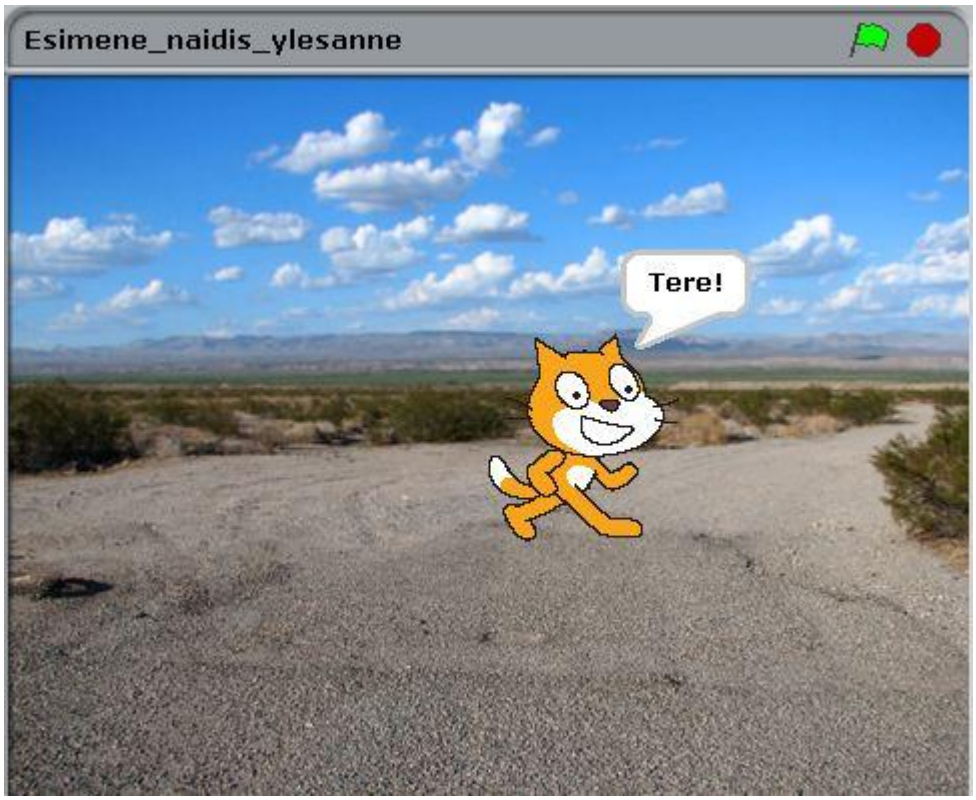
(Vasakult tulbast Juhtimise ja Liikumise nime alt otsida vajalik.)

3. Kui tegelase liikumine on saadud nii kaugele, et ta liigub 10 sammu, proovi ta liikuma panna pidevalt. Ning kui tegelane põrkub seinaga, muutub tegelase suund.
4. Kui tegelane pidevalt liigub, proovi ta kohale tekitada jutumull, kus ilmub tervitus
5. Kui üks tegelane on olemas, proovi tuua sisse ka teine tegelane ning talle panna samamoodi liikumine juurde. Kuid nüüd selle asemel, et kasutada olemasolevat tegelast, joonista ise uus tegelane (lava all on kiri: Uus sprait. Sealt 3-st nupust valida see, mille kohale liikudes tekib kiri: Joonista uus sprait).
6. Pane joonistatud tegelane samamoodi liikuma kui eelmine.
7. Proovi kahe tegelase liikumise suund panna risti. St et esimene tegelane liigub üles-alla ja teine paremale-vasakule (siin aitab kaasa, kui vajutada lava all tegelase peale. Keskele tekib teda kirjeldav tabel ning üleval on kujutis temast, kus peal on üks sinine joon. Seda sinist joont liigutades saab määrata tegelase suuna).
8. Nüüd proovi teha nii, et kui kaks tegelast peaksid kokku juhtuma, siis nad põrkuvad ning suund muutub. Siin aitab kaasa „kui“ , kuhu sisse saab panna tingimuse „puudutab“, mille leiad Andurite alt.
9. Kui nüüd on mõlemad tegelased saadud liikuma, proovi panna taustaks heli. Igale elemendile on võimalik panna oma heli. Antud juhul on meil siis 2 spraiti ja 1 taust. Paneme taustale heli(vajuta tausta peale ning keskmisest tulbast vajuta helid peale)

Näidisülesanne:

http://*Scratch*.mit.edu/projects/mannu3/1747956

Sprite1 skript.



