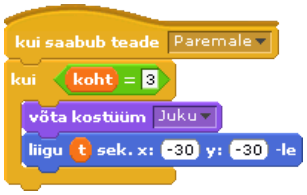
Eesti Teadusagentuur
Estonian Research Council
TeaMe
TTÜ
informaatikainstituut

Rakenduste loomine Scratchiga

Scratch on uue põlvkonna graafiline programmeerimissüsteem, mis võimaldab lihtsalt ja kiiresti omandada algoritmimise ja programmeerimise põhimõtteid ning tüüpilisi vahendeid ja tegevusi, mida kasutatakse ka tavapärastes tekstipõhistes programmeerimiskeeltes.

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



Sisu

Sissejuhatus programmeerimissüsteemi Scratch

Scratch on uue põlvkonna graafiline programmeerimissüsteem, mis on loodud Massachusetts'i Tehnoloogiainstituudis (*Massachusetts Institute of Technology – MIT*) programmeerimise õppimiseks ja õpetamiseks. Scratchi versioon 1.4 on ilmunud 2009. aastal, millest alates algas süsteemi kiire levik.

Scratchi saab tasuta alla laadida aadressilt http://scratch.mit.edu. Seal leidub suurel hulgal õppe- ja abimaterjale tõlgituna erinevatesse keeltesse (sh eesti keelde) ning näiteid. Scratchi serverid pakuvad ka pilveteenuseid – paari hiireklõpsuga saab oma rakenduse (projekti) tõsta „pilve" 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. Uues versioonis on olemas võimalus Scratchi rakenduste loomiseks ja täitmiseks brauseris. Lisaks toimuvad arendustööd selliste vahendite loomiseks, mis võimaldavad kasutada Scratchi rakendusi ka nutiseadmetel nagu tahvelarvutid ja mobiiltelefonid.

Scratchis on lihtne ja mugav rakenduste loomise keskkond ja kasutajaliides. Süsteemiga on kaasas rikkalik valik graafika- ja heliobjekte, kuid neid saab lisada ka oma failidest ja internetist. Pilte saab teha veebikaameraga ja heliklippe salvestada rakenduste loomise käigus, kui arvutil on mikrofon. Oma graafikaobjektide loomiseks või olemasolevate muutmiseks on võimalik kasutada keskkonda sisseehitatud joonistamisredaktorit. Scratchis on pööratud suurt tähelepanu atraktiivsusele ning multimeedia kasutamisele – lihtsalt ja kiirelt on võimalik luua mänge, animatsioone, koomikseid, esitlusi jms. Samas on võimalik lahendada ka arvutuslikke, andmetöötluse- ja graafikaülesandeid: nt realiseerida, uurida ja visualiseerida klassikalisi algoritme massiividega (summade, keskmiste, ekstreemumite leidmine, erinevad otsimise ja sorteerimise meetodid demo), teha joonistusi ja diagramme sealhulgas funktsioonide graafikuid jms.

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 masinkoodi, 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 graafilisi programmeerimissüsteeme, mis on Scratchiga sarnased.

Kõigepealt võiks nimetada suurfirma Google poolt arendatavat süsteemi **Blockly** (http://code.google.com/p/blockly). Ka siin kasutatakse Scratchi käsuplokkidele sarnaseid elemente ning skriptid pannakse hiire abil kokku otse brauseris. Süsteemi üheks huvitavaks eriomaduseks on võimalus protseduure tõlkida erinevatesse tekstipõhistesse programmeerimiskeeltesse nagu Python, JavaScript, Dart ja XML. Üsna sarnane Blocklyga on ka **MIT App Inventor** (http://appinventor.mit.edu), mis on ette nähtud Androidi rakenduste loomiseks mobiilidele. Veel üks omapärane eksperimentaalne süsteem on **Calico** (vt http://calicoproject.org), kus ühes ja samas kasutajaliideses saab kasutada erinevaid programmeerimissüsteeme: graafilist Scratchiga sarnanevat süsteemi, keelt **Jigsaw**, Pythonit, Rubyt, robotite programmeerimise vahendit Myro, …

Scratch ja BYOB leiavad järjest laiemad kasutamisest programmeerimise õpetamise esimeste keeltena. Need sobivad igas vanuses õppuritele, alates põhikoolide algklassidest kuni kolledžite ja ülikoolide programmeerimise sissejuhatavate kursusteni. Käsitletavate teemade ja lahendatavate ülesannete valikul tuleb muidugi arvestada õppijate vanust, taset, õpetamise eesmärke jm. Peab ütlema, et valikuvõimalusi ja variatsioone on siin tunduvalt rohkem kui traditsiooniliste programmeerimiskeelte kasutamisel.

