Scratchis toetatavad programmeerimiskontseptsioonid ja -oskused

Interaktiivsete lugude, mängude ja annimatsioonide loomise protsessis saab algaja teha Scratch'is endale selgeks mitmed olulised rakenduste loomise ja programmeerimise oskused ja kontseptsioonid.

Probleemide lahendamise ja projektide disaini oskused

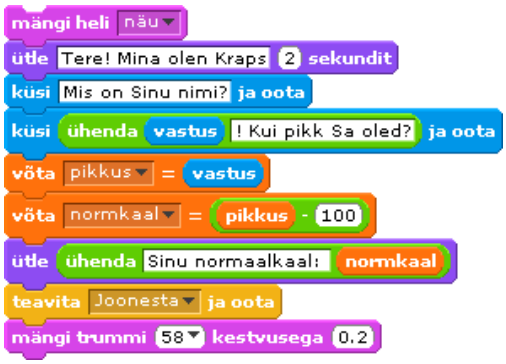
- loogiline ja algoritmiline mõtlemine
- süsteemne lähenemine probleemide lahendamisele
- ideede arendus alates lähtekontseptsioonist kuni projekti lõpplahenduseni
- kasutajaliideste loomise oskused ja kogemused
- silumise ja testimise vilumused
- keskendumisvõime ja visaduse arendamine

Fundamentaalsed ideed arvutitest ja programmeerimisest

- programm ütleb arvutile täpselt, samm-sammult, mida teha
- programmide koostamine ei nõua spetsiaalseid teadmisi, vaid selget ja loogilist mõtlemist

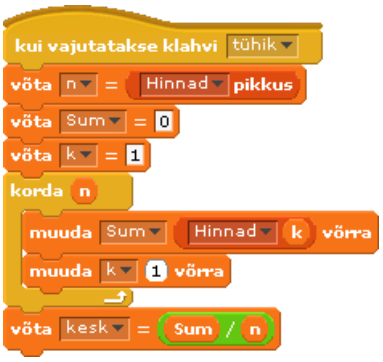
Rakenduste loomise ja programmeerimise põhikontseptsioonid ja põhimõisted

Kontsept	Selgitus	Näide
Kasutajaliides	Kasutajaliides sisaldab vahendeid, mille abil saab kasutaja suhelda programmiga: anda vajalikke korraldusi ja näha nende täitmise tulemusi, muuta algandmeid jms. Scratchis luuakse liides **laval**. Selle elementideks võivad olla taustad, aktiivsed ja passiivsed spraidid, käsunupud, klahvid, muutujate monitorid jms. Teistes süsteemides kasutatakse selleks vorme, ohjureid, dialoogibokse jms. Töö programmiga võib toimuda Scratchi enda keskkonnas, brauseris Java apletiga või Flashi filmiga.	 Näita
Programm. Laused ja **plokid. Programmi-üksused: protseduurid** ja **skriptid**	Programm on **lausete** (korralduste, käskude) kogum, määramaks, milliseid **tegevusi** peab arvuti täitma **andmete** ja/või **objektidega**, tagades ka **kasutajaliides**e töö. Igas keeles on piiratud valik lauseid, mille esitamiseks on kindlad reeglid. Scratch'is kujutavad **käsud** (laused) endast graafilisi **plokk**e, mis on jagatud otstarbe alusel gruppidesse: **Liikumine**, **Juhtimine** jms. Plokiga on lause süntaks määratletud üheselt ja süntaksivigade tekkimine on peaaegu võimatu. Enamikus keeltes võib programm koosneda mitmest üksusest: **protseduurid**, funktsioonid, skriptid jms.	 Programm koosneb neljast skriptist. Kaks on spraidi Kraps omad, Jukul ja piigal on üks skript. Kui klõpsatakse rohelist lippu, algab Krapsu