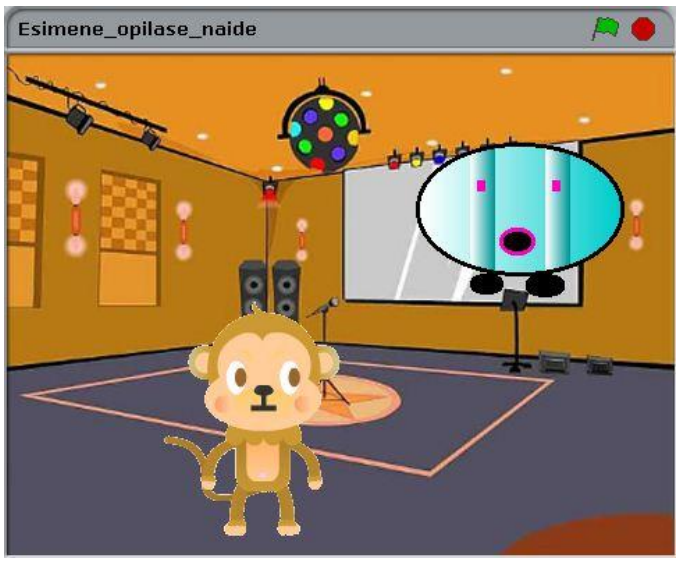
Laval olev pilt.

Õpilaste näited:

http://*Scratch*.mit.edu/projects/mannu3/1747962

Sprite1 skript

Sprite2 skript



Laval olev pilt.

http://*Scratch*.mit.edu/projects/mannu3/1747971

Sprite1 skript.

Sprite2 skript

Laval olev pilt.

8. Teine tund(x90)

Ülesanne:

1. Tegelased: Ema, punamütsike, korv

Tegevus: Ema räägib munamütsikesele, et vanaema on haige. Punamütsike võtab korvi ja hakkab vanaema juurde minema. Ema hoiatab metsas olevate ohtude eest.

2. Tegelased: Punamütsike, hunt ja lilled

Tegevus: Räägivad omavahel ning leppivad kokku teekonna vanaema juurde.

3. Tegelased: Vanaema ja hunt. Taustaks vanaema kodu

Tegevus: Hundi ja vanaema vaheline suhtlus.

4. Tegelased: Hunt, punamütsike, jahimees ja vanaema.

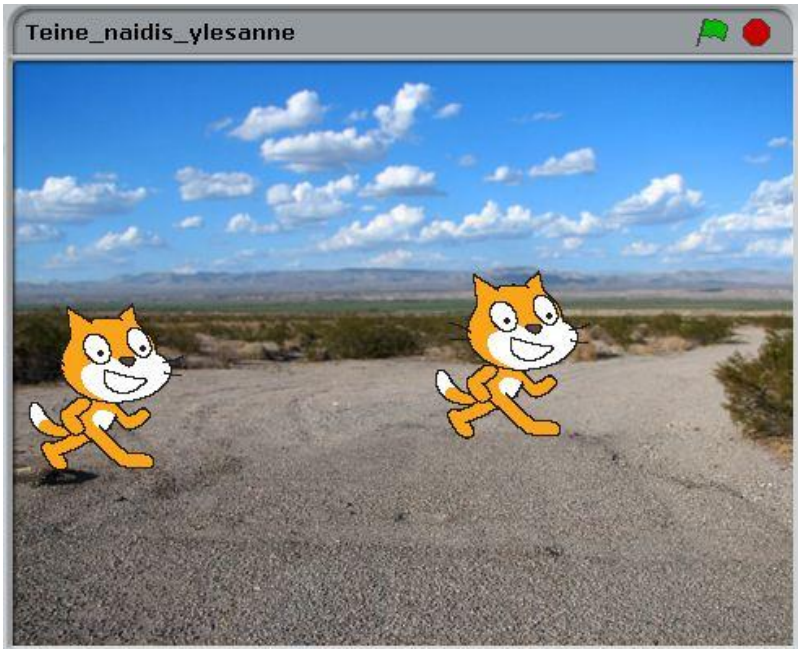
Tegevus: Hunt mängib vanaema ning punamütsike esitab talle küsimusi. Lõpeb sellega et vanaema päästetakse ja kõik on õnnelikud.