USA arvutiõpetajate assotsiatsiooni (CSTA) poolt koostatud arvutiteaduse tüüpõppekavas (ülevaade, kirjeldused) põhineb aine „Sissejuhatus programmeerimisse“ täielikult Scratchil. Scratchi 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).

Esitatud kursuse antud osa eesmärgiks ei ole programmeerimiskeele Scratch õppimine. Scratch on siin lihtsalt vahendiks rakenduste loomise ning programmeerimise ja algoritmimise aluste omandamisel. Scratchi abil on võimalik omandada kiirelt kõik peamised programmeerimise, algoritmimise, modelleerimise ja rakenduste loomise põhimõtted ja tüüptegevused: eri liiki 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 ei toeta antud momendil kõiki objektorienteeritud (OO) programmeerimise võimalusi (tegemist on nn objektipõhise süsteemiga), võimaldab ta tutvustada ja osaliselt rakendada OO lähenemisviisi.

Scratchi materjalide esitatamine kahel tasemel

Võimaldamaks õppijal saada kiiresti ülevaade Scratchist, on esimesel tasemel toodud lühivaade Scratchi kasutamisest rakenduste loomiseks. Sellega on soovitatav 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 praktikumide töölehtedele.

Modelleerimine	Rakenduste loomine	Algoritmimine
Scratchi põhiobjektid	Skriptide loomine	Protsesside juhtimine
Muutujad	Rakenduste loomine Scratchiga	Andmed ja avaldised
Joonistamine	Loendid	Algoritmid loenditega

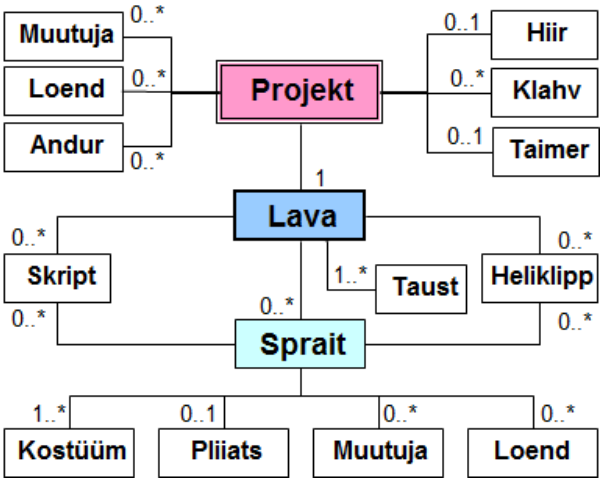
Ülevaade Scratchi rakenduste komponentidest ja vahenditest

Selles jaotises on lühiülevaade Scratchi põhikontseptsioonidest, komponentidest ja vahenditest. Vajadusel saab, kasutades tekstis toodud linke, põhjalikumat infot lisamaterjalidest.

Scratchi rakendusi nimetatakse **projektideks**. Projekti põhikomponentideks on spraidid ja skriptid, mille abil saab luua väga erineva otstarbe, sisu ja struktuuriga projekte (vt näiteid Scratch wiki ja Scratchi koduleht).

Scratchi projekti struktuur ja põhiobjektid

Scratchi projekt koosneb omavahel seotud objektidest. Kasutatavad **objektid** ja **seosed** nende vahel on esitatud 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..*). Selline kordus on mitmel teisel objektil: skript, heliklipp, muutuja jms.

Laval on alati vähemalt üks taust ja spraidil vähemalt üks kostüüm (1..*). Kordus 0..1 näitab, et kasutusel võib olla ainult üks objekt või mitte ühtegi (hiir, taimer, pliiats).

Projekti loomisel ja täitmisel saab kasutada ja muuta objektide **omadusi** (spraidi koordinaadid, suurus, nähtavus, muutuja väärtus jms). Scratchi käsuplokid on objektide **meetodid**, mille abil saab määrata tegevusi spraitide ja nende alamobjektidega (pliiats, heli).

Lisamaterjal: Scratchi põhiobjektid, Scratch wiki.

