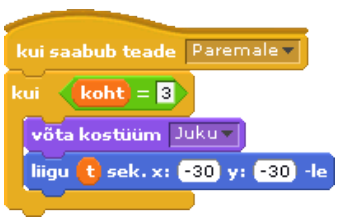
Eesti Teadusagentuur
Estonian Research Council
TeaMe
Europa Lit
Euroopa Sotsiaalfond
Eesti tuleviku heaks
TTÜ
informaatikainstituut
BY NC SA

Rakenduste loomine Scratchiga

Scratch on uue põlvkonna graafiline programmeerimissüsteem, mis võimaldab lihtsalt ja kiiresti omandada algoritmimise ja programmeerimise põhimõtted ning tüüpilised vahendid ja tegevused, mida kasutatakse ka tavapärastes tekstipõhistes programmeerimiskeeltes.

Scratchis pannakse programm kokku käsuplokkidest hiire abil. Süntaksiprobleemid selles keeles praktiliselt puuduvad.



Sisu

Sissejuhatus 3
Ülevaade Scratchi rakenduste komponentidest ja vahenditest 5
Scratchi projekti struktuur ja põhiobjektid 5
Spraidid 5
Skriptid ja käsud 7
Rakenduste loomise ja kasutamise liidesed 7
Lava 8
Andmed 8
Graafikaandmed 8
Heliandmed 8
Märkandmed 9
Konstandid ja muutujad. 9
Operatsioonid märkandmetega. Avaldised 10
Arvavaldised 10
Tekstavaldised 10
Protsesside juhtimine 11
Järjestikune protsess ehk jada 11
Tsüklilised protsessid 11
Hargnevad protsessid 12
Paralleelsed protsessid 13
Sündmused 13
Loendid 13
Scratchi rakenduste kasutamisviisid ja projektide vormingud 14
Näide: Tutvus 15
Ülevaade rakendusest 15
Sissejuhatavad tegevused 16
Kasutaja loobus jätkamisest 16
Kasutaja jätkab 16
Palli skriptid 18
Saleduse leidmine 19
Lava skriptid 20
Rakenduste disainist ja dokumenteerimisest 20
Andmed 21
Rakenduse struktuur 21
Algoritmid 21

Sissejuhatus

Scratch on uue põlvkonna graafiline programmeerimissüsteem. See on loodud Massachusetts'i Tehnoloogiainstituudis (MIT) programmeerimise õppimiseks/õpetamiseks. Scratchi viimane versioon Scratch 1.4 on ilmunud 2009. aastal ning alates sellest algas süsteemi kiire levik. Varsti peaks ilmuma versioon 2.0.

Scratchi saab tasuta alla laadida selle kodusaidilt http://scratch.mit.edu, kus on suurel hulgal õppe- ja abimaterjale erinevates keeltes (sh ka eesti) ning näiteid. Scratchi kodusaidi serverid pakuvad ka pilveteenuseid: paari hiireklõpsuga saab oma rakenduse (projekti) „tõsta pilvedesse“ ja täita seal Java apletina või Flashi klipina. Vastavad teisendused tehakse serveris automaatselt. Praegu on serveris juba mitu miljonit projekti, mida võib vabalt alla laadida, uurida ja remiksida. Järgmises versioonis on ette nähtud võimalus Scratchi rakenduste loomiseks ja täitmiseks otse brauseris. Toimuvad arendustööd vahendite loomiseks, mis võimaldavad kasutada Scratchi rakenduste loomiseks erinevatel mobiilseadmetel: tahvelarvutid, mobiilid jm.

Scratchil on lihtne ja mugav rakenduste loomise keskkond ja kasutajaliides. Süsteemiga on kaasas rikkalik valik graafika- ja heliobjekte, neid saab lisada ka oma failidest ja Internetist. Pilte saab teha veebikaameraga ja heliklippe saab lindistada rakenduste loomise ajal, kui arvutil on mikrofon. On võimalus kasutada sisse-ehitatud joonistamisredaktorit oma graafikaobjektide loomiseks või olemasolevate muutmiseks. Scratchis on pööratud suurt tähelepanu atraktiivsusele ning multimeedia kasutamisele: lihtsalt ja kiiresti on võimalik luua mänge, animatsioone, koomikseid, esitlusi jm. Samas on võimalik lahendada ka arvutuslikke, andmetöötluse- ja graafikaülesandeid. Näiteks realiseerida, uurida ja visualiseerida klassikalisi algoritme massiividega: summade, keskmiste, ekstreemumite leidmine, erinevad otsimise ja sorteerimise meetodid (demo), teha joonistusi, diagramme ja funktsioonide graafikuid jm.

Scratchil on juba üsna mitmeid järglasi ja analooge. Neist tähelepanuväärseim on Berkeley ülikoolis arendatav süsteem **BYOB** (*Build Your Own Blocks*) (http://byob.berkeley.edu).

See on ühilduv Scratchiga ja momendil on oma võimaluste poolest isegi viimasest veidi ees. BYOB võimaldab luua oma plokke (käske), mis vastavad oma olemuselt Scratchi plokkidele ja kujutavad endast parameetritega funktsioone ja protseduure (skripte). BYOBis saab luua ka oma objekte ja klasse.

BYOB võimaldab transleerida enda ja ka Scratchi projekte masinakoodi, saades Windowsi EXE-faili. Praegu on lõpetamisel BYOBi versioon, mis võimaldab luua rakendusi brauseris. Süsteemi nimeks peaks siis saama SNAP!. Uues versioonis peaks ka Scratchil olema analoogsed võimalused.

Veel üheks Scratchi modifikatsiooniks, mis omab võrreldes Scrathiga täiendavaid võimalusi, on **Panther** (http://pantherprogramming.weebly.com).

On tekkinud mitmeid Scratchiga sarnaseid graafilisi programmeerimissüsteeme.

Kõigepealt võiks nimetada suurfirma Google poolt arendatavat süsteemi **Blockly** (http://code.google.com/p/blockly). Ka siin kasutatakse Scratchi käsuplokkidele sarnaseid elemente, skriptid pannakse hiire abil kokku otse brauseris. Süsteemi üheks huvitavaks eriomaduseks on võimalus protseduure tõlkida erinevatesse tekstipõhistesse programmeerimiskeeltesse: Python, JavaScript, Dart ja XML. Üsna sarnane Blocklyga on **MIT App Inventor** (http://appinventor.mit.edu), mõeldud Androidi rakenduste loomiseks mobiilidele. Ning üks omapärane eksperimentaalne süsteem **Calico** (vt http://calicoproject.org), kus ühes ja samas kasutajaliideses saab kasutada erinevaid programmeerimis-süsteeme: graafiline, Scratchiga sarnanev, keel **Jigsaw**, Python, Ruby, robotite programmeerimise vahend Myro jm.

Scratch ja BYOB leiavad järjest laiemad kasutamisest programmeerimise õpetamisel esimese keelena. Need sobivad igas vanuses algajatele: alates koolide algklassidest kuni kolledžite ja ülikoolide programmeerimise sissejuhatavate kursusteni välja. Käsitletavate teemade ja lahendatavate ülesannete valikul tuleb muidugi

arvestada õppijate vanust, taset, õpetamise eesmärke jm, kusjuures valikuvõimalusi ja variatsioone on siin tunduvalt rohkem, kui traditsiooniliste keelte kasutamise korral.

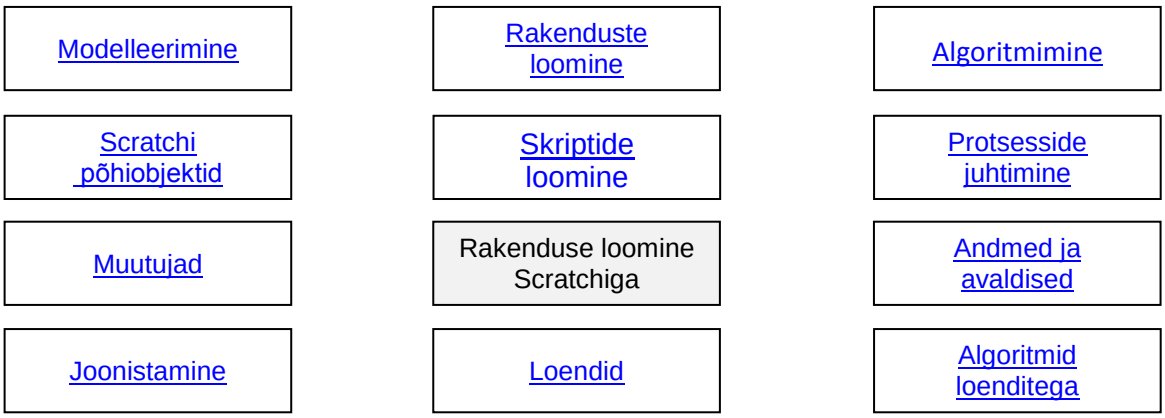
USA arvutiõpetajate assotsiatsiooni (CSTA) poolt koostatud arvutiteaduse tüüpõpekavas (ülevaade, kirjeldused) põhineb aine „Sissejuhatus programmeerimisse“ täielikult Scratchil. Scratchiga või BYOBiga alustatakse programmeerimise kursust näiteks CS50 Harvardi Ülikoolis (vt https://www.cs50.net/psets), Wisconsini Ülikoolis (vt link) jt. Berkeley Ülikoolis põhineb mitteinformaatikute kursus „The Beauty and Joy of Computing“ BYOBil ja Scratchil (vt http://bjc.berkeley.edu).