Näidisülesanne:

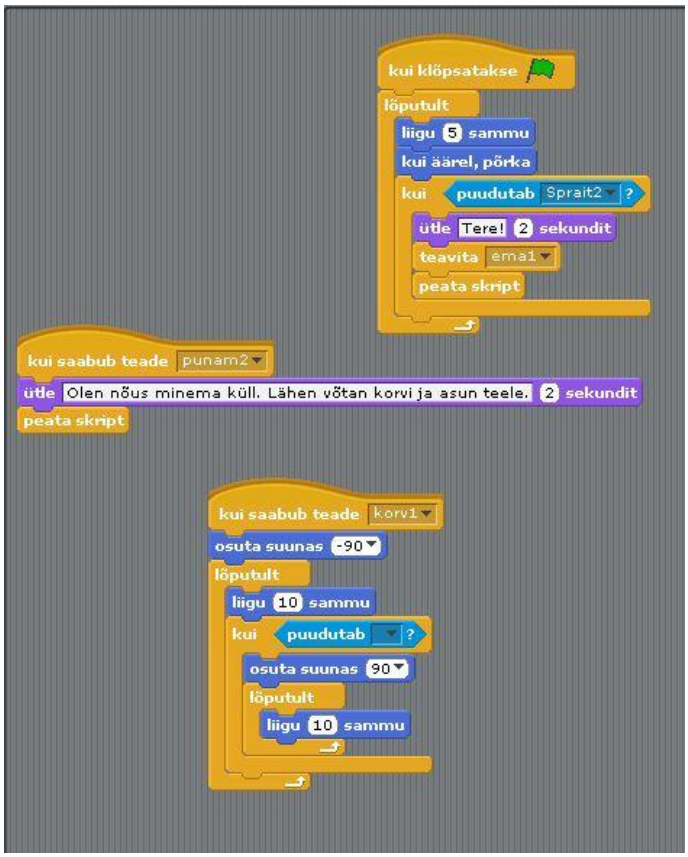
http://*Scratch*.mit.edu/projects/mannu3/1754526

Sprite1 skript

Skripte2 skript



http://*Scratch*.mit.edu/projects/mannu3/1754523



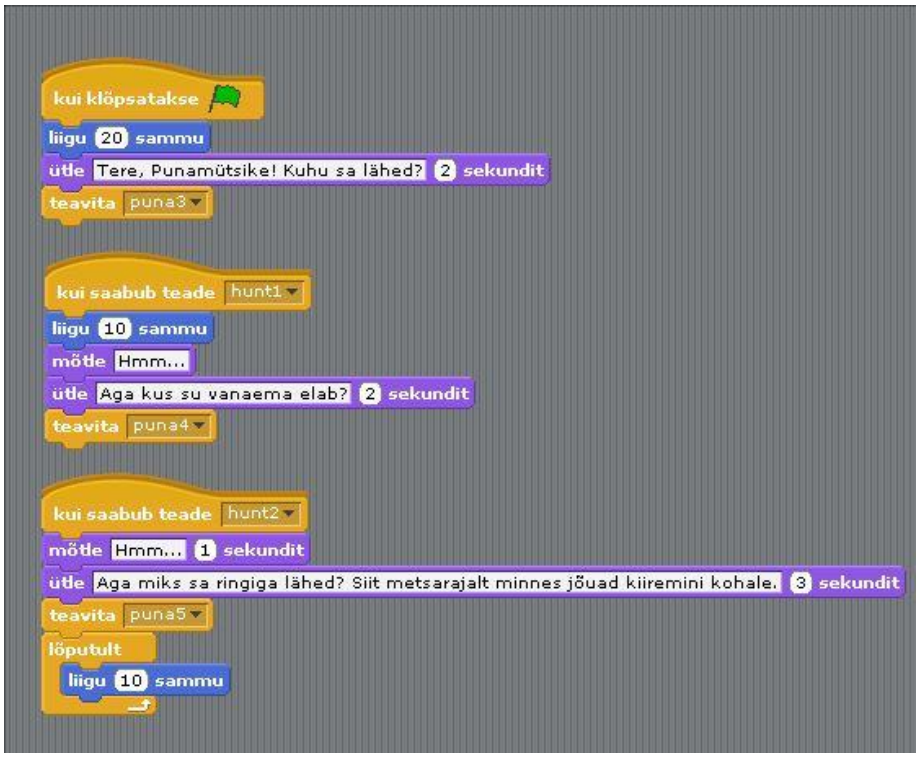
Sprite1 skript.

Sprite2 skript.