Spraidid

Scratchi rakenduste kesksteks komponentideks on **objektid**, mida nimetakse **spraitideks** (inglise k ***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 on süstematiseeritud liikide kaupa (inimesed, loomad, asjad, …) erinevatesse kaustadesse. Iga spraidiga on seotud teatud valik **omadusi** nagu nimi, asukoht laval (x, y), suurus, värvus, … ning **meetodeid**, mis on realiseeritud käsuplokkidena. Spraitide loomisel saab kasutada ka oma graafikaobjekte PNG, GIF, JPG või BMP ning animeeritud GIF-faile. Spraitide kostüüme võib luua ja muuta **joonistamisredaktori** abil (vt Kasutamisjuhend ja Scratch wiki).

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

Spraidil võib olla mitu **kostüümi** ehk **kuvandit** (*costume*) – tüüpiliselt spraidi erinevad variandid e teisikud. Praktiliselt kujutab sprait endast graafiliste kujundite (kostüümide) kogumit. Kui sprait on nähtav, siis on nähtav üks tema kostüümidest e kuvanditest. Vahetades kostüüme programmi käskude abil teatud

sagedusega, saab tekitada animatsiooniefekte nagu kõndimine, tantsimine või muud. Kostüüme võib kasutada ka muul viisil, näiteks erinevate teadete kuvamiseks.

Lisamaterjal: Objektid Scratchis, Kasutamisjuhend, Scratch wiki.

Skriptid ja käsud

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

Plokid ühendatakse omavahel hiire abil tegevuste toimumise järjekorras. Skript algab **päiseplokiga**, mis on seotud skripti käivitamisviisiga: rohelise lipu klõpsamine, vajutus klahvile klaviatuuril, teate saabumine jms.

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 programmeerimises väga tähtsaks põhimõtteks. See võimaldab luua paindlikke ja ökonoomseid rakendusi, kasutada üksusi korduvalt nii samas kui ka teistes rakendustes, täita tegevusi paralleelselt, …

Lisamaterjal: Skriptid Scratchis, Kasutamisjuhend, Scratch wiki.

Scratchis on koostöö skriptide vahel rajatud **teate saatmisele** ühe skripti poolt ja selle **vastuvõtmisele** teiste skriptide poolt. Käsuga [**teavita *nimi***] või [**teavita *nimi* ja oota**] väljastatakse teade (signaal), mille võib vastu võtta ja käivituda suvaline arv skripte päisega [**kui saabub teade *nimi***], kus ***nimi*** on sama mis oli käsus [**teavita** …]. Tegemist võib olla sama spraidi skriptidega või teise spraidi või lava skriptidega. Teate vastuvõtnud skriptid alustavad tööd paralleelselt.

Näiteks võib mitmel spraidil olla pildil esitatud skript, mis kõik hakkavad 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 oma tööd kohe pärast teate saatmist ja töötab paralleelselt teatele reageerinud skriptidega.

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**, **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**. Suuruse valik muudab liidese erinevate alade jaotust. Esitluse olekus on täisekraanil nähtav ainult lava ning käivitamise ja peatamise nupud. Selles olekus ei saa lisada uusi spraite ja skripte.

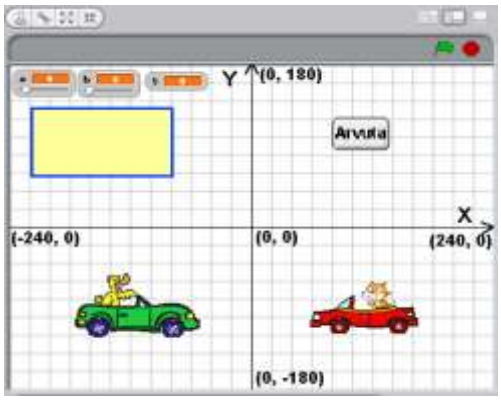
Rakenduse kasutamise liides luuakse laval. Liidese elementideks võivad olla taustad, aktiivsed ja passiivsed spraidid, käsunupud, klahvid, muutujate monitorid, …

Teistes süsteemides kasutatakse liidesena tavaliselt vorme ja dialoogibokse.

Lava

Lava (*Stage*) on ekraani piirkond, kus tegutsevad kõik spraidid.

Lisamaterjal: Objektid Scratchis, Kasutamisjuhend, Scratch wiki.



Laval võib olla mitu **tausta**. Vaikimisi on selleks valge taust. Taustu saab lisada, eemaldada, muuta. Taust on graafikaobjekt (pilt) ning oma olemuselt analoogne spraidi kostüümiga.

Koordinaattelgede algus on lava keskel. Lava on laiuseks on 480 pikselit (x_min = -240, x_max = 240) ja kõrguseks 360 pikselit (y_min = -180, y_max = 180).

