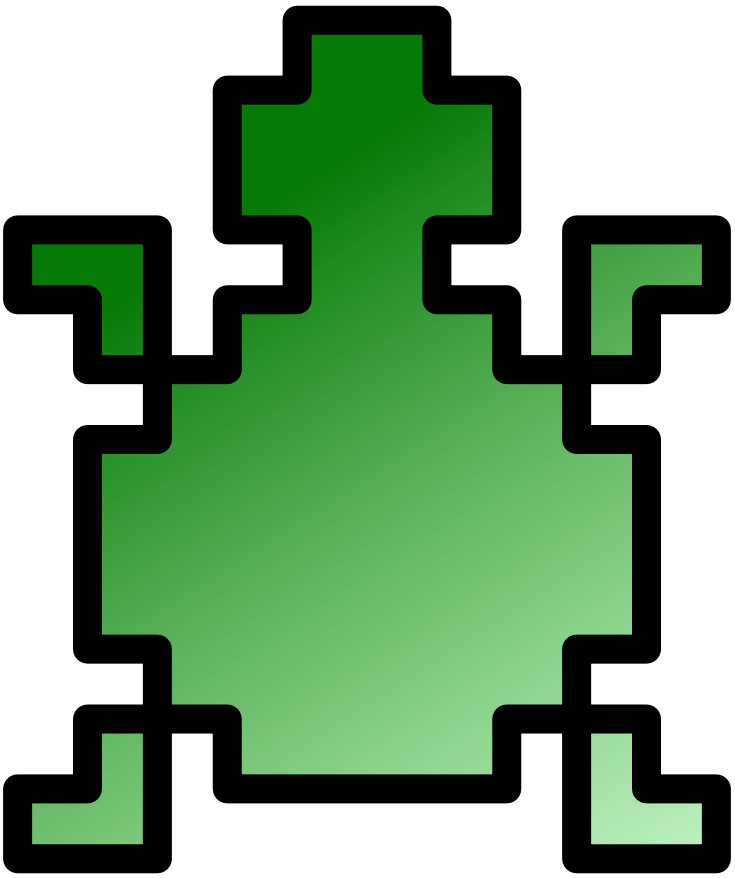
KTurtle'i käsiraamat

Cies Breijs
Anne-Marie Mahfouf
Mauricio Piacentini
Tõlge eesti keelde: Marek Laane



Sisukord

Tabelid

Kokkuvõte

KTurtle on programmeerimise õpikeskkond, mille eesmärk on muuta programmeerimine võimalikult lihtsaks kõigile huvilistele. Sel eesmärgil on KTurtle muutnud programmeerimistööriistad kättesaadavaks otse kasutajaliideses. Rakenduses kasutatakse programmeerimiskeelt TurtleScript, mis lubab käske ka tõlkida.

Peatükk 1

Sissejuhatus

KTurtle on haridusalane programmeerimiskeskkond, mis kasutab programmeerimiskeelt TurtleScript, mille puhul on eeskuju võetud programmeerimiskeelest Logo. KTurtle'i mõte on muuta programmeerimine nii lihtsaks ja arusaadavaks, kui vähegi võimalik. Nii sobib KTurtle ka lastele matemaatika, geomeetria ja ... programmeerimise õpetamiseks. TurtleScripti põhiomaduseks on see, et käske saab tõlkida ka programmeerija emakeelde.

KTurtle on saanud nime ingliskeelse sõna 'turtle' ehk 'kilpkonn' järgi, sest just kilpkonnal on meie programmeerimiskeskkonnas keskne osa: kasutaja programmeerib kilpkonna TurtleScripti käskudega joonistama lõuendile vajalikku pilti.

1.1 Mis on TurtleScript?

KTurtle'is kasutatav programmeerimiskeel TurtleScript on tugevasti inspireeritud programmeerimiskeele Logo mitmest põhimõttelisest kontseptsioonist. Logo esimese versiooni lõi Seymour Papert 1967. aastal MIT tehisintelligentsi laboris, võttes aluseks programmeerimiskeele LISP. Sealtpeale on ilmavalgust näinud arvukalt Logo versioone. Aastaks 1980 olid olemas Logo MSX, Commodore'i, Atari, Apple II ja IBM PC versioonid. Peamiselt olid need mõeldudki õppevahendiks. MIT haldab seniajani Logo veebilehekülge, mis sisaldab mitmeid selle keele populaarseid versioone.

TurtleScript on ühes mõttes sarnane teiste Logo keele variantidega: ka see võimaldab tõlkida käske, et õppurid saaksid neid kasutada oma emakeeles. Sellest on palju abi eriti nooremate laste puhul, kelle inglise keel ei pruugi olla kuigi hea, aga käepärast on see ka vanematele inimestele. Lisaks sellele on KTurtle'il veel rohkelt omadusi, mis muudavad rakenduse kasutamise hõlpsaks neile, kelle teadmised programmeerimisest on alles tillukesed.

1.2 KTurtle'i omadused

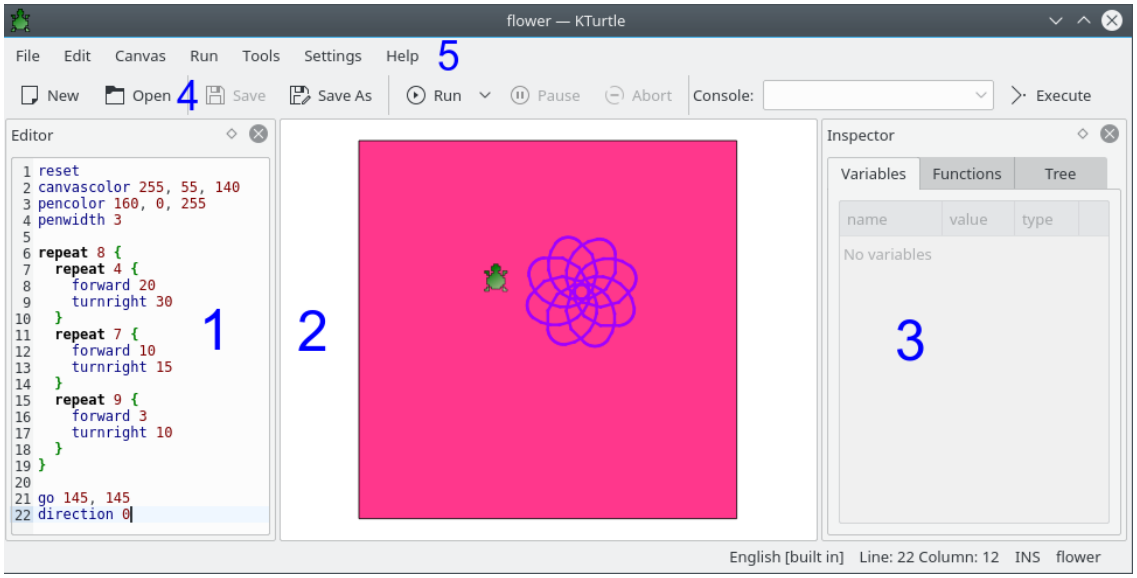
KTurtle pakub mitmeid vahvaid ja võimsaid omadusi, mis lubavad igaühel kerge vaevaga programmeerimisega tegelda. Mõned neist:

- Lõimitud keskkond, kus ühes rakenduses on koos TurtleScripti interpretaator, redaktor, lõuend ja kõik muud tööriistad ilma mingite lisasõltuvustega.
- Võimalus tõlkida TurtleScripti käske KDE tõlkimisraamistikus.
- TurtleScript toetab kasutaja määratud funktsioone, rekursioone ning dünaamilist tüüpide lülitamist.

- Käskude täitmist saab aeglustada, peatada või katkestada igal ajahetkel.
- Võimas redaktor, mis lubab süntaksit esile tõsta, nummerdab ridu, märgib vigu ja teeb veel palju muud toredat.
- Lõuendi, millel kilpkonn joonistab, võib salvestada pildina (PNG) või joonistusena (SVG).
- Kontekstiabi alati, kui seda vajad. Vajuta lihtsalt **F2** (või vaata **Abi** → **Abi x kohta...**), kui soovid saada midagi teada parajasti kursori all asuva koodi kohta.
- Veadialoog, mis seob veateated programmis esinevate vigadega ja märgib nad punase värviga.
- lihtsustatud programmeerimisterminoloogia.
- Lõimitud näidisprogrammid, mis hõlbustavad alustamist. Näidised on tõlgitud KDE tõlkimisraamistiku abil.