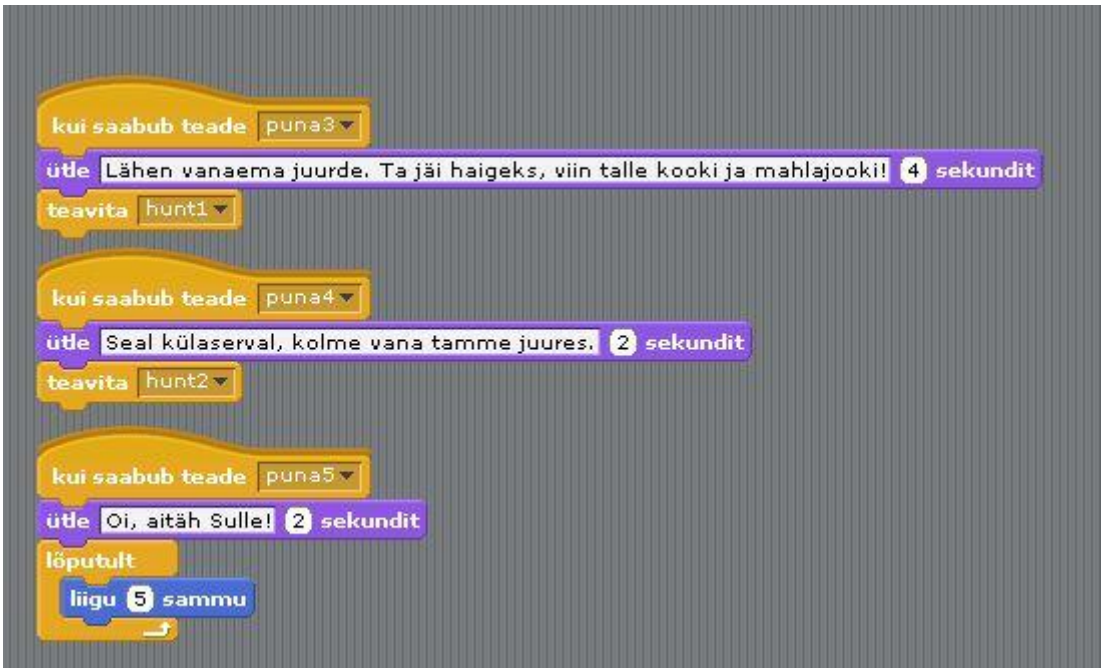
Laval olev pilt.

Õpilaste näited:

http://*Scratch*.mit.edu/projects/mannu3/1747995



Sprite1 skript.



Sprite2 skript.

Laval olev pilt.

Kolmas tund(x90)

Ülesanne:

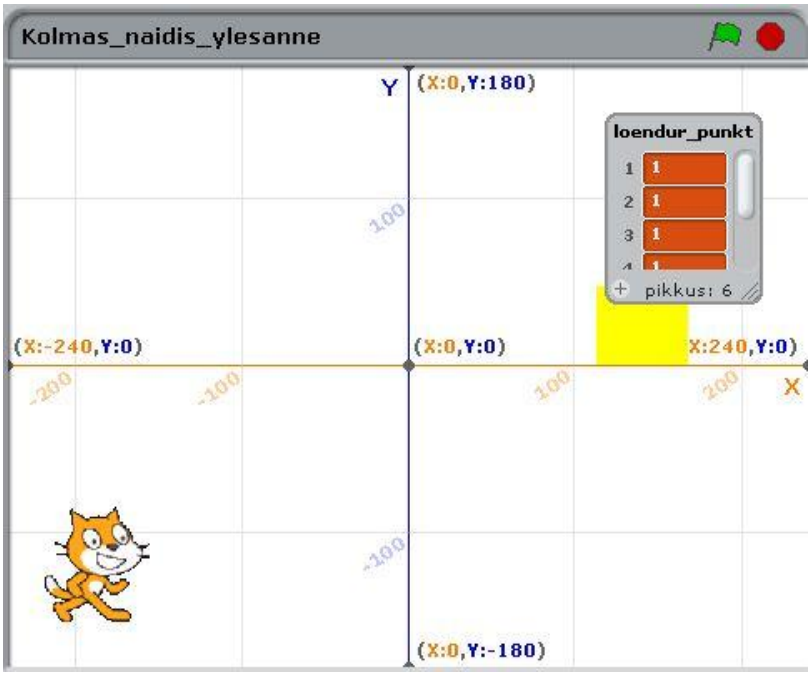
Tegelase liikuma panemine juhtnuppudega. Loenduri lisamine.

Näidisülesanne:

http://*Scratch*.mit.edu/projects/mannu3/1754542

Sprite1 skript.