Tegemist on tingliku ühikuga, mille suurus sõltub ekraani suurusest ja kasutatavast eraldusvõimest.

Andmed

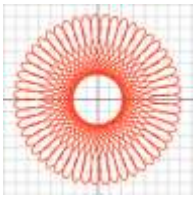
Scratchi rakendustes saab kasutada **graafika**-, **heli**- ja **märkandmeid**. Lisamaterjal: Andmed ja avaldised.

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 (pliiats üleval või all) jms. Grupi **Liikumine** käskude abil muudetakse joonistava spraidi asukohta.

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

Lisamaterjal: Joonistamine, Scratch wiki.

Heliandmed

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

Heliobjektid on heliklipid, mis imporditakse või salvestatakse 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, mis on iseloomulikud teistele programmeerimiskeeltele. Tegemist on tavaliste täis- ja reaalarvudega, kusjuures formaalset erinevust nende vahel ei ole. Murdosa eraldamiseks täisosast tuleb reaalarvudes kasutada punkti. Sisendväljas, kus peaks olema arv (näiteks [liigu ... sammu], [oota ... sek]) võib reaalarvu sisestamisel kasutada ka koma, mis muudetakse automaatselt punktiks. Arve võib esitada käsuplokkides ja avaldistes konstantidena ja sisestada klaviatuurilt. Arvude jaoks on neli põhitehet: liitmine, lahutamine, korrutamine, jagamine (**+** , **-** , ***** , /), võrdlustehted, mis esitatakse vastavate plokkide abil, ning 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, ... Tekst sisestatakse klaviatuurilt konstandina otse plokki, omistatakse muutujale või loendi elemendile. Tekstide jaoks on olemas ka mõned tehted: ühendamine, märkide eraldamine, võrdlemine.

Tõeväärtuste ehk loogikasuuruste jaoks on ainult kaks väärtust ehk loogikakonstanti: tõene (*true*), väär (*false*). Ilmutatud kujul neid Scratchis praktiliselt ei kasutata. Loogikaväärtused tekivad võrdlustes ja loogikaavaldistes, mida kasutatakse tingimuste esitamiseks.

Konstandid ja muutujad

Konstandi väärtus kirjutatakse otse käsuplokki. Programmi täitmisel ei saa konstandi väärtust muuta. Arvus võivad olla ainult numbrid, miinusmärk ja murdosa eraldaja. 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 on muutujal matemaatikas. Programmeerimises mõistetakse muutuja all **nimega varustatud kohta arvuti mälus** (**mälupesa** või mäluvälja), 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**. Muutuja loomisel kuvatakse dialoogiboks, milles saab näidata muutuja nime ja skoobi. Skoop määrab, kas muutuja on kättesaadav kõikidele spraitidele ja lavale või ainult ühele spraidile. Iga muutuja jaoks luuakse automaatselt monitor ehk näidik, mis kuvab laval muutuja jooksva väärtuse arvu või tekstina. Monitori saab ka peita. Monitori näit võib olla tavaline, suur või liuguriga (kerimisribaga), mille abil saab muutuja arvväärtust muuta 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 järgmiselt:

muutuja = ***avaldis***

näiteks: vahe = pikkus – kaal, P = 2 * (a + b)

Operatsioonid märkandmetega. Avaldised

Avaldise abil kirjeldatakse 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

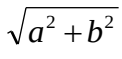
Arvude jaoks on neli põhitehet, mis esitatakse Scratchis vastavate plokkide abil:

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 Scratchis teatud valik **matemaatikafunktsioone**, mis esitatakse avaldistes taoliste plokkide abil nagu nt  .

Vasakpoolses väljas saab valida funktsiooni nime: abs, sqrt, sin, cos, tan, asin, acos, atan, ln, log, e^, 10^.

Argumendiks võib üldjuhul olla avaldis, erijuhul muutuja või konstant:

   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, st 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. Sõltumata kasutatavatest vahenditest võib ülesannete lahendamisel 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 need on esitatud. Käske ei jäeta vahele ega toimu ka kordamisi. Mingeid erilisi vahendeid protsessi määratlemisel ei ole vaja, sest täitmine toimub nö loomulikus järjekorras.

Toodud näide kujutab 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 pindala **S** ja ümbermõõdu **P** arvutamise järjekord.