	Scratch'is nimetakse programmiüksusi **skriptideks**. Iga skript on seotud ühe kindla spraidiga, määrates selle tegevusi. Ühel spraidil võib olla mitu skripti. Skript saab käivitada (öeldakse ka pöörduda, kutsuda välja) teisi skripte, mis kuuluvad samale või teistele spraitidele. Pöördumiseks kasutakse plokke **teavita *nimi*** või **teavita *nimi* ja oota**.	esimese skripti täitmine. Peale "teretamist" käivitab antud skript käsuga **teavita *start* ja oota** korraga Juku ja piiga skriptid, mis algavad plokkidega **kui saabub teade *start***, ning ootab kuni nende täitmine jõuab lõpuni. Seejärel jätkub Krapsu skripti täitmine. Kraps liigub lava keskpunkti ja siis käivitatakse sama spraidi teine skript. Peale selle täitmist, jätkatakse esimese skripti täitmist, mis viib Krapsu ette antud punkti. Näita
Objektid (spraidid). Objektide Omadused, meetodid ja sündmused	Scratch'is on kesksel kohal **graafilised objektid**, mida nimetakse **spraitideks** (sprite) ning nende kostüümid. Objektideks on samuti ka **lava** ja selle taustad. Kuigi Scratch ei ole formaalselt objektorienteeritud süsteem, on sageli otstarbekas seda käsitleda, tuginedes objektorienteeritud lähenemisviisile. Iga objektiga on seotud teatud valik **omadusi:** nimi, asukoht laval (x-y), suurus, värvus jms ning **meetodeid**, mille abil määratakse tegevusi antud tüüpi objektiga: asukoha muutmine, pööramine, suuruse ja värvuse muutmine jms. Käsuplokid sisuliselt vastavad meetoditele. Objekt võib reageerida **sündmustele** nagu nt hiireklõps, vajutus etteantud klahvile, kokkupuude teise objektiga…	Skripti täitmine algab, kui klõpsatakse spraiti Kraps (reaktsioon sündmusele). Näita  Skripti plokid muudavad objekti (spraidi) omadusi: suund, asukoht, suurus, värvus, keere.
Andmete liigid	Scratch'is saab kasutada märk-, graafika- ja heliandmeid. **Märkandmed**: arvud ja tekstid ehk stringid. Neid saab kasutada paljudes plokkides (käskudes) konstantidena, muutujatena ja loendite elementidena. Väärtusi saab leida (tuletada) avaldiste ja funktsioonide abil. **Graafikaandmed** võivad esineda rakendustes kahes variandis: **Spraidid** ja lava **taustad** – imporditakse või luuakse graafikaredaktori abil. Käskude abil saab määrata erinevaid tegevusi nendega. **Pliiatsi** käskudega tehtavad **joonised**	 Skriptid demonstreerivad erinevat liiki andmete kasutamist. Märkandmed (arvud ja stringid) on kasutusel konstantide ja muutujatena. Näita

	Heliandmed. Saab kasutada erinevaid vahendeid helide tekitamiseks (plokid **mängi nooti...**, **mängi trummi...** jms), importida ja lindistada heliklippe: kõne, muusika jms.	 Graafikaandmeid esindab sprait Kraps ja pliiatsiga tehtav joon spraidi liikumisel. Heliandmetega on tegemist plokkide **mängi heli** ... ja **mängi trummi**... kasutamisel.
Andmete organisatsioon	Organisatsiooni järgi eristatakse kõikides programmeerimiskeeltes **skalaarandmeid**: konstandid ja muutujad **struktuurandmeid**: tabelid, massiivid, loendid jm **Konstantide** väärtused esitatakse vahetult programmis. Ülaltoodud skripti näites on mitu arv- ja stringkonstanti. Need on plokkide väljades asuvad konkreetsed arvud: 2, 100, 58, 0.2, 0 jne; ja tekstid: ”Tere! Mina olen Kraps!”, ”Mis on Sinu nimi?” jne. **Muutujad** esitatakse programmis nimede abil, nende väärtusi programmis ei näe – väärtused tekivad tavaliselt programmi täitmise ajal. Näiteks muutuja *pikkus* väärtus tekib sisestamisel käsuga **küsi**, muutuja *normkaal* väärtus tekib arvutuste tulemusena (*pikkus* – 100). Muutujate olemusest ja käsitlemisest vt allpool. Tabelid, massiivid, **loendid** kujutavad endast kindla struktuuriga väärtuste kogumeid. Scratch’is saab kasutada **ainult loendeid**, mis on massiivide erijuht (vt allpool)	
Muutujad ja omistamine	Muutuja on **mälupesa**, kuhu programm saab salvestada väärtusi: arve ja stringe ning kasutada (lugeda) neid hiljem näiteks uute väärtuste leidmiseks. Scratch’is saab muutujaid luua ja kasutada grupis **Muutujad** olevate plokkide abil. Loomisel saab muutujale anda **nime**, mida kasutatakse käskudes viitamiseks tema jooksvale väärtusele. Loomisel saab ka määrata, kas muutuja on kättesaadav kõikidele spraitidele (globaalne muutuja) või ainult ühele konkreetsele spraidile (lokaalne muutuja). Muutujatele väärtuste omistamiseks ja väärtuste muutmiseks kasutatakse plokke **võta** *muutuja = avaldis* ja **muuda** *muutuja avaldis* võrra Muutuja väärtust saab kuvada laval nn **monitori** abil  . Väärtust saab muuta ka ”käsitsi” **liuguri**ga.	Programm imiteerib korvpalli viskeid  **Näited:** Korvpall **ja** Ideaal

