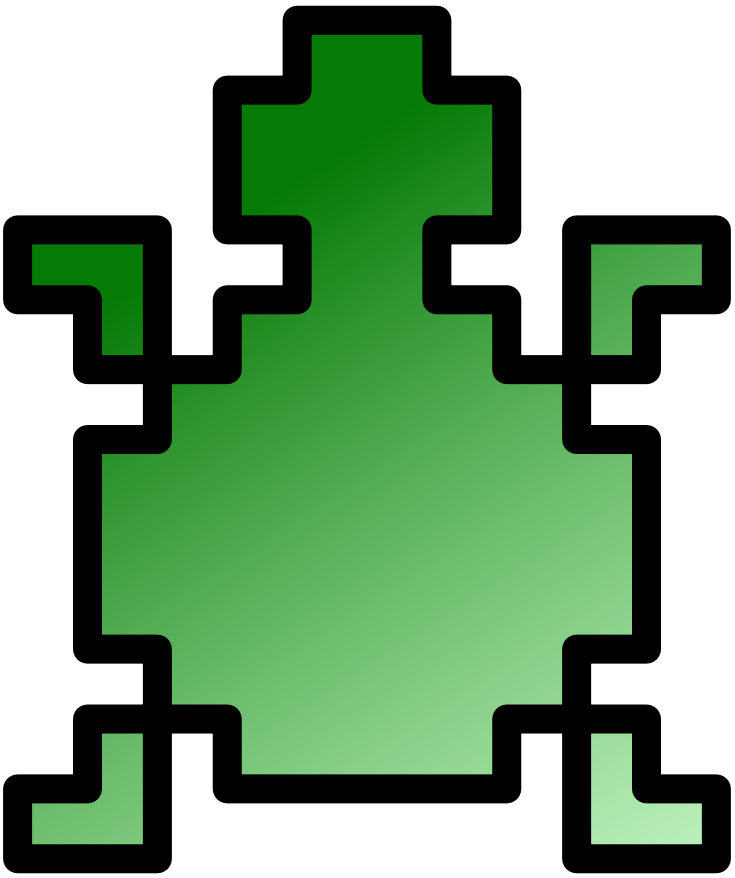
Het handboek van KTurtle

Cies Breijs
Anne-Marie Mahfouf
Mauricio Piacentini
: Jaap Woldringh



Het handboek van KTurtle

Inhoudsopgave

Lijst van tabellen

Samenvatting

KTurtle is een educatieve programmeeromgeving, met als doel het leren programmeren zo eenvoudig mogelijk te maken. Om dit doel te bereiken zijn in KTurtle alle programmeerhulpmiddelen beschikbaar in de interface. De gebruikte programmeertaal is TurtleScript, waarvan de commando's vertaald kunnen worden.

Hoofdstuk 1

Inleiding

KTurtle is een educatieve programmeeromgeving waarin gebruik wordt gemaakt van de programmeertaal TurtleScript, een programmeertaal die enigszins lijkt op en geïnspireerd is door Logo. Het doel van KTurtle is het programmeren zo eenvoudig en toegankelijk mogelijk te maken. Dit maakt KTurtle geschikt om kinderen de basiskennis van wiskunde, meetkunde en... programmeren bij te brengen. Een belangrijke eigenschap van TurtleScript is dat de commando's kunnen worden vertaald naar de spreektaal van de programmeur.

KTurtle is genoemd naar 'de schildpad' (turtle) die een centrale rol speelt in de programmeeromgeving. De gebruiker programmeert de schildpad, met behulp van de TurtleScript-commando's, om een tekening te maken op het canvas.

1.1 Wat is TurtleScript?

TurtleScript, de programmeertaal die in KTurtle wordt gebruikt, is geïnspireerd op de Logo-familie van programmeertalen. De eerste versie van Logo werd in 1967 gemaakt door Seymour Papert van MIT Artificial Intelligence Laboratory, als een afgeleide van de programmeertaal LISP. Vanaf dat moment zijn er veel versies van Logo uitgebracht. Omstreeks 1980 werd Logo steeds meer bekend met versies voor MSX, Commodore, Atari, Apple II en IBM-PC's. Deze versies werden vooral voor educatieve doeleinden uitgebracht. MIT onderhoudt nog steeds een website over Logo. Deze vindt u op http://el.media.mit.edu/logo-foundation/ . Hier vindt u een lijst van een aantal populaire implementaties van de taal.

TurtleScript heeft een eigenschap gemeen met vele andere implementaties van Logo namelijk dat de commando's naar de moedertaal van de leerling kunnen worden vertaald. Dit maakt het gemakkelijker om er mee te werken door leerlingen die weinig of geen Engels kennen. Hiernaast heeft KTurtle vele andere eigenschappen met het doel de eerste programmeerervaring van de leerlingen gemakkelijker te maken.

1.2 Eigenschappen van KTurtle

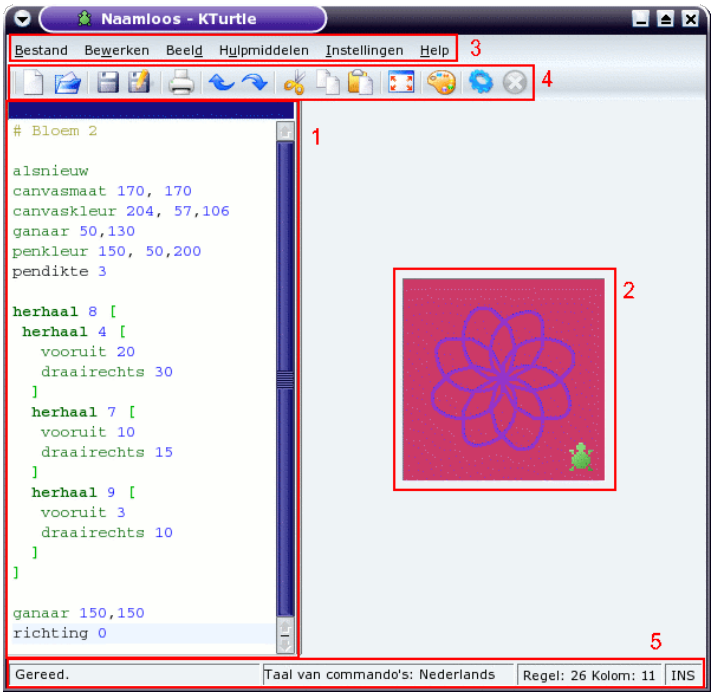
KTurtle heeft enkele leuke eigenschappen die het beginnen met programmeren eenvoudig maken. Hier volgen enkele van de belangrijkste eigenschappen van KTurtle:

- Een geïntegreerde omgeving, met de TurtleScript-interpreter (programma dat de programmacode vertaalt naar machinecode), codebewerker, canvas en andere hulpmiddelen, alle in een applicatie (zonder extra afhankelijkheden).

- De mogelijkheid de commando's van TurtleScript te vertalen, waarbij de vertaalmogelijkheden van KDE worden benut.
- TurtleScript ondersteunt door de gebruiker gemaakte functies, recursie en het dynamisch wijzigen van het (gegevens)type.
- De uitvoering kan op elk moment worden vertraagd, tijdelijk onderbroken of gestopt.
- Een krachtige codebewerker, met intuïtieve syntaxisaccentuering, regelnummering, merkpunten voor fouten, visuele uitvoering, en meer.
- Het canvas waarop de schildpad tekent, kan worden afgedrukt, of opgeslagen als een afbeelding (PNG) of tekening (SVG).
- Contextgevoelige hulp: hulp daar waar die nodig is. Druk op **F2** (of zie **Help** → **Hulp bij: ...**) om hulp te krijgen bij de programmacode onder de muisaanwijzer.
- Een foutdialoog die de foutmeldingen koppelt aan de fouten in het programma, en die met rood markeert.
- Vereenvoudigde programmeerterminologie.
- Geïntegreerde voorbeeldprogramma's die het makkelijker maken om te beginnen. Deze voorbeelden zijn vertaald, gebruikmakend van de vertaalmogelijkheden van KDE.