Peatükk 2

KTurtle'i kasutamine



KTurtle'i peaaken koosneb kolmest põhiosast: redaktor (1) vasakul, kus saab kirja panna TurtleScripti käsud, lõuend (2) paremal, kus kilpkonn sinu joonistust valmistab, ja inspektor (3), mis annab teavet programmi täitmise kohta. Peaaknal on veel neli osa: menüüriba (5), mis võimaldab kasutada kõiki saadaolevaid käske, tööriistariba (4), kus saab kiiresti valida sagedamini vaja mineva toimingu, **konsool**, kus saab testimiseks sisestada ühereal käsu. ja olekuriba (akna allservas), kus näeb KTurtle'i tagasisidena rakenduse olekut.

2.1 Redaktor

Redaktor on koht, kus panna kirja TurtleScripti käsud. See pakub kõiki võimalusi, mida üks tänapäevane redaktor ikka pakkuma peab. Enamik neist on kasutatavad menüüst Fail või Redigeerimine. Redaktori võib dokkida ükspuha millisele peaakna küljele või ka hoopis lahti haakida ja töölaual sootuks mujale asetada.

Koodi saab redaktorisse mitmel moel sisestada. Lihtsaim meetod on kasutada juba valmis tehtud näidiseid. Selleks vali **Fail** → **Näidised** failimenüüs ja klõpsa vajalikul failil. Valitud fail avatakse redaktoris, pärast mida saab menüükäsuga **Käivitamine** → **Käivita** või tööriistariba nupuga **Käivita** hakata seda ellu viima.

TurtleScripti faile saab avada menüükäsuga **Fail** → **Ava...**

Kolmas võimalus on kirjutada kood ise redaktorisse või kopeerida ja asetada kood kuskilt mujalt.

2.2 Lõuend

Lõuend on ala, kus käsud visualiseeritakse, kus nendega 'joonistatakse' pilt. Ehk siis - see on kilpkonna mängumaa. Kui oled sisestanud mingi koodi redaktorisse ja selle käivitanud, võib juhtuda üks kahest: 1) kood täidetakse edukalt ja sa näed midagi lõuendil muutumas või 2) koodis on viga ja ilmub teade, mis ütleb, millise veaga on tegemist.

Hiirerattaga saab lõuendit suurendada ja vähendada.

2.3 Inspektor

Inspektor annab programmi töötamise ajal teada muutujatest ja õpitud funktsioonidest ning näitab koodipuud.

Inspektori võib dokkida ükspuha millisele peaakna küljele või ka hoopis lahti haakida ja töölaual sootuks mujale asetada.

2.4 Tööriistariba

Selle abil saab kiiresti ette võtta sagedamini vaja minevaid toiminguid. Tööriistariba sisaldab ka **konsooli**, mis võimaldab kiiresti käske anda, millest võib olla kasu, kui soovid käsku katsetada ilma redaktori sisu muutmata.

Tööriistariba saab oma maitse järgi kohandada menüükäsuga **Seadistused** → **Tööriistaribade seadistamine...**

2.5 Menüüriba

Menüüribal leiad kõik toimingud, mida KTurtle'is on võimalik kasutada. Need on jagatud järgmistesse rühmadesse: **Fail**, **Redigeerimine**, **Lõuend**, **Käivitamine**, **Tööriistad**, **Seadistused** ja **Abi**. Kirjeldame siin neid kõiki.

2.5.1 Failimenüü

Fail → **Uus (Ctrl-N)**
: Loob uue tühja TurtleScripti faili.

Fail → **Ava... (Ctrl-O)**
: Avab TurtleScripti faili.

Fail → **Ava viimati kasutatud**
: Avab viimati kasutatud TurtleScripti faili.

Fail → **Näidised**
: Avab TurtleScripti näidisfailidega kataloogi. Need peaksid olema keeles, mis on määratud seadistustedialoogis (**Seadistused** → **Skriptikeel**).

Fail → Hangi rohkem näidiseid...

Avab **uue kuuma kraami** hankimise dialoogis, mille abil saab internetist alla laadida uusi TurtleScripti faile.

Fail → Salvesta (Ctrl-S)

Salvestab parajasti avatud TurtleScripti faili.

Fail → Salvesta kui...

Salvestab parajasti avatud TurtleScripti faili uude asukohta või uue nimega.

Fail → Ekspordi HTML-ina...

Ekspordib redaktori aktiivse sisu HTML-failina, mis sisaldab ka esiletõstmise värve.

Fail → Trüki... (Ctrl-P)

Trükib koodiredaktoris kirja pandud käsud.

Fail → Välju (Ctrl-Q)

Lõpetab KTurtle'i töö.

2.5.2 Menüü Redigeerimine

Redigeerimine → Võta tagasi (Ctrl-Z)

Tühistab viimase koodi tehtud muudatuse. KTurtle ei tunne tagasivõtmisel piire.

Redigeerimine → Tee uuesti (Ctrl-Shift-Z)

Taastab koodis tagasivõetud muudatuse.

Redigeerimine → Lõika (Ctrl-X)

Lõikab redaktoris valitud teksti ja asetab selle lõikepuhvrisse.

Redigeerimine → Kopeeri (Ctrl-C)

Kopeerib redaktoris valitud teksti lõikepuhvrisse.

Redigeerimine → Aseta (Ctrl-V)

Asetab teksti lõikepuhvrist redaktorisse.

Redigeerimine → Vali kõik (Ctrl-A)
Valib kogu redaktori teksti.

Redigeerimine → Otsi... (Ctrl-F)
Selle käsuga saab koodis teatud sõna või väljendit otsida.

Redigeerimine → Otsi järgmine (F3)
Sellega saab leida otsitava fraasi järgmise esinemiskorra.

Redigeerimine → Otsi eelmine (Shift-F3)
Sellega saab leida otsitava fraasi eelmise esinemiskorra.

Redigeerimine → Ülekirjutamisrežiim (Ins)
Lülitab lisamis- ja ülekirjutamisrežiimi vahel.

2.5.3 Menüü Lõuend

Lõuend → Ekspordi pildina (PNG)...
Ekspordib aktiivse lõuendi sisu PNG rasterpildina.

Lõuend → Ekspordi joonistusena (SVG)...
Ekspordib aktiivse lõuendi sisu SVG vektorjoonistusena.

Lõuend → Trüki lõuend...
Ekspordib aktiivse

2.5.4 Menüü Käivitamine

Käivitamine → Käivita (F5)
Käivitab redaktoris kirja pandud käsud.

Käivitamine → Paus (F6)
Peatab käskude täitmise. Seda saab kasutada ainult siis, kui käske parajasti täidetakse.

Käivitamine → Katkesta (F7)
Katkestab käskude täitmise. Seda saab kasutada ainult siis, kui käske parajasti täidetakse.

Käivitamine → Töökiirus
Näitab võimalikke täitmise kiirusi: **täiskiirus (esiletõstmiseta ja inspektorita)**, **täiskiirus**, **aeglane**, **aeglasem**, **kõige aeglasem** ja **sammhaaval**. Kui kiiruseks valida **täiskiirus** (vaikeväärtus), on vaevu võimalik jälgida, mis toimub. Sageli on see kasulik, kuid vahel võib tekkida soov käskude täitmist üksikasjalikult jälgida. Sel juhul tuleks valida kiiruseks **aeglane**, **aeglasem**, **kõige aeglasem**. Mõne aeglase kiiruse valimisel näidatakse redaktoris täitmise hetkeasukohta. **Sammhaaval** täidab korraga ühe käsu.

2.5.5 Tööriistade menüü

Tööriistad → Suunavalija...
Avab suunavalija dialoogi.

Tööriistad → värvivalija...
Avab värvivalija dialoogi.

2.5.6 Menüü Seadistused

Seadistused → Skriptikeel
Valib koodi keele.

Seadistused → Redaktori näitamine (Ctrl-E)
Näitab või peidab redaktori.