Kursuse antud osa eesmärgiks ei ole programmeerimiskeele Scratch õppimine. Scratch on siin lihtsalt vahendiks rakenduste loomise ning programmeerimise ja algoritmimise aluste omandamiseks. Scratchi abil on võimalik kiirelt omandada kõik peamised programmeerimise, algoritmimise, modelleerimise ja rakenduste loomise põhimõtted ja tüüptegevused: eriliiki andmete (märk-, graafika-, heli- ja videoandmed) olemus ja kasutamise iseärasused, operatsioonid andmetega ja avaldised, andmete sisestamine ja väljastamine, joonistamise põhimõtted, muutujad ja massiivid, protsesside liigid ja nende kirjeldamine: jada, kordus, valik ja paralleelsus, programmide jagamine üksusteks ning koostöö ja andmevahetuse korraldamine nende vahel, rakenduste kasutajaliideste loomise põhimõtted, objektorienteeritud lähenemisviisi ja modelleerimise olemus. Kuigi Scratch praegu ei toeta kõike objektorienteeritud (OO) programmeerimise võimalusi (tegemist on nn objektipõhise süsteemiga), võimaldab ta tutvustada ja osaliselt rakendada OO lähenemisviisi.

Scratchi materjalid on esitatud kahel tasemel.

Võimaldamaks õppijal saada kiiresti ülevaade Scratchist, on esimesel tasemel toodud lühiülevaade Scratchi kasutamisest rakenduste loomiseks. Sellega on soovitav tutvuda kohe alguses, paralleelselt esimeste harjutustega.

Antud osaga on linkidega seotud lisamaterjalid (teine tase), kust saab täiendavat informatsiooni vastava teema kohta. Lisamaterjalidele võivad olla viited ka praktikumide töölehtedel.



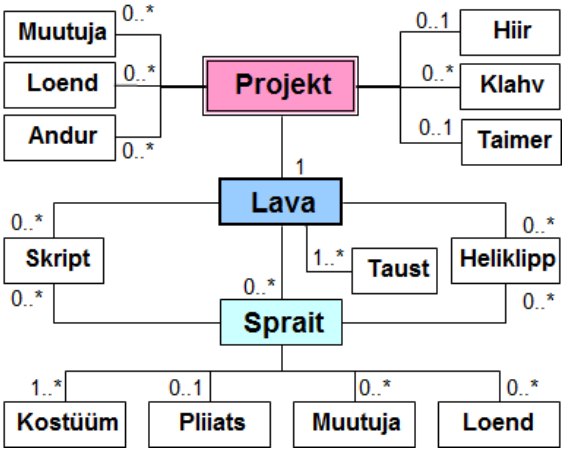
Ülevaade Scratchi rakenduste komponentidest ja vahenditest

Siin on toodud lühiülevaade Scratchi põhikontseptsioonidest, komponentidest ja keelevahenditest. Vajadusel saab tekstis toodud linke kasutades põhjalikumat infot lisamaterjalidest.

Scratchi rakendusi nimetatakse **projektideks**. Projekti põhikomponentideks on spraidid ja skriptid. Saab luua väga erineva otstarbega, sisuga ja struktuuriga projekte (vt näiteks Scratch wiki ja Scratchi kodusait).

Scratchi projekti struktuur ja põhiobjektid

Scratchi projekt koosneb omavahel seotud objektidest. Kasutatavad **objektid** ja **seosed** nende vahel on toodud UML klassidiagrammina (vt modelleerimine)



Seostel toodud kordused näitavad, mitu objekti ühest klassist võivad olla seotud teise klassi objektiga. Vaikimisi on kordus 1.

Projektis on alati üks ja ainult üks lava (kordus 1), millel võib asuda suvaline arv spraite, sh ka mitte ühtegi (kordus 0..*). Sama tüüpi kordus on mitmel teisel objektil: skript, heliklipp, muutuja jm. Laval on alati vähemalt üks taust ja spraidil vähemalt üks kostüüm (1..*). Kordus 0..1 näitab, et kasutusel võib olla üks objekt või ei ühtegi (hiir, taimer, pliiats).

Projekti loomisel ja täitmisel saab kasutada ja muuta objektide **omadusi** (spraidi koordinaadid, suurus ja nähtavus, taimeri ja muutuja väärtus jm). Scratchi käsuplokid kujutavad endast objektide **meetodeid**, mille abil saab määrata tegevusi spraitide ja nende alamobjektidega (pliiats, heli jm).

Lisamaterjal: Scratchi põhiobjektid, Scratch wiki.

Spraidid

Scratchi rakenduste kesksteks komponentideks on **graafikaobjektid**, mida nimetakse **spraitideks** (ingl. ***sprite*** – haldjas või vaim). Programmide (skriptide) abil saab määrata mitmesuguseid tegevusi spraitidega: muuta asukohta, suurust, värvust, panna kõndima, tantsima, tekitama helisid, arvutama, joonistama jms. Süsteemiga tuleb kaasa üsna suur valik spraite, mis asuvad liikide kaupa (inimesed, loomad, asjad jm) erinevates kaustades. Iga spraidiga on seotud teatud valik **omadusi**: nimi, asukoht laval (x-y), suurus, värvus jm, ning **meetodeid**, mis on realiseeritud käsuplokkidena. Spraitide loomisel saab kasutada ka oma graafikaobjekte: PNG, GIF, JPG, BMP ning animeeritud GIF faile. Spraite kostüüme võib luua ja muuta **joonistamisredaktori** abil (vt Kasutamisjuhend ja Scratch wiki).

Spraidid tegutsevad ekraanil piirkonnas, mida nimetatakse **lavaks**.

Spraidil võib olla mitu **kostüümi** (*costume*): tüüpiliselt spraidi erinevad variandid (teisikud). Praktiliselt kujutab endast sprait graafiliste kujundite (kostüümide) kogumit. Kui sprait on nähtav, siis on nähtav üks selle kostüümidest (kujutistest). Vahetades kostüüme programmi käskude abil teatud sagedusega, saab tekitada animatsiooniefekte: kõndimine, tantsimine vm. Kostüüme võib kasutada ka muul viisil, näiteks erinevate teadete kuvamiseks.

Lisamaterjal: Objektid Scratchis, Kasutamisjuhend, Scratch wiki.

Skriptid ja käsud

Scratchi projektis võivad olla kasutusel programmiüksused, mida nimetakse skriptideks. **Skript** (***script***) koosneb **käskudest** ehk **lausetest**, mis on realiseeritud graafiliste plokkidena. Plokid (käsud) on jagatud funktsioonide alusel (liikumine, juhtimine, helid jm) gruppidesse. Käsud jagunevad liht- ja **liitkäskudeks**. Viimased sisaldavad teisi liht- ja/või liitkäske.

Plokid ühendatakse omavahel hiire abil järjekorras, milles peavad toimuma tegevused. Skript algab **päiseplokiga**, mis on seotud skripti käivitamisviisiga: rohelise lipu klõpsamine, vajutus klahvile klaviatuuril, teate saabumine jm.

Skript kuulub alati ühele kindlale spraidile või lavale. Ühel spraidil võib olla suvaline hulk skripte. Skriptid saavad teha omavahel koostööd teadete abil.

Programmi jagamine mitmeks omavahel koostööd tegevaks üksuseks on väga tähtsaks põhimõtteks ükskõik millistes süsteemides programmeerimises. See võimaldab luua paindlikke ja ökonoomseid rakendusi, kasutada üksusi korduvalt nii samas kui ka teistes rakendustes, täita tegevusi paralleelselt jm.

Lisamaterjal: Skriptid Scratchis, Kasutamisjuhend, Scratch wiki.