Hoofdstuk 2

Hoe KTurtle te gebruiken



Het hoofdvenster van KTurtle bestaat uit drie onderdelen: links de codebewerker (1) waarin de commando's voor TurtleScript worden getypt, rechts het canvas (2), waar de schildpad tekent, en de inspectie (3) waarin informatie staat tijdens uitvoeren van je programmacode. Bovendien vind je er de menubalk (5), waarin je alle acties kunt doen, de werkbalk (4) waarin je de meest gebruikte acties kunt doen, de **Console**, waarin je een regel met een commando kunt toetsenom die uit te proberen, en de statusbalk (aan de onderkant van het venster), met daarin wat informatie over de toestand van KTurtle.

2.1 De codebewerker

In de codebewerker type je de commando's van TurtleScript. De meeste functies van de code bewerker vind je in de menu's Bestand en Bewerken. De codebewerker kan op elke rand van het scherm worden ge˝plakt˝, of op elke plaats op het bureaublad worden geplaatst.

Er zijn verschillende manieren om code in de bewerker te krijgen. Het eenvoudigste is met een voorbeeld: kies **Bestand** → **Voorbeelden**, en selecteer een voorbeeld. Het eerst gekozen voor-

beeld wordt in de bewerker geopend, waarna u de opdracht**Laten werken** → **Laten werken** kunt geven (sneltoets: **Alt**+**F2**) of de knop **Laten werken** in de taakbalk als u hetprogramma wilt zien wat het programma doet.

Je kunt TurtleScript-bestanden openen met behulp van de menuoptie **Bestand** → **Openen...**.

De derde manier is zelf je eigen code in de codebewerker in te typen, of door programmacode te kopiëren en te plakken.

2.2 Het canvas

Het canvas is het speelterrein van de schildpad, waarin de schildpad tekent door de commando's die hij krijgt. Nadat er wat programmacode staat in de codebewerker, die wordt uitgevoerd, kunnen er twee dingen gebeuren: of de code wordt goed uitgevoerd, je zal dan hoogst waarschijnlijk iets zien gebeuren op het canvas; of je hebt een fout gemaakt, er komt dan een foutmelding waarin wordt uitgelegd welke fout er is gemaakt.

Met je muiswiel kun je op het canvas in- en uitzoomen.

2.3 De inspectie

In de inspectie krijg je informatie over de variabelen tijdens het lopen van je programmacode.

De inspectie kan worden vast˝geplakt˝ op elke zijde van het hoofdscherm, of hij kan worden losgemaakt en op elke plaats op het bureaublad worden geplaatst.

2.4 De werkbalk

Hierin kan je de meestgebruikte acties vinden. In de taakbalk vind je de **Console**, waarin je snel even commando's kunt uitproberen. Dit is nuttig als je een commando wilt proberen zonder de programmacode in de codebewerker te veranderen.

Je kunt de werkbalk zelf instellen met **Instellingen** → **Werkbalken instellen...**

2.5 De menubalk

In de menubalk vind je alle acties in KTurtle. Zij zijn ingedeeld in de volgende groepen: **Bestand**, **Bewerking**, **Canvas**, **Uitvoeren**, **Hulpmiddelen**, **Instellingen**, en **Help**. In dit deel worden ze allemaal beschreven.

2.5.1 Het menu Bestand

Bestand → **Nieuw (Ctrl-N)**

Maakt een nieuw, leeg TurtleScript-bestand aan.

Bestand → **Openen... (Ctrl-O)**

Opent een bestaand TurtleScript-bestand.

Bestand → **Recent geopend**

Opent een TurtleScript-bestand dat je eerder hebt geopend.

Bestand → **Voorbeelden**

Opent voorbeeldprogramma's voor TurtleScript. De voorbeelden zijn in je eigen taal, die je kunt kiezen in in menu **Instellingen** → **Scripttaal...**.

Bestand → **Meer voorbeelden ophalen...**

Open de dialoog **Vers van de pers** om extra TurtleScript-bestanden van het internet te downloaden.

Bestand → **Opslaan (Ctrl-S)**

Bewaart het momenteel geopende TurtleScript-bestand.

Bestand → **Opslaan als... (Ctrl-Shift-S)**

Bewaart het momenteel geopende TurtleScript-bestand op een opgegeven plaats.

Bestand → **Exporteren naar HTML...**

Exporteert de huidige inhoud van de codebewerker naar een HTML-bestand, inclusief de accentueringskleuren.

Bestand → **Afdrukken... (Ctrl-P)**

Drukt de huidige programmacode in de codebewerker af.

Bestand → **Afsluiten (Ctrl-Q)**

Beëindigt KTurtle.

2.5.2 Het menu Bewerken

Bewerken → **Ongedaan maken (Ctrl-Z)**

Maakt de laatste wijzigingen in de programmacode ongedaan. In KTurtle kun je onbeperkt veel wijzigingen weer ongedaan maken.

Bewerken → **Opnieuw (Ctrl-Shift-Z)**

Voert een wijziging in de programmacode die ongedaan is gemaakt, opnieuw door.

Bewerken → **Knippen (Ctrl-X)**

Knipt de in de codebewerker geselecteerde tekst, en kopieert die naar het klembord.

Bewerken → Kopiëren (Ctrl-C)
Kopieert de geselecteerde tekst in de codebewerker naar het klembord.

Bewerken → Plakken (Ctrl-V)
Plakt de tekst op het klembord in de codebewerker.

Bewerken → Alles selecteren (Ctrl-A)
Selecteert alle tekst in decodebewerker.

Bewerken → Zoeken... (Ctrl-F)
Hiermee kun je zoeken naar tekst in de programmacode.

Bewerken → Volgende zoeken (F3)
Hiermee kun je verder zoeken naar je zoekterm.

Bewerken → Vorige zoeken (F3)
Hiermee kun je de vorige plaats vinden waar je zoekterm in de tekst voorkomt.

Bewerken → Invoegen (Ins)
Schakelen tussen 'Invoegen' en 'Overschrijven'.

2.5.3 Het menu Canvas

Canvas → Exporteren naar afbeelding (PNG)...
Exporteert de inhoud van het Canvas als een raster-afbeelding van het type PNG.

Canvas → Exporteren naar tekening (SVG)...
Exporteert de inhoud van het Canvas als een SVG-tekening (Scalable Vector Graphics).

Canvas → Druk canvas af...
Drukt de inhoud van het huidige Canvas af.

2.5.4 Het menu Uitvoeren

Uitvoeren → Uitvoeren (F5)

Start het uitvoeren van de commando's in de codebewerker.

Uitvoeren → Pauze (F6)

Onderbreekt het uitvoeren van de programmacode. Deze actie is alleen actief tijdens het uitvoeren van programmacode.

Uitvoeren → Afbreken (F7)

Stopt het uitvoeren van de programmacode. Deze actie is alleen actief tijdens het uitvoeren van programmacode.

Uitvoeren → Snelheid...

Presenteert een lijst van mogelijke uitvoersnelheden, waarin: **Volle snelheid (geen markering in de code en geen inspector)**, **Volle snelheid**, **Langzaam**, **Langzamer**, **Erg langzaam** en **Stap-voor-stap**. Als de uitvoersnelheid **Volle snelheid** is (standaard) zien we nauwelijks wat er gebeurt. Soms is dit gewenst, maar soms willen het uitvoeren kunnen volgen. In dat geval kan de snelheid worden ingesteld op **Langzaam**, **Langzamer** of **Erg langzaam**. Wanneer een van de langzame snelheden is geselecteerd wordt het verloop van het programma in de codebewerker getoond. Bij **Stap-voor-stap** wordt er steeds één commando uitgevoerd.

2.5.5 Het menu Hulpmiddelen

Hulpmiddelen → Richting kiezen...

Hiermee wordt de dialoog geopend voor het kiezen van een richting.

Hulpmiddelen → Kleur kiezen...

Hiermee wordt de dialoog geopend voor het kiezen van een kleur.

2.5.6 Het menu Instellingen

Instellingen → Scripttaal

Kies de taal voor de programmacode.