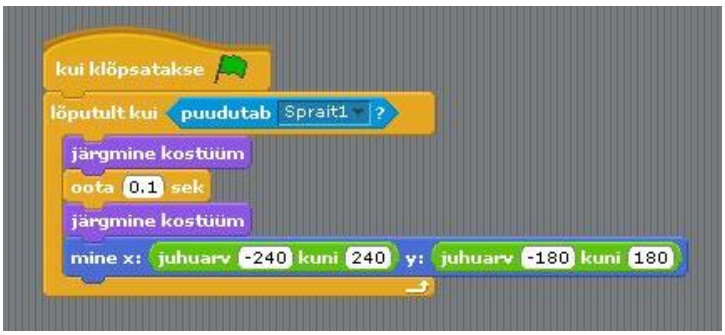
Sprite2 skript.



Laval olev pilt.

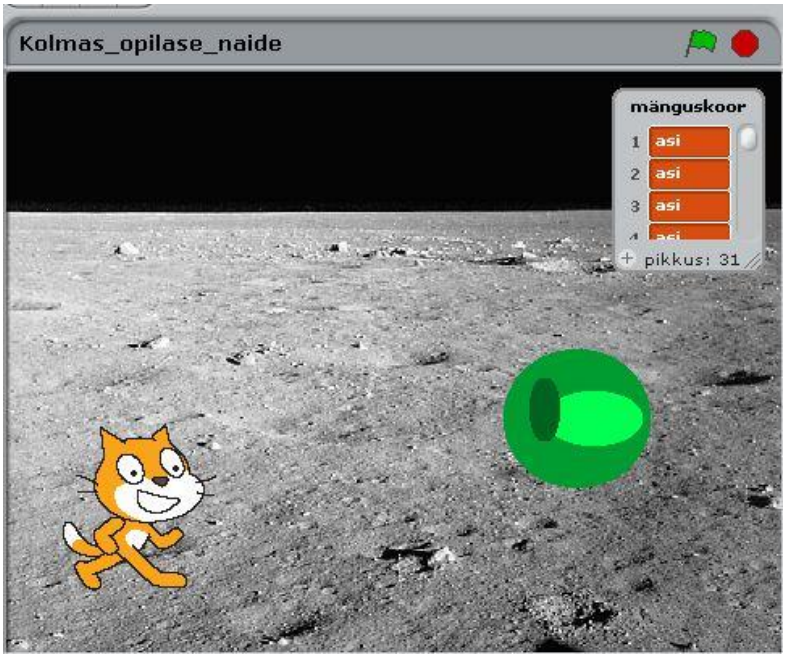
Õpilaste näited:

http://*Scratch*.mit.edu/projects/mannu3/1748039



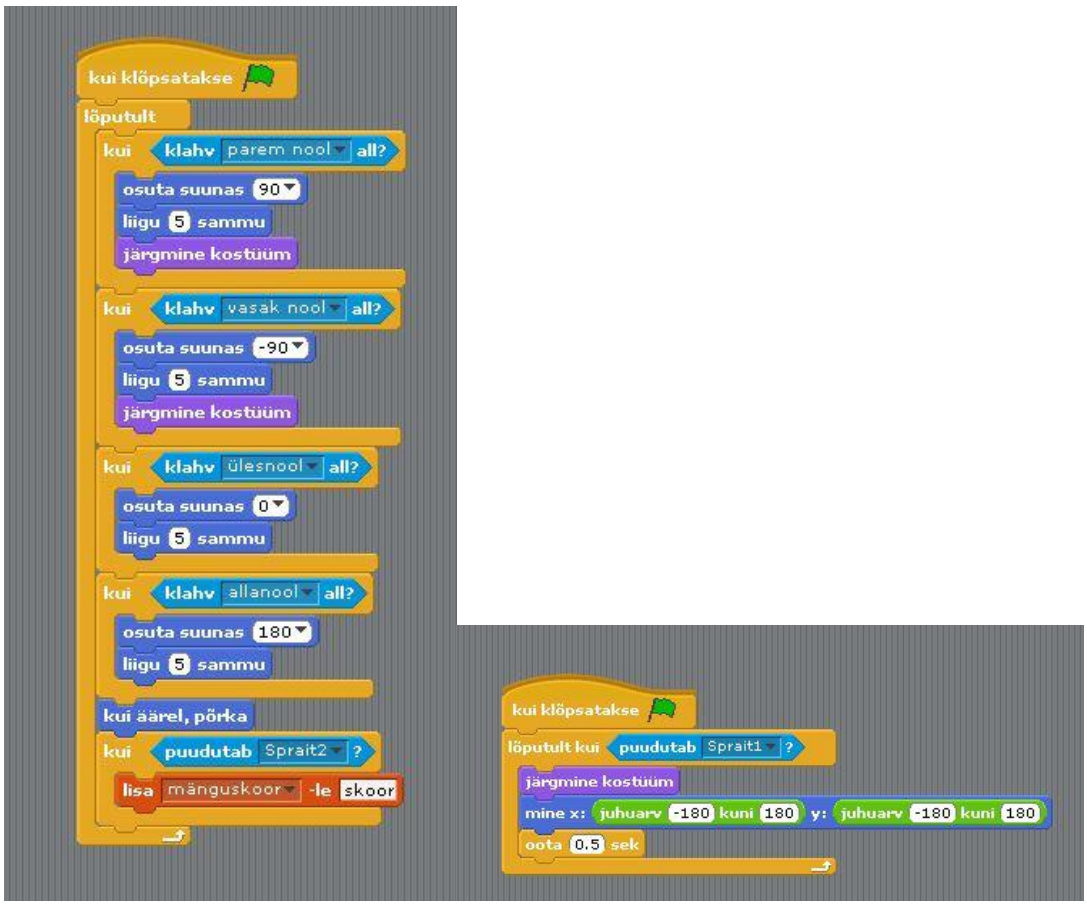
Sprite1 skript.