Seadistused → Inspektori näitamine (Ctrl-I)
Näitab või peidab inspektori.

Seadistused → Vigade näitamine
Näitab koodi täitmisel tekkinud vigade nimekirja **vigade** või peidab selle. Kui see valik on sisse lülitatud, klõpsa kilpkonna taasnägemiseks **lõuendil**.

Seadistused → Näita reanumbreid (F11)
Sellega saab lasta redaktoris reanumbreid näidata. See on kasulik näiteks vigade leidmiseks.

Seadistused → Tööriistariba näitamine
Lülitab peamise tööriistariba näitamise sisse-välja.

Seadistused → Olekuriba näitamine
Lülitab olekuriba näitamise sisse-välja.

Seadistused → Kiirklahvide seadistamine...
KDE tavapärane kiirklahvide seadistamise dialoog.

Seadistused → Tööriistaribade seadistamine...
Tavapärane KDE tööriistaribade seadistamise dialoog.

2.5.7 Abimenüü

Abi → Sisukord... (F1)

Käivitab KDE abisüsteemi ja avab KTurtle abimaterjali (käesoleva käsiraamatu).

Abi → Mis see on? (Shift+F1)

Muudab hiirekursori noole ja küsimärgi kombinatsiooniks. Klõpsates nüüd mõnel KTurtle elemendil, avaneb väike abiaken (kui see on antud elemendi kohta olemas), mis selgitab elemendi funktsiooni.

Abi → Vaheta rakenduse keelt...

Avab dialoogi, kus saab muuta rakenduse **esmast keelt** ja **tagavarakeelt**, kui esmane keel pole saadaval.

Abi → Saada vearaport...

Avab vearaporti dialoogi, mille abil saab teada anda veast või esitada oma 'soov' millegi parandamiseks.

Abi → KTurtle info

Näitab versiooni ja infot autori kohta.

Abi → KDE info

Näitab KDE versiooni ja muud olulisemat põhiinfot.

Abi → Abi x kohta... (F2)

See on väga kasulik võimalus, näidates abi koodi selle koha kohta, kus parajasti asub kursor. Kui nt. kasutasid koodis käsku **`print`**, saad selle abil teada, mida käsiraamat vastava käsu kohta ütleb. Vii kursor käsu **`print`** peale ja vajuta **F2**. Seejärel näed käsiraamatus käsust **`print`** rääkivat kohta.

See on päris kasulik ja oluline TurtleScripti õppimisel.

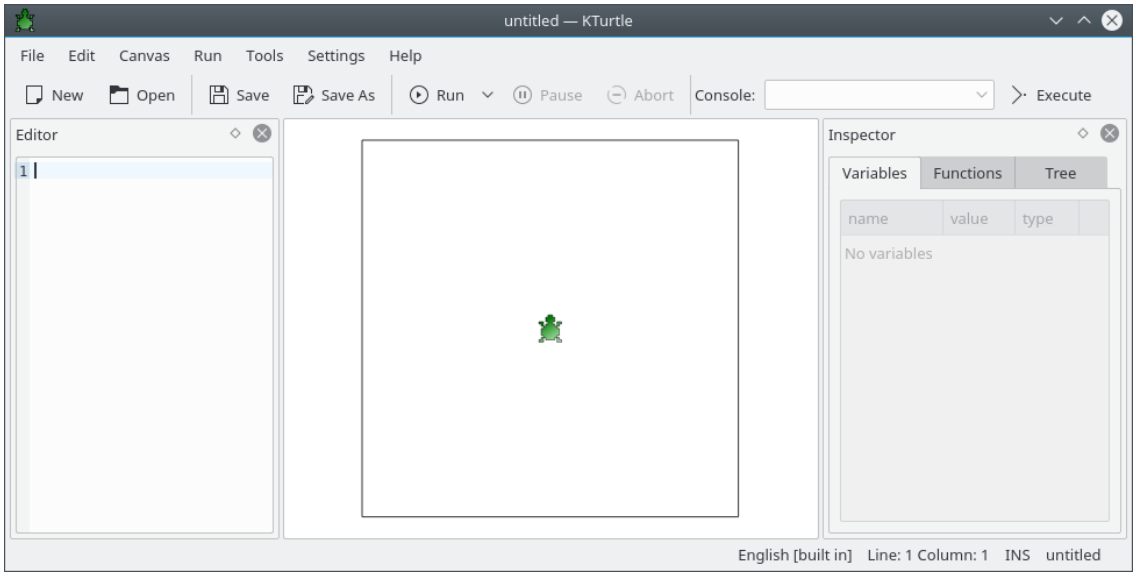
2.6 Olekuriba

Olekuribal annab KTurtle sulle tagasisidet oma oleku kohta. Vasakul on näha viimane toiming. Paremal näeb kursori hetkeasukohta (rea ja veeru number). Keskel näeb, millist keelt käskude puhul parajasti kasutatakse.

Peatükk 3

Alustamine

KTurtle'i käivitamisel ilmub nähtavale midagi sellist:



Me eeldame siinses tutvustuses, et TurtleScripti käskude keel on inglise keel. Kasutatavat keelt saab muuta menüükäsuga **Seadistused** → **Skriptikeel**. Arvesta aga, et keel, mille sa KTurtle'ile valid, peab olema sama keel, milles sa TurtleScripti käsud kirjutad, mitte aga keel, mida kasutab sinu arvutis KDE ja milles näidatakse KTurtle'i liidest ja menüüsid.

3.1 Esimesed sammud TurtleScriptis: tutvume Kilpkonnaga!

Ilmselt märkasid, et keset lõuendit tudub kilpkonn. Nüüd aga vaatame, kuidas teda redaktoris kirjutatud käskudega liikuma panna.

3.1.1 Kilpkonna liikumine

Paneme nüüd kilpkonna liikuma. Ta võib liikuda kolme moodi: (1) edasi ja tagasi, (2) pöörata paremale ja vasakule ning (3) minna (hüpata) otse mingisse ekraanipunkti. Proovime näiteks nii:

```
forward 100
turnleft 90
```

Kirjuta see redaktorisse või kopeeri ja aseta ning seejärel käivita käsud (**Käivitamine** → **Käivita**).

Kui kirjutasid toodud käsud redaktorisse ja käivitasid need, nägid ilmselt vähemalt üht järgmistest asjadest:

1. Pärast käskude käivitamist liikus kilpkonn üles, joonistas sirge ja pööras siis oma liikumissuunaga risti vasakule. Seda põhjustasidki käsud **`forward`** ja **`turnleft`**.
2. Koodi värv muutus kirjutamisel. Seda nimetatakse *intuitiivseks esiletõstmiseks* — erinevat laadi käskudele antakse erinev värv. Nii on eriti suuremaid koodiblokke märksa lihtsam lugeda. Samuti leiab hõlpsamini üles vead, sest sellistele sõnadele värvi ei anta.
3. Kilpkonn joonistas peenikese musta joone.
4. Võis juhtuda ka, et said veateate. See tähendab üht kahest: tegid käskude kopeerimisel mingi vea või siis ei ole sul veel määratud TurtleScripti käskude korrektne keel. Viimast saab teha menüükäsuga **Seadistused** → **Skriptikeel**.

Igaks juhuks kordame üle: **`forward 100`** andis kilpkonnale korralduse liikuda joont maha jättes edasi ning **`turnleft 90`** käsu pöörata 90 kraadi vasakule.

Palun uuri ka käskude kirjeldusi, mida tutvustame lähemalt käesolevas raamatus edaspidi: **`forward`**, **`backward`**, **`turnleft`** ja **`turnright`**.

3.1.2 Veel näiteid

Esimene näide oli imelihtne, nii et vaatame nüüd edasi!

```
reset

canvassize 200,200
canvascolor 0,0,0
pencolor 255,0,0
penwidth 5

go 20,20
direction 135

forward 200
turnleft 135
forward 100
turnleft 135
forward 141
turnleft 135
forward 100
turnleft 45

go 40,100
```

Kirjuta taas toodud kood redaktorisse või kopeeri see siit või ava näidis `arrow` menüüs **Näidised** ning käivita käsud (**Käivitamine** → **Käivita**). Edasistes näidetes me enam seda ei ütle, eeldades, et sa oled selle juba meelde jätnud...