Instellingen → Codebewerker tonen (Ctrl-E)

Tonen/verbergen van de Codebewerker.

Instellingen → Inspectie tonen (Ctrl-I)

Tonen/verbergen van de the inspectie.

Instellingen → Fouten tonen

Tonen/verbergen van het tabblad **Fouten** met een lijst van fouten die optreden bij het uitvoeren van de programmacode. Als deze optie aan staat, moet je op **Canvas** klikken om de schildpad terug te krijgen.

Instellingen → Regelnummers tonen (F11)

Hiermee zie je regelnummers in de codebewerker. Dit kan helpen bij het vinden van fouten.

Instellingen → Werkbalk tonen

Tonen/verbergen vande Hoofdwerkbalk

Instellingen → Statusbalk tonen

Tonen/verbergen van de Statusbalk

Instellingen → Sneltoetsen instellen...

Standaard dialoog van KDE voor het instellen van de sneltoetsen.

Instellingen → Werkbalken instellen...

Standaard dialoog van KDE voor het instellen van de werkbalken.

2.5.7 Het menu Help

KTurtle heeft een standaard KDE **Help**-menu zoals beschreven in de KDE Basisdocumentatie, met nog een extra ingang:

Help → Hulp bij: ... (F2)

Dit is een handige functie, en geeft informatie over de programmacode op de plaats van de cursor in de codebewerker. Stel, je hebt de opdracht **`druk`** in je code staan, en je wilt weten wat in het handboek over deze opdracht staat. Zet dan je muisaanwijzer op de opdracht **`druk`**, en druk op de toets **F2**. Het handboek toont dan alle informatie over de opdracht **`druk`**.

Deze functie is erg belangrijk bij het leren programmeren in TurtleScript.

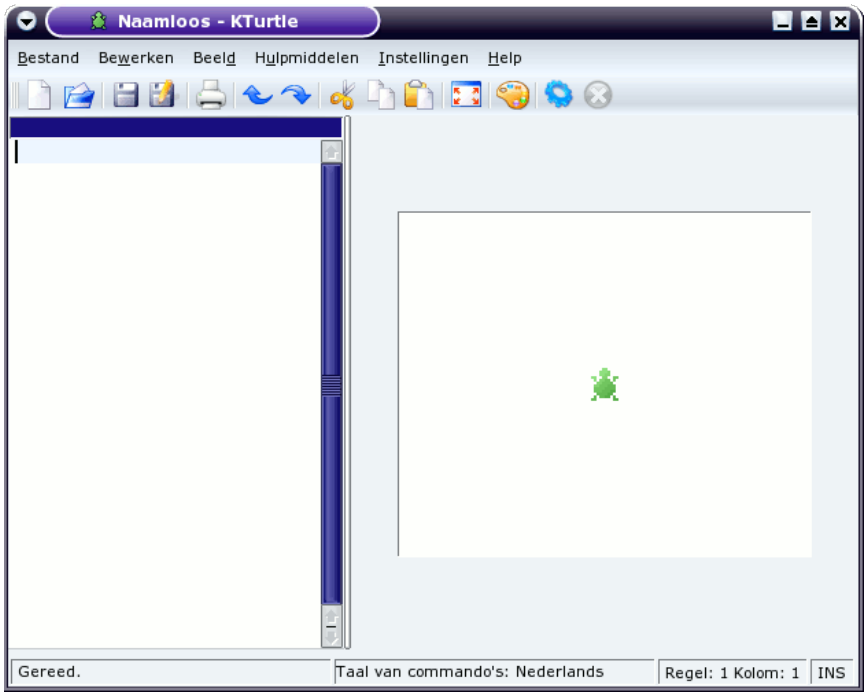
2.6 De statusbalk

In de statusbalk vind je informatie over de status (toestand) van KTurtle. Links zie je de informatie over de laatste actie. Rechts vind je de huidige locatie van de cursor (regel- en kolomnummers). In het midden van de statusbalk staat wat de huidige taal is die gebruikt wordt voor de opdrachten.

Hoofdstuk 3

Aan de slag

Als je KTurtle start, dan zie je iets dat hier op lijkt:



In deze beginnersgids wordt aangenomen dat de TurtleScript-opdrachten in het Engels zijn. Je kunt dit veranderen in het menu **Instellingen** → **Scripttaal**. Let erop dat de taal die je hier instelt voor KTurtle de taal is waarin je de TurtleScript-opdrachten typt, en niet de taal is die door KDE op je computer wordt gebruikt, in de interface en menu's van KTurtle.

3.1 De eerste stappen met TurtleScript: ontmoet de schildpad!

Je hebt vast het kleine schildpadje gezien in het midden van het canvas. Je staat op het punt te leren hoe je die kunt bedienen met behulp van commando's in de codebewerker.

3.1.1 De schildpad beweegt

Laten we beginnen bij het laten bewegen van de schildpad. Onze schildpad kan op 3 manieren bewegen: (1) vooruit en achteruit, (2) naar links en naar rechts en (3) hij kan direct naar een (andere) positie op het scherm gaan. Probeer bijvoorbeeld dit eens:

```
vooruit 100
draailinks 90
```

Type of kopieer en plak de code in de bewerker en voer deze uit (gebruik **Uitvoeren** → **Uitvoeren**) en bekijk het resultaat.

Als je de commando's zoals hierboven in de codebewerker hebt ingetypt en uitgevoerd, dan heb je een of meer van de volgende acties gezien:

1. Dat — na het uitvoeren van de commando's — de schildpad omhoog gaat, een lijn tekent en een kwartslag (90 graden) naar links draait. Dit doet hij omdat je de commando's **`vooruit`** en **`draailinks`** hebt gegeven.
2. Dat de kleur van de code veranderde terwijl je die intypte: deze functie heet *intuïtieve accentuering* — verschillende soorten commando's worden in verschillende kleuren geaccentueerd. Dit maakt het lezen van grote stukken programmacode heel wat eenvoudiger.
3. Dat de schildpad een dunne zwarte lijn heeft getekend.
4. Mogelijk kreeg je een foutmelding. Dit kan twee dingen betekenen: je hebt een fout gemaakt bij het kopiëren van de commando's, of je hebt de juiste taal voor de TurtleScript-commando's nog niet goed ingesteld. Je kunt de taal instellen via het menu **Instellingen** → **Scripttaal**.

Je begrijpt wel dat het commando **`vooruit 100`** de schildpad vooruit laat gaan en een lijn tekenen, en dat **`draailinks 90`** de schildpad 90 graden naar links laat draaien.

Bekijk de volgende verwijzingen naar de programmahandleiding voor een complete uitleg van deze commando's: **`vooruit`**, **`achteruit`**, **`draailinks`**, and **`draairechts`**.

3.1.2 Meer voorbeelden

Het eerste voorbeeld was vrij eenvoudig, dus laten we verder gaan!

```
canvasmaat 200,200
canvaskleur 0,0,0
penkleur 255,0,0
pendikte 5

ganaar 20,20
richting 135

vooruit 200
draailinks 135
vooruit 100
draailinks 135
vooruit 141
draailinks 135
vooruit 100
draailinks 45

ganaar 40, 100
```

Ook hier kun je de programmacode intypen of kopiëren en plakken in de codebewerker, of door het voorbeeld `pijl` in het menu **Voorbeelden** te openen en dat uit te voeren (gebruik **Uitvoeren → Uitvoeren**) en het resultaat te bekijken. In de volgende voorbeelden wordt verwacht dat je weet hoe dit moet.

Je hebt vast gemerkt dat er in het tweede voorbeeld veel meer programmacode is. Ook heb je enkele nieuwe commando's gezien. Hier is een korte uitleg van alle nieuwe commando's:

Na het commando **`alsnieuw`** wordt alles weer als toen KTurtle net was gestart (ja, inderdaad, als nieuw).

`canvasmaat 200,200` zet de breedte en hoogte van het canvas (het speelterrein voor de schildpad) op 200 pixels. De breedte en hoogte zijn gelijk aan elkaar, dus krijgen we een vierkant canvas.