Sprite2 skript.



Laval olev skript.

http://*Scratch*.mit.edu/projects/mannu3/1748048



Sprite1 skript. *Sprite2* skript

Laval olev pilt.

9. Neljas tund(x90)

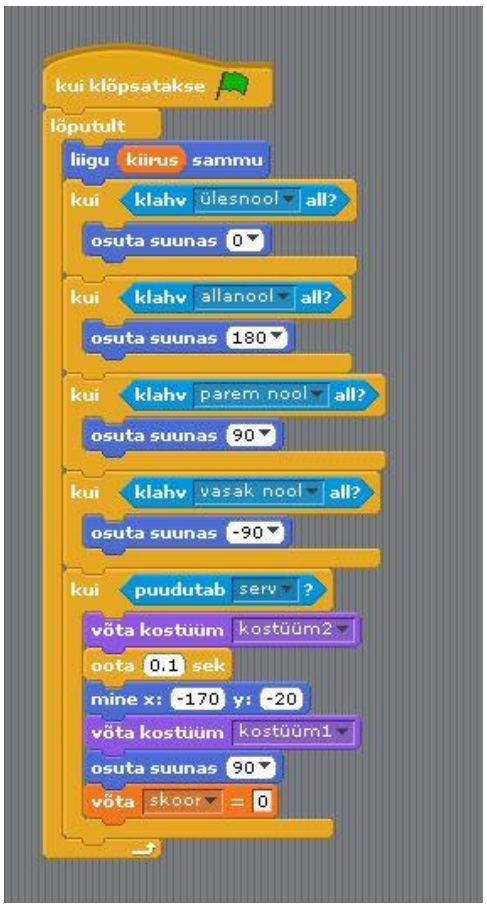
Ülesanne:

Loo ussimäng, millel on järgmised reeglid:

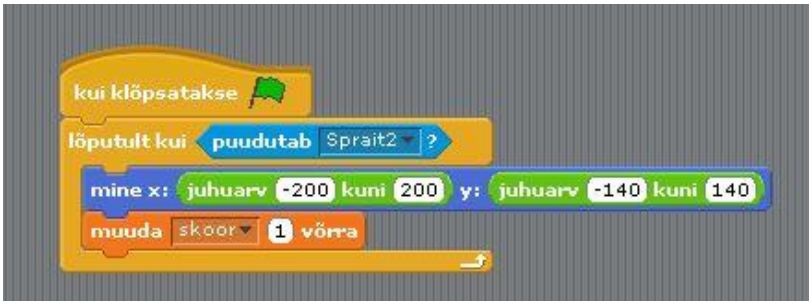
1. Ekraanil on uss, toit, muutuja nimega kiirus, muutuja nimega skoor.
2. Muutuja nimega kiirus on liuguri kujul ning selle värtusi suurendades/vähendades, muutub ussi kiirus.
3. Iga kord, kui uss saab kätte toidu, suureneb muutuja skoor ühe võrra.
4. Iga kord, kui uss põrkub seinaga, muutub skoor nulliks ja uss paigutatakse alguskohale tagasi.
5. On võimalus mängia kahemängija mängu.

