Töölehed Scratch 2.0  
 materjali juurde

**Sisu**

[Kraps123 3](#_Toc408196171)

[Veemaailm 5](#_Toc408196172)

[Efektid 8](#_Toc408196173)

[Joonistamine 9](#_Toc408196174)

[Nimi 10](#_Toc408196175)

[Muutuja kasutamine 11](#_Toc408196176)

[Kraps, pall ja papagoi 13](#_Toc408196177)

[Ristkülik 18](#_Toc408196178)

[Ristkülik ja ring 19](#_Toc408196179)

[Lilleaed 20](#_Toc408196180)

[Ilutulestik 21](#_Toc408196181)

[Juhtimisplokid 22](#_Toc408196182)

[Autod 26](#_Toc408196183)

[Rehmaatika 27](#_Toc408196184)

[Ideaal 29](#_Toc408196185)

[Arvu arvamine 30](#_Toc408196186)

[Ülesanne Kolm modelli 31](#_Toc408196187)

[Ülesanne Mündivisked 32](#_Toc408196188)

[Ülesanne Mündivisked 33](#_Toc408196189)

[Ülesanne Aja määramine 34](#_Toc408196190)

[Ülesanne Kolmnurk 35](#_Toc408196191)

[Labürint 36](#_Toc408196192)

[Ülesanne Päringud loenditest 38](#_Toc408196193)

[Tekst tagurpidi - parandada 39](#_Toc408196194)

[Tekst tagurpidi (ka loendi abil) muuta! 40](#_Toc408196195)

[Autod 2 41](#_Toc408196196)

[Vestlus 42](#_Toc408196197)

[Sõnastik. Keeleõppe test 43](#_Toc408196198)

[Sõnastik. Tõlkimine 44](#_Toc408196199)

[Algoritme loenditega 45](#_Toc408196200)

[Loendite kasutamine spraitide asukoha juhuslikul määramisel 46](#_Toc408196201)

[Kasutajaplokid (protseduurid) 47](#_Toc408196202)

[Rekursioon 48](#_Toc408196203)

[Kloonimine 49](#_Toc408196204)

[Tuli ja vesi 50](#_Toc408196205)

[Kloonimine paigutamisega (lokaalne ID) 51](#_Toc408196206)

[Kaamera ja heli võimaluste kasutamine 52](#_Toc408196207)

# Kraps123

## Harjutus 1

Uue Scratchi rakenduse loomisel on algul laval vaid Kraps – kass kahe **kostüümiga** ja heliga ’näu’. Objekti nime kasutatakse spraidile viitamisel teiste spraitide skriptidest; seda võib muuta igal ajahetkel. Kiisule võiks anda nime: **nimekasti** spraidi **info alas** (lava all aktiivse spraidi kõrval klõpsata nuppu **i** ) kirjutada **Kraps**.

Lihtsa skripti loomiseks lohistada **skriptide alasse** allolevad käsud ja moodustata nendest kõrvalolev **skript**. Seda saab käivitada rohelise lipukesega lava ülaservas.

|  |  |  |
| --- | --- | --- |
|  | cat1-a  cat1-b |  |

Kraps peaks liikuma (suuna määrab spraidi omadus **suund**), vahetama kostüümi, et jääks mulje astumisest ja tegema seda korduvalt (vt. kordused [**korda lõputult**] või [**korda …**]). Esialgu võib korraldust [**oota …**] mitte lisada, siis sibab Kraps väga kiiresti. Lavalt välja sprait ei liigu, servani jõudes pöördub Kraps tagasi (korraldus [**kui äärel, põrka**]). Et ta ei liiguks tagasi jalad ülespidi, tuleks spraidi info alas valida pööramis­stiil vasak-parem.

**NB!** Liikumise kiirus ja sujuvus oleneb sammu pikkusest ja vahepealsest ooteajast.

Sammu pikkus ei pruugi olla konstantne suurus. Selle võib määrata juhuslikult: lohistada väärtuse 10 asemele grupist **Tehted** plokk  ja valida juhusliku väärtuse piirideks näiteks -5 kuni 10. Tekib tantsuline liikumine, Kraps liigub tasapisi siiski edasi.



Animatsiooni teeb (peale liikumise vilkumise ja helide) atraktiivseks juhuslikkus ja kasutaja võimalus kuidagi sekkuda.

Sammu pikkuse võib lasta kasutajal valida tegevuse käigus. Üks võimalus selleks on luua **muutuja** **samm**, mille väärtuse võrra astub Kraps. Muutuja väärtust saab kasutaja muuta laval **liuguri** abil.

Valida käsugrupist **Andmed** korraldus **Loo muutuja**, panna selle nimeks **samm**. Muutuja **monitor** on näha laval, paremklõps sellel avab menüü, kust valida monitori esitusviisiks **liugur** ja seejärel **määra liuguri min ja maks**, näiteks 1 kuni 10. Korraldusse [**liigu …**] tuleks lohistada käsugrupist **Andmed** muutuja **samm**.



Valida Krapsule liikumise taustaks ka ilus lavapilt: aktiveerida lava (vajutada lava ikoonikesele lava all), valida vahekaardil **Taustad** nupp **Vali taust teegst**.

## Harjutus 2

Huvitava animatsiooni saab ka lihtsalt kostüüme (kuvandeid) vahetades.

Lisada uus **sprait** nupuga **Vali sprait teegist**. Valida kindlasti selline, millel oleks mitu kuvandit (kostüümi), näiteks **Ballerina**. Kõik selle spraidi kostüümid lisanduvad automaatselt (vt. vahekaart (tab) **Kostüümid**). „Tantsimiseks“ koostada skript:

## Harjutus 3

Spraidi kostüümide saamiseks on võimalik kasutada **animeeritud GIF**-i.

Selleks lisada suvaline sprait (’vali uus sprait failist’ või ’Vali üllatussprait’), valida vahekaardilt **Kostüümid** nupp **Impordi** ja viidata **animeeritud GIF-failile**. Iga kaader (kiht) lisandub eraldi kostüümina. Vajadusel saab kostüüme parandada ja täiendada Scratchi joonistusredaktori abil.

Lisada spraidid tiiger (tiiger.gif) ja hunt (hunt.gif).

Nende liikumapanemiseks võib mõlemale kopeerida **Krapsu skripti** (lohistada aktiivse spraidi skript lava all olevale ikoonikesele). Et nendel spraitidel on kostüüme tunduvalt rohkem, tuleb määrata nii samm kui ooteaeg väiksemad ning muidugi pööramise stiil ainult nägu vasakule/paremale.

Kui spraidi liikumapanemiseks soovitakse kasutada korraldust [**liigu …**], tuleb juba kostüümide loomisel arvestada, et liikumissuund on originaalkujutisel alati paremale (90 kraadi).

Tiiger hakkab liikuma ilusasti, hunt aga kulgeb kummaliselt – tagurpidi. Hundi kõiki kostüüme tuleb joonistamisredaktori abil redigeerida, peegeldades neid horison­taalselt.

Spraidist **Hunt** võiks eelnevalt teha koopia. Kui kõikide kostüümide redigeerimine õnnestus, saab ülearuse koopia kustutada.

# Veemaailm

Hajutus Scratchi liikumiskäskudega tutvumiseks.

Veemaailmas on võimalikud väga erinevad liikumised. Iga sprait tuleb panna tegutsema erinevalt.

Skriptide tööd on mugav kontrollida, kui panna need algul tööle ükshaaval (hiireklõps skriptil). Valmis skripti (või selle osa) saab kopeerida lohistades teisele spraidile, siis ei pea kõiki käske uuesti otsima ja kokku lohistama.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| crab1-a.png crab1-b.png  Krabi nihkub merepõhjal edasi-tagasi.  Algul ’liigu 10 sammu’ ja ’oota 1 sekund’. Sujuvama liikumise saamiseks teha samm ja ooteaeg väiksemaks. Pööramise stiil ainult nägu vasakule/paremale. | | | | | |  | |
| fish3.png  Kalake liigub juhuslikult, põrgates servadelt.  Aeg-ajalt muudab kalake ootamatult suunda (teine skript). |  | | | | | | |
|  | | | | shark1-a  Hai jälitab kalakest.  Kui Hai puudutab kalakest, väljastab tema skript teate ’kala käes’, seepeale võib käivituda mingi teise spraidi skript (vt. Kaheksajalg). | | | |
| fish1-a.gif Kala järgneb hiirekursorile,  aga ei lähe liiga lähedale. | | | |  | | | |
| fish2.png Kala2 liigub juhusliku arvu sekunditega juhuslikku kohta ekraanil.  Kasutades käsku [liigu ... sek. x: .. y: ..] ei muutu kahjuks spraidi suund. | | | | | | | |
| fish4.png  Kala3 teeb ujudes ringe, kui hoida all klahvi p. | |  | | | | | |
|  | | | clam1.png  Merikarp teeb paar hüpet, kui kasutaja vajutab tühikule.  Järjest tehtavate hüpete arvu võib määrata juhuslikult (1..5) | | | | |
| octopus1-a.pngoctopus1-b.png  Kui Hai on puudutanud Kalakest, ärkab ka Kaheksajalas jahimeheinstinkt. Ta lõbustab ennast Kala2 hirmutamisega hüpates Kala2 asukohta.  Enne hüpet näidatakse hetkeks tema teist kostüümi. | | | | | | |  |
|  | | | | | *whale1.gif*  Vaal ujub tagaplaanil aeglaselt paremalt vasakule, lava vasakus servas kaob mõneks ajaks ja ilmub uuesti nähtavale paremal (kui tema x-asukoht on väiksem, kui -240, peidetakse ta mõneks sekundiks ja x-asukoha uueks väärtuseks võetakse 240). | | |
| *starfish1-a.png*  Meritäht pöörleb ja liigub edasi-tagasi.  Muutuja **s** on liikumise samm, mis määrab liikumissuuna, see muutub servani jõudes vastupidiseks.  Muutuja lisamine: Andmed – Loo Muutuja. Anda muutujale nimi. Võib olla lokaalne muutuja – ainult selle spraidi jaoks.  Tehted ja tingimuse plokid vt. Tehted. | | | |  | | | |

**Lisaülesanded:**

* Et hiirekursorit jälitav kala kursorit kätte saades kummaliselt käituma ei hakkaks, tohib ta liikuda ainult siis, kui tema kaugus hiirekursorist on suurem, kui liikumissamm.
* Kalakese puutumise järel võiks Hai avada ähvardavalt suu (teine kostüüm).
* Kalakese kiirust võiks muuta (muutuja 'samm' liuguri abil).
* Kalake võiks suunda muuta ka siis, kui tema kaugus Haist on liiga väike (~50 sammu).
* Merikarbi hüppe kõrguse võiks valida juhuslikult.
* Kuidas saaks meritäht pöörelda selles suunas, kuhu ta liigub?

# Efektid

Harjutus on Scratchi graafikaefektidega tutvumiseks.

**Luua paar suvalist erinevat spraiti ja proovida nende peal korraldusi efektide (värv, kalasilm, keere, pikselda, mosaiik, heledus, kummitus) ning omaduste suurus ja nähtavus muutmiseks.**

Efektide paremaks mõistmiseks võib koostada lihtsaid skriptikesi:

NB! efektide puhul võib muutus olla ka miinusmärgiga (-10); mõne efekti jaoks (näiteks pikselda) pole muutuse väärtusel tähtsust.

**Lisada ühele spraidile mingi hulk kostüüme (pilte, ka tekste) ja koostada skript, kus valitakse spraidi jaoks juhuslik kuvand (kostüümi nr piirides 1 kuni kostüümide arv), määratakse sellele juhuslik suurus ja demonstreeritakse mitmeid õpitud efekte. Lõpuks liigub sprait kaugusse ja kaob.**

# Joonistamine

Harjutus on Scratchi joonistamisvõimalustega tutvumiseks.

Iga sprait saab liikudes jätta lavale värvilise joone. Spraidist võib jääda lavale **jäljend**. Joone suurust, värvi ja varjundit, pliiatsi seadeid (all, üleval) jm on võimalik muuta käskudega grupist Pliiats. Lava puhastamiseks on käsk **kustuta**.

1. **Leida teegist joonistamiseks sobiv sprait (näiteks *Pencil*). Joone jätab spraidi keskpunkt, joonistamis­redaktori abil võiks määrata pliiatsi kostüümi keskme pliiatsi teraviku otsa.**
2. **Koostada allolev skript ja proovida, mis värvi joone jätab pliiats lavale.** 
3. **Lohistada skriptide alasse PLIIATSi käsk [võta pliiatsi värviks 0]. Klõpsata käsuplokil, et see käivituks.** Mis värvi joone jätab pliiats nüüd?
4. **Proovida mitmeid värvitoone.** Mis värvid on 22, 35, 60, 80, 100, 120, 140, 160, 180, 200, 220?

Erinevaid värve on 0 kuni 200, siis hakkavad värvid korduma.

1. **Proovida muuta ka pliiatsi varjundit.**
2. **Luua endale töövahendid (skriptide käivitamiseks noole- ja täheklahvid):**

|  |  |
| --- | --- |
| [**Kui vajutatakse klahvi** *paremnool*]  [muuda pliiatsi värvi 1 võrra] | [**Kui vajutatakse klahvi** *vasaknool*]  [muuda pliiatsi värvi -1 võrra] |

1. **Teha sellised skriptid ka pliiatsi varjundi ja suuruse muutmiseks.**
2. **Tühikuklahviga oleks mugav kogu joonistus kustutada – käsk [kustuta].**
3. **Lisada uued spraidid: joonistada ise värvipaanid (näiteks ristkülikud). Värvi valimiseks ristküliku värvi järgi koostada skriptid:**

|  |  |
| --- | --- |
| [**Kui klõpsatakse** *spraiti* ]  [teata *kollane*] | pliiats[**Kui saabub teade** kollane]  vali_varviks |
|  |  |

kui klõpsata käsus [vali pliiatsi värviks ] oleval ruudukesel, muutub hiiremarker pipetiks, millega saab ise valida õige värvi vastavalt värvipaanilt.