`canvaskleur 0,0,0` maakt het canvas zwart. **`0,0,0`** is een zogenaamde RoodGroenBlauw-kleurcombinatie (RGB). Door alle waarden op **`0`** te zetten krijg je een zwart resultaat.

`penkleur 255,0,0` maakt de kleur van de pen rood. **`255,0,0`** is een RGB-kleurcombinatie waarbij alleen de rode waarde op **`255`** (volledig aan) is ingesteld. De andere waarden (groen en blauw) krijgen de waarde **`0`** (volledig uit). Dit geeft een heldere kleur rood.

Als je de kleurwaarden niet begrijpt, lees dan even in de woordenlijst wat RGB-combinaties zijn

`pendikte 5` zet de dikte van de pen op **`5`** pixels. Van nu af zal elke lijn die de schildpad tekent de dikte **`5`** hebben, totdat we de **`pendikte`** weer veranderen.

`ganaar 20,20` vertelt de schildpad dat hij naar een bepaalde plek op het canvas moet gaan. Gerekend vanuit de linker bovenhoek ligt deze plek 20 pixels naar rechts, en 20 pixels naar beneden. Je ziet dat met het commando **`ganaar`** de schildpad geen lijn tekent.

`richting 135` bepaalt de richting van de schildpad. De commando's **`draailinks`** en **`draai rechts`** draaien de schildpad naar een andere richting. Het commando **`richting`** verandert de hoek van de schildpad vanuit positie nul, en is dus niet afhankelijk van de vorige richting van de schildpad.

Na het commando **`richting`** volgen een aantal **`vooruit`**- en **`draailinks`**-commando's. Met deze commando's wordt de tekening gemaakt.

Tot slot wordt het commando **`ganaar`** gebruikt om de schildpad naar ergens anders te verplaatsen.

Volg alle verwijzingen naar de programmeerhandleiding. Hier vind je een grondige uitleg van elk commando.

Hoofdstuk 4

Programmeren met TurtleScript

Dit is het overzicht van TurtleScript van KTurtle. In het eerste deel van dit hoofdstuk kijken we naar enkele aspecten van de grammatica van TurtleScript-programma's. Het tweede deel behandelt uitsluitend rekenkundige bewerkingen, booleaanse (waar/onwaar) bewerkingen en vergelijkingsbewerkingen. Het derde deel is eigenlijk een lange lijst van alle opdrachten, die een voor een worden uitgelegd. In deel vier wordt uitgelegd hoe waarden kunnen worden toegekend (gegeven) aan variabelen. Tenslotte leggen we uit hoe de uitvoering van de opdrachten kan worden gestuurd met programmastuuropdrachten in deel vijf en hoe zelf opdrachten te maken met leer in deel zes.

4.1 De grammatica van TurtleScript

Zoals elke taal heeft ook TurtleScript verschillende soorten woorden en tekens. In het Nederlands kennen we werkwoorden (zoals 'lopen' of 'zingen') en zelfstandige naamwoorden (zoals 'zuster' of 'huis'), zij worden verschillend gebruikt. TurtleScript is een programmeertaal, en wordt gebruikt om KTurtle te vertellen wat er moet gebeuren.

In dit deel worden de verschillende soorten woorden en tekens in TurtleScript kort uitgelegd. We leggen uit wat commentaar is, opdrachten en de drie verschillende soorten van waarden: getallen, tekenreeksen en booleaanse (waar/onwaar) waarden.

4.1.1 Commentaar

Een programma bestaat uit instructies die worden uitgevoerd als het programma loopt, én zogenoemd commentaar. Commentaar wordt niet uitgevoerd, maar wordt door KTurtle domweg genegeerd bij het uitvoeren van het programma. Commentaar dient om andere programmeurs (en je zelf, op een later tijdstip) je programma beter te laten begrijpen. Alles dat op een regel volgt na een hekje (**#**), wordt in TurtleScript opgevat als commentaar.

```
# Dit kleine programma bevat alleen commentaar en doet dus helemaal niets!
```

. Het is wat nutteloos, maar geeft een en ander goed weer.

Commentaar is erg nuttig wanneer een programma wat ingewikkelder gaat worden. Hierin kan aan andere programmeurs iets worden verteld. In het volgende programma zie je hoe commentaar wordt gebruikt bij het commando druk.

```
# dit programma werd gemaakt door Cies Breijs.
druk "deze tekst wordt op het canvas afgedrukt"
# de vorige regel was geen commentaar,maar de volgende wel:
# druk "deze tekst wordt niet afgedrukt!"
```

. De eerste regel geeft wat toelichting bij het programma. De tweede wordt door KTurtle uitgevoerd, en drukt **deze tekst wordt op het canvas afgedrukt** af op het canvas. De derde regel bevat commentaar. En de vierde regel bevat commentaar meteen stukje TurtleScript. Als het teken **#** dat ervoor staat wordt weggehaald, zal KTurtle ook deze regel uitvoeren, maar met het hekje ervoor niet. Door een hekje ervoor wordt een programmaregel dus uitgeschakeld.

Commentaarregels worden geaccentueerd met lichtgrijs in de codebewerker.

4.1.2 Opdrachten

Met opdrachten kun je de schildpad of KTurtle opdragen iets te doen. Bij sommige opdrachten is invoer nodig, en andere geven uitvoer.

```
# vooruit is een commando dat invoer nodig heeft, in dit geval het getal ↩
    100:
vooruit 100
```

De eerste regel is commentaar. De tweede bevat het commando **vooruit**, en de invoer is het getal **100**. Het getal wordt niet beschouwd als een deel van het commando, maar als invoer.

Voor sommige opdrachten bijv. **ganaar** zijn meerdere invoerwaarden nodig. Deze moeten door een **,** (komma) worden gescheiden.

Meer informatie over alle opdrachten die KTurtle van huis uit kent vind je hier. Deze ingebouwde opdrachten worden met donkerblauw geaccentueerd

4.1.3 Getallen

Getallen ken je vast wel. De manier waarop getallen worden gebruikt in KTurtle is niet veel anders dan in een spreektaal, of bij het rekenen. Maar er is wel een verschil: ze worden niet met komma's, maar met punten geschreven. Dit is omdat met een komma twee getallen van elkaar worden gescheiden, zodat 2,5 twee getallen, 2 en 5, zijn. Je moet dus 2.5 schrijven, als je maar één getal bedoelt.

Er zijn ook de natuurlijke getallen: **0**, **1**, **2**, **3**, **4**, **5**, etc. De negatieve getallen: **-1**, **-2**, **-3**, etc.. En de kommagetallen zoals: **0,1**, **3,14**, **33,3333**, **-5,05**, **-1,0**. De **.** punt is het scheidingsteken voor decimalen: de cijfers achter de komma.

Getallen kunnen worden gebruikt in rekenkundige bewerkingen en vergelijkingsbewerkingen. Ze kunnen ook worden bewaard in variabelen. Getallen worden geaccentueerd in donkerrood.

4.1.4 Tekenreeksen

Eerst een voorbeeld:

```
druk "Hallo, ik ben een tekenreeks."
```

In dit voorbeeld is **druk** een commando en **´´Hallo, ik ben een tekenreeks´´** een tekenreeks. Een tekenreeks begint en eindigt met het teken **´´**. Hierdoor weet KTurtle dat dit een tekenreeks is.

Tekenreeksen kunnen in variabelen worden bewaard, net zoals getallen. Maar, anders dan getallen, kunnen tekenreeksen niet in berekeningen of vergelijkingsbewerkingen worden gebruikt. Tekenreeksen worden met rood geaccentueerd.

4.1.5 Booleaanse waarden (waar/onwaar)

Er zijn maar twee booleaanse waarden: **waar** en **onwaar**. Ze worden ook wel genoemd: '**aan**' en '**uit**', '**ja**' en '**nee**', '**een**' en '**nul**'. Maar in TurtleScript worden zij altijd , **waar** en **onwaar** genoemd. Bekijk dit stukje TurtleScript:

```
$a = waar
```

In de Inspector kun je zien dat de variabele **$a** de waarde **waar** heeft, en booleaans is.