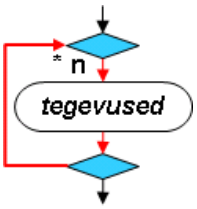
Tsüklilised protsessid

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

Lisamaterjal: Algoritmimine.

Etteantud korduste arvuga kordus

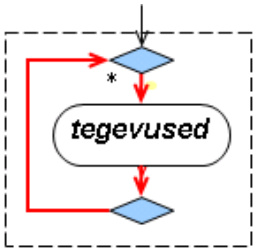
See esitatakse Scratchis plokiga [**korda** *n*], kus ***n*** on kordamiste arv. Viimane võib olla esitatud konstandi, muutuja või avaldise abil. Allpool on toodud selle kordusetüübi esitus ka UML diagrammide ja algoritmikeele (pseudokoodi) abil.



kordus *n* **korda**
tegevused
lõpp kordus

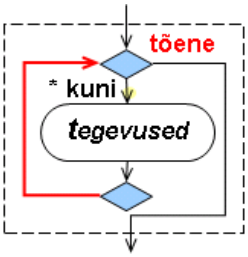
Lõputu kordus

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.

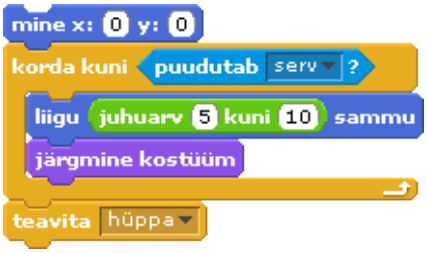


kordus
tegevused
lõpp kordus

Eelkontrolliga tingimuslik kordus



kordus kuni *tingimus*
tegevused
lõpp kordus



Selle korduseliigi 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*).

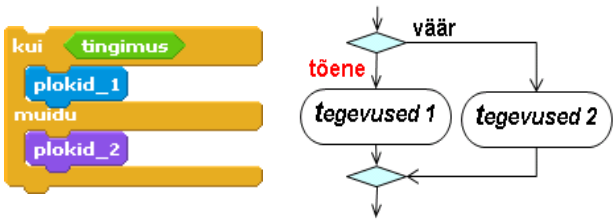
Hargnevad protsessid

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

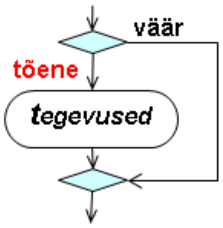
Vt ka Algoritmimine.

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:

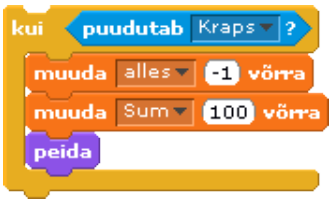
Loogigaavaldise väärtuseks saab olla ainult tõeväärtus: tõene (*true*) või väär (*false*).



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 [**kui klõpsatakse** ...] või [**kui vajutatakse klahvi** …]. Lisaks juba töötavatele skriptidele võib kasutaja käivitada (kui rakenduses on ette nähtud) täiendavaid skripte, mis alustavad tööd paralleelselt eelmistega.

Skriptidest saab paralleelseid protsesse käivitada käsuplokiga [**teavita** *nimi*] ja [**teavita** *nimi* **ja oota**]. Korraga 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: 

.

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

Selliseid andmekogumeid nimetatakse sageli ka ühemõõtmelisteks massiivideks.

Näites 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** n] liidetakse järjest loendi **hinnad** elemendid (**hinnad** järjenumbriga **k**) muutujale **Sum**, saades hindade summa. Selle jagamisel muutuja **n** väärtusega saadakse keskmine.

Lisamaterjalid: Loendid 1, Loendid 2, Scratch wiki.

Scratchi rakenduste kasutamisviisid ja projektide vormingud

Scratchi rakendusi saab kasutada (täita) erineval viisil.

Keskkond võib asuda kliendi oma arvutis või serveris. Scratchi projekti fail on laiendiga **sb**. Programmi täitmist korraldab **interpretaator**, mis transleerib (tõlgib) skriptid ühekaupa masinkeelde 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 aplet. Näide: hunt kits kapsas.html.

Projekti täitmine „pilves“ – Scratchi kodulehel *MIT* serveris. Scratchi projekti saab serverisse üles laadida menüükäsuga „jaga projekti internetis“. Käsuga „mine Scratchi kodulehele“ saab 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 MS Windows). Tegemist on masinkeelse 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 põhilisi tegevusi nagu graafika- ja heliobjektide kasutamine, märkandmete väljastamine ja sisestamine, joonistamine, korduste ja valikute kirjeldamine, muutujate, loendite ja juhuslike arvude kasutamine, lihtsate tekst- ja arvavaldiste kasutamine, tegevuste jagamine ning koostöö korraldamine skriptide vahel.