Nimi

Harjutus on Scratchi liikumiskäskude ja graafikaefektide kordamiseks ning skriptide käivitamisvõimaluste proovimiseks.

Luua spraidid tähtedega (kaust Letters) ja moodustada nendest oma nimi. Panna tähed liikuma (liuglema, pöörlema, hüppama) ja muutuma (värvuma, keerduma, kaduma jm). Valida oma rakendusele sobiv taust (taustad) ja lisaelemendid.

Projekti märkustesse (või eraldi tekstifaili) kirjutada, kuidas spraidid mingitele sündmustele reageerivad.

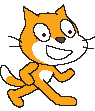
Skripte on võimalik käivitada:

* rohelise lipukesega
* klaviatuuri klahvidega
* hiireklõpsuga mõnel spraidil või laval
* reaktsioonina teatele, mille tekitab mõni teine skript.

Näiteid skriptidest:

Hiireklõps spraidil Kraps käivitab skripti, milles on (muuhulgas) teade ’laiali’, millele võivad reageerida suvaliste spraitide skriptid. Näiteks pärast Krapsu klõpsamist punuvad kõik tähed laiali.

 l

|  |  |
| --- | --- |
| m  r | Hiireklõpsule tähega spraidil võib reageerida hoopis mingi teine sprait. |

|  |  |
| --- | --- |
| Kusagil võiks kasutaja jaoks olla info ka selle kohta, millisele krahvile või spraidile on mõtet vajutada. Näites jalutab Kraps ekraanil ja ütleb aeg-ajalt midagi kasulikku. | cat1-a |

# Muutuja kasutamine

*Muutuja on* ***koht*** *(pesa, väli)* ***arvuti mälus*** *mingi* ***väärtuse salvestamiseks****.*

*Muutuja* ***omadused*** *on* ***nimi****,* ***väärtus****,* ***tüüp****,* ***skoop*** *ehk mõjupiirkond (kõikide spraitide või ainult ühe jaoks).*

*Muutuja loomiseks Scratchis tuleb klõpsata käsugrupis* ***Andmed*** *nuppu* ***Loo muutuja****.*

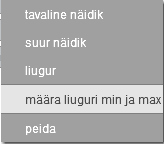
Harjutus on muutuja lihtsamate kasutamisvõimaluste proovimiseks.

### costume_kiirusMuutuja rakenduse tegevuste (kiiruse, sammu vm) muutmiseks

1. Luua uus sprait **Auto** ja muutuja **kiirus** (ainult selle spraidi jaoks).

Kus asub selline muutuja muutujate nimekirjas?

Kontrollida: kui aktiivne (valitud) on lava või mõni teine sprait, siis seda muutujat muutujate nimekirjas näha ei ole; muutuja monitori juurde laval on kirjutatus ka spraidi nimi.

1. Valida muutuja jaoks liuguriga monitor ja määrata liuguri väikseim ja suurim väärtus (min ja max).   
   Näiteks -5 kuni 15.
2. Koostada auto jaoks skript, mis käivituks rohelise lipuga:

[lõputult]

[liigu *kiirus* sammu]

[oota *0.1* sek]

[kui äärel, põrka]

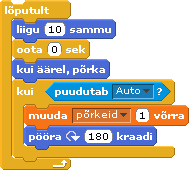
Liuguri abil muutuja monitoril saab muuta auto liikumise kiirust. Nii saab panna auto liikuma ka tagurpidi.

### Muutuja loendurina

1. Luua uus sprait **Pall**. Luua muutuja **põrkeid** (kõikide spraitide jaoks).

Muutuja väärtust saab muuta suuremaks (või väiksemaks) käsuplokiga porkeid1.

1. Koostada **palli** jaoks skript, mis paneb palli laval liikuma (lendama) ja põrkuma lava servadelt. Kui **pall** puudutab **autot**, tuleb muutuja **põrkeid** väärtus muuta 1 võrra suuremaks. Ka see skript peaks käivituma rohelise lipuga. Palli liikumise suuna võiks skripti algul valida juhuslikult.

 [lõputult]

[liigu *10* sammu]

[oota *0.1* sek]

[kui äärel, põrka]

[kui puudutab *Autot*]

[muuda *põrkeid* *1* võrra]

[pööra *180* kraadi]

Muutuja ja selle väärtus jäävad alles ka pärast rakenduse töö lõppu, seepärast peab mängu uuesti alustades loenduri „nullima“.

1. Skripti algusesse (enne korduseplokki) tuleks panna käsk porkeid0.

Eelneva tegevuse tulemuseks on mäng, milles kasutaja peab auto eemale juhtima palli eest.

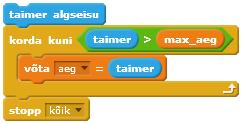
Mängus loendatakse tabamuste arv.

Lisame mängule ka ajaarvestuse.

Kui ületatakse lubatud aeg (max\_aeg), saab mäng otsa (lõpeb auto ja palli liikumine).

### Taimeri kasutamine

*muut_taimerTaimer on Scratchi sisemine muutuja, milles näidatakse aega (sekundites) alates mingist ajahetkest. Taimeri saab viia algseisu (nullida) käsuga [taimer algseisu].   
Taimeri monitori saab näidata laval (vt märkeruut). Taimerit peatada ei ole võimalik.*

1. Luua muutujad **aeg** ja **max\_aeg**. Rakenduse töötamise aja **max\_aeg** võiks määrata kasutaja, muutuja **aeg** monitoris kuvatakse jooksev aeg, seni kuni kestab mäng.
2. Koostada lava jaoks skript, mis käivitub rohelise lipukesega:

[taimer algseisu]

[korda kuni taimer > max\_aeg]

võta aeg = taimer

[peata kõik]

**NB!** Selle skripti käivitamisel saab mäng kohe otsa, sest muutujale **max\_aeg** ei ole antud veel väärtust (vaikimisi on muutuja väärtus 0).

### Muutuja väärtuse küsimine kasutajalt

*kysiKasutajalt saab küsida väärtuse käsuploki [****küsi*** *....] abil.*

*Nii sisestatud väärtus paigutatakse sisemisse muutujasse (mäluvälja) nimega* ***vastus****, sealt võib selle hiljem võtta mingi teise (omaloodud) muutuja väärtuseks.*

1. Lisada eelmise skripti algusesse käsud:

[küsi (mitu sekundit kestab mäng?) ja oota]

[võta max\_aeg = vastus]

[teata start]

1. Muidugi ei tohi auto ja pall enne mängu aja määramist liikuma hakata. Asendada Auto ja Palli skriptides päiseplokk [kui klõpsatakse (roheline lipp)] päiseplokiga [kui teade *start*]

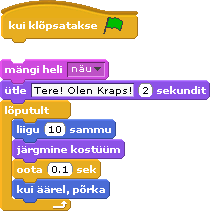
# Kraps, pall ja papagoi

Koostada rakendus (mäng) järgmise stsenaariumi alusel:

Kraps jalutab rahulikult palliga rannas edasi-­tagasi. Järsku ilmub taevasse papagoi, kes lendab ringi ja narrib Krapsu. Kraps üritab mängija abiga tabada palliga papagoid: palli lennutamiseks peab kasutaja vajutama tühikuklahvi. Mängu eesmärgiks on tabada papagoid võimalikult rohkem kordi etteantud aja jooksul (max\_aeg).

Rakendust arendada järk-järgult.

### Krapsu liikumine.

Liikumise ja välimuseplokid; skriptide käivitamine.

Esialgu võib Kraps liikuda ilma pallita.

Krapsul on kaks kostüümi. Liikumiseks koostada skript:

Määrata pööramise stiil ’ainult nägu vasakule/paremale’.

Kasutajal võiks olla võimalik juhtida Krapsu nooleklahvide abil.

Koostada skriptid, mis käivituvad vastavalt klahvidega ’vasak nool’ ja ’parem nool’.



### Papagoi tegevused

Spraitide ja kostüümide lisamine, skriptide kopeerimine ja redigeerimine, tingimused ja valikud, spraitide asukoha määramine, spraitide peitmine ja kuvamine, spraitide „ringliiklus“, juhuslike arvude kasutamine.

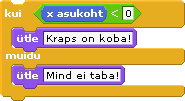
Lisada sprait **papagoi** ja selle teine kostüüm. Kopeerida **Krapsu** skript **papagoile** ja proovida.

Ringliikluse korraldamiseks muuta: kui papagoi jõuab lava parema servani, peita sprait (käsk[**peida**]) korraks (1-3 sek), viia lava vasakusse serva ja tuua nähtavale ([**näita**]).

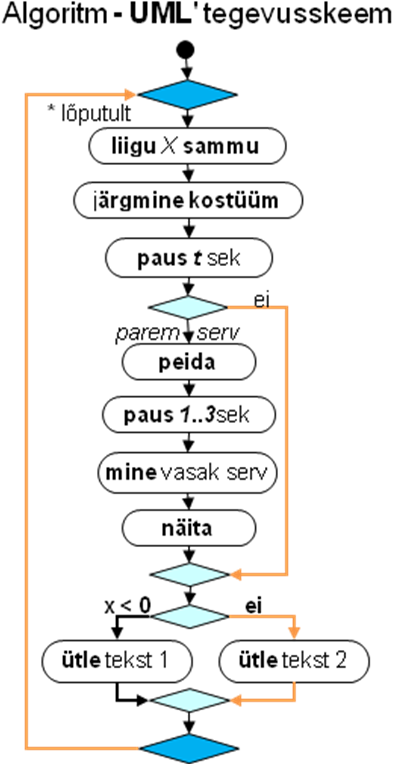
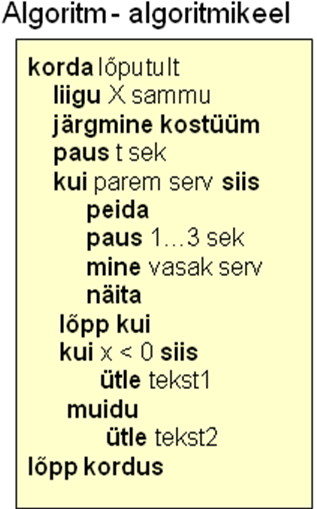
Pausi tekitamiseks kasutada **juhuarve**.

Et teha tabamine raskemas, võiks käsu [**pane x**-240] asemel võtta näiteks käsu [**pane x** juhuarv -240 kuni 120]. Papagoi ilmub nüüd nähtavale juhuslikus kohas antud vahemikus.

Lisada **papagoi** "jutt" (narrimine): **kui** asukoht < 0-st jutt1 **muidu** jutt2



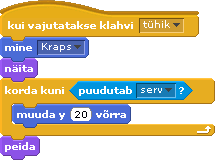
Tegevuste kirjeldamisel on juhendmaterjalis edaspidi kasutusel ka algoritmikeel ja UML tegevus­diagrammid. Võrrelda alltoodud algoritme papagoi skriptiga. Kas on sarnane?

### Palli lisamine ja selle tegevused.

Joonistamisredaktor, sündmus ’puudutab’, objekti viimine teise spraidi asukohta, spraidi kese, tingimuslik kordus, skripti peatamine, teadete saatmine ja vastuvõtmine**.**

Palli jaoks joonistada kujutis joonistamisredaktori abil: ring (shift-klahvi abil ellipsist) ja sellele mõned värvilised laigud. Spraidi (kostüümi) kese on tavaliselt keskel, võimalik määrata ka mujale.

Koostada tühikuga käivitatav skript palli lendamiseks Krapsu juurest üles lava servani.

**mine** Kraps

näita

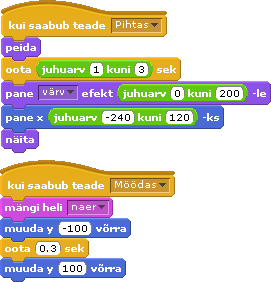
kordus kuni puudutab serv

**muudaY** h võrra

peida

|  |  |
| --- | --- |
| Nüüd tuleb kontrollida, kas pall tabab papagoid. Seda on võimalik teha nii palli kui papagoi skriptis. Otstarbekam lisada palli skripti, sest sel pole mõtet, kui pall ei lenda.  Tabamusest tuleks kindlasti teavitada (teade …), et ka teised spraidid (Kraps ja papagoi) saaks rea­geerida (ühe spraidi skript ei saa määrata tegevusi teiste spraitidega). Teavitama peaks ka mitte­tabamisest. | **mine** Kraps  näita  kordus kuni *serv*  **muudaY** h võrra  kui puudutab papagoi  peida  **teavita** Pihtas  peata skript  lõpp kui  lõpp kordus  peida  teavita Möödas |

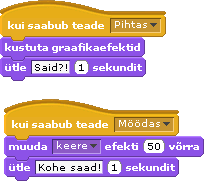
Kassi ja papagoi reaktsioonid:



**Papagoi**

**Pihtas**: papagoi peidetakse korraks, Kraps ütleb midagi

**Möödas**: papagoi naerab, laskub hetkeks allapoole, Kraps - mingid graafikaefektid (keere jm).



**Kraps**

### Visete ja tabamuste loendamine

Muutuja, omistamine

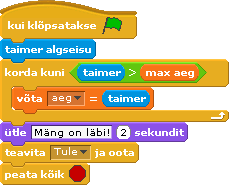
Luua kaks globaalset (nende väärtusi saavad kasutada ja muuta kõigi spraitide skriptid) muutujat loendamiseks: ***viskeid*** ja ***tabas***. Praegu on mõistlik teha nende muutujate väärtuste muutmiseks kas eraldi skriptid või lisada vastavad käsud palli skripti. Visete arv ***viskeid*** peaks muutuma palli lenduminekul ja tabamuste arv ***tabas*** pärast papagoi puudutamist.