Vaak zijn booleaanse waarden het resultaat van vergelijkingsbewerkingen, zoals in het volgende stukje TurtleScript:

```
$antwoord = 10 > 3
```

De variabele **$antwoord** krijgt de waarde **waar** omdat **10** inderdaad groter is dan **3**.

Booleaanse waarden, **waar** en **onwaar**, worden met donkerrood geaccentueerd.

4.2 Rekenkundige, booleaanse en vergelijkingsbewerkingen

De naam van dit deel klinkt misschien moeilijk, maar het valt allemaal best mee.

4.2.1 Rekenkundige bewerkingen

Dit zijn de basisbewerkingen van het rekenen: optellen (**+**), aftrekken (**−**), vermenigvuldigen (*****), delen (**/**) en machtsverheffen (**^**).

Hier is een klein voorbeeld van het gebruik van rekenkundige bewerkingen in TurtleScript:

```
$optellen                = 1 + 1
$aftrekken              = 20 - 5
$vermenigvuldigen = 15 * 2
$delen                     = 30 / 30
$machtsverheffen   = 2 ^ 2
```

De waarden die deze bewerkingen opleveren worden toegekend aan de verschillende variabelen (die je willekeurige $namen kunt geven). In de inspectie kun je deze waarden zien.

Als je alleen maar een eenvoudige berekening wilt laten doen, dan kan dat bijvoorbeeld zo:

```
druk 2010-12
```

Nu een voorbeeld met haakjes:

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

Het gedeelte tussen de binnenste haakjes wordt het eerst uitgerekend. In dit voorbeeld wordt 20 - 5 berekend, dan wordt de uitkomst vermenigvuldigd met 2, en daarna wordt er gedeeld door 30. Tot slot wordt er 1 bij opgeteld (het antwoord is 2). Haakjes kunnen ook in andere gevallen worden gebruikt.

KTurtle heeft ook nog wat wiskundige eigenschappen in de vorm van opdrachten. Kijk maar eens naar de volgende opdrachten, die, let op, wel wat gevorderd zijn: afronden, gok, wortel , pi, sin, cos, tan, arcsin, arccos, arctan.

4.2.2 Booleaanse bewerkingen (waar/onwaar)

Waar wiskundige bewerkingen vooral werken metgetallen, werken booleaanse bewerkingen met de booleaanse waarden (**waar** en **onwaar**). Er zijn slechts drie booleaanse bewerkingen, namelijk: **en**, **of**, en **niet**. In het volgende stuk TurtleScript kan je zien hoe dit te gebruiken:

```
$and_1_1 = waar en waar          # -> waar
$and_1_0 = waar en onwaar      # -> onwaar
$and_0_1 = onwaar en waar      # -> onwaar
$and_0_0 = onwaar en onwaar  # -> onwaar

$or_1_1 = waar of waar              # -> waar
$or_1_0 = waar of onwaar          # -> waar
$or_0_1 = onwaar of waar          # -> waar
$or_0_0 = onwaar of onwaar      # -> onwaar

$not_1 = niet waar      # -> onwaar
$not_0 = niet onwaar  # -> waar
```

Met behulp van de inspector kan je de waarden zien, maar we geven toch de resultaten aan als commentaar aan het einde van de regels. **en** wordt alleen **waar** als beide booleaanse waarden**waar** zijn. **of** wordt alleen **onwaar**als beide booleaanse waarden **onwaar** zijn. En **niet** verandert een **waar** in **onwaar** en een **onwaar** in **waar**.

Booleaanse bewerkingen worden geaccentueerd met roze.

4.2.2.1 Enkele meer gevorderde voorbeelden

Kijk naar het volgende voorbeeld met **en**:

```
$a = 1
$b = 5
als (($a < 10) en ($b == 5)) en ($a < $b) {
  druk "hallo"
}
```

In dit stukje TurtleScript worden de antwoorden van drie vergelijkingsbewerkingen samengevoegd door bewerkingen met **en**. Dit betekent dat ze alle drie ˝waar˝ moeten zijn, voordat de tekst ˝hallo˝ wordt afgedrukt (op het scherm).

Een voorbeeld met **of**:

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

In dit stukje TurtleScript is links van **of** de vergelijking 'waar', en rechts 'onwaar'. Omdat één van de twee zijden van **of** 'waar' is, is hier het resultaat van **of** de waarde 'waar'. En dus wordt ˝hallo˝ afgedrukt (op het scherm).

En tenslotte een voorbeeld met **niet**, die de waarde 'waar'verandert naar 'onwaar' en 'onwaar' naar 'waar'. Kijk maar:

```
$n = 1
als niet ($n == 3) {
  druk "hallo"
} anders {
  druk "niet hallo ;-)"
}
```

4.2.3 Vergelijkingsbewerkingen

Kijk naar deze eenvoudige vergelijking:

```
$antwoord = 10 > 3
```

Hier wordt **10** vergeleken met **3** met de bewerking 'groter dan'. Het resultaat van deze vergelijking, de booleaanse waarde **waar** wordt bewaard in de variabele **$antwoord**.

Alle getallen en variabelen (die getallen bevatten) kunnen met deze vergelijkingsbewerkingen met elkaar worden vergeleken.

Hier vind je alle vergelijkingsbewerkingen:

$A == $B	gelijk aan	antwoord is is 'waar' als **$A** gelijk is aan **$B**
$A != $B	niet gelijk is	antwoord is 'waar' als**$A** niet gelijk is aan **$B**
$A > **$B**	groter dan	antwoord is 'waar' als **$A** groter is dan **$B**
$A < **$B**	kleiner dan	antwoord is 'waar' als **$A** kleiner is dan **$B**
$A >= $B	groter dan of gelijk aan	antwoord is 'waar' als **$A** groter is dan of gelijk aan **$B**
$A <= $B	smaller than or equals	answer is 'waar' als **$A** kleiner is dan of gelijk aan **$B**

Tabel 4.1: Vragen:

Let erop dat **$A** en **$B** getallen moeten zijn of variabelen met een getalswaarde.

4.3 Opdrachten

Met behulp van opdrachten vertel je de schildpad of KTurtle iets te doen. Sommige opdrachten hebben een invoer nodig, anderen geven een uitvoer. In dit deel zullen we alle ingebouwde opdrachten die in KTurtle kunnen worden gebruikt, uitleggen. Met het commando leer kunnen overigens zelf opdrachten worden gemaakt. De hier besproken ingebouwde (dus oorspronkelijke) opdrachten worden met donkerblauw geaccentueerd

4.3.1 De schildpad bewegen

Er zijn verschillende opdrachten waarmee je de schildpad over het scherm kunt bewegen.

vooruit (vt)

```
vooruit X
```

vooruit X verplaatst de schildpad X pixels vooruit. Als de pen neer is laat de schildpad een spoor achter. **vooruit** kan worden afgekort met **vt**

achteruit (at)

```
achteruit X
```

achteruit X verplaatst de schildpad X pixels achteruit. Als de pen neer is laat de schildpad een spoor achter. **achteruit** kan worden afgekort met **at**.

draailinks (dl)

```
draailinks X
```

draailinks X laat de schildpad X graden naar links draaien. **draailinks** kan worden afgekort met **dl**.

draairechts (dr)

```
draairechts X
```

draairechts X laat de schildpad X graden naar rechts draaien. **draairechts** kan worden afgekort met **dr**.

richting (rt)

```
richting X
```

richting X zet de richting van de schildpad op X graden, geteld vanaf 0 (recht naar boven). Dus niet ten opzichte van de vorige positie van de schildpad. **richting** kan worden afgekort met **rt**.

haalrichting

```
haalrichting
```

haalrichting X geeft de ricting terug van de schildpad, in de vorm van een aantal graden, geteld vanaf nul. Nul is de richting recht naar boven, dus als de schildpad recht naar boven loopt.

thuis

```
thuis
```

thuis plaatst de schildpad in het midden van het canvas.

ganaar

```
ganaar X,Y
```

ganaar plaatst de schildpad op een bepaalde plaats op het canvas. De plaats is op X pixels vanaf de linkerkant en Y pixels vanaf de bovenkant van het canvas.

ganaarx

```
ganaarx X
```

ganaarx X laat de schildpad naar een positie gaan op X pixels vanaf de linkerkant van het canvas. De hoogte blijft hetzelfde. **gox** kan wordenafgekort tot **gx**.

ganaary

```
ganaary Y
```

ganaary laat de schildpad naar een positie gaan die Y pixels vanaf de bovenkant van het canvas ligt. De afstand vanaf de linkerkant van het canvas blijft hetzelfde. **ganaary** kan wordenafgekort tot **gy**.