Scratchis on koostöö skriptide vahel rajatud **teadete saatmisele** ühe skripti poolt ja nende **vastuvõtmisele** teiste skriptide poolt. Käsuga [**teavita *nimi***] või [**teavita *nimi*** ja oota] väljastatakse teade (signaal), mille võib võtta vastu ja hakata tööle suvaline arv skripte päisega [**kui saabub teade *nimi***], kus ***nimi*** on sama nagu käsus [**teavita** …]. Tegemist võib olla sama spraidi skriptidega või teise spraidi omadega. Teate vastuvõtnud skriptid alustavad tööd paralleelselt. Näiteks võib mitmel spraidil olla kõrvalolev skript, need hakkavad kõik tööle, kui mingi skript saadab teate [**teavita *hüppa***…]. Sama päisega (nimega) skriptid võivad olla ka erineva sisuga.

Kui käsus [**teavita** …] on täiend **oota**, siis peatatakse teate saatnud skripti täitmine, kuni lõpeb kõikide teate vastuvõtnud skriptide täitmine ning peale seda jätkub antud skripti töö. Kui täiendit **oota** käsus ei ole, jätkab teate saatnud skript kohe oma tööd.

Rakenduste loomise ja kasutamise liidesed

Scratchi kasutajaliides (projekti loomise liides) kujutab endast ekraanipiirkondade kogumit, kus asuvad vahendid rakenduste loomiseks. Selle abil realiseeritakse tavaliselt rakenduse kasutajaliides, mille abil kasutaja suhtleb valmisrakendusega. Lisamaterjal: Kasutamisjuhend, Scratch wiki.

Pildil on Scratchi rakenduste loomise liidese põhielemendid. Akna ülaservas on menüü ja tööriistariba.

Liidese aknas võib eristada järgmisi alasid: **lava**, **spraitide loetelu** (kasutatud spraitide ja lava ikoonid), **aktiivse spraidi jooksev info** (siit on valitavad **skriptide**, **kostüümide** ja **helide vahelehed**), **skriptide** ala, **kostüümide** ala, **helide** ala, **käskude plokid** ja vastavad **plokkide grupid**.

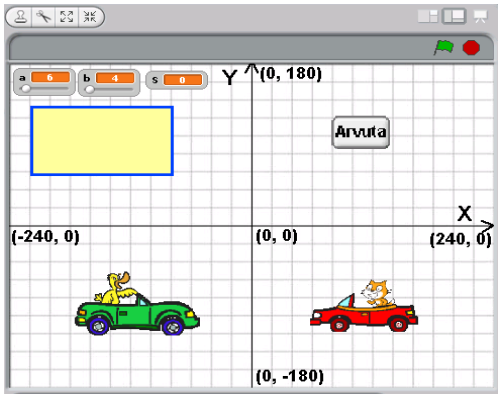
Lava kohal asuvate nuppude abil saab valida lava kolme oleku (suuruse) vahel: **normaalne**, **väike** ja **esitlus**. Koos sellega muutub jaotus liidese erinevate alade vahel. Esitluse olekus on täisekraanil nähtav ainult lava ning käivitamise ja peatamise nupud. Selles olekus ei saa lisada uusi spraite ja skripte.

Rakenduse kasutamise liides luuakse laval. Selle elementideks võivad olla taustad, aktiivsed ja passiivsed spraidid, käsunupud, klahvid, muutujate monitorid jm. Teistes süsteemides kasutatakse selleks tavaliselt vorme ja dialoogibokse.

Lava

Lava (*Stage*) on ekraani piirkond, kus tegutsevad kõik spraidid. Siin asub ka suurem osa kasutajaliidese elementidest: muutujate monitorid, käsunupud jm.

Lisamaterjal: Objektid Scratchis, Kasutamisjuhend, Scratch wiki.



Laval võib olla mitu **tausta**. Vaikimisi on tavaline valge taust. Taustu saab lisada, eemaldada, redigeerida. Taust kujutab endast graafikaobjekti (pilti) ning on oma olemuselt analoogne spraidi kostüümiga.

Koordinaattelgede algus on lava keskel. Lava on 480 pikselit lai (x_min = -240, x_max = 240) ja 360 pikselit kõrge (y_min = -180, y_max = 180). Tegemist on tingühikuga, mille suurus sõltub ekraani suurusest ja kasutatavast eraldusvõimest.

Andmed

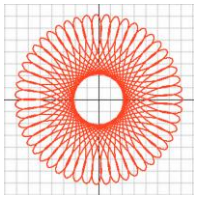
Scratchi rakendustes saab kasutada **graafika**-, **heli**- ja **märkandmeid**. Lisamaterjal: Andmed ja avaldised.

Graafikaandmed

Scratchis võib esineda kahte liiki graafikaandmeid: **graafikaobjektid** ja **graafilised kujutised**.

Graafikaobjektid imporditakse graafikafailidest või luuakse Scratchi joonistamisredaktoriga projekti loomise faasis. Formaalselt on tegemist kas lava taustadega või spraitide kostüümidega. Objektide loomiseks võib kasutada PNG, GIF, JPG, BMP faile.

Lisamaterjal: Scratchi põhiobjektid



Graafilised kujutised luuakse projekti täitmise ajal. Selleks kasutatakse käske gruppidest **Pliiats** ja **Liikumine**. Grupi **Pliiats** käsud võimaldavad määrata pliiatsi omadusi: värv, joone paksus, olek (üleval või all) jm. Grupi **Liikumine** käskude abil muudetakse joonistava spraidi asukohta vastavalt kujutise geomeetriale.

Scratchis kasutatakse nn **kilpkonnagraafikat**. Meetod ja nimetus võeti kasutusele aastaid tagasi programmeerimissüsteemis **Logo**, mis leiab laialdast kasutust, eriti programmeerimise õpetamisel, ka praegu. Kilpkonnagraafika on kasutusel mitmetes programmeerimissüsteemides: Python, BYOB, Basicu mitmed versioonid jm. Selle elemente kasutatakse praktiliselt kõikides programmeerimissüsteemides, tegemist on arvutigraafika ja eriti vektorgraafika algelementidega.

Lisamaterjal: Joonistamine, Scratch wiki.

Heliandmed

Scratchis võib eristada ka kahte liiki heliandmeid: **heliobjektid** ja hetkhelid ehk **helindid.**

Heliobjektid on heliklipid, mis imporditakse või lindistatakse projekti loomise ajal. Saab kasutada WAV ja MP3 vormingus faile. Heliklippe saab esitada grupi **Heli** käskudega [**mängi heli** *nimi*] ja [**mängi heli** *nimi* kuni valmis].

Hetkhelid tekitatakse arvuti poolt vastavate käskude toimel. Scratchis kasutatakse selleks käske mängi nooti ja mängi trummi. Mõlemal juhul saab valida erinevaid instrumente, heli tugevust ja tempot.

Lisamaterjal: Scratchi põhiobjektid, Scratch wiki.

Märkandmed

Scratchis kasutatakse järgmisi märkandmeid: **arvud**, **tekstid** ja **tõeväärtused**.

Arvude jaoks ei ole Scratchis erinevaid vorminguid ja tüüpe, nagu on iseloomulik teistele programmeerimiskeeltele. Tegemist on nö tavaliste täis- ja reaalarvudega, kusjuures formaalset vahet nende vahel ei tehta. Murdosa eraldamiseks täisosast võib reaalarvudes kasutata koma või punkti. Arve võib esitada käsuplokkides ja avaldistes konstantidena ning sisestada klaviatuurilt. Arvude jaoks on ette nähtud neli põhitehet: **+** , **-** , ***** , **/** ning võrdlustehted, mis esitatakse vastavate plokkide abil. On olemas teatud valik matemaatilisi funktsioone.

Tekstandmete väärtuseks on Scratchis, nagu ka teistes programmeerimissüsteemides, suvaliste märkide jada. Tekste saab kasutada ainult mõnedes plokkides: teadete väljastamiseks, dialoogide korraldamiseks jmt. Tekst kirjutakse konstandina otse plokki, sisestatakse klaviatuurilt, omistatakse muutujale või loendi elemendile. Tekstide jaoks on olemas mõned tehted: ühendamine, märkide eraldamine, võrdlemine jmt.
Tõeväärtuste ehk loogikasuuruste jaoks on olemas ainult kaks väärtust ehk loogikakonstanti: tõene (*true*) ja väär (*false*). Ilmutatud kujul neid Scratchis praktilisel ei kasutata. Loogikaväärtused tekivad võrdlustes ja loogikaavaldistes, mida kasutatakse tingimuste esitamiseks.

Konstandid ja muutujad.