**NB!** Muutujate algväärtustamine!

Korduval käivitamisel jätkatakse eelmisest väärtusest. Uue mängu algul tuleks loendurid nullida. Kui neid muutujaid palli skriptide juures juba kasutusel on, võiks ka algväärtustamised olla palli teises skriptis, mis käivitub mängu alustamisel (rohelise lipukesega).

### Aja kuvamine ja mängu lõpetamine.

Taimer, Käsk [**peata kõik**], liugur (kerimisriba) muutujate väärtuste määramiseks,“ teatespraidid“

Luua kaks muutujat: ***aeg*** (jooksev aeg) ja ***max\_aeg*** (mängu aeg). max\_aeg monitorile laval lisada **liugur** (kerimisriba), mille abil saab "käsitsi" muuta muutuja väärtust, ja määrata min ja max väärtused. Valida max\_aeg väärtuseks esialgu 10 (sek).

Skript taimeri nullimiseks, aja näitamiseks ja mängu lõpetamiseks võib olla suvalise spraidi või ka lava oma.

Taimeri plokid on grupis **Andurid.**

Kui klõpsatakse rohelist lippu

taimer algseisu

**korda kuni** *aeg* > *max\_aeg*

*aeg* = taimer

peata kõik

Mängu võib lõpetada ka lubatud visete arvu möödumisel. Luua muutuja **m\_viskeid**, lisada liugur (ka min ja max), valida sobiv väärtus.

Käsk ’oota kuni …’ seab skripti ooteseisu (kaudselt kordus).

Skripti võib panna suvalise spraidi moodulisse: näiteks lava või Kraps



Teadete, selgituste jmt kuvamiseks võib ploki **ütle** asemel kasutada nö "tekstispraite". Spraidil võib olla mitu kostüümi: igaüks neist kujutab omaette teadet. Vastavalt olukorrale võetakse sobiv kostüüm ja tehakse sprait nähtavaks.

Siin on kasutusel kaks teadet:

**Aeg on läbi!** (kostüüm Aeg)

**Visete arv on täis!** (kostüüm Visked)

Sõltuvalt lõpetamise põhjusest valitakse ja kuvatakse vastav teade.

### Palli liikumine koos Krapsuga. Olekumuutuja.

Pall peaks liikuma koos Krapsuga ja võiks ka pöörelda (NB! **Lisaülesanne** kiirematele). Sellise liikumiste sünkroniseerimise saab teha erinevalt. Näiteks võib Kraps iga kord pärast sammu astumist „kutsuda“ palli enda juurde.

Lisada teavitamise käsk [teavita *tule* ja oota] Krapsu põhiskripti.

Koostada palli jaoks skript, mis käivitub teatele **Tule**: viia pall **Krapsu** juurde ja alla (valida sobiv nihe) ning pöörata palli, arvestades Krapsu liikumise suunda.

**Mis juhtub, kui nüüd pall pannakse lendama? Tuleb tagasi!**

Võtta kasutusele nö olekumuutuja (tunnus) ***lendab****:*

* *k*ui **pall** lendab, on muutuja väärtus **1** muidu **0** (kokkulepe)
* muutuja väärtust muudab **palli** põhiskript
* algväärtus 0 (koos teiste muutujate algväärtustamisega).
* lisada Krapsu skripti valikuplokk kui lendab=0 teatega 'tule'.



### Rakenduse kirjeldus (dokumentatsioon)

Rakendusest arusaamiseks, vajadusel muutmiseks ja otstarbekaks kasutamiseks koostatakse alati ka projekti dokumentatsioon:

* kasutusjuhend – eesmärgid, tegevuste käivitamise võimalused
* spetsifikatsioon – selgitused spraitide ja nende skriptide (tegevuste) ning muutujate kohta.

Skriptide juures võivad olla kommentaarid - lühikesed selgitavad tekstid. (Loomiseks paremklõps skriptide alas, ilmuvast menüüst valida "**lisa kommentaar**"; saab muuta asukohta, siduda konkreetse käsuga jm.)

Kasutusjuhend (näiteks eraldi spraidina) võiks olla kättesaadav ka projekti kasutajaliideses.

Suuremate projektide korral võivad taolised märkused olla vormistatud eraldi dokumentidena.

Allpool on näitena toodud vaadeldava projekti (rakenduse) kirjeldus.

#### Kasutusjuhend

Kraps jalutab palliga edasi-tagasi. Papagoi lendab ringiratast ülalpool.

Eesmärk on tabada palliga papagoid võimalikult palju kordi valitud aja jooksul: max\_aeg (saab muuta). Programm näitab aega mängu algusest, tehtud visete ja tabamuste arvu. Visete arvu saab piirata muutuja m\_viskeid väärtusega.

Uue mängu algus: **Roheline Lipp.** Krapsu pööramine: vasaknool ja paremnool. Pall läheb lendu tühiku­klahvile vajutamisel.

#### Selgitused

**Põhispraidid**: Kraps, papagoi ja pall. Lisaks sprait **Teated**.

**Kraps** (2 kostüümi, 5 skripti). Jalutab palliga edasi-tagasi.

Põhiskript (käivitub rohelise lipuga) juhib Krapsu (ja ka palli) liikumist edasi-tagasi. Kui pall ei lenda, tuuakse see igal sammul Krapsu juurde (palli vastava skripti reaktsioon teatele **Tule** Krapsu skriptis).

Nooleklahvide (parem ja vasak) skriptid võimaldavad Krapsu pöörata ja liigutada vastavas suunas.

Palli teadete Pihtas ja Möödas toimel käivitatavad skriptid määravad reaktsioonid nendele sündmustele.

**Papagoi** (2 kostüümi, 3 skripti).

Põhiskript: juhib papagoi "ringliiklust" ja kuvab tekstid Krapsu narrimiseks.

Palli skriptist saadetud teadete Pihtas ja Möödas toimel käivitatavad skriptid määravad papagoi reaktsioonid nendele sündmustele.

**Pall** (1 kostüüm, 3 skripti).

Põhiskript: juhib palli lendu, kontrollib puudet papagoi või servaga, mõlemal juhul lend katkestatakse. Skript muudab ka muutujate viskeid, tabas ja lendab väärtusi. Muutujate algväärtustamine toimub eraldi skriptiga (käivitub rohelise lipuga). Krapsu põhiskriptis oleva teate Tule toimel käivituv skript viib palli Krapsu juurde ja pöörab veidi Krapsu liikumise suunas.

**Lava** (1 taust, 2 skripti). Taustaks ranna pilt või ka mingi muu.

Üks skript muudab ja kontrollib taimeri abil aega. Kui aeg ületab muutuja max\_aeg väärtuse, initsialiseerib vastava teate kuvamise ja lõpetab kogu projekti töö. Teine skript kontrollib lõpetamist visete arvu järgi.

**Muutujad:**

***viskeid*** – tehtud visete arv; muudab palli põhiskript

***tabas*** – tabamuste arv; muudab palli põhiskript

***aeg*** – aja jooksev väärtus; muudab lava skript

***max\_aeg*** – lubatud mängu aeg, kasutaja saab valida (muuta) liuguri abil

***m\_viskeid*** – visete maksimaalne arv, valib kasutaja

***lendab*** – palli olekumuutuja: 1 - lendab, 0 - ei lenda; muudab palli skript

# Ristkülik

Koostada rakendus, mis arvutab ristküliku pindala ja ümbermõõdu suhte.

Lisada animatsioon, kus õpetaja küsib kasutajalt ristküliku külgede pikkused ja seejärel „küsib“ õpilaselt vastused.

Kasutajaliides peaks välja nägema selline:

### Analüüs

Antud on ristküliku küljed (tähistame c ja d). Leida tuleb pindala S, ümbermõõt P, nende suhe ja ristküliku diagonaal. Kasutada saab järgmisi arvutusvalemeid:

S = c \* d; P = 2(c + d); suhe = S / P; Diag =

### Disain

**Muutujad**: c, d, S, P, suhe, Diag; muutujate monitorid suurte näitudena.

**Spraidid**: Õpetaja, Õpilane, tekst1 (1. külg), tekst2 (2. külg), tekst3 (tulemused).

Skriptide loomisel võiks üldjoontes järgida sellist skeemi:

**Õpetaja tegevused**

**Õpilase tegevused**

arvuta pindala

arvuta ümbermõõt

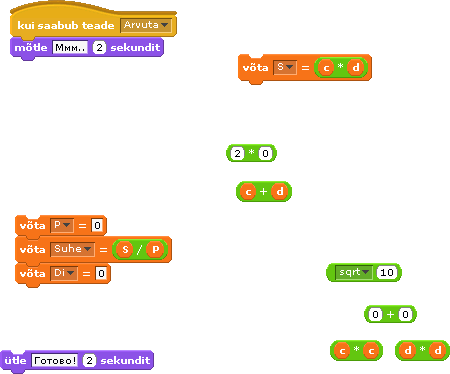
arvuta suhe

arvuta diagonaal

Kui teade *Arvuta*

ütle ***Valmis*** *2 sek*

mõtle ***Hmmm*** 2 sek



kustuta kõik vanad väärtused

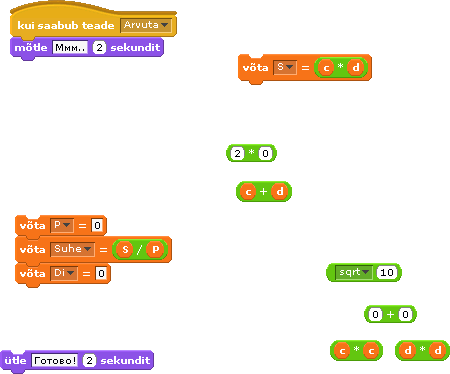
küsi c ja d väärtused ja omista need muutujatele

tervita õpilasi 2 sek

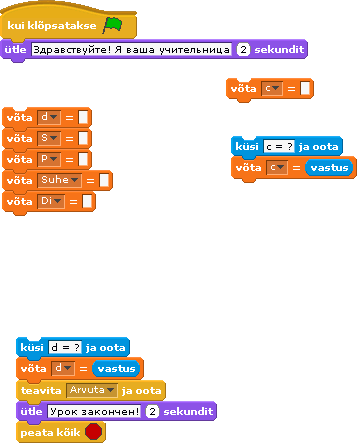
teade ***Arvuta*** ja oota

Ütle ***Tund on läbi*** *2 sek*

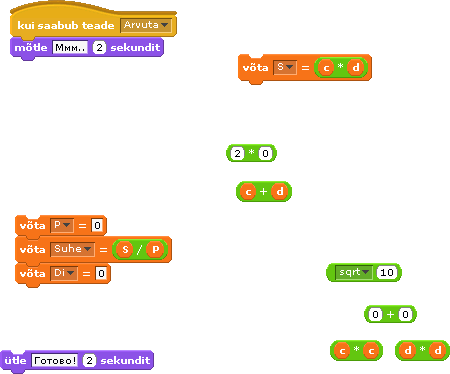
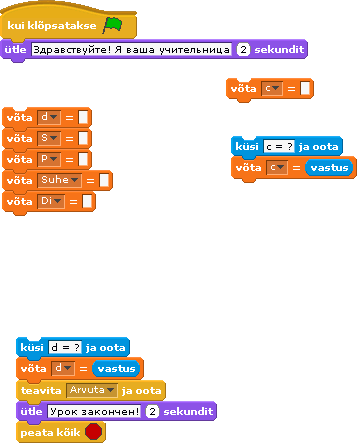
Peata kõik



veel tuleb küsida ***d***



ja kõik ülejäänud



# Ristkülik ja ring

Koostada rakendus, mis leiab ristküliku külgede abil sama suure pindalaga ringi raadiuse ja joonestab mõlemad kujundid.

### Analüüs:

Antud on ristküliku küljed **a** ja **b**. Leida tuleb ristküliku pindala **S**, mis on ühtlasi ka ringi pindala, ja ringi raadius **r**.

Tuletada meelde valemid ristküliku ja ringi pindalade leidmiseks. Kuidas leida ringi pindala järgi ringi raadius?

Ringi saab joonestada hulknurgana, kui **n** (nurkade arv) on piisavalt suur.

Polaarkoordinaatides on ringjoone iga [punkt](http://et.wikipedia.org/wiki/Punkt_(matemaatika)) üheselt määratud [kaugusega](http://et.wikipedia.org/wiki/Kaugus) fikseeritud punktist ([koordinaatide alguspunktist](http://et.wikipedia.org/w/index.php?title=Koordinaatide_alguspunkt&action=edit&redlink=1)) ning nurgaga fikseeritud suunast (nurk määratakse polaarteljest vastupäeva). Punktide koordinaadid saab leida valemitega:

**x = r · cos(nurk)**

**y = r · sin(nurk)**

Ringjoone joonistamisel anname muutujale **nurk n** väärtust vahemikus 0 … 360.

### Disain:

Vajalikud muutujad: **a**, **b**, **S**, **r**, **nurk**, **n**. Kõikide muutujate monitorid peaks olema näha ka laval. Muutujad **a** ja **b** saavad väärtuse rakenduse kasutamisel, muutuja väärtust saab kasutaja muuta kas liuguri abil või küsitakse need skriptis (võib realiseerida erinevad variandid). **S** ja **r** tuleb arvutada, kui **a** ja **b** väärtused on teada. **Nurk** ja **n** on vajalikud ringi joonestamisel.

Objektid (spraidid): **rist** ristküliku joonestamiseks ja **ring** ringi joonestamiseks (nii saab moodustada mõlemad kujundid samaaegselt). Rakenduse ilmestamiseks võiks olla veel mingi tegelane, näiteks **Kraps**

**Kraps** küsib kõigepealt kasutajalt ristküliku külgede väärtused ja arvutab väärtused **s** ja **r**. Seejärel käivituvad kujundite joonistamise skriptid.