OPMERKING
Met de opdrachten **ganaar**, **ganaarx**, **ganaary** en **thuis** tekent de schildpad geen lijn, onafhankelijk van of de pen op is of neer.

4.3.2 Waar is de schildpad?

Er zijn twee opdrachten die de positie van de schildpad op het scherm doorgeven (teruggeven).

haalx

haalx geeft het aantal pixels door dat de huidige positie van de schildpad is vanaf de linkerkant van het scherm.

haaly

haaly geeft het aantal pixels door dat de huidige positie van de schildpad is vanaf de bovenkant van het scherm.

4.3.3 De schildpad heeft een pen

De schildpad heeft een pen die een lijn tekent als de schildpad beweegt. Er zijn een aantal opdrachten voor het besturen van de pen. In dit deel leggen we deze opdrachten uit.

penop (po)

```
penop
```

penop tilt de pen op van het canvas. Als de pen 'op' is, wordt er geen lijn getekend als de schildpad beweegt. Kijk ook bij **penneer**. **penop** kan worden afgekort met **po**.

penneer (pn)

```
penneer
```

penneer zet de pen neer op het canvas. Als de pen 'neer' is op het canvas, wordt er een lijn getekend als de schildpad beweegt. Zie ook **penop**. **penneer** kan worden afgekort met **pn**.

pendikte (pd)

```
pendikte X
```

pendikte X stelt de dikte van de pen (de breedte van de lijn) op X pixels. **pendikte** kan worden afgekort met **pd**.

penkleur (pk)

```
penkleur R,G,B
```

penkleur stelt de kleur in van de pen. **penkleur** heeft een RGB-combinatie (drie getallen tussen 0 en 255) nodig als invoer. **penkleur** kan worden afgekort met **pk**.

4.3.4 Opdrachten voor het canvas

Er zijn verschillende opdrachten voor de besturing van het canvas

canvasmaat (cm)

```
canvasmaat X,Y
```

Met het commando **canvasmaat** kan de grootte van het canvas worden ingesteld. Invoer is X en Y, waarbij X is de nieuwe canvasbreedte is in pixels, en Y de nieuwe hoogte van het canvas, ook in pixels. **canvasmaat** kan worden afgekort tot **cs**.

canvaskleur (ck)

```
canvaskleur R,G,B
```

canvaskleur stelt de kleur in van het canvas. **canvaskleur** heeft een RGB-combinatie (drie getallen tussen 0 en 255) nodig als invoer. **canvaskleur** kan worden afgekort met **ck**.

4.3.5 Opdrachten om schoon te maken

Er zijn twee opdrachten waarmee je het canvas kunt schoonmaken nadat je er een bende van hebt gemaakt.

schoon (cs)

```
schoon
```

Met **schoon** kunnen alle tekeningen op het canvas worden gewist. De rest blijft staan, de positie en de hoek van de schildpad, de kleur van het canvas, de zichtbaarheid van de schildpad en de grootte van het canvas blijven zoals ze waren.

alsnieuw

```
alsnieuw
```

alsnieuw maakt nog meer schoon dan het commando **schoon**. Na het commando **alsnieuw** is alles weer zoals het was direct na het starten van KTurtle. De schildpad wordt in het midden van het scherm gezet (met de neus naar boven), de canvaskleur is wit, de schildpad tekent een zwarte lijn op het canvas, en de grootte van het canvas is weer 400 x 400 pixels.

4.3.6 De schildpad is een sprite (zeg maar: sprait)

Eerst volgt hier een korte uitleg van wat sprites zijn: sprites zijn kleine plaatjes die je kunt verplaatsen op het scherm, zoals je vaak in computerspelletjes ziet. Onze schildpad is zo'n sprite. Voor meer info, zie de woordenlijst bij linkend=˝sprites˝ >sprites. Noot vertaler: eigenlijk zijn sprites zoiets als geesten, feeën of kabouters.

Hierna volgt een volledig overzicht van alle opdrachten die je kunt gebruiken bij sprites.

> OPMERKING
> [De huidige versie van KTurtle heeft alleen een schildpad als sprite. Andere sprites zijn er dus niet. In toekomstige versies van het programma zul je de schildpad kunnen veranderen naar je eigen ontwerp]

toon (ts)

```
toon
```

toon maakt de schilpad weer zichtbaar nadat die is verborgen. **toon** kan worden afgekort met **ts**.

verberg

```
verberg
```

verberg verbergt de schildpad. Dit is handig als de de schildpad niet in de tekening past. **verberg** kan worden afgekort met **vs**.

4.3.7 Kan de schildpad schrijven?

Het antwoord is: 'ja'. De schildpad kan schrijven. Hij schrijft zo'n beetje alles wat je hem opdraagt.

druk

```
druk X
```

Het commando **druk** kan worden gebruikt om de schildpad iets op het canvas te laten schrijven. **druk** heeft getallen en tekenreeksen nodig als invoer. Je kunt verschillende getallen en stukken tekst **laten schrijven** met behulp van het '+'-symbool. Hier is een klein voorbeeld:

```
$jaar = 2003
$auteur = "Cies"
druk $auteur + " begon het KTurtle-project in " + $jaar + " en vindt ↩
    het nog steeds leuk om er aan te werken!"
```

lettermaat

```
lettermaat X
```

lettermaat bepaalt de grootte van de letters voor het commando **druk**. **lettermaat** heeft een getal nodig als invoer. De grootte geef je op in pixels.

4.3.8 Wiskundige opdrachten

Hier volgen de wiskundige opdrachten die KTurtle kent.

afronden

```
afronden X
```

afronden van het gegeven getal naar het dichtstbijzijnde gehele getal.

```
druk afronden 10.8    #denk erom: een punt
   vooruit 20
druk afronden 10.3
```

Hiermee drukt de schildpad de getallen 11 en 10 af.

gok (gk)

```
gok X,Y
```

gok is een commando met invoer en uitvoer. Als invoerheeft het commando twee getallen nodig, het eerste getal (X) is de minimum grootte van de uitvoer, en het tweede (Y) de maximum grootte. De uitvoer is een willekeurig getal dat gelijk is aan of groter dan het minimum en gelijk aan of kleiner dan het maximum. Hier is een klein voorbeeld:

```
herhaal 500 [
 $x = gok 10,200
 vooruit $x
 draailinks 10 - $x
]
```

Met het commando **gok** kun je wat chaos aan je programma toevoegen.

mod

```
mod X,Y
```

Met het commando **mod** wordt de rest berekend vande deling van het eerste getal door het tweede getal.

wortel

```
wortel X
```

Met het commando **wortel X** wordt de wortel van het getal X berekend.

pi

```
pi
```

Dit commando geeft de constante pi, **3.14159**, door.

sin, cos, tan

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

Met deze drie opdrachten worden de welbekende goniometrische functies **sinus**, **cosinus** en **tangens** berekend. De invoer van deze opdrachten, X, is eengetal (in graden).

arcsin, arccos, arctan

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

Deze opdrachten zijn de inverse (dat is hier: terug-) functies van sin, cos en tan. De invoer van deze functies, X, is een getal. Die kan overigens niet elke waarde hebben, alléén die waarden die sin X, cos X en tan X kunnen hebben.