Näidisülesanne:

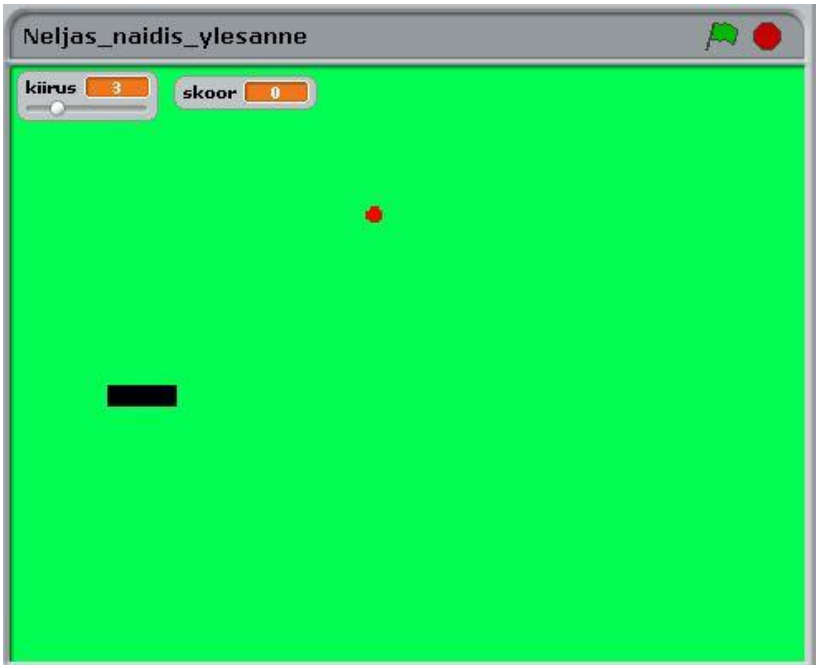
http://*Scratch*.mit.edu/projects/mannu3/1748053



Sprite1 skript.



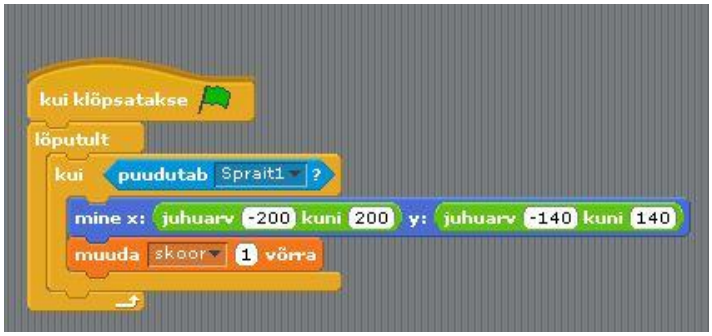
Sprite2 skript.



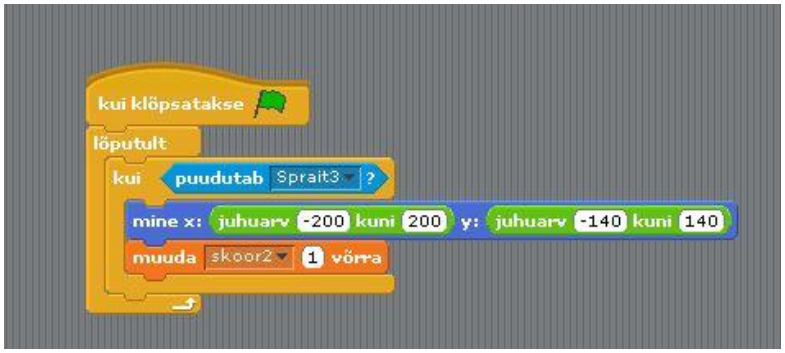
Laval olev pilt

Õpilaste näited:

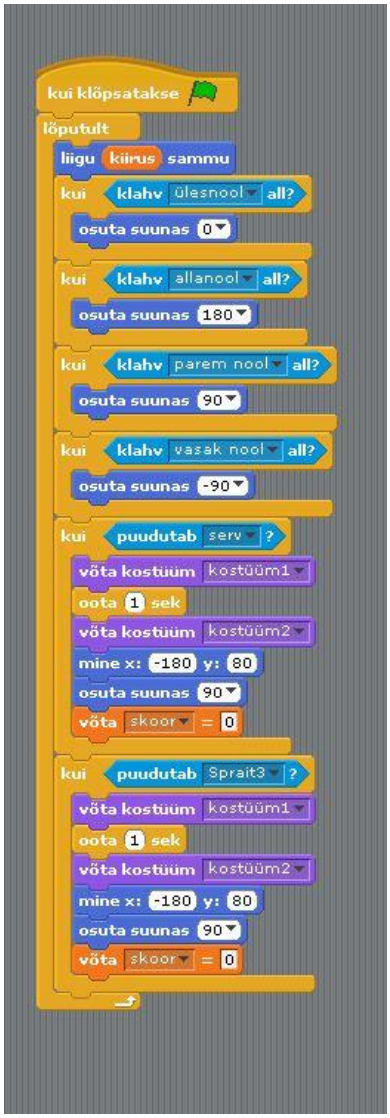
http://*Scratch*.mit.edu/projects/mannu3/1748056