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

Ülevaade rakendusest

Stsenaarium*: Laval on tegelased: 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 abispraidil joonistada andmete ümber raami. Kui kasutaja ei soovi teada saledust, eemaldab abisprait muutujate 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 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 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 lava taustavärve ja töötab kogu aeg paralleelset teiste skriptidega.

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

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

Sissejuhatavad tegevused

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

Käsk [**teavita** *hei_hop* **ja oota**] saadab süsteemile teate, käivitamaks skripte, mille alguses on plokk [**kui saabub teade** *hei_hop*]. Praegu on Krapsul üks selline skript, mis paneb ta tegema saltot. Alamskripti käskude täitmise järel jätkub peaskripti töö.

Andmete sisestamise käsk [**küsi** *tekst* **ja oota**] peatab skripti täitmise, kuvab *teksti* spraidi juures olevas jutumullis ning kuvab lava alumises servas tekstivälja, kuhu sisestada väärtus. Kui kasutaja tipib selle 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* väärtus muutujale *nimi*.

Järgneb liitkäsk [**kui** *tingimus* (käsud1) **muidu** (käsud2)], mille abil saab kirjeldada **valikuid** ehk hargnevaid protsesse. Praegu on tegemist **kahendvalikuga** – kahest alternatiivsest võimalusest valitakse üks, olenevalt võrdluse abil esitatud tingimuse väärtusest. Kui tingimus on **tõene**, täidetakse käsud1 ning käsud2 jäetakse vahele. Vastupidisel juhul täidetakse käsud2 ning käsud1 jäetakse vahele. Antud juhul tehakse võrdluse abil kindlaks, kas muutujal *nimi* on väärtus või mitte.

Kasutaja loobus jätkamisest

Kui tingimus on tõene (muutuja *nimi* väärtus on tühi), 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 värvikoodi (väärtused on vahemikus 0 kuni 200). Praegu kasutatakse sinist värvi. Käsk [**ütle** Kahju!…] kuvab teate ja järgnev käsk [**peida**], mis 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 

ning kus ühendatakse tekstkonstant „Tubli, “ ja muutuja **nimi** väärtus.

Tekstide ühendamiseks tekstipõhistes süsteemides kasutatakse märki **+** või **&**. Näiteks Pythonis ja Visual Basicus pannakse selline avaldis kirja nii: „Tubli, “ + nimi, VBs ka „Tubli, “ & nimi.

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

Järgmine käsutüüp [**teavita** alustame **ja oota**] esines juba ka eespool, kui käivitus üks Krapsu skript. Praegu on tegemist üldisema juhuga, mil 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 peatatakse peaskripti täitmine, kuni lõpeb kõikide alamskriptide töö, ning peale seda jätkub antud skripti täitmine. Vaatleme nüüd nendes protsessides osalevate spraitide vastavaid skripte ja tegevusi, mis 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“, mis on mõeldud tema enda skriptile – see paneb Juku hüppama. Skriptis on plokiga [**korda** 3] määratud etteantud korduste arvuga kordus, st et ploki sees olevaid käske täidetakse kolm korda. Iga kord suurendatakse Juku y-koordinaati 40 pikseli võrra, tehakse väike paus, vähendatakse koordinaati 40 pikseli võrra ja tehakse jälle paus. Tulemusena liigub Juku kolm korda üles ja alla, imiteerides hüppamist. Käsk [**teavita** löök **ja oota**] käivitab palli skripti, mis viib palli Juku juurest Krapsu juurde ja sealt tagasi. Vahepeal lendab pall ka üles.

Juku **Robi**

Robi skriptid on analoogsed. Käsuga [**mängi heli** robi] käivitatakse salvestatud heliklipp. Salvestada saab vahelehel **Helid** korraldusega **Lindista**, kui arvutil on mikrofon. Kui see 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 saadud esimesest kasutades Scratchi joonistamisredaktorit.

Alustame. Krapsu ja Mari skriptid

Kraps **Mari**

Muusika ja tantsulembese **Mari** skriptides demonstreeritakse heliandmete kasutamist ja tantsu imiteerimist kostüümide vahetamise teel. Skriptis **alustamine** mängitakse lihtsalt üks noot, kasutades Scratchis olevat lihtsat helide süntesaatorit. Käsus [**mängi nooti** *number* **kestvusega** *aeg*] saab näidata noodi numbri (48 kuni 72) ja kestvuse sekundites. Kes tunneb noote, võib proovida ka komponeerida meloodiat. Plokiga [**mängi trummi** …] saab imiteerida ka löökpillide helisid.