4.3.9 Invoer en terugkoppeling met behulp van dialogen

Een dialoog is een venstertje met daarin een boodschap, of waarin om invoer wordt gevraagd. KTurtle kent twee opdrachten voor een dialoog, namelijk **bericht** en **vraag**

bericht

```
bericht X
```

De opdracht **bericht** heeft als invoer een tekenreeks. Er wordt een venstertje getoond waarin een dialoog met de tekst van deze tekenreeks.

```
bericht "Cies begon het KTurtle-project in 2003 en vindt het nog steeds ↩
    leuk om er aan te werken!"
```

vraag

```
vraag X
```

vraag heeft als invoer een tekenreeks. Er wordt een venstertje getoond met de tekst van deze tekenreeks, net als bij het commando bericht. Maar in deze dialoog is ook een invoerveld. In dit invoerveld kan een gebruiker een getal of een tekenreeks invoeren die in een variabele kan worden opgeslagen, of aan een commando kan worden doorgegeven. Bijvoorbeeld

```
$in = vraag "Hoe oud ben je?"
$uit = 5 + $in
druk "Over 5 jaar ben je " + $uit + " jaar."
```

. Als de gebruiker de invoerdialoog annuleert of niets invult wordt de variabele leeg gemaakt.

4.4 Toewijzingen aan variabelen

Eerst bekijken we variabelen, daarna hoe we daar waarden aan kunnen toewijzen.

Variabelen zijn woorden die met een '$' beginnen, in de codebewerker worden ze met paars geaccentueerd.

In een variabele kan elk getal, tekenreeks of booleaanse waarde (waar/onwaar) worden bewaard. Met behulp van de toewijzing, **=**, kan de variabele zijn inhoud krijgen. Die inhoud blijft hierin aanwezig, totdat het programma eindigt, of als de variabele een andere waarde krijgt.

Variabelen, mits met inhoud, kunnen in de plaats van hun inhoud worden gebruikt. Zie bijvoorbeeld het volgende stukje TurtleScript:

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

Eerst krijgt de variabele **$x** de waarde**10**. Daarna krijgt **$x** een nieuwe waarde, van zichzelf, gedeeld door **3** — dit betekent dus dat **$x** daarna de waarde **10 / 3** heeft. Tenslotte wordt **$x** afgedrukt (op het scherm). In regels 2 en 3 zie je dat **$x** in de plaats van zijn inhoud wordt gebruikt.

Variabelen meten een waarde hebben gekregen voordat ze kunnen worden gebruikt. Bijvoorbeeld:

```
druk $n
```

zal een foutmelding geven.

Beschouw het volgende stukje TurtleScript:

```
$a = 2004
$b = 25

# Het volgende commando drukt "2029" af
druk $a + $b
achteruit 30
# Het volgende commando drukt "2004 plus 25 is gelijk aan 2029" af
print $a + " plus " + $b + " is gelijk aan " + ($a + $b)
```

In de eerste twee regels krijgen de variabelen **$a** en **$b** de waarden 2004 en 25. Daarna volgen twee **druk**-opdrachten met daartussen in een **achteruit 30**. In het commentaar voor de **druk**-opdrachten wordt uitgelegd wat er gebeurt. Zoals je ziet worden variabelen precies zo gebruikt als hun inhoud, je kunt ze met elke soort bewerkingen gebruiken, of als invoer voor opdrachten.

Nog een voorbeeld:

```
$naam = vraag "Hoe heet je?"
print "Ha die " + $naam + "! Veel plezier met het leren programmeren..."
```

Tamelijk recht zo die gaat. Ook hier kun je zien dat de variabele **$naam** precies als zijn inhoud, een tekenreeks, wordt gebruikt.

Bij het gebruik van variabelen is de inspectie erg nuttig. Hierin kun je de inhoud zien van alle huidige variabelen.

4.5 Sturen van het programma

De programmastuuropdrachten maken het mogelijk, —zoals de naam al zegt— het verloop van een programma te sturen.

Opdrachten die het programmaverloop sturen worden geaccentueerd met donkergroen in een vet lettertype. De accolades worden meestal samen met de programmastuuropdrachten gebruikt en worden geaccentueerd met zwart.

4.5.1 De schildpad laten wachten

Als je al wat hebt geprogrammeerd in KTurtle, dan heb je vast gemerkt dat de schildpad erg snel tekent. Dit commando laat de schildpad steeds een opgegeven tijdsduur te wachten.

wacht

```
wacht X
```

wacht X laat de schildpad X seconden wachten.

```
herhaal 36{
 vooruit 5
 draairechts 10
 wacht 0.5           # Denk erom: een punt
}
```

Deze programmacode tekent een cirkel, maar de schildpad zal na elke stap een halve seconde wachten. Hier door lijkt het net of de schildpad kruipt.

4.5.2 Uitvoeren 'indien'

als

```
als boolean { ... }
```

De programmacode tussen de accolades wordt alleen uitgevoerd **als** de booleaanse waarde de waarde 'waar' krijgt.

```
$x = 6
als $x > 5 {
  druk "$x is groter dan 5!"
}
```

Op de eerste regel krijgt **$x** de waarde 6. Op de tweede regel wordt een vergelijkingsbewerking gebruikt om de waarde van **$x** > **5** te bepalen. Omdat die 'waar' is, 6 is groter dan 5, zal het programmastuurcommando **als** de programmacode tussen de accolades laten uitvoeren.

4.5.3 Als niet, of in andere woorden: 'anders'

anders

```
als booleaanse waarde { ... } anders { ... }
```

anders kan worden gebruikt bij het programmastuurcommando **als**. De programmacode tussen de accolades na **anders** wordt alleen uitgevoerd als de booleaanse waarde de waarde 'onwaar' heeft.

```
alsnieuw
 $x = 4
als $x > 5 {
  druk "x is groter dan vijf!"
}
anders
{
  druk "x is kleiner dan zes!"
}
```

De vergelijkingsbewerking bepaalt de waarde van **$x** > **5**. Omdat 4 niet groter is dan 5 is die waarde 'onwaar'. Dit betekent dat de programmacode tussen de accolades na **anders** wordt uitgevoerd.

4.5.4 De 'terwijl'-lus

terwijl

```
terwijl booleaanse bewerking { ... }
```

Het programmastuurcommando **terwijl** lijkt veel op als. Het verschil is dat **terwijl** de programmacode tussen de accolades blijft herhalen totdat de booleaanse waarde 'onwaar' is geworden.

```
$x = 1
terwijl $x < 8 {
  vooruit 10
  wacht 1
  $x = $x + 1
}
```

Op de eerste regel krijgt **$x** de waarde 1. Op de tweede regel wordt de waarde bepaald van**$x** < **8**. Omdat die waarde 'waar' is, start het programmastuurcommando**terwijl** het uitvoeren van de code tussen de accolades totdat de waarde van **$x** < **8** 'onwaar' is geworden. In dit voorbeeld wordt de code tussen de accolades 7 keer uitgevoerd, omdat telkens als de vijfde regel wordt uitgevoerd de waarde **$x** met 1 wordt vermeerderd.

4.5.5 De 'herhaal'-loop

herhaal

```
Herhaal getal { ... }
```

Het programmastuurcommando **herhaal** lijkt veel op **terwijl**. Het verschil is dat **herhaal** de programmacode tussen de accolades het in het getal opgegeven aantal keren blijft herhalen

4.5.6 De 'voor'-lus, een tellende lus

voor

```
voor variabele = getal tot getal { ... }
```

De lus **voor** is een 'tellende lus', en telt dus voor jou.

```
voor $x = 1 tot 10 {
 druk $x * 7
 vooruit 15
}
```

Telkens als de programmacode tussen de accolades wordt uitgevoerd wordt de waarde **$x** met 1 vermeerderd (dit gaat dus van zelf), totdat **$x** de waarde 10 heeft bereikt. De programmacode tussen de accolades schrijft de waarde van **$x** (op het scherm), vermenigvuldigd met 7. Nadat deze opdrachten zijn uitgevoerd zie je de tafel van 7 op het canvas staan.
De standaard stapgrootte in een lus is 1, maar een andere waarde kan worden ingesteld met

```
voor variabele = getal tot getal  stap getal { ... }
```

4.5.7 Een lus verlaten

kap

```
kap
```

Stopt onmiddellijk met het uitvoeren van de huidige lus en ga verder met het uitvoeren van de programmaregel direct na de lus.