Nagu näed, on teises näites koodi märksa enam. Samuti on siin mitu uut käsku. Seletame need nüüd lühidalt lahti:

Pärast käsku **`reset`** on kõik selline, nagu KTurtlei käivitamisel.

`canvassize 200,200` määrab lõuendi laiuseks ja kõrguseks 200 pikslit. Laius ja kõrgus on võrdsed, see tähendab, et lõuend on ruudukujuline.

canvascolor 0,0,0 muudab lõuendi mustaks. **0,0,0** on RGB värvikood, milles kõik väärtused on **0**, mis tähendabki kokkuvõttes musta.

pencolor 255,0,0 määrab pliiatsi värviks punase. **255,0,0** on RGB värvikood, kus ainult punase väärtus on maksimaalne **255**, teistel (rohelisel ja sinisel) aga **0**, mis annab kokku erepunase.

Kui sa ei ole kursis, mis on värviväärtused, loe kindlasti sõnastikust artiklit RGB värvikoodide kohta.

penwidth 5 määrab pliiatsi laiuseks **5** pikslit. See tähendab, et siitpeale on kõik kilpkonna tõmmatavad jooned paksusega **5**, kuni me anname **penwidth** väärtuseks midagi muud.

go 20,20 annab kilpkonnale korralduse minna lõuendi kindlasse punkti. Ülemisest vasakust nurgast arvestades asub see koht lõuendi vasakust servast 20 piksli ning ka lõuendi ülaservast 20 piksli kaugusel. Pane tähele, et käsu **go** kasutamisel kilpkonn joont ei tõmba.

direction 135 määrab kilpkonna suuna. Käsud **turnleft** ja **turnright** muudavad kilpkonna nurka selle praeguse suuna suhtes. Käsk **direction** muudab kilpkonna suunda alates nullist, see ei ole seotud kilpkonna varasema suunaga.

Pärast käsku **direction** tuleb mitu käsku **forward** ja **turnleft**. Nendega käibki tegelik joonistamine.

Uus **go** annab lõpuks kilpkonnale käsu eemale minna.

Uuri kindlasti ka käskude seletusi, kus on kõik põhjalikult lahti seletatud.

Peatükk 4

TurtleScript'i käskude seletused

Alljärgnev on KTurtle'i TurtleScripti seletus. Peatüki esimeses osas tutvustame TurtleScripti programmide grammatika mõningaid külgi. Teises osas on juttu eranditult matemaatilistest tehetest, tõeväärtustehetest ja võrdlustehetest. Kolmas osa kujutab endast hiiglaslikku kõigi käskude seletust. Neljandas osas selgitatakse, kuidas omistada väärtusi muutujatele. Lõpuks seletame viiendas osas, kuidas korraldada käskude täitmist täitmise kontrollimise lausetega, ning kuuendas osas, kuidas luua omaenda käske käsuga learn.

4.1 TurtleScripti grammatika

Nagu igas keeles, on ka TurtleScriptil omad sõnad ja sümbolid. Tavalises keeles eristatakse näiteks tegusõnu (˝käima˝, ˝laulma˝) ja nimisõnu (˝õde˝, ˝maja˝), mida kasutatakse eri eesmärgil. TurtleScript on programmeerimiskeel, mida kasutatakse KTurtle'i juhendamiseks, mida ette võtta.

Käesolevas osas selgitatakse lühidalt TurtleScripti sõnade ja sümbolite tüüpe. Me tutvustame kommentaare, käske ja kolme eri laadi literaale: arvud, stringid ja tõeväärtused (tõene/väär).

4.1.1 Kommentaarid

Programm koosneb instruktsioonidest, mis täidetakse programmi töötamisel, ja niinimetatud kommentaaridest. Kommentaare ei täideta, KTurtle lihtsalt eirab neid töötamise ajal. Kommentaarid on mõeldud teistele programmeerijatele, et nad saaksid paremini programmist aru. Kõik, mis järgneb sümbolile **#**, on TurtleScripti seisukohalt kommentaar. Näiteks see pisike programm ei tee midagi:

```
# see väike programm ei tee midagi, see ongi ainult kommentaar!
```

See on vahest mõttetu, aga selgitab kenasti asja olemust.

Kommentaarid on väga kasulikud, kui programm on vähegi keerukam. Need on suureks abiks teistele programmeerijatele. Järgmises programmis kasutatakse kommentaare koos käsuga print.

```
# selle programmi tegi Cies Breijs.
print "seda teksti näidatakse lõuendil"
# eelmine rida ei olnud kommenaar, aga järgmine rida on:
# print "seda teksti ei näidata!"
```

Programm esimene rida kirjeldab programmi. Teise rea KTurtle täidab ning näitab lõuendil teksti **seda teksti näidatakse lõuendil**. Kolmas rida on kommentaar. Neljas rida on samuti kommentaar, mis aga sisaldab TurtleScripti, nii et kui sümbol **#** neljanda rea eest eemaldada, täidab KTurtle ka selle näitamiskäsu. Programmeerijate kõnepruugis on neljanda rea näitamiskäsk ˝välja kommenteeritud˝.

Kommenteeritud read on redaktoris esile tõstetud helehalli värviga.

4.1.2 Käsud

Käskudega antakse kilpkonnale ehk KTurtle'ile korraldus midagi teha. Mõned käsud vajavad sisendit, teised annavad väljundit.

```
# forward on käsk, mis vajab sisendit, antud juhul arvu 100:
forward 100
```

Esimene rida on kommentaar. Teine rida sisaldab käsku **forward** ja arvu **100**. Arv ei ole käsu osa, seda käsitletakse käsu ˝sisendina˝.

Mõned käsud, nt. **go**, vajavad rohkem kui üht sisendväärtust. Mitut väärtust saab sisestada neid komaga (**,**) eraldades.

Täpsemalt kirjeldatakse kõiki käske, mida KTurtle toetab, siin. Sisseehitatud käsud on esile tõstetud tumesinise värviga.

4.1.3 Arvud

Arvatavasti oled juba koolis õppinud, mis asjad on arvud või mingil muul moel osanud selle välja uurida. KTurtle kasutab arve üsna sarnaselt sellele, kuidas kasutatakse neid matemaatikas või kas või tavalises kõnepruugis.

Me kasutamine niinimetatud naturaalarve: **0**, **1**, **2**, **3**, **4**, **5** jne. Samuti negatiivseid arve: **−1**, **−2**, **−3** jne. Aga ka murdarve, näiteks: **0.1**, **3.14**, **33.3333**, **−5.05**, **−1.0**. Kümnendkoha eraldajana kasutatakse punkti (**.**).

Arve saab pruukida matemaatilistes tehetes ja võrdlustehetes. Neid saab lisada ka salvestada muutujatesse. Arvud on esile tõstetud tumepunase värviga.

4.1.4 Stringid

Kõigepealt näide:

```
print "Hei, mina olen string."
```

Selles näites on **print** käsk, **´´Hei, mina olen string.´´** aga string. Stringide alguses ja lõpus seisab **´´**, mille järgi KTurtle tunneb ära, et tegemist on stringiga.

Stringid võivad esineda muutujates nagu arvudki. Aga erinevalt arvudest ei saa stringe kasutada matemaatilistes tehetes ega võrdlustehetes. Stringid on esiletõstetud punase värviga.

4.1.5 Tõeväärtused (tõene/väär)

Tõeväärtusi on ainult kaks: **tõene** ja **väär**. Mõnikord kasutatakse nende kohta ka muid nimetusi: sees ja väljas, jah ja ei, üks ja null. Kuid TurtleScriptis on nad alati **tõene** ja **väär**. Vaatame killukest TurtleScripti:

```
$a = true
```

Kui vaadata inspektorit, näeb, et muutuja **$a** väärtuseks on määratud **tõene** ja see on tõeväärtus.

Tõeväärtused on sageli võrdlustehte tulemus, nagu järgmises TurtleScripti killukeses:

```
$answer = 10 > 3
```

Muutuja **$answer** väärtuseks on määratud **tõene**, sest **10** on suurem kui **3**.

Tõeväärtused **tõene** ja **väär** on esile tõstetud tumepunase värviga.