### Realisatsioon:

Luua muutujad **a**, **b**, **S**, **r**, **nurk**, **n** ja paigutada monitorid sobivatesse kohtadesse laval (muutujat **nurk** pole vaja näidata, **n** optimaalse väärtuse väljaselgitamiseks võiks esialgu võimaldada seda muuta liuguri abil). Luua spraidid **Rist**, **Ring**, **Kraps** (on niigi olemas). Määrata lava taustaks koordinaatteljestik ’xy-grid’.

Koostada skriptid (erinevatele spraitidele):

|  |  |  |
| --- | --- | --- |
| **Kraps**  **Kui klõpsatakse rohelist lippu**  kustuta varasem joonis  küsi a  küsi b  S = a\*b  r = sqrt(S/3,141592)  teavita joonista | **Ristkülik**  **Kui saabub teade joonista**  mine[–a/2; -b/2]  osuta suunas 90  pliiats alla  liigu a sammu ja pööra 90 kraadi  liigu b sammu ja pööra 90 kraadi  liigu a sammu ja pööra 90 kraadi  liigu b sammu  pliiats üles | **Ring**  **Kui saabub teade joonista**  nurk = 0  mine r; 0  pliiats alla  korda n korda  nurk=nurk+360/n  mine [r\*cos(nurk); r\*sin(nurk)]  pliiats üles |

### Lisaülesanded:

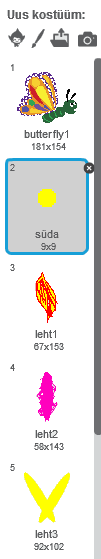
* Mõlema kujundi joonestamine võiks toimuda aeglaselt (jälgitavalt) ja lõppeda samal ajal.
* Kuidas saaks ringi ja/või ristküliku joonestada laval mingisse teise kohta?
* Edaspidiseks: kujundi (ringi, ristküliku jm) joonistamiseks on mugav luua lisaplokk, millele asukoht ja mõõdud antakse parameetrite abil.

# Lilleaed

Koostada rakendus, mis sarnaneks Scratch 1.4 demole *GardenSecret* (näidete alamkaustas *InterActive Art*).

Hiireklõps laval käivitab tegevuse, mille tulemusena „kasvab“ hiirekursori asukohta lill: kõigepealt joonistatakse vars alates lava allservast kuni hiirekursorini, seejärel moodustub (õielehe kujutise jäljenditest) varre otsa õis.

Lilled peaksid olema erinevad (juhuslikult valitud suurus, kuju, värv, õielehtede arv), erineda võiks ka nende ilmumisviis.



Objektid: varre joonistaja, õieleht (pöördub ja jätab jäljendi), südamik.

Kui lillede ilmumisviisis pole plaanis väga suuri erinevusi, võivad need olla ka ühe spraidi erinevad kostüümid: üks vars, üks südamik ja mitu õielehte, mille jaoks valitakse iga kord erinev värv ja suurus, ning kogu tegevus on ühes skriptis või on erinevad tegevused igaüks eraldi skriptis (nii on neid mugavam testida) ja skript valitakse juhuslikult.

Keerulisemate erisuste jaoks (kui lilleõis moodustub erinevast hulgast erinevatest kujunditest), on parem luua mitu spraiti (üks iga õiekomplekti jaoks).

Tegevused varre joonistamiseks:

Mine (hiire x; -180)

Võta lillevarrejoonistamiskostüüm

Võta sobiv suurus

Vali pliiatsi värv ja suurus

Pliiats alla

Liigu (hiire x; hiire y)

Pliiats üles

Tüüptegevused õie joonistamiseks:

Vali sobiv kostüüm, suurus ja värv (juhuslikult)

Võta <arv> = juhuarv (mõistlikes piirides)

Korda <arv> korda

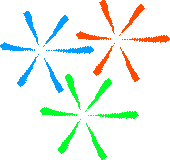
Jäta jälg

Pööra 360/<arv>

# Ilutulestik

### Idee <http://scratch.mit.edu/projects/johndo77/1503800>

### tapidÜlesanne1: hiirega klõpsatud kohal laval plahvatab rakett

1. Luua uus sprait **rakett** joonistades tema kostüümiks 6-7 punktikest.
2. Koostada **Lava** jaoks skript, mis käivitub **hiireklõpsuga laval** ja väljastab teate, et sprait **rakett** võib tegutsema asuda [**teata** *start*].
3. Koostada spraidi **rakett** skript, mis käivitub, **kui saabub teade *start***, viib spraidi hiirekursori asukohta, valib sellele juhusliku värvuse, teeb spraidi algul pisikeseks (5%) ja siis muudab tasapisi suuremaks pidevalt jäljendeid jättes.

[**mine** hiirekursor]

[**pane** värv **efekt** (**juhuarv** .....)]

[**võta** **suuruseks** 5 %]

[**korda** 20]

[**jälg**]

[**muuda** **suurust** 10 **võrra**]

Käsk [**pööra** ....] teeb plahvatuse huvitavamaks. Kuhu see paigutada? Kui suur valida pööramisnurk?

tapid2Lisada spraidi kostüümile veel mõned täpid. Katsetada.

Luua võimalus moodustunud kujutise **kustutamiseks**   
(näiteks tühikuklahvile vajutades).

### Ülesanne 2: juhuslikus kohas laval plahvatab rakett, sädemed langevad allapoole.

Luua muutujad **Rx** ja **Ry** raketi juhusliku asukoha meeldejätmiseks ning muutuja **muut**, mis on abiks sädemete allalangemise juures. Koostada spraidile **rakett** skript, mis käivitub mingile täheklahvile vajutades (võib teha koopia eelmisest skriptist ja seda muuta ning täiendada):

[**võta** *Rx* = (**juhuarv** -200 kuni 200)] - muutujates salvestatakse juhusliku punkti asukoht

[**võta** *Ry* = (**juhuarv** -150 kuni 150)]

[**mine** **x:** *Rx* **y:** *Ry*] - sprait paigutatakse varemmääratud juhuslikku punkti

[**pane** värv **efekt** (juhuarv .....)] - juhuslik värv

[**võta** **suuruseks** 5 %] - algul väike suurus

[**võta** *muut* = -1]

[**korda** 20]

[**jälg**] - jäljendi jätmine

[**muuda** **suurust** 10 **võrra**] - sprait muutub suuremaks

[**muuda** **y** (*muut*) **võrra**] - muutuja **muut** väärtuse võrra liigub sprait (allapoole)

[**võta** *muut* = (1.1 \* *muut*)] - **muut** väärtus suureneb veidi

**Lisaülesanne:** lisada viimasesse skipti käsud, et enne plahvatamist algaks raketi teekond lava allservast punktist x: Rx ja y: -180. Ülespoole kerkides peaks ta jätma maha jäljed oma teekonnast. Teha selliste jäljendite jätmiseks spraidile teine kostüüm – üksik punktike. Kui rakett on jõudnud õigele kohale, tuleb kostüüm vahetada.

**Idee**: <http://scratch.mit.edu/projects/15033002/>

Luua sprait – must ristkülik, sama suur kui lava. Koostada skript, milles kummitusefekt võetakse umbes 90, seejärel tekitab sprait lõputult lavale jäljendeid, kustutades nii tasapisi rakettidest jäänud kujutised.

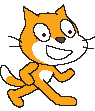
# Juhtimisplokid

Harjutus tutvumiseks protsesside juhtimisega (valik, kordus) programmides ning ülevaate saamiseks Scratchi juhtimisplokkidest ja skriptide käivitamisest.

Lisaks veel:

* viitamine spraidile muutuja abil
* spraidi dublandi kasutamine.

Tegelased: kassike Kraps, kummitus Kaspar, lillevaas ja väike vihane lillehaldjas Piksi.

Kassike Kraps (kaks tavapärast kostüümi) soovib saada kummitus Kaspari sõbraks, jookseb tema suunas ning hüpleb (lihtsam) või teeb kukerpalli (keerukam), kui jõuab piisavalt lähedale.

Kummitus Kasparit võiks juhtida nooleklahvidega. Kui kassike trügib liiga lähedale, muutub ta tasapisi läbipaistvaks ja läheb kiiresti mujale.

Laval on lillevaas, mis läheb ümber, kui Kraps (või Kaspar) seda puudutab.

Lillehaldjas Piksi, kes lendleb mesilasena ümber vaasi, pahandab vaasi ümberajaja peale, muutub pisikeseks vihaseks nõiaks ja hakkab süüdlast jälitama.

Kogu tegevus kestab ettemääratud aja jooksul.

#### Käivitamine:

Skripte on võimalik käivitada rohelise lipukesega lava ülanurgas (Kraps), nooleklahvide abil (Kaspar), hiireklõpsuga spraidil (Lilled) või reaktsioonina teatele, mille käivitab teine skript.

|  |  |  |  |
| --- | --- | --- | --- |
| cat1-a | ghost1 | flower-vase |  |

Teateid, millele teised skriptid saavad reageerida, on kahesuguseid:  ja 

Käsu [teata *...***]** korral jätkub antud skripti täitmine kohe, teatega käivituvad skriptid ja teataja (välja­kutsuja) töötavad paralleelselt.

Käsk [teata *...* ja oota**]** peatab antud skripti töö ja see jätkub alles siis, kui on lõpetanud kõik skriptid, mis teatele reageerisid.

#### Valikud:

Hargnevad protsessid.

|  |  |  |  |
| --- | --- | --- | --- |
| j_kui | Sõltuvalt tingimuse väärtusest täidetakse valikuplokis olevad käsud või jäetakse vahele (valik ühest – täidetakse/ei täideta) | j_kui_muidu | Kahest võimalikust käsugrupist valiku­­plokis täidetakse üks, teise käsud jäetakse vahele (valik kahest, täidetakse üks või teine käsugrupp). |

|  |  |
| --- | --- |
| j_kui_n | j_kui_muidu_n |

#### Kordused:

Tegevuste kordamist (tsükliline protsess) on selles näites vaja tegelaste liikumise tekitamiseks. Kordused on **etteantud korduste arvuga kordus** (Kraps hüppab 3 korda), **lõputu kordus** (Kraps liigub Kaspari suunas) ja **eelkontrolliga kordus**, mis kestab kuni etteantud tingimus saab täidetud.

|  |  |  |
| --- | --- | --- |
| j_kordaj_korda_n  j_korda_kuni | j_korda3 | cat1-aj_loputult |
| juhtimine_aeg | Muutuja **aeg** väärtuseks võetakse **taimer**i väärtus kuni ette­antud aeg otsas; seejärel lõpeb kõigi rakenduse skriptide töö. | |

#### Muud juhtimisplokid Scratchis:

|  |  |  |  |
| --- | --- | --- | --- |
| j_oota | tekitab skripti töös antud pikkusega katkestuse (pausi); sobib kasutada tegevuste (spraitide liikumise/muutumise) kiiruse määramiseks. | | |
| j_oota_kuni | Kontrollitakse lõputult tingimust; kui tingimus on tõene, täidetakse järgnevad tegevused. Selle asemel oleks võimalik kasutada lõpmatut kordust, milles on kui-plokk. Kui tingimus on tõene, täidetakse kui-ploki käsud, siis lõpetatakse töö. | | **Kordus lõputult**  **Kui** tingimus tõene  ……  **Peata skript**  **Lõpp kui**  **Lõpp kordus** |
|  |  | |  |
| **Kaspar** liigub laval noole­klahvide abil (võib lisada eraldi skripti, et ta liiguks pidevalt aeglaselt ja põrkuks seintelt).  Kõrvalolev skript viib Kaspari juhus­likku kohta ekraanil, kui Kraps jõuab häirivalt lähedale. | | ghost1 | |

|  |  |
| --- | --- |
| Kraps ja Kaspar võivad ringi liikudes kogemata ümber ajada **lilled**. Lillevaas peaks seatama püsti rakenduse töö alguses. Hiljem võib kasutaja mahaaetud lillevaasi püsti tõsta sellel klõpsates (roheline lipp ja hiireklõps spraidil käivitavad mõlemad sama tegevuse, mis on vormistatud eraldi skriptina).  Mahaaetud lillevaasi veel kord ümber ajada ei saa, seepärast on kasutusel plokk [**oota kuni ...**]  Sprait **lilled** omistab pahategija nime globaalsele muutujale **lõhkuja**. Muutujat, mille väärtuseks on spraidi nimi, saab mitmetes käsuplokkides kasutada spraidile viitamiseks.  Lillede kukkumisel arvestatakse seda, kustpoolt lõhkuja lähenes.  Teatele **appi** peaks reageerima Piksi (võivad ka teised). | flower-vase |

Lillehaldjas Piksi lendleb mesilasekostüümis ümber lillede, kui klaviatuuril on vajutatud tühik.

Lillevaasi võib asetada ka mõnele teisele riiulile, Piksi tuleb alati lillede juurde tagasi.

Kui lillevaas ümber lükatakse (teade appi), muutub Piksi vihaseks nõiaks (vahetab kostüümi) ja hakkab pahategijat sõimeldes jälitama nii kaua kui selle kätte saab (**eelkontrolliga kordus**); sajatused lõpevad alles pärast seda.

|  |  |
| --- | --- |
|  |  |

Sama spraidi erinevaid kostüüme saab kasutada objekti muutumiseks. Näiteks auto ja avarii, pomm ja plahvatus jm.

Peategelane Kraps liigub kogu rakenduse töö jooksul Kaspari suunas (vt. skript eespool). Et tal vasakule liikudes ei oleks jalad ülespidi, on tema jaoks valitud režiim (vt. spraidi omadused), kus ta saab pöörata vaid näo vasakule või paremale.

Kui Kraps jõuab piisavalt kummituse lähedale, üritab ta Kasparit mängima meelitada hüpates (vt. skript eespool) või kukerpalli tehes (pööreldes) soovitavalt selles suunas, kuhu ta liigub.

Scratch 2 võimaldab ka programselt juhtida spraidi omadust, mis määrab, kas omaduse suund muutumisel saab sprait pöörelda 360°, pöördub ainult nägu vasakule/paremale või ei pöördu üldse.