Konstandi väärtus kirjutatakse (esitatakse) otse käsuplokis. Programmi täitmisel konstandi väärtust muuta ei saa. Arvus võivad olla ainult numbrid, miinusmärk ja murdosa eraldaja (koma või punkt). Tekstkonstante ei paigutata mingite piirajate (jutumärgid või ülakomad) vahele nagu seda peab tegema tekstipõhistes programmeerimiskeeltes.

Muutuja on üheks kesksemaks mõisteks kõikides programmeerimissüsteemides. Selle tähendus programmeerimises ei ole päris sama, mis muutujal matemaatikas. Programmeerimises mõistetakse muutuja all **nimega varustatud kohta arvuti mälus** (**mälupesa** või mäluväli), kuhu programm saab salvestada väärtusi. Salvestatud väärtust saab hiljem kasutada näiteks uue väärtuse leidmiseks.

Lisamaterjal: Muutujad Scratchis.

Scratchis luuakse muutujad „käsitsi“ enne programmi täitmist grupis **Muutujad** asuva korraldusega **Tee muutuja**. Loomisel kuvatakse dialoogiboks, milles saab näidata muutuja nime ja skoobi. Skoop määrab, kas muutuja on kättesaadav kõikidele spraitidele ja lavale või ainult ühele spraitidele. Iga muutuja jaoks luuakse automaatselt monitor ehk näidik, mis esitab laval muutuja jooksva väärtuse: arv või tekst. Monitori saab ka peita. On kolm monitori tüüpi: tavaline monitor, suur monitor ja liuguriga ehk kerimisribaga monitor. Viimasega saab muuta muutuja arvväärtust nö „käsitsi“. Monitoride näited:

tavalised:

,

; suured:

,

; liuguriga:

.

Scratchi skriptides saab muutujatele omistada väärtusi **omistamiskäskudega**:

Avaldis määrab väärtuse leidmise eeskirja. Selle erijuhtudeks on ka konstant ja muutuja. Käsu täitmisel leitakse avaldise väärtus ja tulemus võetakse muutuja väärtuseks. Mõned näited:

,

,

.

Enamikes tekstipõhistes programmeerimiskeeltes ja algoritmides esitatakse omistamiskorraldus taoliselt:

muutuja = ***avaldis***

näiteks: vahe = pikkus - kaal; P = 2 * (a + b), olemus aga on sama.

Operatsioonid märkandmetega. Avaldised

Avaldise abil antakse eeskiri vajaliku väärtuse leidmiseks: määratakse tehted ja operatsioonid ning nende täitmise järjekord. Avaldiste koostamiseks kasutatakse vastavaid tehete ja funktsioonide plokke.

Lisamaterjal: Andmed ja avaldised.

Arvavaldised

Arvude jaoks on ette nähtud **neli põhitehet**, mis esitatakse vastavate plokkide abil.

Tehete järjekord määratakse plokkide paigutusega avaldises. Oluline on arvestada, et ühes plokis olev avaldise osa vastab sulgudes olevale avaldisele. Näiteks plokk

vastab avaldisele (a + b).

Arvandmete jaoks on olemas teatud valik **matemaatikafunktsioone**. Need esitatakse avaldistes taoliste plokkide abil:

.

Vasakpoolses väljas saab valida funktsioon nime: abs, sqrt, sin, cos, tan, asin, acos, atan, ln, log, e^, 10^. Argument võib üldjuhul olla avaldis, erijuhul muutuja või konstant:

$\sqrt{a^2+b^2}$

sin(x + 45) – argument kraadides.

Tekstavaldised

Scratchis ei ole eriti palju vahendeid tegevuste täitmiseks tekstidega.

Tekstide ühendamiseks saab kasutada plokki

Mõlemas väljas võivad olla omakorda ühendamise plokid.

,

,

Plokk

võimaldab leida antud teksti pikkuse: märkide arvu tekstis.

Plokk

tagastab antud numbriga (nr) sümboli.

Protsesside juhtimine

Scratchis kujutab protsessi kirjeldus käsuplokkide gruppi, milleks võib olla terve skript või osa sellest. Ülesannete lahendamisel, sõltumata kasutatavatest vahenditest, võib eristada nelja liiki protsesse:

- järjestikune protsess ehk jada
- tsükliline protsess ehk kordus
- hargnev protsess ehk valik
- paralleelsed protsessid.

Lisamaterjal: Protsesside juhtimine, Scratch wiki.

Järjestikune protsess ehk jada

Järjestikuse protsessi korral täidetakse kõik käsud täpselt sellises järjekorras nagu nad on esitatud. Käske ei jäeta vahele ega toimu kordamisi. Mingeid erilisi vahendeid protsessi määratlemisel ei ole vaja, sest täitmine toimub nö loomulikus järjekorras.

Järgnevalt näitena toodud ülesande lahendamine kujutab oma olemuselt järjestikust protsessi. Programm küsib ja loeb ristküliku külgede pikkused (*a* ja *b*), arvutab selle pindala (*S*) ja ümbermõõdu (*P*) ning leiab pindala ja ümbermõõdu suhte (*suhe*).

NB! käskude järjekorra valimisel peab arvestama, et neid täidetakse täpselt selles järjekorras, nagu need on esitatud programmis. Algandmete lugemise käsud peavad eelnema pindala ja ümbermõõdu arvutamisele ning viimased omakorda suhte leidmisele. Samal ajal ei oma tähtsust *a* ja *b* lugemise omavaheline järjekord ning *S* ja *P* arvutamise järjekord.

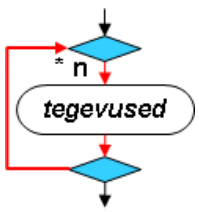
Tsüklilised protsessid

Tsükliline protsess seisneb tegevuste (käskude) korduvas täitmises. Programmis tuleb määrata reeglid ja tingimused, kuidas toimub protsessi täitmine. Scratchis on korduste kirjeldamiseks mitu erinevat tüüpi käsuplokki, mis võimaldavad arvestada erinevat tüüpi protsessidega.

Lisamaterjal: Algoritmimine.

Etteantud korduste arvuga kordus

esitatakse Scratchis plokiga [**korda** *n*], kus *n* on kordamiste arv. Viimane võib olla esitatud konstandi, muutuja või avaldise abil. Allpool on toodud ka antud korduse tüübi esitus UMLi diagrammidel ja algoritmikeeles / pseudokoodis.



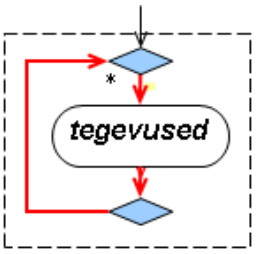
kordus *n* **korda**

tegevused

lõpp kordus

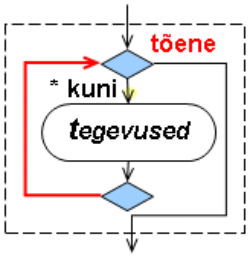
Lõputu kordus

Formaalselt täidetakse ploki sees olevaid käske lõputult, kuid praktiliselt katkestatakse kordus ühel või teisel viisil. Korratavate käskude seas võib olla näiteks käsk [**peata skript**], mis täidetakse teatud tingimuse täitumisel ja see lõpetab lõputut kordust sisaldava skripti töö. Korduse võib katkestada käsk [**peata kõik**], mis asub samas või mõnes teises skriptis. Alati saab korduse katkestada punase nupuga, mis lõpetab terve rakenduse töö.

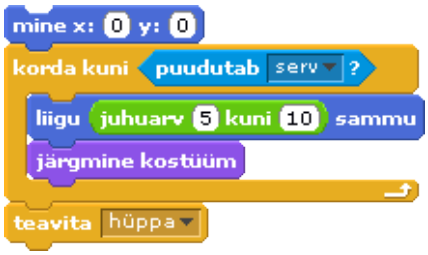


kordus
tegevused
lõpp kordus

Eelkontrolliga tingimuslik kordus



kordus kuni *ting*
tegevused
lõpp kordus



Selle korduse liigi korral **korratakse** tegevusi seni, **kuni** antud tingimus **saab tõeseks**.

Tingimuste esitamiseks kasutatakse loogikaavaldisi, mis esitatakse võrdluste ja loogikatehete plokkide abil:

Loogikaavaldise tulemuseks saab olla ainult tõeväärtus: tõene (*true*) või väär (*false*).

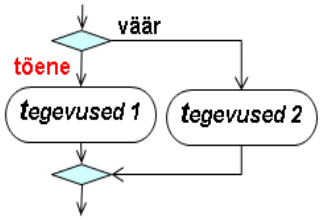
Hargnevad protsessid

Hargnevas protsessis valitakse üks mitmest võimalikust tegevusest, sõltuvalt antud tingimustest.