4.2 Matemaatika-, tõeväärtus- ja võrdlustehted

Selle osa pealkiri võib tunduda raskepärasena, aga tegelikult pole siin midagi väga keerulist.

4.2.1 Matemaatilised tehted

Need on põhilised matemaatikasümbolid: liitmine (**+**), lahutamine (**-**), korrutamine (*****), jagamine (**/**) ja astendamine (**^**).

Väike näide matemaatikatehetest TurtleScriptis:

```
$add      = 1 + 1
$subtract = 20 - 5
$multiply = 15 * 2
$divide   = 30 / 30
$power    = 2 ^ 2
```

Matemaatikatehete tulemused omistatakse eri muutujatele. Väärtusi näeb inspektori abil.

Kui soovid väga lihtsat tehet teha, siis see käib umbes nii:

```
print 2010-12
```

Ja nüüd näide sulgudega:

```
print ( ( 20 - 5 ) * 2 / 30 ) + 1
```

Sulgudes olevad tehted sooritatakse esimestena. Antud näites arvutatakse 20-5, siis korrutatakse 2-ga, jagatakse 30-ga ning seejärel liidetakse 1 (tulemuseks on 2). Sulge võib kasutada ka muudel juhtudel.

KTurtle pakub käskudes teistegi matemaatiliste võimaluste kasutamist. Uuri näiteks järgmisi käske (kuid arvesta, et tegemist on keerukamate tehetega): round, random, sqrt , pi, sin, cos, tan, arcsin, arccos, arctan.

4.2.2 Tõeväärtustehted

Kui matemaatilised tehted on mõeldud eelkõige arvudele, siis tõeväärtustehteid tehakse tõeväärtustega (**tõene** ja **väär**) Tõeväärtustehteid on ainult kolm, nimelt **and**, **or** ja **not**. Järgnev TurtleScripti killuke seletab nende kasutamist:

```
$and_1_1 = true and true    # -> true
$and_1_0 = true and false   # -> false
$and_0_1 = false and true   # -> false
$and_0_0 = false and false  # -> false

$or_1_1 = true or true    # -> true
```

```
$or_1_0 = true or false   # -> true
$or_0_1 = false or true   # -> true
$or_0_0 = false or false  # -> false

$not_1 = not true   # -> false
$not_0 = not false  # -> true
```

Inspektoris näeb ka väärtusi, kuid me andsime tulemused ka lühikommentaarina ridade lõpus. **and** on **tõene** ainult siis, kui mõlemad pooled on **tõesed**. **or** on **tõene** siis, kui üks pooltest on **tõene**. **not** aga muudab **tõese vääraks** ja **väära tõeseks**.

Tõeväärtustehted on esile tõstetud roosa värviga.

4.2.2.1 Mõned keerukamad näited

Üks näide tehte **and** kohta:

```
$a = 1
$b = 5
if (($a < 10) and ($b == 5)) and ($a < $b) {
  print "hello"
}
```

Selles TurtleScripti killukeses ühendatakse kolm võrdlustehet tehte **and** abil. See tähendab, et kõik kolm peavad olema ″tõesed″, et näidataks teksti ″hello″.

Näide tehtega **or**:

```
$n = 1
if ($n < 10) or ($n == 2) {
  print "hello"
}
```

Selles TurtleScripti killukeses on **or** vasak pool ″tõene″, parem pool aga ″väär″. Et **or** üks pool on siiski ″tõene″, on ka tehte **or** tulemus ″tõene″ ning järelikult näidatakse teksti ″hello″.

Lõpuks ka näide tehtega **not**, mis muudab ″tõese″ ″vääraks″ ja ″väära″ ″tõeseks″:

```
$n = 1
if not ($n == 3) {
  print "hello"
} else {
  print "not hello ;-)"
}
```

4.2.3 Võrdlustehted

Vaatleme lihtsat võrdlust:

```
$answer = 10 > 3
```

Siin võrreldakse arvu **10** arvuga **3** ″suurem kui″ tehtega. Võrdluse tulemus, tõeväärtus **tõene** salvestatakse muutujasse **$answer**.

Võrdlustehetega saab omavahel võrrelda kõiki arve ja muutujaid (mis sisaldavad arve).

Toome ära kõik võimalikud võrdlustehted:

$A == $B	võrdub	vastus on 'tõene', kui **$A** võrdub **$B**
$A != $B	ei võrdu	vastus on 'tõene', kui **$A** ei võrdu **$B**-ga
$A > **$B**	suurem kui	vastus on 'tõene', kui **$A** on suurem kui **$B**
$A < **$B**	väiksem kui	vastus on 'tõene', kui **$A** on väiksem kui **$B**
$A >= $B	suurem kui või võrdne	vastus on 'tõene', kui **$A** on suurem kui või võrdub **$B**-ga
$A <= $B	väiksem kui või võrdne	vastus on 'tõene', kui **$A** on väiksem kui või võrdub **$B**-ga

Tabel 4.1: Küsimuste tüübid

Palun arvesta, et $A ja $B peavad olema arvud või muutujad, mis sisaldavad arve.

4.3 Käsud

Just käskudega saab panna kilpkonna või KTurtle'i midagi tegema. Osa käske eeldab sisendit, osa annab väljundi. Selles osas kirjeldame kõiki KTurtle'i sisseehitatud käske. Lisaks saab oma käske luua käsuga learn. Käsitletavad sisseehitatud käsud on esile tõstetud tumesinise värviga.

4.3.1 Kilpkonna liigutamine

Kilpkonna saab lõuendil liigutada mitme käsuga.

forward (fw)

```
forward X
```

forward paneb kilpkonna otse edasi liikuma X piksli võrra. Kui pliiats on sees, jätab kilpkonna ka jälje maha. **forward** asemel võib kasutada lühendit **fw**.

backward (bw)

```
backward X
```

backward paneb kilpkonna otse tagasi liikuma X piksli võrra. Kui pliiats on sees, jätab kilpkonna ka jälje maha. **backward** asemel võib kasutada lühendit **bw**.

turnleft (tl)

```
turnleft X
```

turnleft paneb kilpkonna X kraadi võrra vasakule pöörama. **turnleft** asemel võib kasutada lühendit **tl**.

turnright (tr)

```
turnright X
```

turnright paneb kilpkonna X kraadi võrra paremale pöörama. **turnright** asemel võib kasutada lühendit **tr**.

direction (dir)

```
direction X
```

direction määrab kilpkonna suunaks X kraadi arvestades nullist, see ei ole seotud kilpkonna varasema suunaga. **direction** asemel võib kasutada lühendit **dir**.

getdirection

```
getdirection
```

getdirection tagastab kilpkonna suuna kraadidena arvestades nullist, kus null on suund, millega kilpkonn suundub üles.

center

```
center
```

center sunnib kilpkonna liikuma lõuendi keskpunkti.

go

```
go X,Y
```

go paneb kilpkonna liikuma lõuendi konkreetsesse punkti. See asub X piksli kaugusel lõuendi vasakust servast ja Y piksli kaugusel lõuendi ülaservast.

gox

```
gox X
```

gox paneb kilpkonna samal kõrgusel liikuma X piksli kaugusele lõuendi vasakust servast. **gox** asemel võib kasutada lühemat **gx**.

goy

```
goy Y
```

goy paneb kilpkonna samal laiusel (kaugusel vasakust servast) liikuma Y piksli kaugusele lõuendi ülaservast. **goy** asemel võib kasutada lühemat **gy**.

MÄRKUS
Käskudega **go**, **gox**, **goy** ja **center** ei joonista kilpkonn joont, olgu pliiats üleval või all.

4.3.2 Kus on kilpkonn?

Kilpkonna asukohta lõuendil saab teada kahe käsuga.

getx

getx tagastab pikslite arvu lõuendi vasakust servast kilpkonna asukohani.

gety

gety tagastab pikslite arvu lõuendi ülemisest servast kilpkonna asukohani.