<table>
<tr>
<td>Loendid (massiivid)</td>
<td>Scratch’i loend vastab enam-vähem teistes keeltes kasutatavatele dünaamilistele ühemõõtmelistele massiividele. Taoline massiiv kujutab endast järjestatud mälupesade (elementide) kogumit. Elementidele saab viidata massiivi nime ja indeksite abil.
Scratch’is saab luua ja kasutada loendeid grupi Muutujad plokkidega. Viimaste abil saab lisada elemente loendi lõppu ja vahele, asendada ja eemaldada elemente, viidata elementidele jms.
 
</td>
<td> 

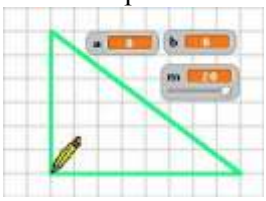
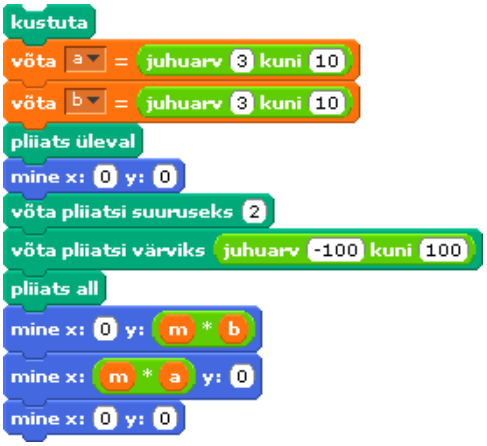
Skript keskmise hinna leidmiseks
 

Näita</td>
</tr>
<tr>
<td>Avaldised, tehted ja funktsioonid</td>
<td>Avaldise abil saab anda eeskirja vajaliku väärtuse leidmiseks. Avaldiste loomiseks kasutatakse Scratch’is grupi Tehted plokke.
Avaldis koosneb operandidest ja tehetest.
Operandideks võivad olla konstandid, muutujad, funktsioonid ja loendite elemendid.
Sõltuvalt tehetest võib avaldised jagada järgmistesse rühmadesse:
• arvavaldised: +, -, *, /
• stringavaldised: ühenda, eralda
• võrdlused: <, =, >
• loogikaavaldised: ja, või, mitte
Avaldistes saab kasutada mitmeid funktsioone: abs (absoluutväärtus), sqrt (ruutjuur), sin, cos, asin, log, ln …
Sisuliselt on funktsioonideks ka eraldi esitaud plokid; mod (jääk), ümarda ja juhuarv.
NB! Sulgusid tehete järjekorra määramiseks Scratchis kasutada ei saa. Iga tehte plokki võib lugeda sulgudes oleva avaldise liikmeks</td>
<td> 

$\frac{2(a-x)}{a+b}$
 

$c = \frac{a+13}{\log|b|}$ c ümardakse kahe kohani peale koma
 

palli asukohta laval muudetakse juhuarvude abil
Näide Ideaal</td>
</tr>
</table>

<table>
<tr>
<td>Joonistamine, joonestamine</td>
<td>Enamikus programmeerimiskeeltes on olemas vahendid, mis võimaldavad luua programmil täitmise ajal jooniseid. Scratchis on peamised vahendid selleks gruppides Pliiats ja Liikumine.
Kõrvalolev skript joonistab täisnurkse kolmnurga, mille külgede pikkused a ja b tekitatakse juhuarvude abil.
Täisnurk pannakse lava keskele punkti  (0,0). Muutuja m abil määratakse mastaap: punktide arv ühes pikkuse ühikus.</td>
<td> 