Siin on probleemiks asjaolu, et kujundi vaba pöörlemist lubades oleks vasakule liikunud Kraps enne ja pärast pööret pea alaspidi, vertikaalsihis peegeldatud lisakostüümi aga ei võimalda mugavalt sammumist imiteerivate kostüümide vahetamine.

Kasutusele võiks võtta veel ühe Krapsu sarnase tegelase sobivate kostüümidega pööramiseks mõlemas suunas. Krapsu põhiskriptis antakse teade **pöörle** ja peidetakse see sprait. Reageerides teatele ilmub samale kohale teine sprait (Kraps1), valib sobiva kostüümi ja teeb oma pöörde. Seejärel ta peidetakse ning lavale ilmub uuesti esialgne Kraps.

|  |  |
| --- | --- |
| cat1-a | cat1-acat1-a |

NB! Kui liikumise simuleerimiseks on kasutusel mitu kostüümi, mida järjest vahetatakse, oleks sama tegelase tavaolukorrast suuresti erineva käitumise jaoks paremaks lahenduseks formeerida veel teine sprait, mis vahetab sobival hetkel esimesega kohad.

# Autod

Ülesanne: koostada rakendus kolme auto võidusõidu imiteerimiseks. Kohtunik (näiteks Kraps) teatab sõidu alustamisest ja kuulutab lõpuks välja võitja. Autode kiiruse (sammu pikkuse) võiks valida juhuslikult, et tulemus poleks ette määratud. Kasutaja peaks saama ette anda sõidetavate ringide arvu.

Iga auto puhul peab olema teada sõidetava ringi number ja sõiduaeg, need võiks olla auto omadused – lokaalsed muutujad, mida saab muuta ainult selle spraidi skriptides (lugeda saab lokaalse muutuja saab ka teistes skriptides).

|  |  |  |
| --- | --- | --- |
| Protseduur **Start**  x = -230  ring = 0  **Kordus** ringe **korda**  **Kordus kuni** x > 240  **Liigu** 10 kuni 15 sammu  **Lõpp kordus**  **Peida**  **Oota** 1 kuni 3 sek  x = -230  **näita**  aeg = taimer  ring = ring + 1  **Lõpp kordus** | Kõikide autode tegevused on ühesugused, seega saab ühest autost pärast skriptide loomist teha vajaliku arvu koopiaid ja seejärel muuta veidi nende välimust.  Algul võiks kõigi muutujate väärtused olla laval näha, et rakenduse tööd oleks parem kontrollida ja võimalikke vigu lihtsam leida. Hiljem võib osa väärtusi peita.  Kõik autod sõidavad nõutud arvu ringe ja peatuvad siis. Lokaalses muutujas **aeg** salvestatakse iga auto lõppuaeg. | |
| Kraps juhatab võidusõitu, tema skriptist tulevad ka teated teiste tegevuste käivitamiseks.  Korraldus [**Teata** Start **ja oota**] tagab järgnevate tegevustega (võitja selgitamine, autasustamine) jätkamise alles pärast seda, kui kõik autod on sõitmise lõpetanud.  Keerukamad tegevused/arvutused võiks vormistada eraldi skriptidena. | | Protseduur **Kraps**  võitja = ’’  **taimer algseisu**  **Teata** Start **ja oota**  **Teata** Kes\_võitis **ja oota**  **Mine** võitja  **Ütle** ’Võitja on tubli’ |

Alternatiivne protseduur **kes\_võitis2**, mis kasutab lisaks muutujat **min** väikseima sõiduaja meeles­pidamiseks, on kompaktsem ja mugavamalt muudetav kui peaks lisanduma veel võistlejaid.

|  |  |  |
| --- | --- | --- |
| Protseduur **kes\_võitis1**  **Kui** (auto1.aeg < auto2.aeg) **ja** (auto1.aeg < auto3.aeg)  võitja = auto1  **Muidu**  **Kui** auto2.aeg < auto3.aeg  võitja = auto2  **muidu**  võitja = auto3  **lõpp kui**  **Lõpp kui** |  | Protseduur **kes\_võitis2**  min = auto1.aeg  võitja = auto1  **Kui** auto2.aeg < min  min = auto2.aeg  võitja = auto2  **Lõpp kui**  **Kui** auto3.aeg < min  min = auto3.aeg  võitja = auto3  **Lõpp kui** |

Lisaks:

* auto kiirus võiks olla spraidi omadus (lokaalne muutuja), mis määratakse (juhuslikult) võistluse alguses
* Kraps võiks rohkem esineda: kamandada kõigepealt võistlejad kohale, lõpus liikuda võitja juurde jmt
* Lisada võistlusele veel üks auto

# Rehmaatika

Koostada rakendus arvutamise harjutamiseks. Kuvatakse tehteid liitmine-lahutamine ja kasutaja peab võimalikult kiiresti sisestama vastuseid.

Arvud (liidetavad) ja tehe tekitatakse juhuarvude abil etteantud piirides min...max. Et rakendus sobiks kasutamiseks algklassiõpilaste jaoks, ei tohiks lahutamistehe tulemus olla negatiivne arv.

Kasutaja peab saama valida taset, millest sõltub tehetes kasutatavate arvude suurus ja punktide arv ühe tehte kohta ning katse aega.

Lisada võib ka sobiva animatsiooni.

Tegevuste üldine skeem on järgnevalt toodud UML tegevusdiagrammidena ja ka algoritmikeeles.

**Muutujad:**

**a** ja **b** – juhuarvud vahemikus min...max,

**c** – a ja b summa (a + b)

**tehe** – juhuarv: 1 - liitmine, 2 - lahutamine

**tulem** – õige vastus: c (kui liitmine) või b (kui lahutamine)

**vastus** – kasutaja vastus

**punkte** – punktide jooksev ja lõplik summa

**tase** – ülesannete tase, võtame 3 taset

**min** ja **max** – min ja maks väärtus, sõltuvad tasemest näiteks: 1. tase: 1...10, 2. tase: 5...20, 3. tase: 10...50

**p** – punkte ühe tehte eest, sõltub tasemest, näiteks: 1. tase: p = 2, 2. tase: p = 5; 3. tase: p = 10

**max\_aeg** – katse maksimaalne aeg, laseme valida kasutajal, näiteks vahemikus 20 ... 60 sek

**aeg** – jooksev aeg katse lõpuni: max\_aeg – taimer

|  |  |
| --- | --- |
|  | **Algoritm** (algoritmikeel)  **protseduur** **Liida\_lahuta**  punkte = 0  **kutsu** **Tee\_tase**  **kordus lõputult**  a = **juhuarv**(min, max)  b = **juhuarv**(min, max)  c = a + b  tehe = **juhuarv**(1, 2)  **kui** tehe = 1 **siis**  **kuva** a & “ + “ & b  tulem = c  **muidu**  **kuva** c & “ - “ & a  tulem = b  **lõpp kui**  **loe** vastus  **kui** vastus = tulem **siis**  punkte = punkte + p  **muidu**  **kuva** “Vale!“  **lõpp kui**  **lõpp kordus** |

Rakendust võiks hakata ehitama järk-järgult:

* tehte koostamine
* tehte vastuse küsimine, vastuse kontroll (esialgu ei pruugi punktiarvestust toimuda)
* raskusastme valik ning miinimumi, maksimumi, ühe tehte eest antavate punktide muutmine vastavalt valitud tasemele.

Realiseerimisel võib kasutada lisaks abimuutujaid näiteks küsimuse (tehte teksti) säilitamise jaoks.

|  |  |
| --- | --- |
| Raskusastme kohandamine vastavalt eelnevalt määratud tasemele | |
| Variant1 | |
| min = 1; max = 10; p = 2  min = 5; max = 20; p = 5  tase =1  tase =2  min = 10; max = 50; p = 10  tase =3  ei  Tegevusskeem - **UML**    ei  ei | **protseduur Tee\_Tase**  **kui** tase = 1 **siis**  min = 1; max = 10  p = 2  **kui** tase = 2 **siis**  min = 5; max = 20  p = 5  **kui** tase = 3 **siis**  min = 10; max = 50  p = 10  **Algoritmikeel**    Kõiki kolme tingimust kontrollitakse ka siis,  kui tase on 1. Parem on variant 2. |
| Variant2 | |
| tase =1  tase =2  ei  ei  min = 1  max = 10  p = 2  min = 5  max = 20  p = 5  min = 10  max = 50  p = 10 | **protseduur Tee\_Tase**  **kui** tase = 1 **siis**  min = 1; max = 10; p = 2  **muidu**  **kui** tase = 2 **siis**  min = 5; max = 20; p = 5  **muidu**  min = 10; max = 50; p = 10  **lõpp kui**  **lõpp kui** |

Lisaülesandeid:

* Loendada ka tehete ja õigete (ja valede) vastuste arv
* Väikseima ja suurima liidetava asemel võiks olla ette antud väikseim liidetav ja suurim summa.
* Lisada ka korrutamise ja jagamise tehted.
* Vale vastuse puhul võiks sama tehet korrata.
* Raskusaste (tase) võiks punktisumma kasvades suureneda automaatselt.

# Ideaal

Koostada rakendus inimese keha omaduste leidmiseks.

Inimese soo (mees/naine), vanuse *t* (aastates), pikkuse *l* (cm) ja kaalu *m* (kg) alusel tuleb alltoodud valemite abil arvutada ideaalne mass *mid* (kg), rasvaprotsent *r*, kehamassi indeks *kind*, tihedus *ρ* (kg/m3), ruumala *V* (dm3), pindala *S* (m2) ning anda kehamassi indeksi alusel sõnaline *hinnang*.

* Keerulisemate avaldiste väärtuste leidmisel on otstarbekas kasutada abimuutujaid.
* Muutujaid võiks näidata laval alles siis, kui neile on saadud väärtused kas kasutajalt või arvutuste teel.
* Väärtused tuleks sobivalt ümardada. Liigne täpsus ei ole hea!
* Lisada sobiv animatsioon.

Saadud tulemusi võib võrrelda alloleva tabeli andmetega.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sugu** | N | M | M | N |  |
| **Vanus** | 22 | 20 | 60 | 37 |  |
| **Pikkus** | 165 | 180 | 175 | 160 | cm |
| **Kaal** | 48 | 70 | 90 | 78 | kg |
| **Ideaalne mass** | 55,6 | 72,5 | 78,8 | 55,6 | kg |
| **Rasvaprotsent** | 6,2 | 11,4 | 27,5 | 50,8 | % |
| **Kehamassiindeks** | 17,6 | 21,6 | 29,4 | 30,5 |  |
| **Tihedus** | 1086,9 | 1076,0 | 1042,3 | 993,4 | kg/m3 |
| **Ruumala** | 44,2 | 65,1 | 86,4 | 78,5 | dm3 |
| **Pindala** | 1,5 | 1,9 | 2,1 | 1,9 | m2 |
| **Hinnang** | kõhn | normaalne | ülekaalus | ei või olla! |  |

**NB!** Astendamist Scratch ei tunne, aga on võimalus eksponendi ja logaritmi leidmiseks ning 

Internetis on hulk lehekülgi, kus logaritmi kohta selgitusi saab. Näiteks <http://et.wikipedia.org/wiki/Logaritm>

Scratchis on kümnendlogaritmi leidmiseks funktsioon (plokk) log. log(100) = 2 , sest 102 = 100

Naturaallogaritmi (logaritm alusel e, e väärtus on umbes 2,72) leidmiseks on Scratchis funktsioon ln() 

Scratchis puudub astendamise operatsioon, aga on funktsioonid (plokid) kümne astme arvutamiseks  ja e astme arvutamiseks .

Logaritmi omadustest tuleneb, et astendamine taandub korrutamiseks. Näiteks 24 = 10log(2)\*4 = 16

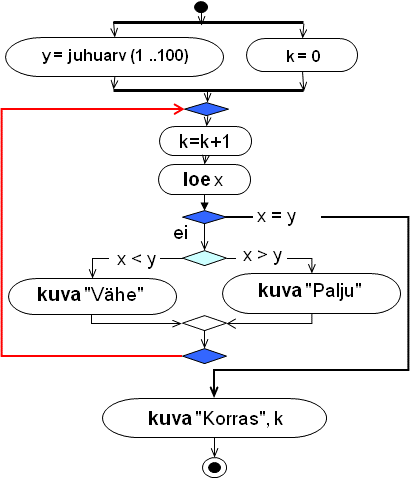
ab = 10log(a)\*b  ehk ab = eln(a)\*b 

Inimese keha pindala arvutamine (m mass (kg) ja L pikkus (cm)) saaks Scratchi skriptis sellise kuju:



# Arvu arvamine

Koostada skeemi järgi rakendus, mis võimaldaks kasutajal ära arvata vahemikus 1…100 juhuslikult valitud arvu, loendada tuleb ka katsete arv.



Rakenduse täitmisel tuleb anda kasutajale piisavalt infot (mida ta peaks sisestama, kas varemsisestatud arv oli liiga väike või suur, mitu arvamiskatset on tehtud jm). Rakenduse kirjeldus lisada projekti märkustesse või eraldi tekstifaili.

### Lisaülesanded:

* Lisada rakendusele mingi asjakohane animatsioon
* Kui vahemikus 1..100 genereeritud arv õnnestub kasutajal ära arvata kuni seitsme katsega, kuvada vastav teade või/ja käivitada mingi lisategevus (animatsioon).
* Lisada võimalus juhusliku arvu genereerimiseks mingis teises etteantud arvuvahemikus. Kasutajat tuleb min ja max väärtusest teavitada kas arvamise alguses või iga küsimuse korral.

# Ülesanne Kolm modelli

Laval on kolm ühesugust Modelli, kellel on kolm kostüümi. Modellid vahetavad iga paari sekundi tagant kostüüme ja hüppavad juhuslikult valitud asukohta laval. Mängija ülesanne on klõpsata hiirega ühte neist modellist siis, kui kõigil kolmel on seljas ühesugune kostüüm. Iga selline klõps suurendab mängija punktisummat 13 punkti võrra.