4.3.3 Kilpkonn pliiatsiga

Kilpkonnal on pliiats, millega ta liikudes joont veab. Pliiatsit saab juhtida mitme käsuga, mida siin tutvustamegi.

penup (pu)

```
penup
```

penup tõstab pliiatsi lõuendilt ja kuni see on 'üleval', ei tõmba kilpkonn liikudes ühtki joont. Vaata ka **pendown**. **penup** asemel võib kasutada lühendit **pu**.

pendown (pd)

```
pendown
```

pendown surub pliiatsi lõuendi vastu ja kuni see on 'all', tõmbab kilpkonn liikudes joone. Vaata ka **penup**. **pendown** asemel võib kasutada ka lühendit **pd**.

penwidth (pw)

```
penwidth X
```

penwidth määrab pliiatsi tõmmatava joone laiuseks X pikslit. **penwidth** asemel võib kasutada ka lühendit **pw**.

pencolor (pc)

```
pencolor R,G,B
```

pencolor määrab pliitsi tõmmatava joone värvi. **pencolor** tahab saada sisendina RGB värvikoodi. **pencolor** asemel võib kasutada ka lühendit **pc**.

4.3.4 Lõuendiga seotud käsud

Lõuendi omadusi saab muuta mitme käsuga.

canvassize (cs)

```
canvassize X,Y
```

Käsk **canvassize** määrab lõuendi suuruse. Käsu sisendiks on X ja Y, kus X on uue lõuendi laius pikslites ja Y uue lõuendi kõrgus samuti pikslites. **canvassize** asemel võib kasutada lühendit **cs**.

canvascolor (cc)

```
canvascolor R,G,B
```

Käsk **canvascolor** määrab lõuendi värvi. **canvascolor** tahab saada sisendina RGB värvikoodi. **canvascolor** asemel võib kasutada ka lühendit **cc**.

4.3.5 Puhastamiskäsud

Kahe käsuga saab lõuendi puhastada sellest, mida sa sinna oled valmis meisterdanud.

clear (ccl)

```
clear
```

Käsk **clear** puhastab lõuendilt kogu joonistatu. Muud asjad jäävad püsima: kilpkonna asukoht ja nurk, lõuendivärv, kilpkonna nähaolek ja lõuendi suurus.

reset

```
reset
```

Käsk **reset** puhastab palju rohkem asju kui **clear**. Pärast käsku **reset** on kõik taas selline, nagu KTurtle'i käivitamisel: kilpkonn on lõuendi keskel, lõuendi värv on valge, kilpkonn hakkab liikudes vedama musta joont ja lõuendi suurus on 400 x 400 pikslit.

4.3.6 Kilpkonn on sprait

Vaevalt et paljud teavad, mis õieti on sprait, nii et ütleme siis saladuse välja: sprait on väike pilt, mida saab ekraanil liigutada. Täpsemalt vaata sõnastikust mõiste sprait kirjeldust. See tähendab, et meie kilpkonn on sprait!

Toome siin ära kõik käsud, mida saab spraidi puhul kasutada.

[KTurtle'i praegune versioon toetab spraidina ainult kilpkonna kasutamist. Aga tulevikus võib olla vägagi võimalik, et võid kilpkonna oma soovi kohaselt millegi muuga asendada.]

spriteshow (ss)

```
spriteshow
```

Käsk **spriteshow** muudab kilpkonna uuesti nähtavaks pärast tema peitmist. **spriteshow** asemel võib kasutada ka lühendit **ss**.

spritehide (sh)

```
spritehide
```

spritehide peidab kilpkonna. Seda on mõtet kasutada siis, kui kilpkonn kohe kuidagi ei taha sinu joonistusele sobida. **spritehide** asemel võib kasutada ka lühendit **sh**.

4.3.7 Kas kilpkonn oskab ka kirjutada?

Õige vastus: 'jah, muidugi'. Kilpkonn võib kirjutada küll, aga mõistagi pead sina talle ütlema, mida kirjutada.

print

```
print X
```

print annab kilpkonnale käsu midagi lõuendile kirjutada. **print** sisendiks võivad olla nii numbrid kui stringid. Sümbolit '+' kasutades saab käsuga **print** lasta kirjutada mitmesuguseid numbreid ja stringe. Toome siin väikese näite:

```
$year = 2003
$author = "Cies"
print $author + " alustas KTurtle'i projekti aastal " + $year + "  ja  ↩
    ta töötab endiselt rõõmuga selle kallal!"
```

fontsize

```
fontsize X
```

fontsize määrab fondi suuruse, mida **print** kirjutamisel kasutab. **fontsize** sisend peab olema number. Suurus määratakse pikslites.

4.3.8 Matemaatilised käsud

Järgmised käsud on KTurtle'i keerukamad matemaatilised käsud.

round

```
round(x)
```

round ümardab antud arvu lähima täisarvuni.

```
print round(10.8)
forward 20
print round(10.3)
```

Selle koodiga kirjutab kilpkonn arvud 11 ja 10.

random (rnd)

```
random X,Y
```

random on käsk, mis tahab saada sisendit ja annab kohe ka väljundi. Sisendiks on kaks arvu, milles üks (X) määrab minimaalse ja teine (Y) maksimaalse väljundi. Väljund on juhuslik arv, mis on miinimumiga võrdne või suurem ning maksimumiga võrdne või väiksem. Üks väike näide:

```
repeat 500 {
  $x = random 1,20
  forward $x
  turnleft 10 - $x
}
```

Käsuga **random** saab oma programmi veidi kaootilisemaks muuta.

mod

```
mod X,Y
```

Käsk **mod** tagastab esimese arvu jagamisel teisega tekkiva jäägi.

sqrt

```
sqrt X
```

Käsk **sqrt** peab leidma arvu X ruutjuure.

pi

```
pi
```

See käsk tagastab konstandi pii **3,14159**.

sin, cos, tan

```
sin X
cos X
tan X
```

Need kolm käsku esindavad maailmakuulsaid trigonomeetrilisi funktsioone **sin**, **cos** ja **tan**. Nende käskude sisendargument X on arv.

arcsin, arccos, arctan

```
arcsin X
arccos X
arctan X
```

Need käsud on funktsioonide sin, cos and tan pöördväärtused. Nende käskude sisendargument X on arv.

4.3.9 Sisend ja tagasiside dialoogide kaudu

Dialoog on väike hüpikaken, mis annab teatud tagasisidet või soovib saada mingit sisendit. KTurtle kasutab dialoogidele kaht käsku: **`message`** ja **`ask`**.

message

```
message X
```

Käsk **`message`** vajab sisendina stringi. See näitab hüpikdialoogi, mis sisaldab stringi teksti.

```
message  "Cies alustas KTurtle'iga aastal 2003  ja ta töötab endiselt  ↩
    rõõmuga selle kallal!"
```

ask

```
ask X
```

Käsk **`ask`** vajab sisendina stringi. See näitab hüpikdialoogi, mis sisaldab stringi nagu ka message. Kuid lisaks sellele on dialoogis ka sisendiväli. Selle abil saab kasutaja sisestada arvu või stringi, mis on võimalik salvestada muutujasse või edastada argumendina käsule. Näide:

```
$in = ask "Kui vana sa oled?"
$out = 2003 - $in
print "Aastal 2003 olid sa " + $out + " aastat vana."
```

Kui kasutaja sulgeb sisenddialoogi või ei sisesta midagi, on muutuja tühi.

4.4 Muutujate omistamine

Kõigepealt vaatleme muutujaid, seejärel neile väärtuse omistamist.

Muutujad on sõnad, mille alguses seisab '$'. Redaktoris on nad esile tõstetud purpurse värviga.

Muutujad võivad sisaldada suvalist arvu, stringi või tõeväärtust. Omistamise **`=`** abil antakse muutujale sisu. See sisu püsib seni, kuni programm lõpetab käskude täitmise või kuni muutujale pole omistatud midagi muud.

Omistatud muutujaid saab seejärel kasutada vastavalt nende sisule. Võtame näiteks järgmise TurtleScripti killukese:

```
$x = 10
$x = $x / 3
print $x
```

Kõigepealt omistatakse muutujale **`$x`** väärtus **`10`**. Seejärel omistatakse muutujale **`$x`** iseenda väärtus jagatud **`3`**-ga — see tähendab, et muutuja **`$x`** väärtuseks saab tehte **`10 / 3`** tulemus. Lõpuks **`$x`** näidatakse. Teisel ja kolmandal real on näha, et muutujat **`$x`** kasutatakse nagu arvu.