Demo Rist_Ring</td>
</tr>
<tr>
<td>Märkandmete sisestamine ja väljastamine</td>
<td>Ülesannete lahendamisel on sageli programmil vaja saada algandmeid ja peegeldada tulemusi. Esimesel juhul räägitakse andmete sisestamisest (lugemisest), teisel juhul – andmete väljastamisest (kirjutamine, kuvamine). Scratch’is saab andmete sisestamiseks kasutada muutujate liuguritega monitore (ainult arvud) ja plokki küsi, mis võimaldab sisestada arve ja tekste dialoogirežiimis.
Tulemusi saab väljastada muutujate ja/või loendite monitoride või plokkidega ütle.</td>
<td> 

Näide Ideaal</td>
</tr>
<tr>
<td>Protsesside juhtimine</td>
<td colspan="2">Väga tähtsal kohal ülesannete lahendamisel on tegevuste täitmise järjekorra määramine. Just selles seisnebki programmeerimise kunst. Vajalikud tegevused ja nende järjekord ei sõltu üldiselt kasutatavast keelest, vaid tulenevad ülesande olemusest ja selle lahendamise algoritmist.
Võib eristada nelja liiki protsesse: jada ehk järjestikune protsess, kordus ehk tsükliline protsess, valik ehk hargnev protsess ning paralleelne protsess.
Programmeerimiskeeltes on olemas vahendid erinevat liiki protsesside kirjeldamiseks.</td>
</tr>
<tr>
<td>Jada ehk järjestikune protsess</td>
<td>Programmi koostamisel peab arvestama, et plokkide poolt määratud tegevusi täidetakse kindlas järjekorras.
Lihtsamal juhul on tegemist järjestikuse protsessiga, kus plokke täidetakse järjest ülevalt alla.</td>
<td> 
</td>
</tr>
</table>

<table>
<tr>
<td>Kordus ehk tsükliline protsess</td>
<td>Korduste kirjeldamiseks on Scratch’is, nagu ka teistes keeltes, mitu lauset (plokki): lõputult, korda n, korda kuni tingimus, lõputult kui tingimus.
Ploki lõputult sees olevaid plokke täidetakse põhimõtteliselt lõpmatult. Skripti töö saab katkestada näiteks punase nupuga. Korduse sees võib olla üks või mitu plokki kui tingimus, mis tingimuse täitumisel katkestab vastavate plokkide abil skripti (plokk peata skript) või terve programmi töö (peata kõik).

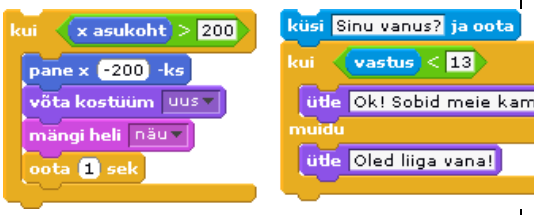
Käsu korda n täitmisel korratakse ploki sees olevaid plokke n korda. Kordamiste arv (n) võib olla antud konstandi, muutuja või avaldise abil.

Käsu korda kuni tingimus täitmisel korratakse ploki sees olevaid plokke seni kuni tingimus saab tõeseks.

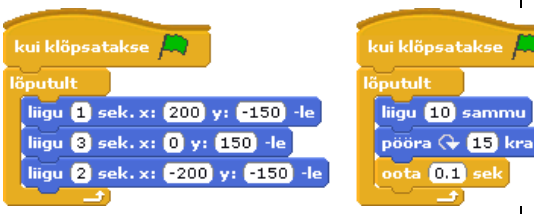
Käsu korda kui tingimus täitmisel korratakse ploki sees olevaid plokke nii kaua, kui tingimus on tõene</td>
<td> 