Sprite1 skript.

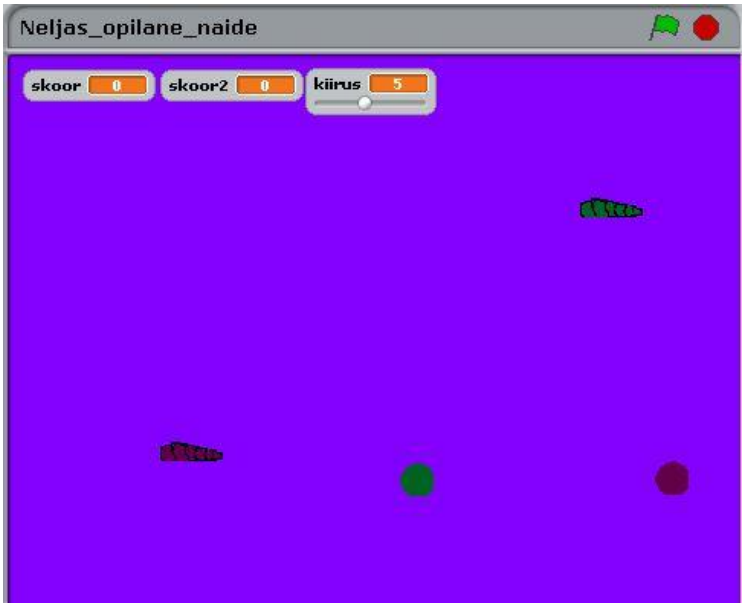


Sprite3 skript.



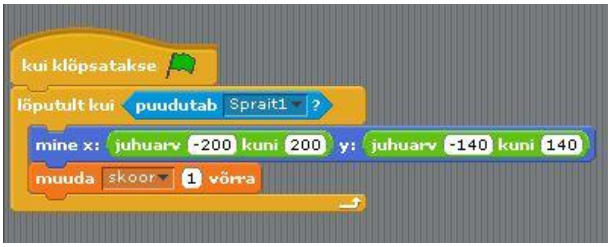
Sprite2 skript

Sprite4 skript



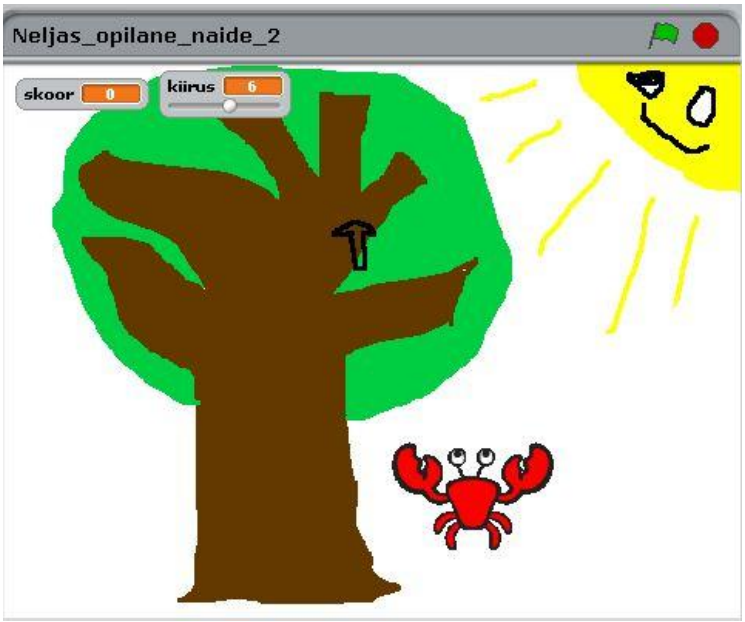
Laval olev pilt.

http://*Scratch*.mit.edu/projects/mannu3/1748059



Sprite1 skript

Sprite2 skript.



Laval olev pilt.