Et muutujaid kasutada, tuleb neile väärtus omistada. Näide:

```
print $n
```

See ei näita mitte midagi ja annab vaid veateate.

See oli lihtne, aga võtame kohe TurtleScripti raskema näite:

```
$a = 2004
$b = 25

# the next command prints "2029"
print $a + $b
backward 30
# the next command prints "2004 plus 25 equals 2029"
print $a + " plus " + $b + " equals " + ($a + $b)
```

Esimesel kahel real määratakse muutujate **$a** ja **$b** väärtuseks 2004 ja 25. Ülejäänud osa näitest koosneb kahest käsust **print**, mida eraldab käsk **backward 30**. Kommentaarid selgitavad, mida käsud teevad. Käsk **backward 30** on selleks, et iga väljund oleks ikka uuel real. Toodud näites on ka näha, et muutujaid saab kasutada nende sisuga mis tahes tehete sooritamiseks või sisendina, kui kutsuda välja mõni käsk.

Veel üks näide:

```
$name = ask "What is your name?"
print "Hi " + $name + "! Good luck while learning the art of programming ↩
    ..."
```

Siin on kõik päris selge ning ka siin võib näha, et muutujat **$name** kasutatakse nagu stringi.

Muutujate kasutamisel on suureks abiks inspektor. See näitab kõigi kasutusel olevate muutujate sisu.

4.5 Täitmise kontrollimine

Täitmise kontrollerid lubavad — nagu nimetuski ütleb — kontrollida käsu täitmist.

Täitmise kontrollimise käsud on esile tõstetud tumerohelise rasvase kirjaga. Looksulud, mida väga sageli koos täitmise kontrolleritega pruugitakse, on esile tõstetud musta värviga.

4.5.1 Kilpkonna ootelejätmine

Kui oled juba natuke KTurtle'iga harjutanud, paned tähele, et kilpkonn võib päris kiiresti joonistada. See käsk muudab tema tegevuse aga veidi aeglasemaks.

wait

```
wait X
```

wait paneb kilpkonna X sekundiks ootama.

```
repeat 36 {
  forward 5
  turnright 10
  wait 0.5
}
```

Selle koodiga joonistatakse ringjoon, aga kilpkonn ootab igal sammul pool sekundit. See jätab mulje nii enam-vähem normaalse kiirusega liikuvast kilpkonnast - vähemalt kilpkonna kohta...

4.5.2 Tingimuse ˝kui˝ (if) täitmine

if

```
if tõeväärtus { ... }
```

Kood, mis seisab sulgudes, täidetakse ainult siis, kui (**if**) tõeväärtuse väärtus on 'tõene'.

```
$x = 6
if $x > 5 {
  print "$x is greater than five!"
}
```

Esimesel real määratakse **$x** väärtuseks 6. Teisel real kasutatakse võrdlustehet **$x** > **5** hindamiseks. Et hinnang on 'tõene' ehk 6 on suurem kui 5, lubab täitmise kontroller **if** looksulgudes oleva koodi täita.

4.5.3 Vastasel juhul ehk ˝else˝

else

```
if tõeväärtus { ... } else { ... }
```

Kontrollerit **else** saab kasutada koos kontrolleriga **if**. Kood, mis seisab looksulgudes **else** järel, täidetakse ainult siis, kui tõeväärtuse väärtus on 'väär'.

```
reset
$x = 4
if $x > 5 {
  print "$x is greater than five!"
} else {
  print "$x is smaller than six!"
}
```

Võrdlustehe hindab avaldist **$x** > **5**. Et 4 ei ole suurem kui 5, on väärtuseks 'väär'. See tähendab, et täidetakse kood, mis seisab looksulgudes **else** järel.

4.5.4 Silmus ˝while˝

while

```
while tõeväärtus { ... }
```

Täitmise kontroller **while** on üsna sarnane kontrolleriga **if**. Nende peamine erinevus seisab selles, et **while** kordab looksulgudes olevat koodi seni, kuni vastus tõeväärtus on lõpuks 'väär'.

```
$x = 1
while $x < 5 {
  forward 10
  wait 1
  $x = $x + 1
}
```

Esimesel real määratakse **$x** väärtuseks 1. Teisel real hinnatakse avaldist **$x** < **5**. Et vastus sellele küsimusele on 'tõene', alustab täitmise kontroller **while** looksulgudes oleva koodi täitmist ja teeb seda seni, kuni **$x** < **5** on lõpuks 'väär'. Antud juhul täidetakse looksulgudes olevat koodi neli korda, sest iga viienda rea täitmisega suureneb **$x** ühe võrra.

4.5.5 Silmus ˝repeat˝

repeat

```
repeat arv { ... }
```

Täitmise kontroller **repeat** on üpris sarnane kontrolleriga **while**. Erinevus seisab selles, et **repeat** kordab sulgudes olevat koodi määratud arv kordi.

4.5.6 Arvestussilmus ˝for˝

for

```
for muutuja = arv to arv { ... }
```

for on 'arvestussilmus', mis teeb midagi teatud arv kordi. Esimene arv määrab muutujale esimese silmuse väärtuse. Iga silmusega arv suureneb, kuni jõuab teise arvuni.

```
for $x = 1 to 10 {
  print $x * 7
  forward 15
}
```

Igal looksulgudes oleva koodi täitmisel antakse **$x** väärtus ühe võrra suuremana, kuni **$x** võrdub lõpuks 10. Looksulgudes olev kood antud juhul kirjutab tehte '**$x** korda 7' tulemuse. Kui programm on töö lõpetanud, on tulemuseks lõuendil ilutsev numbri 7 korrutustabelike.
Silmuse sammu vaikesuurus on 1, aga kasutada võib mis tahes väärtust.

```
for muutuja = arv to arv step arv { ... }
```

4.5.7 Silmusest väljumine

break

```
break
```

Katkestab otsekohe aktiivse silmuse ning annab järje üle vahetult silmuse järel seisvale lausele.

4.5.8 Programmi täitmise peatamine

exit

```
exit
```

Lõpetab programmi täitmise.

4.5.9 Eelduse kontrollimine käitusajal

assert

```
assert tõeväärtus
```

Seda saab kasutada programmi või sisendi korrektsuse kontrollimiseks.

```
$in = ask "Mis on sinu sünniaasta?"
# aasta peab olema positiivne
assert $in
> 0
```

4.6 Omaenda käskude loomine käsuga 'learn'

Käsk **learn** on väga eriline, sest selle abil saab luua omaenda käske. Sinu loodud käsk võib kasutada nii sisendit kui anda väljundit. Vaatamegi nüüd, kuidas uus käsk luua.

```
learn ringjoon $x {
  repeat 36 {
    forward $x
    turnleft 10
  }
}
```

Uus käsk kannab nime **ringjoon**. **ringjoon** kasutab sisendina argumenti, mis määrab ringjoone suuruse. **ringjoon** ei tagasta mingit väljundit. Nüüd võime käsku **ringjoon** kasutada täiesti tavalise käsuna, nagu selles näites:

```
learn ringjoon $X {
  repeat 36 {
    forward $X
    turnleft 10
  }
}

go 200,200
ringjoon 20

go 300,200
ringjoon 40
```

Järgmises näites aga loome väljundväärtusega käsu.

```
learn faculty $x {
  $r = 1
  for $i = 1 to $x {
    $r = $r * $i
  }
  return $r
}

print faculty 5
```

Selles näites on uue käsu nimeks **faculty**. Kui käsu sisend on on **5**, siis väljund on **5*4*3*2*1**. Käsk **return** määrab kindlaks väärtuse ning täitmisega tagastataksegi väljundväärtus.

Käskudel võib olla enam kui üks sisend. Järgmises näites luuakse käsk, mis toob kaasa ristküliku joonistamise.

```
learn box $x, $y {
  forward $y
  turnright 90
  forward $x
  turnright 90
  forward $y
  turnright 90
  forward $x
  turnright 90
}
```

Nüüd võid käivitada **box 50, 100** ning kilpkonn joonistab lõuendile ristküliku.