Skriptis **tantsi** võetakse Mari jaoks esimene kostüüm ballerina-a (igal kostüümil on nimi). Seda tehakse igaks juhuks, arvestades et programmi eelneval katkestamisel võib Mari jääda suvalisse olekusse. Käsuga [**mängi heli** *Triumph*] käivitatakse heliklipp ning skripti töö jätkub muusika saatel. On olemas ka käsk [**mängi heli** *nimi* **kuni valmis**]. Selle käsu korral skripti töö peatatakse, mängitakse heliklipp lõpuni ning seejärel 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 liigub üles 30 pikselit, teeb ühe tiiru ja liigub tagasi alla. Paneme tähele, et 360-kraadine pööre ei ole määratud ühe käsuga: [**pööra** 360 **kraadi**], vaid 30 korda täidetava käsuga [**pööra** 12 **kraadi**] (30 * 12 = 360), et pööre oleks nähtav (jälgitav). Kui kasutada esimest varianti, siis pööret näha ei oleks. Realiseeritud variandi korral, kus igale pöördenurga muutusele (12 kraadi) järgneb ka väike paus, jaguneb liikumine kaadriteks. Taoline põhimõte on animatsioonides üldkasutatav. Liikumise kiirust ja sujuvust saab reguleerida vaadeldaval juhul nurga muutusega ja pausi pikkusega. Paneme tähele, et pause 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 määratud nö puutesündmustega – puudutab Kraps, puudutab Juku. Teises on tingimus esitatud lava serva puudutamisega (üles liikudes) ja jõudmisega antud kõrguseni (alla liikudes). Pallil on ka mõned klahvisündmustele reageerivad skriptid: [**kui vajutatakse klahvi** vasak nool], [kui **vajutatakse klahvi** ülesnool] ja [**kui klõpsatakse** pall], millest on näidatud ainult esimene.

Saleduse leidmine

Käsk [**teavita** *saledus* **ja oota**] käivitab rakenduse teise osa, kus põhitegelaseks on Juku. Skriptis **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 **pöörle** 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 tegemist ei olnud 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 kasutatakse taolist skaalat:

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

Siin kasutavas skriptis ei arvestata viimast kriteeriumi ($k_{\text{ind}} > 30$ – suur ülekaal). Lugeja võiks proovida see lisada.

Muutujate monitorid on sageli olulisteks kasutajaliidese elementideks. Antud rakenduses on ette nähtud monitoride peitmine ja kuvamine vastavalt olukorrale. Seda teevad spraidi **abi** skriptid: **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 lavale joon. Joonistamisel kasutatakse peamiselt käske gruppidest **Pliiats** ja **Liikumine**.

Pliiatsi käskudega saab määrata joone värvi, varjundi ja paksuse ning muuta pliiatsi olekut, kas üleval või all. Praegu valitakse pliiatsi värv ja suurus juhuslikult.

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

Hüvastijätt

Selles tegevuses osalevad kõik tegelased, kusjuures tegu on paralleelsete protsessidega. Siin demonstreeritakse ka lihtsamaid võimalusi loendite kasutamiseks.

Loendis nimega **T** on tekstid, millest valitakse iga spraidi jaoks juhuslik väärtus, mis kuvatakse 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 ning see leitakse spraidi **abi** skriptis.

Lava skriptid

Laval on kaks skripti. Üks käivitub koos Krapsu põhiskriptiga, kui klõpsatakse rohelist lippu, muutes lava värvi.

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

Värvi muudetakse sagedusega üks kord sekundis. Värvi kood valitakse juhuslikult.

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

Andmed

Järgnevalt vaadeldakse objekte, põhimuutujaid ja rakenduse struktuuri.