Kui modellil klõpsamise hetkel ei ole kõigil neil ühesugune kostüüm, siis väheneb mängija punktisumma 7 punkti võrra.

Laval on veel üks tegelane – kohtunik Kraps, kes teatab mängija punktisumma igast muutumisest.

**Vihje1**: Tegelaste kostüüminumbrite kontrolliga võiks tegeleda Kohtunik.

**Vihje2**: Märguande kostüümide vahetamiseks võiks ka anda Kohtunik.

**Lisaülesanne**: Kui kõigil modellidel on ühesugune kostüüm ja mängija ei klõpsa ühtegi modelli, siis väheneb mängija punktisumma 5 punkti võrra.

**Lisaülesanne**: Ühe muutumiskorra jooksul ei tohi jagada punkte (ei plusse ega miinuseid) rohkem kui üks kord.

# Ülesanne Mündivisked

Koostada Scratchis rakendus, mis modelleerib kulli ja kirja mängu.

**Tegevuse kirjeldus:**

Kasutajal on algul mingi hulk raha (küsitakse kasutajalt, kui palju tal raha on, või määratakse summa konstandina või juhuslikult).

Kasutajalt küsitakse ühekordse panuse suurust (maksimaalne panus peaks olema ette antud). Kindlasti tuleks kontrollida, kas kasutaja öeldud panus ei ületa maksimumpanust ja kas tal üldse nii palju raha on.

Visatakse münt. Mündi viskamine: juhuarv 1 või 2. 1 on kull ja 2 on kiri (või vastupidi).

Kui tuli kiri, saab kasutaja panuse topelt tagasi, muidu jääb ta mängupandud rahast ilma.

Kindlasti tuleb kasutajat iga mängukorra järel teavitada mündiviske tulemusest, tema võidust või kaotusest ning allesolevast rahasummast.

Mäng kestab seni, kuni kasutajal raha otsa saab.

Lisada rakendusele sobiv animatsioon.

**Lisaülesanne**: Kasutaja võiks ise valida, kumma tulemuse (kull või kiri) puhul ta võidab.

**Lisaülesanne**[[1]](#footnote-1): visete andmete (panus, mündiviske tulemus, rahajääk jm) säilitamiseks võiks kasutada loendeid.

# Ülesanne Mündivisked

Koostada Pythonis rakendus, mis modelleerib kulli ja kirja mängu.

**Tegevuse kirjeldus:**

Kasutajal on algul mingi hulk raha (küsitakse kasutajalt, kui palju tal raha on, või määratakse summa konstandina või juhuslikult).

Kasutajalt küsitakse ühekordse panuse suurust (maksimaalne panus peaks olema ette antud). Kindlasti tuleks kontrollida, kas kasutaja öeldud panus ei ületa maksimumpanust ja kas tal üldse nii palju raha on.

Visatakse münt. Mündi viskamine: juhuarv 1 või 2. 1 on kull ja 2 on kiri (või vastupidi).

Kui tuli kiri, saab kasutaja panuse topelt tagasi, muidu jääb ta mängupandud rahast ilma.

Kindlasti tuleb kasutajat iga mängukorra järel teavitada mündiviske tulemusest, tema võidust või kaotusest ning allesolevast rahasummast.

Mäng kestab seni, kuni kasutajal raha otsa saab.

**Lisaülesanne**: Kasutaja võiks ise valida, kumma tulemuse (kull või kiri) puhul ta võidab.

**Lisaülesanne**[[2]](#footnote-2): kasutaja pakkumised, visete tulemused, raha jääk jm salvestada iga viske järel loendi(te)sse, mis väljastatakse mängu lõppedes tabeli kujul *shelli* aknasse.

# Ülesanne Aja määramine

Koostada rakendus, milles kasutaja saab kontrollida, kui täpselt ta oskab määrata etteantud ajavahemikku. Ajavahemik (mitte liiga pikk) valitakse juhuslikult.

Piisavalt heaks loetakse tulemus, kus eksimus on väiksem kui 5% (võib olla ka suurem).

Iga katse järel väljastatakse teade „vajutasid … sek liiga vara“ või „vajutasid … sek liiga hilja“ ja kas tulemus on arvestatud või mitte.

Loendada tuleb katsete arv ja õnnestumiste arv.

Lisada sobiv animatsioon.

**Lisaülesanne:** Kontrollitava ajavahemiku saab kasutaja ise määrata.

**Lisaülesanne**: katsete andmete säilitamiseks võiks kasutada loendeid.

# Ülesanne Kolmnurk

Kolm punkti (spraiti) määravad kolmnurga. Kasutaja peab saama punkte laval nihutada.

Joonistada see kolmnurk.

Leida ja näidata laval pidevalt kolmnurga külgede pikkuseid, kolmnurga ümbermõõtu ja pindala.

**Vihje1**: joonistamiseks võiks olla eraldi sprait, mida laval ei näidata.

**Vihje2**: et joonis peab muutuma (punkte liigutatakse, muutub joone värv), tuleb pidevalt vana joonis kustutada ja joonistada uuesti.

**Lisaülesanne1**: Olenevalt kolmnurga pindalast peab astmeliselt (hüppeliselt) muutuma kolmnurka joonistava spraidi pliiatsi värv.

**Lisaülesanne2:** panna üks või mitu punkti juhuslikus suunas liikuma (ja põrkuma lava servadelt) ning ka peatuma mingi klahvivajutuse peale.

# Labürint

Koostada PacMan’i laadne mäng, kus rakenduse peategelane, keda kasutaja saab juhtida noole­klahvidega, korjab labürindis kullakange. Teda jälitab koll. Pacman läbi seinte ei saa, aga koll saab. Rakenduses tuleb leida kogutud kulla maksumus (kullakangide maksumused võivad olla erinevad). Mäng lõpeb, kui kogu kuld on kätte saadud või ületatakse etteantud aeg või koll saab Pacmani kätte.

#### Objektid:

**Labürint** – joonistatakse lava taustana; ühte värvi seinad, millest Pacmani läbi ei lasta

**Pacman** – võib luua joonistusredaktoriga, kaks (või rohkem) kostüümi, mida vahetatakse liikumisel. Liigutamine nooleklahvidega (4), ei tohi minna läbi seinte (kui puudutab seina värvi, liikumine peatub).



**Kullakangid** – luuakse joonistamisredaktoriga, omadus ’hind’ võiks olla lokaalne muutuja; koostatakse vajalikud skriptid, paljundatakse ühest valmis spraidist vajalik hulk objekte, paigutatakse labürinti laiali, iga kullakangi omadusele ’hind’ antakse erinevad väärtused. NB! Kulla hulk sõltub arendaja suvast, seetõttu on mõistlik, et kulla skriptis kontrollitakse, kas see sprait puudutab Pacmani

**Koll** – jälitab Pacmani (st. liigub kogu aeg Pacmani poole); kui tabab (puudutab), siis lõpeb mäng.

**Teade** – sprait, mille kostüümideks on teated (’Kogu kuld leitud’, ’Koll sai kätte’, ’Aeg läbi’). Selle (või lava) juurde võib koondada kõik üldised skriptid (muutujate algväärtustamised, ajaarvestus, kullaarvestus)

#### Muutujad:

***hp*** – Pacmani liikumise samm (määratakse ja kasutatakse Pacmani skriptides)

***hk*** – kolli liikumise samm (võib muuta kolli skriptis sõltuvalt tasemest või kogutud kullast)

***max\_kuld*** – otsitavate kullakangide arv. Väärtus omistatakse rakenduse loomise ajal.

***alles*** –alles olevate kullakangide arv (algväärtus ***max\_kuld***), muudavad kullakangide skriptid

***summa*** – kogutud kulla väärtuse jooksev summa (algväärtus 0), muudavad kulla skriptid

***max\_aeg*** – mängu max. aeg, määratakse rakenduse loomise ajal (või määrab kasutaja mängu alguses)

***aeg*** – mängu lõpuni jääv aeg: ***max\_aeg* – taimer** (algväärtus ***max\_aeg*)**

#### Võimalikud lõpuolukorrad: (seejärel peatatakse kõik skriptid)

Tehtud – kui kogu kuld on korjatud (alles = 0). Pacman rõõmustab, kuvatakse teade ’Kogu kuld leitud!’

Käes – teatab koll, kui Pacmani kätte saab. Koll rõõmustab, Pacman kurb, kuvatakse ’Koll sai kätte’

Aeg läbi – kui taimer > max\_aeg. Kuvatakse ’Aeg läbi!’

#### Olulisemad algoritmid:

|  |  |  |
| --- | --- | --- |
| **Kui vajutatakse nooleklahvi**  **Osuta suunas** {90; -90; 0; 180}  **Liigu** hp sammu  **Järgmine kostüüm**  **Kui puudutab** värvi (sein)  **Pööra** 180  **Liigu** hp **sammu**  **Lõpp kui** | **Pacman**  Pacmani nooleklahvi­skriptidel on üsna suur ühisosa, mis oleks mõistlik vormistada eraldi skriptina. | **Kui vajutatakse nooleklahvi** **…**  **Osuta suunas** {90; -90; 0; 180}  **Teata** **tee samm**  **tee samm**  **Liigu** hp sammu  **Järgmine kostüüm**  **Kui puudutab** värvi (sein)  **Pööra** 180  **Liigu** hp **sammu**  **Lõpp kui** |

Kulla ja kolli skriptid ning algseaded ja ajaarvestus (ükskõik millise spraidi juures) käivituvad mängu alguses.

|  |  |
| --- | --- |
| **Protseduur** **ajaarvestus**  **taimer algseisu**  **Kordus** **kuni** taimer > max\_aeg  aeg = taimer  **Lõpp kordus**  **teata** Aeg läbi! **ja oota**  **Peata kõik** | **Protseduur algseaded**  alles = max\_kuld  summa = 0  **oota kuni** alles = 0  **teavita tehtud**  **Peata kõik** |
| **Protseduur** **koll**  **Kordus** **kuni** puudutab Pacman  **osuta** Pacman  **liigu** hk sammu  **oota** …  **Lõpp kordus**  **teata** Käes! **ja oota**  **Peata kõik** | **Protseduur** **kuld**  **näita**  **Oota kuni** puudutab Pacman  summa = summa + hind  alles = alles -1  **peida** |

Lisaülesandeid:

* liikuvad uksed ja muud takistused
* rohkem kolle erineva käitumisega
* mängul võiks olla mitu taset (kolli liikumine muutub kiiremaks vm)
* Pacman liigub ise, nooleklahvidega saab muuta vaid suunda
* kui mäng ei käi, ei tohi Pacman liikuda.

# Tekst tagurpidi - parandada

Koostada rakendus kasutajalt küsitud teksti esitamiseks tagurpidi.

1. sümbolid vastupidises järjestuses
2. sõnad vastupidises järjestuses.

Tekstist on võimalik eraldada sümboleid t_symbol,

tekste saab ühendada t_yhenda,

saab teada teksti pikkuse (sümbolite arvu) t_pikkus.

|  |  |
| --- | --- |
| Ühest tekstist tuleb eraldada sümbolid (korrata iga sümboli jaoks, alustada võib nii teksti algusest kui lõpust) ja lisada teise teksti nii, et järjestus oleks vastupidine.  Andmeteks (säilitamist vajavad väärtused) on: kasutajalt küsitud tekst **tekst1**, tulemus (koostatav tekst) **tekst2**, jooksva sümboli järjenumber **nr**. Muutujatesse võivad olla salvestatud ka teksti pikkus **mitu** ja jooksev sümbol **s**. | **protseduur Tagurpiditekst**  mitu = pikkus(tekst1)  tekst2 = ’’  nr = mitu  **kordus** mitu **korda**  s = tekst1 nr-s täht  tekst2 = ühenda tekst2 ja s  nr = nr – 1  **lõpp kordus** |

Teksti sõnade kaupa töötlemisel oleks hea eraldada tekstist sõnad (salvestada iga sõna eraldi muutujas). Sõnade eraldajaks on tühik. Sõnade arv tekstis ei ole teada, see on muutuv.

Probleemi saab lahendada loendi kasutamisega. Loend – muutujate järjestatud kogum.

|  |  |
| --- | --- |
| Nii nagu teisedki muutujad, tuleb Scratchis tekitada loendid rakenduse loomisel.  Programselt saab väärtusi lisada korraldusega l_lisa ja eemaldada käsuga l_eemaldakoik (kas kõik või ükshaaval järje­numbri abil).  Teada saab väärtuste arvu loendis l_pikkus ja etteantud järjenumbriga elemendi väärtuse l_el_nr.  Teksti jaotamine sõnadeks (enne luua loend **sõnad**):  Tekstist **tekst1** eraldatakse sümbol **s** (korrata iga sümboli jaoks). Iga kord kontrollitakse, kas eraldatud sümbol on tühik. Kui tegemist pole tühikuga, lisatakse see **sõna**-le. Kui eraldatud sümbol on tühik, siis seda ei lisata vaid muutuja **sõna** väärtus lisatakse loendile **sõnad** ja **sõna** tühjendatakse. Lõpuks lisatakse loendile veel kord **sõna** väärtus (sümbolid pärast viimast tühikut). | **protseduur Tagurpidi2**  mitu = pikkus(tekst1)  tekst2 = ’’  sõna = ’’  nr = 1  **kordus** mitu **korda**  s = tekst1 nr-s täht  **kui** s = ’ ’  Lisa sõnad-le sõna  sõna = ’’  **muidu**  sõna = ühenda sõna ja s  **lõpp kui**  nr = nr + 1  **lõpp kordus**  tekst2 = sõnad(1)  nr = 1  **kordus** pikkus(sõnad)-1 **korda**  Nr = nr + 1  Tekst2 = ühenda sõnad(nr) ja (ühenda ’ ’ ja tekst2)  **lõpp kordus** |

**NB!** Et mõlemad protsessid võivad töötada paralleelselt, võiks abimuutujate (nr, s) segaminimineku vältimiseks kasutata erinevate spraitide lokaalseid muutujaid.