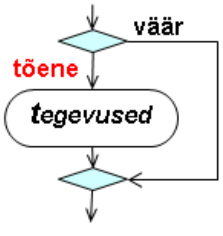
Vt ka Algoritmimine.

Scratchis on kaks valikuplokki: valik kahest ja valik ühest. Esimesel juhul täidetakse kahest võimalikust plokkide grupist üks. Teisel juhul, sõltuvalt tingimuse väärtusest, antud plokid kas täidetakse või jäetakse vahele. Iga plokk võib omakorda sisaldada valikuplokke. Tingimuste esitamiseks kasutatakse loogikaavaldisi, mis esitatakse võrdluste ja loogikatehete plokkide abil:

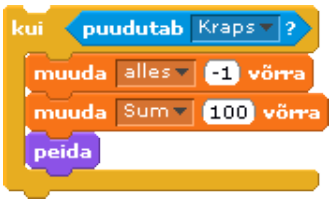
Loogigaavaldise väärtuseks saab olla ainult tõeväärtus: tõene (*true*) või väär (*false*).



kui *tingimus* **siis**
tegevused 1
muidu
tegevused 2
lõpp kui



kui *tingimus* **siis**
tegevused
lõpp kui



Paralleelsed protsessid

Kasutaja saab korraga käivitada mitu skripti, mille päises on plokk rohelise lipuga või käsuga [**kui vajutakse klahvi** …]. Lisaks juba töötavatele skriptidele võib kasutaja käivitada, kui rakenduses on ette nähtud, täiendavaid skripte, mis alustavad tööd paralleelselt eelmistega. Nendeks võivad olla skriptid, mis reageerivad klahvisündmusele või hiireklõpsule: algavad päisega [**kui vajutatakse klahvi** …] või [**kui klõpsatakse *sprait***].

Skriptidest saab paralleelseid protsesse käivitada käsuplokiga [**teavita *nimi***] ja [**teavita *nimi* ja oota**]. **K**orraga hakkavad tööle kõik skriptid, mille päises on plokk [**kui saabub teade *nimi***], kus nimi on sama nagu käsus **teavita**. Tegemist võib olla sama spraidi skriptidega või teise spraidi omadega.

Sündmused

Süsteemis on ette nähtud erinevate sündmuste kontroll ja reaktsioon nendele. Sündmusi saab tekitada kasutaja hiire ja klahvidega ning skriptid teadete saatmisega. Üheks enim esinevaks reaktsiooniks sündmustele on skriptide käivitamine, mis on seotud päiseplokkidega. Scratchi põhiobjektid

- kui klõpsatakse rohelist lippu
- kui klõpsatakse spraiti
- kui klõpsatakse lava,
- kui vajutatakse klahvi
- kui saabub teade

Sündmuste kontrolli ja reaktsioone neile saab skriptides teostada andurite ja vastavate tingimuste kasutamisega:

- puudutab objekti
- hiir all?
- klahv all?
- taimer

Loendid

Loend (*List*) on järjestatud mäluväljade kogum, mis tähistatakse ühe nimega. Loendi elementidele viidatakse nime ja indeksi abil: arvutid 1, hinnad k.

Käsuga **Tee loend** luuakse tühi loend ja laval tekkib tekib loendi monitor. Loendi elemente saab lisada, muuta ja eemalda nö "käsitsi". Neid saab importida tekstifailist ja eksportida tekstifaili. Elemente saab lisada, eemaldada ja muuta skriptides vastavate käskude abil.

Taolisi andmekogumeid nimetatakse sageli ka ühemõõtmelisteks massiivideks.

Näiteks on toodud kahe loendi monitorid ja skript, mis leiab loendi **hinnad** elementide aritmeetilise keskmise, mis omistatakse muutujale **kesk**. Skript teeb kindlaks elementide arvu loendites (loendite pikkused) ja omistab selle muutujale **n**. Plokis **korda** liidetakse järjest loendi **hinnad** elemendid muutujale **Sum**, saades hindade summa. Selle jagamisel muutuja **n** väärtusega saadakse keskmine.

Lisamaterjalid: Loendid1, Loendid2, Scratch wiki.

Scratchi rakenduste kasutamisviisid ja projektide vormingud

Scratchi rakendusi saab kasutada (täita) erineval viisil.

- Oma keskkond, fail on laiendiga **sb,** võib asuda kliendi arvutis või serveris. Programmi täitmist korraldab **interpretaator**, mis transleerib (tõlgib) skriptid ühekaupa masinakeelde ja suunab koodi täitmiseks protsessorisse. Näide: hunt_kits_kapsas.sb. Fail laetakse serverilt ning automaatselt avatakse Scratchis.
- Scratchi projekti saab siduda html-dokumendiga märgendiga <applet…>. Süsteem loob märgendi alusel projektist Java apleti ja paigutab selle html-dokumenti (vt link). Viimase laadimisel brauserisse aktiveeritakse aplett. Näide: hunt_kits_kapsas.html.
- Projekti täitmine „pilvedes“ – Scratchi kodusaidil MIT-serveris. Menüükäsuga ’jaga projekti internetis’ saab laadida projekti Scratchist serverisse. Käsuga ’mine Scratchi kodulehele’ saab Scratchist otse minna Scratchi avalehele (scratch.mit.edu). Kui kasutaja on loonud serveris konto, kuvatakse tema projektide loetelu, kust saab valida täitmiseks sobiva projekti. See esitatakse Java apletina või flashi klipina. Kasutaja võib lisada mingisse dokumenti lingi serveris asuvale (oma või võõra) projektile. Näiteks http://scratch.mit.edu/projects/vilip/2828630. Aadressi algus on standardne, lõpus on kasutajanimi (vilip) ja projekti järjenumber (2828630), mille süsteem fikseerib automaatselt üleslaadimise hetkel. Aadressi saab näiteks kopeerida, kui projekt muudetakse serveris aktiivseks.
- EXE-faili kasutamine (ainult Windows). Tegemist on masinakeelse programmiga, mille täitmiseks ei ole vaja Scratchi ega muid vahendeid. Scratch ise praegu exe-faile ei tee, kuid selleks saab kasutada BYOBi. Viimane võimaldab avada ja täita Scratchi faile ning kompileerida need exe-failideks. Näide: hunt_kits_kapsas.exe.

Lähitulevikus peaks saama võimalikuks Scratchi rakenduste loomine ja täitmine otse brauseris.

Näide: Tutvus

Rakendus demonstreerib mitmeid Scratchi võimalusi ja vahendeid ning programmeerimise üldiseid põhimõtteid: graafika- ja heliobjektide kasutamine, märkandmete väljastamine ja sisestamine, joonistamine, korduste ja valikute kirjeldamine, muutujate, loendite ja juhuslike arvude kasutamine, lihtsad tekst- ja arvavaldised, tegevuste jagamine ning koostöö korraldamine skriptide vahel.

Demo: tutvus.sb, tutvus.html, tutvus.java, tutvus.exe

Ülevaade rakendusest

Stsenaarium: *Laval on neli tegelast: Kraps, Mari, Juku, robot Robi ja pall. Programmi käivitamisel Kraps teretab, esitleb end, teeb salto ja küsib kasutaja nime. Kasutaja võib sisestada nime või loobuda sellest. Viimasel juhul muudab Kraps pahameelest oma värvi, väljendab kahetsust, kustutab eelmised andmed, peidab ennast ära ning lõpetab programmi töö. Kui aga kasutaja sisestab nime (põhimõtteliselt suvaline tekst: vähemalt üks märk), kiidab Kraps kasutajat ning tegelased esitlevad end, tehes seejuures mõned väikesed trikid: hüpped, saltod, tantsud jm. Seejärel pakub Juku kasutajale teenust: saada teada oma „saledus“. Kui kasutaja soovib seda, küsib Juku tema käest pikkuse ja kaalu (massi), leiab kehamassiindeksi (pikkus/kaal^2) ja selle alusel „saleduse“ ning laseb abilisel joonistada andmete ümber raami. Kui kasutaja ei soovi teada saledust, eemaldab abi muutuja monitorid ja raami. Enne töö lõppu jätavad tegelased hüvasti, kuvades vastavad teated ja tehes veel mõned trikid. Vahepeal poisid veidi togivad ja lennutavad palli.*

Lava kujundus (taust) ei oma erilist tähtsust, see võib ka puududa. Praegu on laval mitu tausta, millest on nähtav üks. Hiireklõpsuga laval, saab neid vahetada. Laval on ka muutujate monitorid, milles kuvatakse kasutaja **nimi**, **pikkus**, **kaal**, **indeks** ja **saledus**. Laval on oma skriptid, üks neist töötab pidevalt paralleelselt teiste skriptidega ja muudab lava tausta värve.