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)

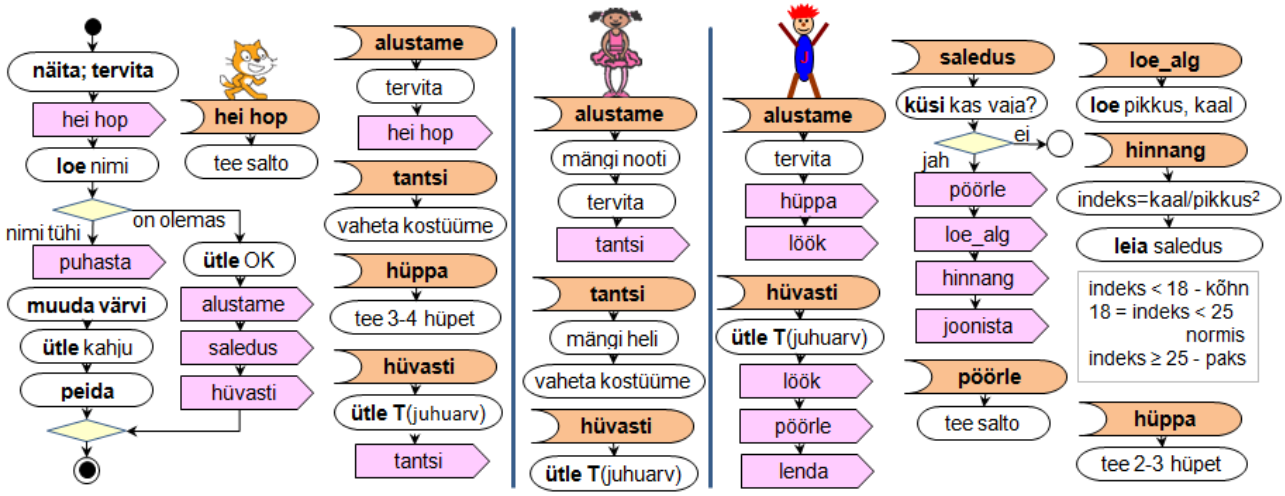
pall – teda togivad ja pilluvad **Juku** ja **Kraps** (üks kostüüm, viis skripti)

abi (peidetud) – joonistab ja teeb muid abitöid (üks kostüüm, neli skripti)

Põhimuutujad: nimi, pikkus, indeks, saledus

Rakenduse struktuur

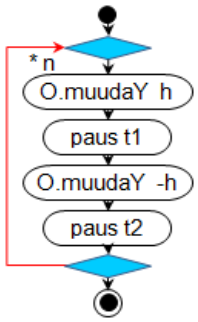
On näidatud üldvaade kolme spraidi skriptidest. Koostööd peegeldavad teavitamise korraldused ja vastuvõtvate skriptide päised. Vt Rakendused.



Algoritmid

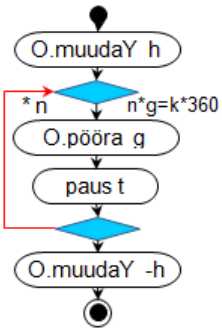
Näiteks on toodud mõned algoritmid UML tegevusdiagrammidena ja pseudokoodis. Vt Algoritmimine.

hüppa



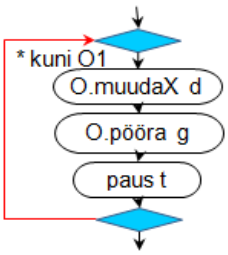

```
prots hüppa
kordus n korda
    objekt.muudaY h
    paus t1
    objekt.muudaY -h
    paus t2
```

salto



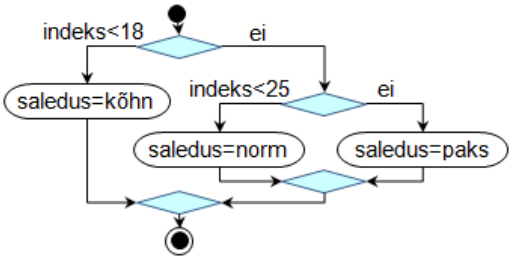

```
prots salto
objekt.muudaY h
kordus n korda
    objekt.pööra g
    paus t
lõpp kordus
objekt.muudaY -h
```

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
```

Rakenduste disainist ja dokumenteerimisest

Suuremate rakenduste korral on otstarbekas enne skriptide koostamist täpsustada ülesande püstitust, viia läbi rakenduse analüüs ja disain (projekteerimine) ning koostada projekti dokumentatsioon:

- määratleda andmed,
- fikseerida spraitide funktsioonid,
- seosed nende vahel ja tegevused,
- koostada algoritmid,
- kavandada kasutajaliides.

Rakenduse realiseerimise käigus dokumentatsioon enamasti täiustub ning viimasena vormistatakse selle lõplik variant, mida läheb vaja rakenduse kasutamisel ja arendamisel.

Lisamaterjal: Rakendused.