Peatükk 5

Sõnastik

Selles peatükis selgitame kõige 'ebatavalisemaid' mõisteid, mida käsiraamat sisaldab.

kraadid

Kraadid mõõdavad nurka ehk pöörde ulatust. Täispööre on 360 kraadi, järelikult poolpööre 180 ja veerandpööre 90 kraadi. Käskude **`turnleft`**, **`turnright`** ja **`direction`** sisend peab olema antud kraadides.

käskude sisend ja väljund

Mõned käsud vajavad sisendit, mõned annavad väljundit, mõned vajavad sisendit ja annavad väljundit ning mõned ei vaja sisendit ega anna ka mingit väljundit.

Mõned näited ainult sisendiga käskudest:

```
forward 50
pencolor 255,0,0
print "tere!"
```

Käsk **`forward`** võtab sisendiks **`50`**. **`forward`** vajab seda sisendit määramaks, mitu pikslit edasi liikuda. **`pencolor`** võtab sisendiks värvikoodi ja **`print`** stringi. Pane tähele, et sisend võib olla ka konteiner, nagu näha järgnevas näites:

```
$x = 50
print $x
forward 50
$str = "tere!"
print $str
```

Nüüd mõned näited väljundit andvate käskude kohta:

```
$x = ask "Palun kirjuta midagi ja vajuta OK... aitäh!"
$r = random 1,100
```

Käsk **`ask`** võtab stringi sisendiks ja annab väljundiks sisestatud arvu või stringi. Nagu näha, on **`ask`** väljund salvestatud konteineris **`x`**. Ka käsk **`random`** annab väljundi. Antud juhul on väljundiks number vahemikus 1 kuni 100. Ka see väljund on salvestatud konteinerisse, milleks siin on **`r`**. Pane tähele, et konteinereid **`x`** ja **`r`** ei ole kasutatud eespool toodud näidiskoodis.

Mõned käsud ei vaja sisendit ega anna ka väljundit, näiteks:

```
clear
penup
```

intuitiivne esiletõstmine

See KTurtle'i omadus muudab koodi kirjutamise veelgi kergemaks. Esiletõstmine annab kirjutatud koodi osadele erinevad värvid, mis märgivad seda, millist tüüpi koodiga kuskil tegemist on. Järgnevas tabelis on ära toodud erinevad kooditüübid ja värv, millega selliseid osi näitab redaktor.

tavalised käsud	tumesinine	Tavalisi käske kirjeldatakse siin.
täitmist kontrollivad käsud	must (rasvane)	Nende spetsiaalsete käskude täitmist kontrollivate käskude kohta saab põhjalikumalt lugeda siit.
kommentaarid	hall	Kommenteeritud ridade alguses seisab vastav sümbol (#). Neid ridu koodi täitmisel ignoreeritakse. Kommentaarid lubavad programmeerijal oma koodi selgitada või siis ajutiselt teatud koodiosa täitmata jätta.
looksulud {, }	tumeroheline (rasvane)	Nurksulge kasutatakse koodiosade grupeerimiseks. Tihtipeale pruugitakse neid koos täitmise kontrolleritega.
käsk learn	heleroheline (rasvane)	Käsuga learn saab luua uusi käske.
stringid	punane	Ega ka (teksti)stringide kohta pole palju muud öelda, kui et nende alguses ja lõpus on alati topeltjutumärgid (˝).
numbrid	tumepunane	Numbrid on numbrid, mis nende kohta ikka rohkem öelda...
tõeväärtused	tumepunane	Tõeväärtusi on kõigest kaks: tõene (true) ja väär (false).
muutujad	purpurne	Algavad märgiga '$' ja võivad sisaldada arve, stringe või tõeväärtusi.
matemaatilised tehted	hall	Matemaatilised tehted on järgmised: **+**, **-**, *****, **/** ja ^.
võrdlustehted	helesinine (rasvane)	Võrdlustehted on järgmised: **==**, **!=**, **<**, **>**, **<=** ja **>=**.
tõeväärtustehted	roosa (rasvane)	Tõeväärtustehted on järgmised: **and**, **or** ja **not**.
tavaline tekst	must	

Tabel 5.1: Erinevad kooditüübid ja nende esiletõstmise värv

pikslid

Piksel on punkt ekraanil. Kui vaatad väga teraselt, näed, et kogu sinu monitori ekraan on piksleid täis. Kõik ekraanil olevad kujutised õigupoolest pikslitest koosnevadki. Piksel ongi kõige väiksem asi, mida saab ekraanile joonistada.

Paljud käsud vajavad sisendina pikslite arvu: **`forward`**, **`backward`**, **`go`**, **`gox`**, **`goy`**, **`canvassize`** ja **`penwidth`**.

KTurtle'i varasemates versioonides oli lõuend põhimõtteliselt rasterpilt, nüüd aga on see vektorjoonistus. See tähendab, et lõuendit saab suurendada ja vähendada, sest piksel ei pea tingimata vastama ühele punktile ekraanil.

RGB värvikoodid

RGB värvikoodid tähistavad värve. 'R' tähendab punast (inglise keeles 'red'), 'G' rohelist ('green') ja 'B' sinist ('blue'). RGB värvikood on näiteks **`255,0,0`**, kus esimene ehk punane väärtus on 255 ja teised 0, mis kokku annab erepunase. Iga RGB värvikoodi väärtus peab olema vahemikus 0 kuni 255. Toome siin ära mõned sagedamini kasutust leidvad värvid:

`0,0,0`	must
`255,255,255`	valge
`255,0,0`	punane
`150,0,0`	tumepunane
`0,255,0`	roheline
`0,0,255`	sinine
`0,255,255`	helesinine
`255,0,255`	roosa
`255,255,0`	kollane

Tabel 5.2: Sagedasemad RGB värvikoodid

Kaks käsku vajavad sisendina RGB värvikoodi: **`canvascolor`** ja **`pencolor`**.

sprait

Sprait on väike pilt, mida saab ekraanil liigutada. Nii on näiteks ka meie armas pisike kilpkonnake sprait.

Märkus: KTurtle'i praeguses versioonis saab sprait olla ainult kilpkonn, aga tulevastes versioonides on loodetavasti võimalik kilpkonn asendada just sulle meelepärase spraidiga.

Peatükk 6

KTurtle sinu emakeeles

Nagu juba arvatavasti oled aru saanud, saab KTurtle programmeerimiskeelt TurtleScript ka tõlkida. See lihtsustab eriti noorematel õpilastel tunduvalt programmeerimise aluste omandamist.

KTurtle tõlkimisel uude keelde leiab standardsetes .pot-failides, mida KDE kasutab tõlgete pakkumiseks, lisaks kasutajaliidese kirjetele ka programmeerimiskäske, näiteid ja veateateid. Kõiki neid tuleb tõlkida nii, nagu KDE-s ikka, kuid siiski on äärmiselt soovitatav eelnevalt tundma õppida, kuidas neid viimaseid korrektselt tõlkida.

Tõlkimisest kõneldakse põhjalikumalt aadressil http://edu.kde.org/kturtle/translator.php. Täname sind su panuse eest juba ette!

Peatükk 7

Autorid ja litsents

KTurtle
Tarkvara autoriõigus 2003-2007: Cies Breijs cies AT kde DOT nl

Dokumentatsiooni autoriõigus 2004, 2007, 2009:

- Cies Briej cies AT kde DOT nl
- Anne-Marie Mahfouf annma AT kde DOT org
- Mõningane korrektuur: Philip Rodrigues phil@kde.org
- Tõlkimise HOWTO uuendamine ja osaline korrektuur: Andrew Coles andrew_coles AT yahoo DOT co DOT uk

Tõlge eesti keelde: Marek Laane bald@starman.ee

Lisa A

Paigaldamine

A.1 KTurtle'i hankimine

KTurtle on osa KDE projektist http://www.kde.org/ .

KTurtle asub paketis kdeedu KDE projekti peamises FTP saidis ftp://ftp.kde.org/pub/kde/ .

A.2 Kompileerimine ja paigaldamine

Et KTurtle oma süsteemis kompileerida ja paigaldada, anna KTurtle baaskataloogis järgmised käsud:

```
% ./configure
% make
% make install
```

Kuna KTurtle kasutab **autoconf**'i ja **automake**'i, ei tohiks kompileerimisel probleeme esineda. Kui neid siiski ette tuleb, anna sellest palun teada KDE meililistides.

Lisa B

Indeks