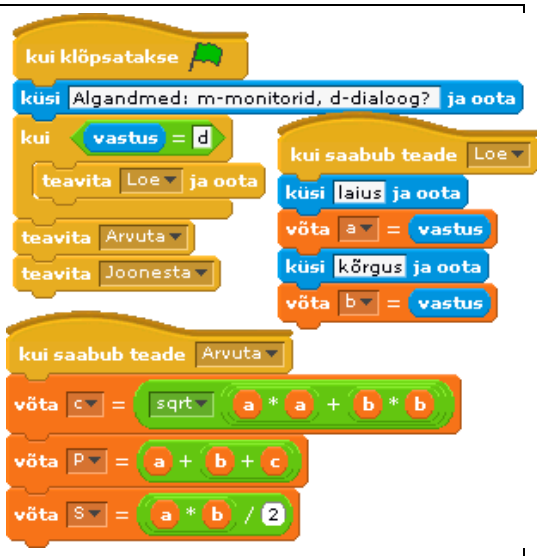
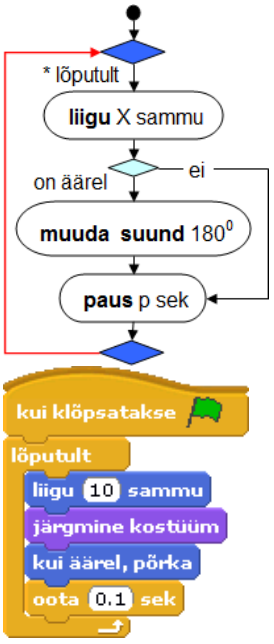
Esimese ploki toimel ”jalutab” sprait lõpmatult laval edasi-tagasi. Katkestada saab punase nupu abil. Teise ploki täitmisel liigub sprait vasakult paremale. Liikumine (kordamine) lõpeb, kui spraidi x-koordinaat saab suuremaks 200st,
 

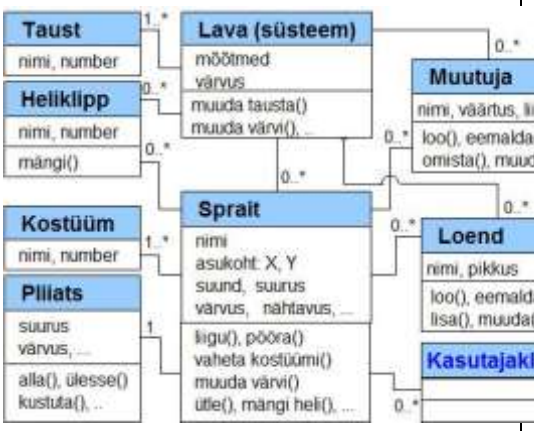
Objekt (näiteks kiisu) ”põristab” trummi ja teeb ringi (360°) laval.
Näita
 


</td>
</tr>
<tr>
<td>Valik ehk hargnev protsess</td>
<td>Plokkidega kui tingimus ja kui tingimus muidu saab kirjeldada valikuid ehk hargnevaid protsesse.
Lause kui ... korral (valik ühest), kui tingimus on tõene täidetakse ploki sees olevad plokid, vastupidisel juhul jäetakse need vahele.
Lause kui ... muidu korral (kahendvalik), kui tingimus on tõene täidetakse esimeses harus olevad plokid, vastupidisel juhul teises harus olevad plokid.</td>
<td> 

Esimeses plokis kontrollitakse tingumust, kas
x-koordinaat on suurem 200-st, kui jah, siis täidetakse sisemised plokid. Teises skripti fragmendis kuvatakse teade, sõltuvalt sisestatud vanusest. Näide Ideaal</td>
</tr>
</table>