**Märkus.** Loendi kasutamine ei ole selle ülesande juures hädavajalik. Sõnade loendisse paigutamise asemel võib esimesest tekstist eraldatud sõna kohe lisada tulemusele.

# Tekst tagurpidi (ka loendi abil) muuta!

Koostada rakendus kasutajalt küsitud teksti esitamiseks tagurpidi.

1. sümbolid vastupidises järjestuses
2. sõnad vastupidises järjestuses.

Tekstist on võimalik eraldada sümboleid t_symbol,

tekste saab ühendada t_yhenda,

saab teada teksti pikkuse (sümbolite arvu) t_pikkus.

|  |  |
| --- | --- |
| Ühest tekstist tuleb eraldada sümbolid (korrata iga sümboli jaoks, alustada võib nii teksti algusest kui lõpust) ja lisada teise teksti nii, et järjestus oleks vastupidine.  Andmeteks (säilitamist vajavad väärtused) on: kasutajalt küsitud tekst **tekst1**, tulemus (koostatav tekst) **tekst2**, jooksva sümboli järjenumber **nr**. Muutujatesse võivad olla salvestatud ka teksti pikkus **mitu** ja jooksev sümbol **s**. | **protseduur Tagurpiditekst**  mitu = pikkus(tekst1)  tekst2 = ’’  nr = mitu  **kordus** mitu **korda**  s = tekst1 nr-s täht  tekst2 = ühenda tekst2 ja s  nr = nr – 1  **lõpp kordus** |

Teksti sõnade kaupa töötlemisel oleks hea eraldada tekstist sõnad (salvestada iga sõna eraldi muutujas). Sõnade eraldajaks on tühik. Sõnade arv tekstis ei ole teada, see on muutuv.

Probleemi saab lahendada loendi kasutamisega. Loend – muutujate järjestatud kogum.

|  |  |
| --- | --- |
| Nii nagu teisedki muutujad, tuleb Scratchis tekitada loendid rakenduse loomisel.  Programselt saab väärtusi lisada korraldusega l_lisa ja eemaldada käsuga l_eemaldakoik (kas kõik või ükshaaval järje­numbri abil).  Teada saab väärtuste arvu loendis l_pikkus ja etteantud järjenumbriga elemendi väärtuse l_el_nr.  Teksti jaotamine sõnadeks (enne luua loend **sõnad**):  Tekstist **tekst1** eraldatakse sümbol **s** (korrata iga sümboli jaoks). Iga kord kontrollitakse, kas eraldatud sümbol on tühik. Kui tegemist pole tühikuga, lisatakse see **sõna**-le. Kui eraldatud sümbol on tühik, siis seda ei lisata vaid muutuja **sõna** väärtus lisatakse loendile **sõnad** ja **sõna** tühjendatakse. Lõpuks lisatakse loendile veel kord **sõna** väärtus (sümbolid pärast viimast tühikut). | **protseduur Tagurpidi2**  mitu = pikkus(tekst1)  tekst2 = ’’  sõna = ’’  nr = 1  **kordus** mitu **korda**  s = tekst1 nr-s täht  **kui** s = ’ ’  Lisa sõnad-le sõna  sõna = ’’  **muidu**  sõna = ühenda sõna ja s  **lõpp kui**  nr = nr + 1  **lõpp kordus**  tekst2 = sõnad(1)  nr = 1  **kordus** pikkus(sõnad)-1 **korda**  Nr = nr + 1  Tekst2 = ühenda sõnad(nr) ja (ühenda ’ ’ ja tekst2)  **lõpp kordus** |

**NB!** Et mõlemad protsessid võivad töötada paralleelselt, võiks abimuutujate (nr, s) sassiminemise vältimiseks kasutata erinevate spraitide lokaalseid muutujaid.

**Märkus.** Loendi kasutamine ei ole selle ülesande juures hädavajalik. Sõnade loendisse paigutamise asemel võib esimesest tekstist eraldatud sõna kohe lisada tulemusele.

# Autod 2

Koostada mitme auto võidusõit. Osalevate autode arv ei ole piiratud.

Kohtunik teatab sõidu alustamisest ja kuulutab lõpuks välja võitja.

Enne võistluse algust võiks kasutaja ennustada, kes võidab. Kui võistlus on lõppenud, tuleb teatada, mitmendana lõpetas kasutaja poolt võitjaks ennustatud auto.

|  |  |
| --- | --- |
| **protseduur** **Start**  x = -230  **kordus** ringe **korda**  **kordus kuni** x > 240  **liigu** 10 kuni 20 sammu  **lõpp kordus**  **peida**  **oota** 1 kuni 3 sek  x = -230  **näit**a  aeg = taimer  **lõpp kordus**  **lisa loendisse** | Autode kiirus (sammu pikkus) valitakse juhuslikult igal sammul või määratakse juhuslikult sõidu algul.  Võistluse lõpetanud auto nimi ja aeg kirjutatakse vastavatesse loenditesse.  Võistluse võitja nimi ja aeg on loendites esimesed.  **Andmed:** objektid **autod** ja **kohtunik**, loendid **Ajad** ja **Autod**, muutujad **ringe**, **ennustus**, **j** (järjenumber); iga auto lokaalsed muutujad: **nimi**, **ring**, **kiirus** |
| j = 1  **kordus** Autod.pikkus **korda**  **kui**  ennustus = Autod(j) **siis**  **stopp**  **lõpp kui**  j = j +1  **Lõpp kordus** | Kasutaja poolt võitjaks ennustatud auto leidmine võistluse lõppjärjestuses.  Selle võiks vormistada eraldi skriptina (sees on katkestus; Scratch ei võimalda korduse katkestamist, seega peatatakse skripti töö). |

### Lahendamist vajavad:

* Kuidas kasutaja ennustab võitja (kohtunik küsib ja kasutaja vastab; hiireklõps valitud autol; vm)
* Kuidas kuulutatakse välja võitja (kohtunik ütleb võitja nime; kohtunik läheb võitja juurde; võitja teeb auringi, pöörleb või hüpleb; vm)
* Kuidas teatatakse kasutaja poolt võitjaks ennustatu koht ja aeg; mis saab, kui ennustus osutus õigeks.

# Vestlus

Koostada rakendus kahe tegelase dialoogi esitamiseks. Eelnevalt moodustada kaks loendit, kuhu salvestada osaliste tekstid.

### Kirjeldus:

Dialoogi alustab üks osalistest öeldes esimese teksti oma loendist, teine vastab oma loendi esimese repliigiga. Tegevust korratakse nii kaua, kui kõik tekstid on ära öeldud. Järjenumbri meelespidamiseks tuleks kasutada muutujat. Ühe spraidi skriptist algab tegevus, teise spraidi teksti esitamiseks väljastatakse teade ja oodatakse, kuni vastus on näidatud.

**Andmed:** kaks tegelast (spraiti), loendid tekstid1 ja tekstid2, j (loendur).

Esialgu võib kõiki tekste kuvada sama aja jooksul, hiljem proovida teksti kuvamise aeg panna sõltuma teksti pikkusest:

aeg = baasaeg + repliigi\_pikkus / k.

Sobiva baasaja ja k väärtuse (5 … 20) saab leida proovimise teel.

|  |  |  |  |
| --- | --- | --- | --- |
| girl4-sitting | j = 1  **kordus** tekstid1.pikkus **korda**  **ütle** tekstid1(j)  **teavita vasta**  j = j + 1  **tagasi** | Skript **vasta**:  **ütle** tekstid2(j) | girl5 |

### Lisaülesanne:

Kõik tekstid võivad olla ka ühes loendis. Sel juhul on paarituarvuliste järjenumbritega repliigid ühe tegelase jaoks ja paarisarvuliste järjenumbritega repliigid teise omad.

Kas nii saaks koostada ka suurema tegelaskonnaga etenduse?

# Sõnastik. Keeleõppe test

Antud on sõnastik – kaks loendit – ühes eestikeelsed, teises inglisekeelsed sõnad. Soovi korral võib mingi hulga sõnu loodud loenditesse importida failidest eesti.txt ja inglise.txt.

Koostada rakendus kasutaja teadmiste kontrollimiseks. Loendada tuleb õigete vastuste arv. Koostada kaks skripti: eestikeelsete sõnade inglisekeelsete vastete küsimine ja inglisekeelsete sõnade eestikeelsete vastete küsimine. Rakendust arendada järk-järgult.

NB! õ ja täpitähtedega sõnu ei saa kasutada, seepärast tuleb eestikeelsete sõnadega loend pärast sõnade lisamist üle vaadata ja teha vajalikud parandused.

### Analüüs:

**Tegevuse kirjeldus:** Kasutajalt küsitakse eestikeelsete sõnade hulgast juhuslikult valitud sõna. Kasutaja peab sisestama õige inglisekeelse sõna. Skript peab kontrollima, kas vastus on õige, vajadusel suurendama õigete vastuste arvu.

Kui eelnev toimib, lisada kordus ja otsustada, millal peaks küsimine lõppema.

**Andmed:** loendid **eesti** ja **inglise**, **j** (järjenumber), **õigeid**.

õigeid = 0

**kordus**

j = juhuarv (1 kuni eesti.pikkus)

**küsi** … **loe** vastus

**kui**  vastus = inglise(j) **siis**

õigeid = õigeid + 1

ütle „Õige vastus!“

**muidu**

ütle „Vale vastus!“

**lõpp kui**

**lõpp kordus**

### Lisaülesandeid:

* realiseerida ka inglisekeelse sõna järgi eestikeelse sõna küsimine
* loendada ka valede vastuste arv
* vale vastuse korral küsitakse sama sõna uuesti
* sama tegevust kasutada küsimustiku juures, loendites küsimused-vastused
* muuta skripti nii, et sõnu (küsimusi) küsitaks järjest – alates esimesest kuni viimaseni.
* Lindistada helindid: küsimused „Kuidas on inglise keeles“, „Kuidas on eesti keeles“ ja loendites olevad sõnad (iga sõna eraldi). Moodustada loend, milles on lindistatud helindite nimed. Esitada küsimus nii tekstina ekraanil kui heli abil.

# Sõnastik. Tõlkimine

Antud on sõnastik – kaks loendit – ühes eestikeelsed, teises inglisekeelsed sõnad. Soovi korral võib mingi hulga sõnu loodud loenditesse importida failidest eesti.txt ja inglise.txt.

Koostada rakendus kasutaja antud sõnade tõlkimiseks ühest keelest teise.

NB! õ ja täpitähtedega sõnu ei saa kasutada, seepärast tuleb eestikeelsete sõnadega loend pärast sõnade lisamist üle vaadata ja teha vajalikud parandused.

### Analüüs:

**Tegevuse kirjeldus:** Kasutajalt küsitakse eestikeelne sõna. Seejärel tuleb ükshaaval järjest vaadata iga eestikeelset sõna (kordus ’loendi pikkus’ korda) ja võrrelda antud sõnaga. Kui leitakse, tuleb inglisekeelsete sõnade hulgast võtta sama järjenumbriga sõna (st. muutujat järjenumbri jaoks – loendur) ja öelda see kasutajale. Kui kogu loend on üle vaadatud, aga sõna ei leitud, tuleb ka seda teatada kasutajale.

**Andmed:** loendid **eesti** ja **inglise**, **j** (järjenumber), **sõna** (küsitud sõna).

**NB!** Kui tõlkimised eesti-inglise ja inglise-eesti võivat töötada paralleelselt, võiks järjenumbri ja küsitud sõna jaoks kasutada erinevaid muutujaid või erinevate spraitide lokaalseid muutujaid (sel juhul võivad olla sama nimega).

**loe** sõna

j = 1

**kordus** eesti.pikkus **korda**

**kui**  sõna = eesti(j) **siis**

**kuva** inglise(j)

**stopp**

**lõpp kui**

j = j + 1

**lõpp kordus**

**kuva** "Sellist sõna minu sõnastikus ei ole"

**NB!** Kui väärtused loendis, milles toimub sõna otsimine, oleks järjestatud, saaks kasutada keerukamaid otsimisalgoritme, mis töötaksid kiiremini.

Lisaülesanne. Lisada võimalus küsida vale vastuse korral sama sõna korduvalt.