Rakenduses on neli põhispraiti: **Kraps**, **Mari**, **Juku** ja **Robi** ning üks abisprait, mis joonistab ja haldab muutujate monitore (kuvab ja peidab). Konkreetsed spraitide kujud ja nimed ei oma erilist tähtsust. **Kraps** on oma kahe kostüümiga kohe laval olemas. **Mari** ja selle kostüümid on imporditud spraitide hoidlast. Samuti on hoidlast imporditud **Robi**, millele on kopeerimise ja redigeerimisega lisatud üks kostüüm. **Juku** on imporditud gif-failist. Kui aga lugeja realiseerib ise antud rakenduse, võib ta valida mõned teised spraidid.

Rakenduses on üsna mitu skripti. Pildil on toodud üks Krapsu skriptidest, mida võib nimetada ka **peaskriptiks**. Sellest algab programmi töö ja siit käivitatakse teised skriptid. Peaskript alustab tööd, kui kõpsatakse rohelist lippu. Samaaegselt käivitub ka lava skript, mis muudab selle tausta värve ja töötab kogu aeg paralleelset teiste skriptidega.

Edasi vaatame programmi tööd, järgides peaskripti. Skripti käske täidetakse järjest. Alguses on mitu lihtkäsku. Põhiosa skriptist moodustab liitkäsk [**kui** *tingimus…*], mis sisaldab mitut lihtkäsku. Sõltuvalt sellest, kas kasutaja sisestas oma nime või mitte, täidetakse ainult üks grupp käskudest. Skriptis on mitu käsku [**teavita** t*eade*…], mis käivitavad ühe või mitu alamskripti ja peatavad vahepeal peaskripti täitmise.

Siin ja edaspidi: viitamisel käskudele tekstis, paigutame need märkide **[ ]** vahele. Mõnikord on näidatud käsu kõik elemendid, mõnikord ainult käsu algus või nimi.

Sissejuhatavad tegevused

Kõigepealt teeb käsk [**näita**] Krapsu nähtavaks (kui ta oli peidus). See on vajalik programmi käivitamisel olukorras, kui eelmisel käivitamisel kasutaja loobus nime sisestamisest ja Kraps peitus. Käsk [**mängi heli** *näu*] käivitab heliklipi **näu** (mis on Krapsul kohe olemas). Vajadusel saab heliklippe importida skriptide piirkonna vahelehelt **Helid**. Käsk [**ütle** …] väljastab Krapsu tervituse, kuvades antud teksti kõneballoonis Krapsu juures.

Käsk [**teavita** *hei_hop* ja oota] saadab süsteemi teate, mille alusel käivitatakse skriptid, kus alguses on plokk [**kui saabub teade** *hei_hop*]. Praegu on üks selline skript Krapsul, mis paneb ta tegema saltot. Peale alamskripti täitmist peaskripti töö jätkub.

Andmete sisestamise käsk [**küsi** *teade* ja oota] peatab skripti täitmise, kuvab teate kõneballoonis spraidi juures ning tekstivälja väärtuse sisestamiseks lava alumises servas. Kui kasutaja tipib vastuse ning vajutab klahvi **Enter**, võetakse antud väärtus sisemuutuja ***vastus*** väärtuseks ja skripti töö jätkub. Kui kasutaja vajutab kohe klahvile Enter on tegemist nn tühja väärtusega. Käsuga [**võta** *nimi* = *vastus*] omistatakse muutujas *vastus* olev väärtus muutujale *nimi*.

Edasi tuleb liitkäsk [**kui** *tingimus* (käsud1) **muidu** (käsud2)], mille abil saab kirjeldada **valikuid** ehk hargnevaid protsesse. Praegu on tegemist nn **kahendvalikuga**: kahest alternatiivsest võimalusest valitakse üks, sõltuvalt tingimuse väärtusest. Tingimus on esitatud võrdluse abil. Kui tingimus on **tõene**, täidetakse käsud1, käsud2 jäetakse vahele. Vastupidisel juhul täidetakse käsud2, käsud1 jäetakse vahele. Praegu tehakse võrdluse abil kindlaks, kas nime väärtus on tühi või mitte.

Kasutaja loobus jätkamisest

Kui tingimus on tõene (nime ei ole), täidetakse **kui**-harus olevad käsud. Käsk [**teavita** puhasta] käivitab spraidi **abi** vastava skripti, mis peidab muutujate **pikkus**, **kaal**, **indeks** ja **saledus** monitorid ning kustutab raami. Käsk [**pane** *värv* efekt 100-le] muudab spraidi värvi, kasutades värvi koodi (väärtused on vahemikus 0 kuni 200). Praegu kasutatakse sinist värvi. Käsk [**ütle** Kahju!…] kuvab teate ja järgnev käsk [**peida**] peidab Krapsu ära. Kuna **muidu**-haru käsud jäetakse vahele, siis järgnevalt täidetakse käsk [**peata kõik**], mis lõpetab kogu rakenduse töö.

Kasutaja jätkab

Käsu [**kui**…] teise haru esimene käsk [**ütle**…] on juba tuttav. Võrreldes eelmiste seda tüüpi käskudega, on siin kasutusel **tekstavaldis**, mis on moodustatud plokiga

, kus ühendatakse tekstkonstant ’Tubli, ’ ja muutuja **nimi** väärtus. Tekstide ühendamiseks tekstipõhistes süsteemides kasutatakse märki **+** või **&**. Näiteks Pythonis ja Visal Basic’us pannakse taoline avaldis kirja nii: “Tubli, “ + nimi.

Käsk [**teavita** alustame] ja selle alusel käivitatud tegevused

Järgmine käsutüüp [**teavita** *alustame* ja oota] esines juba ka eespool, kuid praegu on tegemist üldisema juhuga. Kui käsuga **teavita** käivitus üks Krapsu skript, siis nüüd käivitub korraga neli skripti, sest päisega [**kui saabub teade** *alustame*] algav skript on praegu igal spraidil. Skriptid töötavad paralleelselt. Kuna käsus **teavita** on täiend **oota**, siis peatakse peaskripti täitmine seni, kuni lõppeb kõikide alamskriptide täitmine, ning peale seda jätkub antud skripti töö.

Vaatleme nüüd nendes protsessides osalevate spraitide vastavaid skripte ja tegevusi. Nad on üles ehitatud suhteliselt sarnaselt.

Alustame. Juku ja Robi skriptid

Juku skript **alustame** tekitab käsuga [**oota** 2 sek] kõigepealt väikese pausi, et alustada tegevusi teistest veidi hiljem. Seejärel Juku esitleb end ja saadab teate **hüppa**; see on mõeldud tema enda skriptile, mis paneb Juku hüppama. Selles on plokiga [**korda** 3] määratletud etteantud korduste arvuga **kordus**. Ploki sees olevaid käske täidetakse kolm korda. Iga kord suurendatakse Juku y-koordinaati 40 pikseli võrra, tehakse väikene paus, vähendatakse koordinaati 40 pikseli võrra ja tehakse jälle paus. Tulemusena tõuseb Juku kolm korda ülespoole ja langeb alla, imiteerides hüppamist. Käsk [**teavita** löök ja oota] käivitab palli skripti, mis viib selle Juku juurest Krapsu juurde ja sealt tagasi. Vahepeal lendab pall ka üles.

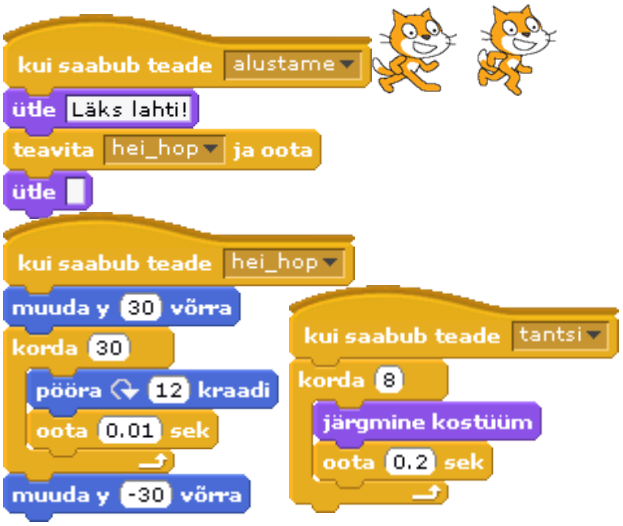
Juku

Robi

Robi skriptid on analoogsed. Käsuga [**mängi heli** robi] käivitatakse lindistatud heliklipp. Lindistada saab vahelehel **Helid** korraldusega **Lindista**, kui arvutil on mikrofon. Kui viimane puudub, võib võtta süsteemi heliklippide hoidlast suvalise klipi. Midagi ei juhtu, kui klippi üldse ei ole: heli lihtsalt ei mängita. Skript **liigu** paneb **Robi** kostüüme vahetama (visuaalsed efektid) ning horisontaalselt liikuma, suurendades ja vähendades x-koordinaati. Ka siin on plokk **korda**, mis tagab sisemiste käskude kordamise kaks korda. Robi teine kostüüm on tehtud esimesest Scratchi joonistamisredaktori abil.