Harud ehk paralleelsed protsessid	Kahte või enamat skripti (protsessi) saab täita samaaegselt ehk paralleelselt. Paralleelset täitmist saab määrata mitmel erineval viisil. Näiteks kõik skriptid, mille esimeseks plokiks on rohelise lipuga plokk, käivitatakse samaaegselt ja täidetakse paralleelselt, kui klõpsatakse rohelist lippu. Paralleelselt täidetakse ka skripte, mis algavad plokiga **kui saabub teade *nimi***, milles on kasutusel sama ***nimi***.	 Üks objekt liigub pidevalt mööda kolmnurkset trajektoori, teine – mööda ringikujulist.  Kui saabub teade ”korras” , üks objekt teeb 3 hüpet, teine vahetab 8 korda kostüümi (näiteks tantsib) Projekt Tantsud
Sündmused	Objektid võivad reageerida teatud sündmustele: vajutus mingile klahvile, objekti klõpsamine hiirega, kokkupuude teise objektiga või lava servaga jms. Skriptides võib ette näha reaktsioone kindlatele sündmustele. Peamisteks vahenditeks sündmuste haldamisele on nn päiseplokid: **kui vajutakse klahvi** ja **kui klõpsatakse spraiti** Skriptide sees leiab sageli kasutust plokk **kui puudutab** {sprait | serv | kursor} Kui antud sprait puudutab teist spraiti, lava serva või hiirekursorit, tagastab plokk väärtuse ***tõene***. Põhimõtteliselt on sündmusega tegemist ka rohelise lipu kasutamisel ja skriptide käivitamisel plokkide **teavita** ja **kui saabub teade** abil (vt allpool).	 Kui klõpsatakse rohelist lippu, võetakse muutujate väärtuseks 0 ja pall algseisu Kui vajutakse tühikuklahvi, muudetakse juhuarvudega palli asukohta ja suurendatakse löökide arvu ühe võrra. Kui pall puudutab väravat, lisatakse 1 muutujale **tabas**. Proovige!
Dünaamiline interaktsioon	Objekte saab mõjutada nö reaalajas hiire kursoriga (hiire liigutamisega), heliga jm., kasutades grupi **Andurid** plokke **hiire *x***, **hiire *y***, **helitugevus** jm	 Kui klõpsatakse lippu, liigub objekt horisontaal-suunas koos hiirekursoriga ja muudab keerme suurust, vastavalt y-i väärtusele. Proovige!

Koostöö spraitide ja skriptide vahel	Kui programm koosneb mitmest skriptis tekkib sageli vajadus nende töö koordineerimiseks ja sünkroniseerimiseks. Üks sprait võib pöörduda teiste poole, need omakorda järgmiste poole jne. Skriptide koostöö korraldamisel kasutatakse plokke **teavita** *teade* ja **teavita** *teade* **ja oota** ning **kui saabub** *teade* Siin vaadeldav programm koosneb neljast skriptist. Rohelise lipuga skript on **peaskript**. Sellest algab programmi töö ning ta käivitab teisi. Kui kasutaja sisestab tähe ”d”, käivitatakse skript **Loe** ning peale selle töö lõppu pannakse tööle skript **Arvuta**. Kui sisestati d-st erinev väärtus, käivitatakse kohe **Arvuta**.	 Ootamata **Arvuta** töö lõppu pannakse tööle ka skript **Joonesta** (ei ole siin näidatud, on analoogiline skriptiga punktis J**oonestamine**). Näide.
Algoritmimine	Algoritmi all mõistetakse eeskirja, mis määrab, milliseid tegevusi ja millises järjekorras tuleb täita antud ülesande lahendamiseks või töö tegemiseks. Vajalikud tegevused ja nende täitmise järjekord ei sõltu üldiselt konkreetsest keelest vaid ülesandest või mudelist. Tegevused algoritmis esitatakse üldisemal kujul. Viimasel ajal kasutatakse algoritmide esitamiseks sageli modelleerimiskeelt UML. Scratch’i skripte võib vaadelda kui ühte algoritmide esitusviisi.	 Demod: Jalka, Ruutvõrrand, Maks_Vek

Modelleerimine	Scratch'i vahendeid saab kasutada objekt-orienteeritud modelleerimise põhimõistete selgitamiseks ja illustreerimiseks. Scratch'i spraidid kuuluvad ühte universaalsesse **klassi** – **Sprait**. Spraitidel kindel valik **omadusi**: nimi, asukoht, suurus, ... ja **meetodeid**: liigu(), pööra(), muudaX(), ... Viimased esitatakse käskude ehk plokkide abil Käskudest (baasmeetoditest) moodustatud skripte võib käsitleda kasutaja **alam-klassidena**. Alamklassidena on käsitleta-vad ka spraidiga seotud kostüümid, pliiats, muutujad, loendid. **Lava** on vaadeldav nö peaklassina – klass **Sprait** on selle alamklass. Kõik spraidid asuvad laval. Klassi **Lava** võib vaadelda süsteemi või projekti nö esindajana.	 Näide. Projekt Tantsud.