# Algoritme loenditega

|  |  |  |
| --- | --- | --- |
| Luua kaks loendit: Puud ja Hinnad (või Nimi ja Vanus, Sportlane ja Tulemus vm) ning sisestada väärtused.  Koostada skriptid, mis leiavad:   * hindade aritmeetilise keskmise * maksimaalse hinna ja sellele vastava puu nimetuse * etteantud puuliigi hinna   Arvestada võimalusega, et skriptid võivad töötada paralleelselt. | **Puud** | **Hinnad** |
| haab | 1200 |
| kask | 1700 |
| kuusk | 2300 |
| lepp | 1600 |
| mänd | 2100 |
| saar | 3100 |
| tamm | 3500 |
| vaher | 3300 |

|  |  |
| --- | --- |
| Hindade **aritmeetilise keskmise** leidmisel tuleb kõigepealt leida hindade summa, liidetavate arvuks on loendi pikkus (väärtuste arv).  Sisend: loend **Hinnad**, n (väärtuste arv loendis)  Väljund: keskmine  Abi: S (summa), i (järjenumber) | …  S = 0  i = 1  **kordus** n **korda**  S = S + Hinnad(i)  i = i +1  **lõpp kordus**  keskmine = S / n  … |
| **Suurima väärtuse** leidmisel peab muutuja **max** väärtuseks saama suurim loendi väärtustest. Algul eeldame, et esimene väärtustest on suurim, seejärel võrdleme **max** kõigi ülejäänud väärtustega. Kui leidub mõni suurem, omistatakse see muutujale **max**. Meelde tuleb jätta ka suurima asukoht. Muutuja **nr** saab uue väärtuse iga kord, kui muutub **max**.  Sisend: loendid **Puud** ja **Hinnad**, n (väärtuste arv loendis)  Väljund: **max**, **puu**  Abi: **k** (järjenumber), **nr** (suurima väärtuse järjenumber loendis) | …  max = Hinnad(1)  nr = 1  k = 1  **kordus** n - 1 **korda**  k = k +1  **kui**  Hinnad(k) > max **siis**  max = Hinnad(k)  nr = k  **lõpp kui**  **lõpp kordus**  puu = Puud(nr)  … |
| Etteantud puuliigi hinna leidmisel küsitakse kasutajalt puuliigi nimetus **liik**. Seejärel tuleb järjest vaadata kõik puu liigid loendis **Puud** (kordus ’loendi pikkus’ korda) ja võrrelda antud liigiga.  Kui selline leitakse, tuleb loendist **Hinnad** võtta sama järjenumbriga **j** väärtus. Kui kogu puuliikide loend on üle vaadatud, aga antud liiki ei leitud, tuleb ka seda teatada kasutajale.  Sisend: loendid **Puud** ja **Hinnad**, n (väärtuste arv loendis), **liik**  Väljund: **hind**  Abi: **j** (järjenumber) | …  **loe** liik  j = 1  **kordus** n **korda**  **kui**  liik = Puud(j) **siis**  hind = Hinnad(j)  **stopp**  **lõpp kui**  j = j + 1  **lõpp kordus**  hind = 0  **kuva** "Sellist puuliiki ei ole"  … |

# Ülesanne Päringud loenditest

Luua Scratchi rakenduses kaks loendit: nimed ja palgad. Nimed võib loendisse kirjutada või importida failist [nimed.txt](http://rlpa.ttu.ee/scratch/h2/nimed.txt), palkade väärtuste hankimiseks koostada eraldi skript, mis käivitub u-tähele vajutades ja tekitab loendisse uued väärtused juhuarvudest vahemikus 400 kuni 3000.

Koostada skriptid:

1. keskmise palga leidmine
2. loenditesse s\_nimed ja s\_palgad inimeste andmete eraldamine, kelle palk on suurem kui keskmine palk
3. loendid nimed ja palgad tuleb sortida palkade kahanemise järjestuses

**Soovitusi:**

* Esialgu võiks rakendust katsetada väiksema hulga väärtustega (15-20 nime ja palka).
* Uute andmete genereerimine ja tulemuste leidmine peaks kindlasti olema eraldi käivitatavad tegevused.
* Loendite sortimise selgitamiseks on demod: [Jadasortimine](http://rlpa.ttu.ee/scratch/d2/sort_jada_demo.html) ja [Valiksortimine](http://rlpa.ttu.ee/scratch/d2/sort_valik_demo.html).
* Loendite sorteerimiseks palkade järgi tuleb ka nimed ümber tõsta vastavalt palgaväärtuste ümber­paigutamisele

# Ülesanne Päringud tekstifailist (Python)

Failis [tabel.txt](http://rlpa.ttu.ee/python/h/tabel.txt) on andmed paljude inimeste lotovõitude kohta: nimi, isikukood, elukoht, võitmise kuu ja võidetud summa. Iga isiku andmed (kirje) on eraldi reas (rea lõpus \n – reavahetussümbol). Ühe isiku andmeväljad on eraldatud tabulaatoriga \t.

Ülesanne

Lugeda andmed failist loendi(te)sse.

Leida järgmised väljavõtted ja kokkuvõtted:

1. Moodustada uus fail inimeste andmetest, kes võitsid mingil kuul (kuu küsida kasutajalt).
2. Moodustada uus fail inimeste andmetest, kes elavad etteantud asumis (asula nimi küsida kasutajalt).
3. Leida ja väljastada võitude summad tabeli kujul shelli aknas
   1. kuude lõikes (kasutada loendeid)
   2. elukohtade lõikes (proovida kasutada sõnastikku)

Teha veel vähemalt kaks kokkuvõtet või päringut, tunnused ja väärtused mõelda ise.

Näiteks: Nimekiri meestest, kelle sünnipäev on lotovõidu saamisega samas kuus.

Tegevuste selgitused tuleb kirjutada kommentaaridena programmi teksti.

Sarnaseid näiteid vt. <http://rlpa.ttu.ee/python/h/paringud.pdf>

**Info isikukoodi kohta:**

Isikukoodis on täpselt 11 numbrit.

Isikukood SAAKKPPxxxC sisaldab:

1. number S näitab sugu ja sünnisajandit.

1, 3, 5 - mehed, 2, 4, 6 - naised;

1 ja 2 - sündinud 18xx,

3 ja 4 - sündinud 19xx,

5 ja 6 - sündinud 20xx

2. ja 3. number AA on sünniaasta kaks viimast numbrit (01 ... 99)

4. ja 5. number KK on sünnikuu (01 ... 12)

6. ja 7. number PP on sündimise kuupäev (01 ... 31)

Viimane number C on kontrollnumber, mis arvutatakse kümne eelneva numbri alusel.

# Loendite kasutamine spraitide asukoha juhuslikul määramisel

Kui rakenduses tuleb spraidid paigutada juhuslikult, aga kindlaksmääratud kohtadele, on üks võimalus kasutada loendite abi järgmiselt.

### Andmed:

loendid **sprait**, **X**, **Y**; muutuja **j** (järjenumber)

Loenditesse **X** ja **Y** salvestatakse võimalike asukohtade koordinaadid: iga paar (xi, yi) vastab ühele asukohale.

### I etapp:

Algul tuleb **loend** **sprait** tühjendada, seejärel peavad kõik spraidid, mis juhuslikult lavale paigutatakse, kirjutama mingi juhusliku ajavahemiku järel sellesse loendisse kas

a) teatise nime, millele nad reageerivad

või

b) mingi sellele spraidile ainuomase väärtuse (spraidi nimi ei ole kahjuks omadus, mida on võimalik skriptis lugeda, aga selleks otstarbeks võib luua kõigi lavale paigutatavate spraitide jaoks samanimelised lokaalsed muutujad (näiteks ID), mille väärtus on igal spraidil unikaalne).

### II etapp:

a) Loendist **sprait** võetakse järjekorras (loendur **j** hoolitseb selle eest) teatised ja teatatakse need. Igale teatisele peab reageerima ühe spraidi skript paigutades spraidi järjenumbriga **j** määratud koordinaatidel **X**(j), **Y**(j)

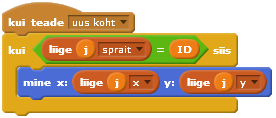
või

b) j väärtuse muutumise järel tuleb teatada „uus koht“. Sellele peavad reageerima kõikide asukohta muutvate spraitide skriptid, mis kontrollivad, kas loendi sprait väärtus järjenumbriga j on sama, mis selle spraidi lokaalse muutuja ID väärtus. Asukohta muudab ainult see sprait, mille muutuja väärtus sobis.

|  |  |
| --- | --- |
| Tegevuste „mootor“ võib olla üks lava skriptidest:  **tühjenda** loend sprait  **teade** segamini - *teatised või väärtused kirjutatakse loendisse*  j = 0  **korda** sprait.pikkus **korda**  j = j + 1  **teade** (sprait(j) või uus koht)  **lõpp kordus**  sun | **Variant a)** |

**Variant b)**





# Kasutajaplokid (protseduurid)

Kasutajaplokid on võimalus suurte samalaadsete skriptide asemel kasutada omaloodud korduvkasutatavaid pinuplokke, mida saab käivitada erinevate sisendandmetega. Ühtlasi on selline plokk Scratchi ainus võimalus tegevuste käivitamiseks ilma pidevalt ekraanikuva värskendamata. See on kasulik näiteks suuremahuliste arvutuste ja jooniste tegemisel.

Teistes programmeerimiskeeltes oleks selliste plokkide vasteks protseduur, funktsioon või meetod.

Kasutajaplokk on kasutatav ainult selle spraidi skriptides, mille juurde kuulub selle kirjeldus. Ploki kirjelduse võib muidugi kopeerida ka teisele spraidile.

Kasutajaplokk koosneb **päisest** ja **kehast**.

**Päise** loomiseks tuleb valida grupist **Lisaplokid** nupp **Loo plokk**. **Päis** sisaldab käsuploki **nime** ja **parameetreid** (muutujad, mis saavad oma väärtused ploki poole pöördumisel), **märgendeid** (aitavad selgitada parameetrite tähendust) ning kas plokki töö jooksul ekraanikuva uuendatakse või mitte. Need saab määrata klõpsates **Valikud**. Parameetreid võivad olla arvud, tekstid või tõeväärtused.

Päise alusel moodustatakse gruppi **Lisaplokid** uus plokk.

Ploki **keha** kujutab endast peaaegu tavalisel viisil loodud skripti. Eripäraks on parameetrite kasutamine, need lohistatakse päisest vajalikku kohta, nagu tehakse muutujatega tavalistes skriptides. Päisesse jääb parameeter alles.

Kasutajaplokkides on võimalik rekursioon (st plokk võib pöörduda iseenda poole).

Ülesanne:

Koostada kujundite joonistamiseks kasutajaplokid, valida nende jaoks sobivad parameetrite komplektid.

Joonistada loodud plokkide abil pilt (maja, päike jm). Kasutada korduseplokke.

Kasutajaplokid võiksid joonistada:

Joon. Parameetriteks alg- ja lõpp-punkti koordinaadid.

Joon. Parameetriteks alguspunkti koordinaadid, suund ja pikkus.

Ristkülik. Parameetriteks alguspunkti koordinaadid, laius ja kõrgus. Ristküliku küljed joonistatakse paralleelselt horisontaal- ja vertikaalteljega.

Ristkülik. Parameetriteks alguspunkti koordinaadid, laius ja kõrgus ning esimese joonistatava külje pöördenurk.

Ring (tegelikult korrapärane hulknurk). Parameetriteks keskpunkti koordinaadid, raadius ja nurkade arv.

Trapets (korrapärane ja täisnurkne). Parameetriteks alguspunkti koordinaadid, aluste pikkused ja kõrgus.

Romb. Parameetriteks alguspunkti koordinaadid, külje pikkus, pööre ja nurk

Selliste plokkide parameetriteks võivad olla ka joone paksus, värv jm

Mõned lihtsamad kujundid (ristkülik, ring) võiksid olla ka täidetud (joonistada tihedalt jooned).

Joonistamise kiirendamiseks võiks sisse lülitada turborežiimi (shift + hiireklõps rohelisel lipul) või määrata ploki päises, et selle täitmisel ei uuendata ekraanikuva.

# Rekursioon

Protseduur (funktsioon) on rekursiivne, kui ta kutsub iseennast välja (sageli selleks, et lahendada protseduurile antud probleemi kergem variant).

Et rekursioon (enesesse pöördumine) mingil hetkel lõppeks, peab rekursiivses funktsioonis toimuma hargnemine (valik) sõltuvalt protseduuri sisendandmetest.

Vähemalt üks haru peab olema ilma rekursiivse väljakutseta – seda nimetatakse rekursiooni baasiks.

Haru, kus toimub rekursiivne pöördumine, nimetatakse rekursiooni sammuks.

Paar näidet rekursiivsetest skriptidest

|  |  |
| --- | --- |
| **Täidetud ring** |  |

**Puu**



# Kloonimine

Kloonimine on võimalus rakenduse töötamise ajal luua spraitidest kloone (teisikuid) programmi abil.

Kloonid pärivad oma „esivanema“ kõik omadused, skriptid, kostüümid ja helid ning võivad seejärel muutuda, olles seega eraldiseisvad üksused.

Ühes Scratchi projektis võib olla kuni 300 klooni.

Kloonimisega seotud käsuplokid (3) asuvad grupis Juhtimine.



Klooni saab luua nii endast (sellest spraidist, mille skriptis on kloonimiskäsk) kui ka igast teisest spraidist.

Ka kloonid võivad luua uusi kloone.

 Selle päiseplokiga skriptid alustavad uue klooni loomise järel ja mõjutavad ainult värskelt loodud klooni, mitte esivanemat (seda tekitanud spraiti) ega varemloodud kloone.

Kloonid eemaldatakse rakendusest rohelist lippu või punast nuppu vajutades (lava ülaservas) ning käsuga [eemalda see kloon]. 

Ülesanne

Koostada rakendus, kus kasutaja ülesandeks on lavale laiali pillutud objektid neil hiirega klõpsates võimalikult kiiresti kokku korjata. Objektide arvu määrab kasutaja mängu alguses. Loendada tuleb klõpsatud spraitide arv ja fikseerida selleks kulutatud aeg.

Lisada rakendusele sobiv animatsioon.

**Objektid:**

**Asi** − sprait, millest mängu algul tekitatakse vajalik hulk kloone (võib olla mitu kostüümi).

Sprait, kes küsib kasutajalt objektide arvu ja teatab pärast tulemuse (võib ka puududa).

**Muutujad:**

**max\_leida** − objektide arv algul

**leitud** − klõpsatud objektide arv (loendur)

**aeg** − muutuja, mille monitori saab kasutada aja kuvamiseks laval

**Tegevuse kirjeldus:**

Selgitatakse muutuja **max\_leida** väärtus. Muutuja **leitud** väärtus on alguses null.

**Max\_leida** korda korratakse: spraidist asi tekitatakse kloon. Kloon valib endale juhusliku kostüümi ja paigutub juhuslikku kohta laval.

Kui klõpsatakse mõnel spraidi **asi** kloonil, liidetakse muutujale **leitud** 1.

Mäng kestab seni, kuni kõik kloonid on klõpsatud.

Lisaülesanne:

kui klõpsatud objekti kohe ei peideta, ei tohi sellel klõpsamist teist korda arvesse võtta (olekumuutuja peaks olema spraidi lokaalne muutuja).

1. Loendite kasutamine on õppematerjali järgmises teemas [↑](#footnote-ref-1)
2. Loendite kasutamine on õppematerjali järgmises teemas [↑](#footnote-ref-2)