Alustame. Krapsu ja Mari skriptid

Kraps



Mari

Muusika ja tantsulembelise **Mari** skriptides demonstreeritakse heliandmete kasutamist ja tantsu imiteerimist kostüümide vahetamise teel. Skriptis **alustamine** mängitakse lihtsalt üks noot. Scratchis on

olemas lihtne muusikahelide (nootide) süntesaator. Käsus [**mängi nooti** *number* kestvusega *aeg*] saab näidata noodi numbri (48 kuni 72) ja kestvuse sekundites. Kes tunneb noote, võib proovida komponeerida. Plokiga [**mängi trummi** …] saab imiteerida ka löökpillide helisid.

Skriptis **tantsi** võetakse Mari jaoks esimene kostüüm (ballerina-a, igal kostüümil on nimi). Seda tehakse nö igaks juhuks, arvestades, et programmi katkestamisel võib Mari jääda suvalisse olekusse. Käsuga [**mängi heli** *Triumph*] käivitatakse heliklipp ning skripti töö jätkub nö muusika saatel. On olemas ka käsk [**mängi heli** *nimi* kuni valmis]. Selle käsu korral skripti töö peatatakse, mängitakse heliklipp lõpuni ja siis jätkatakse tööd.

Tantsu simuleerimiseks kasutatakse siin plokki [**korda** 8]. Igal kordamisel vahetatakse kostüümi ja tehakse väike paus. Kuna kordamisi on praegu 8, käiakse kostüümid kaks korda läbi: kui jõutakse viimase kostüümini, alustatakse uuesti esimesest.

Skript **tantsi**, ilma kostüümi valimiseta ja heliklippi käivitamiseta, on ka **Krapsul**. Kui Mari skript **alustame** saadab teate **tantsi**, hakkab tööle ka Krapsu samanimeline skript.

Krapsu skript **hei_hop** paneb teda tegema saltot. Kraps kerkib üles 30 pikselit, teeb ühe tiiru ja laskub tagasi alla. Paneme tähele, et pööre 360 kraadi ei ole määratud ühe käsuga: [**pööra** 360 kraadi], vaid 30 korda täidetava käsuga [**pööra** 12 kraadi]: 30 * 12 = 360. Kui kasutada esimest varianti, siis pööret näha ei ole. Realiseeritud variandi korral, kus igale pöördenurga muutusele (12 kraadi) järgneb ka väikene paus, jaguneb liikumine (pööre) nö kaadriteks. Taoline põhimõtte on üldiselt kasutatav animatsioonides. Liikumise kiirust ja sujuvust saab reguleerida vaadeldaval juhul nurga muutusega ja pausi pikkusega. Liikumine on visuaalselt jälgitav. Paneme tähele, et pause (kaadreid) kasutatakse ka teistes animatsioonidega seotud skriptides: **hüppa** (Juku), **tantsi** (Mari ja Kraps), **liigu** (Robi).

Skriptides **hüppa**, **liigu**, **tantsi** ja **hei_hop** kasutatakse etteantud korduste arvuga kordust, mis on üks lihtsamaid ja sagedamini kasutatav korduse tüüp kõikides programmeerimiskeeltes. Palli skriptides demonstreeritakse tingimuslike korduste kasutamist. Lava skriptis esineb ka lõputu kordus.

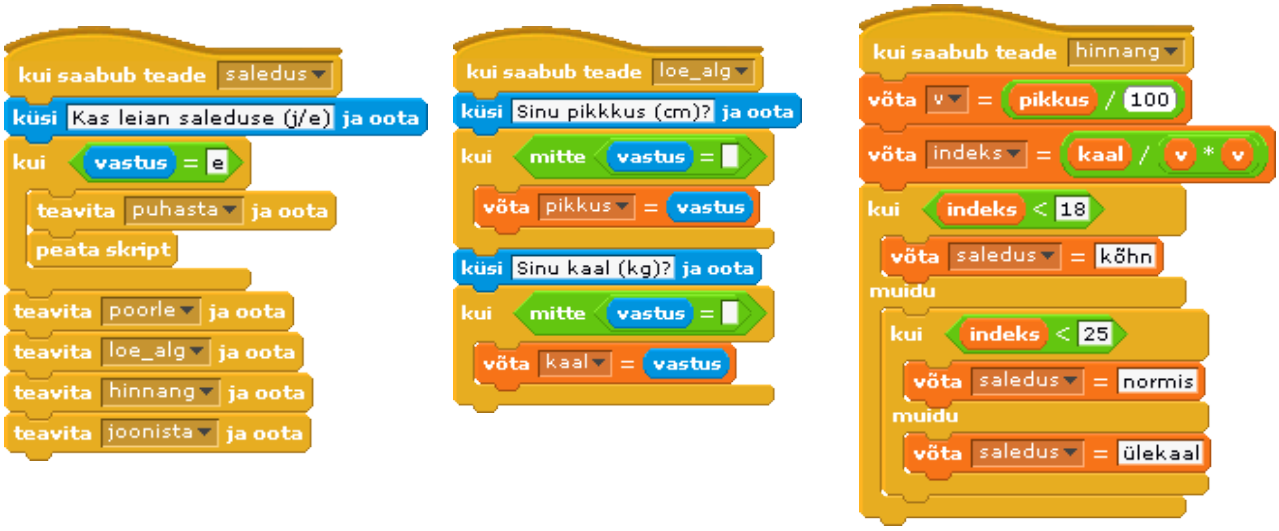
Palli skriptid

Kaheks peamiseks palli skriptiks on **löök** ja **lenda**. Esimene viib palli pööreldes vasakule Krapsuni ja tagasi Juku juurde. Teine viib palli üles lava servani ja alla Juku juurde tagasi. Mõlemas skriptis kasutatakse kaks korda tingimuslikku kordust. Esimeses skriptis on tingimus määratletud nö puutesündmustega: puudutab Kraps ja puudutab Juku. Teises on tingimus esitatud lava serva puutega (üles liikudes) ja jõudmisega antud kõrguseni (alla liikudes). Pallil on ka mõned sündmusskriptid: [kui vajutatakse klahvi vasak nool], [kui vajutatakse klahvi ülesnool] ja [kui klõpsatakse pall]. Neist on näidatud ainult esimene.

Saleduse leidmine

Käsk [**teavita *saledus*** ja oota] käivitab rakenduse teise osa, kus põhitegelaseks on Juku. Skript **saledus** küsitakse, kas kasutaja soovib teada oma saledust. Kui kasutaja sisestab tähe **e**, peidetakse muutujate **pikkus**, **kaal**, **indeks** ja **saledus** monitorid ja skripti töö peatakse. See tähendab, et täitmisjärg läheb tagasi peaskriptile. Kui aga sisestatakse midagi muud, käivitakse järjest käsud **teavita**.

Juku skript **poorle** paneb Juku tegema saltot. Põhimõtelist on tegemist sama asjaga, nagu Krapsu skriptis **hei_hop**, ning skripti pole siin näidatud.



Saleduse leidmisel on kasutusel neli põhimuutujat: **pikkus**, **kaal**, **indeks** ja **saledus**. Esimese kahe väärtused sisestab kasutaja, teised leiab skript **hinnang**.

Algandmete sisestamist korraldab skript **loe_alg**. Kuid muutujate (pikkus ja kaal) väärtusi saab muuta ka liugurite abil. Skriptis **loe_alg** on ette nähtud võimalus, et olemasolevaid muutujate väärtusi ei muudeta. Kui käsu **küsi** täitmisel väärtust ei sisestata, vaid vajutatakse kohe klahvile Enter, jääb muutuja väärtus endiseks. Väärtust muudetakse ainult siis, kui ei olnud tegemist tühja väärtusega.

Saleduse leidmisel leitakse kõigepealt kehamassi indeks (kaal/pikkus^2) ja omistatakse see muutujale **indeks**. Kuna pikkus sisestatakse sentimeetrites, kehamassi indeksi leidmisel aga peab see olema meetrites, jagatakse muutuja **pikkus** väärtus 100-ga. Skriptis kasutatakse abimuutujat **v**, millele omistatakse pikkuse väärtus meetrites, et järgnev käsk oleks lühem.

Kehamassiindeks k_{ind} leiab sageli kasutamist inimese saleduse või täidluse hindamisel. Tavaliselt kasutake taolist skaalat:

$$k_{ind} < 18 - \text{kõhn}; 18 = k_{ind} < 25 - \text{normaalne}; \; 25 = k_{ind} \le 30 - \text{ülekaal}; \;\; k_{ind} > 30 - \text{paks}$$

Siin kasutavas skriptis ei arvestata viimast kriteeriumi (k_{ind} > 30 – paks). Lugeja võiks proovida lisada see.

Muutujate monitorid on sageli olulisteks kasutajaliidese elementideks. Antud rakenduses on ette nähtud monitoride peitmine ja kuvamine vastavalt olukorrale. Seda teevad spraidi **abi** skriptd: **puhasta** ja **loe_alg**, vastavate teadete saamisel.

Skript **puhasta** kustutab ka raami, mis on joonistatud monitoride ümber.

Joonistamine

Joonistada võib suvaline sprait. Iga spraidiga on alati seotud **pliiats**, mis võib olla üleval – spraidi liikumisel joont ei teki, või all – liikumisel tekib laval joon. Joonistamisel kasutatakse peamiselt grupi **Pliiats** ja **Liikumine** käske.

Pliiatsi käskudega saab määrata joone värvi ja jämeduse ning muuta pliiatsi olekut: üleval või all. Praegu valitakse joone värv ja paksus juhuslikult.

Liikumise käskudega määratakse joonte asukoht laval. Käsk [**mine** x: … y: …] viib spraidi momentaanselt antud punkti. Käsuga [**liigu** t sek. x: … y: …] saab määrata spraidi sujuva liikumise. Ühikuks on piksel.

Hüvastijätt

Selles osalevad kõik tegelased ja tegemist on paralleelsete protsessidega. Siin demonstreeritakse ka lihtsamaid võimalusi loendite kasutamiseks.

Loendis nimega **T** on tekstid, millest valitakse iga spraidi jaoks juhuslik väärtus kuvamiseks käsus **ütle**. Kõikide spraitide skriptides päisega [**kui saabub teade** *hüvasti*] on käsk

Loendi **T** elemendi indeksiks võetakse juhuslik arv vahemikust 1 kuni **m**, kus **m** on elementide arv loendis. See leitakse spraidi **abi** skriptis.

Lava skriptid

Laval on kaks skripti. Üks läheb käima koos Krapsu põhiskriptiga, kui klõpsatakse rohelist lippu, ja muudab lava värvi.

Formaalselt on tegemist lõputu kordusega. Praktiliselt aga lõpetab skripti töö käsk [**peata kõik**] Krapsu põhiskriptis. Alati saab kogu rakenduse töö katkestada ka lava kohal asuva punase nupuga.

Värvi muudetakse sagedusega üks sekund. Värvi kood valitakse juhuslikult.

Teine skript, mis käivitub hiireklõpsuga laval, koosneb ühest käsust:

.

Rakenduste disainist ja dokumenteerimisest

Suuremate rakenduste korral on tavaliselt otstarbekas enne skriptide koostamist täpsustada ülesande püstitust ning viia läbi rakenduse analüüs ja disain ning koostada projekti dokumentatsioon: määratleda andmed, fikseerida spraitide funktsioonid, seosed nende vahel ja tegevused, koostada algoritmid, kavandada kasutajaliides. Rakenduse realiseerimise käigus dokumentatsioon tavaliselt täiustub ning lõpuks vormistatakse selle lõplik variant. Seda on vaja rakenduse kasutamisel ja arendamisel.

Lisamaterjal: Rakendused.

Andmed

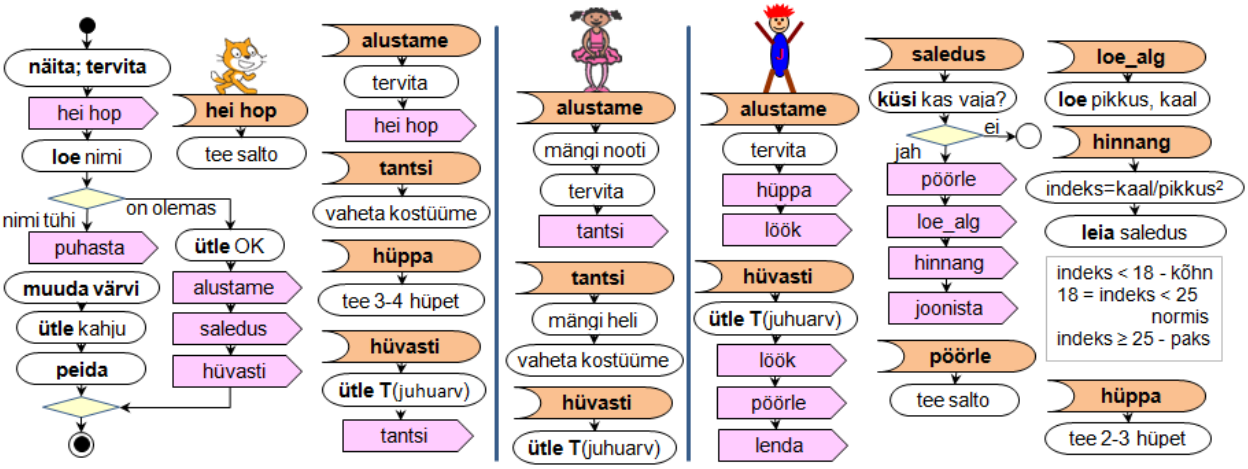
Objektid (spraidid)

- Kraps – juhib ja korraldab rakenduse tööd ja teeb igasuguseid trikke. Kaks kostüümi, seitse skripti
- Mari – kultuuritöötaja: tantsib ja teeb muusikat. Neli kostüümi, kolm skripti
- Juku – sportlane ja matemaatik, arvutab ja hindab saledusi. Üks kostüüm, seitse skripti
- Robi – robot, tutvub inimestega, õpib arvutama faktoriaale. Kaks kostüümi, kuus skripti
- pall – teda togivad ja pilluvad Juku ja Kraps. Üks kostüüm, viis skripti
- abi (peidetud) – joonistab ja teeb muid abitöid. Üks kostüüm, neli skripti

Põhimuutujad: nimi, pikkus, indeks, saledus

Rakenduse struktuur

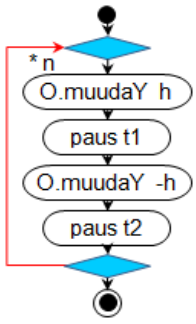
On näidatud üldvaade kolme spraidi skriptidest. Koostööd peegeldavad teavitamise korraldused ja vastuvõtvate skriptide päised. Rakendused.



Algoritmid

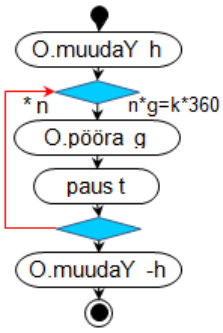
Näiteks on toodud mõned algoritmid UMLi tegevusdiagrammidena ja pseudokoodis. Algoritmimine.

hüppa



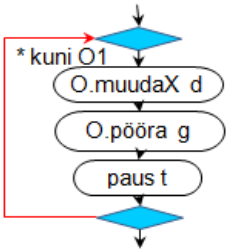

```
prots hüppa
kordus n korda
    objekt.muudaY h
    paus t1
    objekt.muudaY -h
    paus t2
```

salto



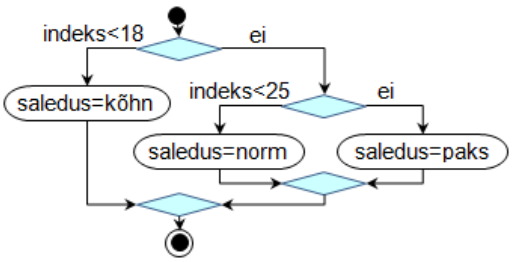

```
prots salto
objekt.muudaY h
kordus n korda
    objekt.pööra g
    paus t
lõpp kordus
objekt.muudaY -h
```

löök




```
prots löök
kordus kuni objekt_1
    objekt.muudaY h
    objekt.pööra g
    paus t
```

hinnang




```
prots hinnang
kui indeks<18 siis
   saledus = kõhn
muidu
   kui indeks<25 siis
      saledus = norm
   muidu
      saledus = paks
```