4.5.8 Het uitvoeren van je programma stoppen

afsluiten

```
afsluiten
```

Beëindigt het uitvoeren van je programma.

4.5.9 Invoer testen

test

```
test boolean { ... }
```

Kan worden gebruikt voor het beredeneren van de juistheid van het programma of van de invoer.

```
$in = ask "In welk jaar ben je geboren?"
# het jaartal moet positief zijn
test $in
> 0
```

4.6 Eigen opdrachten maken met 'leer'

leer is een erg leuk commando, omdat die kan worden gebruikt om je eigen opdrachten te maken. Opdrachten die je maakt kunnen invoer nodig hebben en uitvoer teruggeven. Laten we eens kijken hoe je een nieuw commando kunt maken:

```
leer cirkel $x {
   herhaal 36 {
     vooruit $x
     draailinks 10
  }
}
```

Het nieuwe commando heet hier dus **cirkel**. **cirkel** heeft een invoer nodig, een getal, dat de grootte van de cirkel bepaalt. **cirkel** geeft geen uitvoer terug. Het commando **cirkel** kan nu worden gebruikt als een normaal commando in de rest van de programmacode. Hier een voorbeeld:

```
leer cirkel $x {
  herhaal 36 {
    vooruit $x
    draailinks 10
  }
}

ganaar 200,200
cirkel 20

ganaar 300,200
cirkel 40
```

In het volgende voorbeeld maken we een commando met een .

```
leer faculteit $x {
    $r = 1
    voor $i = 1 tot $x {
        $r = $r * $i
    }
  geefdoor $r
}

druk faculteit 5
```

In dit voorbeeld maken we een nieuw commando met de naam **faculteit**. Als de invoer van dit commando **5** is, dan is de uitvoer **5*4*3*2*1**, en dat is 120. Door **geefdoor** wordt de waarde van de uitvoer doorgegeven (of ook wel: teruggegeven), zodat die bijvoorbeeld kan worden afgedrukt op het scherm, zoals in de laatste regel.

Opdrachten kunnen meer dan een invoer nodig hebben. In het volgende voorbeeld maken we een commando dat een rechthoek tekent:

```
leer rechthoek $x, $y {
  vooruit $y
  draairechts 90
  vooruit $x
  draairechts 90
  vooruit $y
  draairechts 90
  vooruit $x
  draairechts 90
}
```

Nu kun je **rechthoek 50, 100** doen, waarna de schildpad een rechthoek tekent op het canvas.

Hoofdstuk 5

Woordenboek

In dit hoofdstuk vind je een uitleg van de betekenis van de meeste 'vreemde' woorden die in het handboek worden gebruikt.

graden

Graden zijn eenheden die worden gebruikt om de grootte van hoeken of draaibewegingen te meten. Een keer helemaal rond is 360 graden, een halve keer rond is 180 graden (de schildpad gaat daarna in de tegengestelde richting), en een kwart keer rond is 90 graden. De commando's **draailinks**, **draairechts** en **richting** hebben een invoer in graden nodig.

invoer en uitvoer van commando's

Sommige commando's hebben invoer nodig, andere commando's geven uitvoer terug, sommige commando's hebben invoer nodig *en* geven uitvoer terug, en sommige commando's hebben geen invoer nodig en geven ook geen uitvoer terug. Invoer is informatie die het commando nodig heeft en uitvoer is het resultaat dat het commando geeft.

Enkele voorbeelden van commando's die alleen invoer nodig hebben zijn:

```
vooruit 50
penkleur 255,0,0
druk "Hallo!"
```

Het commando **vooruit** heeft **50** als invoer. **vooruit** heeft deze invoer nodig om te weten hoeveel pixels de schildpad vooruit moet lopen. **penkleur** heeft een kleur als invoer en **druk** heeft een tekenreeks (tekst) als invoer. Onthoudt dat de invoer ook een variabele mag zijn. In het volgende voorbeeld wordt zo'n variabele gebruikt:

```
$x = 50
druk $x
vooruit 50
$txt = "hallo!"
druk $txt
```

Nu wat voorbeelden met uitvoer :

```
$x = vraag "Typ iets in en druk daarna op OK... dank je wel!"
$r = gok 1,100
```

Het commando **vraag** heeft als invoer een tekenrij of getal nodig, en voert dat wat is ingevoerd weer uit. De uitvoer van **vraag** wordt opgeslagen in de variabele **$x**. Het commando**gok** geeft ook uitvoer. In dit geval is dat een getal tussen 1 en 100. De uitvoer van

gok wordt ook bewaard in een variabele, met de naam **$r**. Merk op dat de variabelen **$x** en **$r** niet worden gebruikt in het bovenstaande voorbeeld.

Er zijn ook commando's die geen invoer nodig hebben en geen uitvoer geven. Hier zijn enkele voorbeelden:

```
schoon
penop
```

Intuïtieve accentuering

Dit is een eigenschap van KTurtle die het programmeren nog eenvoudiger maakt. Met intuïtieve accentuering krijgt de tekst die je intypt een kleur die het type van de programmeercode aangeeft. In de volgende lijst vind je de verschillende types code en de kleur die die krijgt in de codebewerker.

algemene commando's	donkerblauw	De algemene commando's worden hier beschreven.
commando's waarmee de loop van het programma wordt gestuurd	zwart (vet)	Lees hier meer over deze speciale programmastuur-commando's
commentaren	grijs	Commentaarregels beginnen met een commentaarteken (#). Deze regels worden genegeerd als de programmacode wordt uitgevoerd. Commentaren kun je gebruiken om een uitleg van je programmacode toe te voegen, of een stukje code tijdelijk niet uit te voeren.
accolades {, }	donkergroen (vet)	Accolades worden gebruikt om regels met programmacode te groeperen. Accolades worden vaak gebruikt in combinatie met uitvoercontroles.
het commando leer	lichtgroen (vet)	Het commando leer wordt gebruikt voor het maken van nieuwe commando's.
tekenreeksen	rood	Tekenreeks is een ander woord voor een stukje tekst. Tekenreeksen staan altijd tussen dubbele aanhalingstekens (˝).
getallen	donkerrood	Getallen, je kent ze wel.
booleaanse waarden	donkerrood	Er zijn precies twee booleaanse waarden, namelijk: waar en onwaar.

variabelen	paars	Beginnen met een '$' en kan getallen bevatten, tekenreeksen en booleaanse waarden.
rekenkundige bewerkingen	grijs	Dit zijn de rekenkundige bewerkingen: **+**, **−**, *****, **/** en **^**.
Vergelijkingsbewerkingen	lichtblauw (vet)	Dit zijn de vergelijkingsbewerkingen: **==**, **!=**, **<**, **>**, **<=** and **>=**.
booleaanse bewerkingen	roze (vet)	Dit zijn de booleaanse bewerkingen: **en**, **of**, **niet**.
gewone tekst	zwart	

Tabel 5.1: Verschillende types van programmacode en de kleur waarmee die worden geaccentueerd.

pixels

Een pixel is een stip op het beeldscherm. Als je het beeldscherm van heel dichtbij bekijkt zul je zien dat het beeld is opgebouwd uit allemaal kleine stippen. Deze stippen worden pixels genoemd. Alle afbeeldingen op het beeldscherm zijn uit deze pixels opgebouwd. Een pixel is het kleinste dat je op een beeldscherm kunt tekenen.

Veel commando's vragen om een aantal pixels als invoer. Deze commando's zijn: **vooruit**, **achteruit**, **ganaar**, **ganaarx**, **ganaary**, **canvasmaat** en **pendikte**.

In de vroegere versies van KTurtle was het canvas eigenlijk een rasterbeeld, maar in de meer recente versies is het canvas een vector-afbeelding. Hierdoor kan in het canvas worden in- en uitgezoomd, en komt een pixel niet altijd overeen met één pixel op het scherm.

RGB-combinaties (kleurcodes)

RGB-combinaties worden gebruikt om de kleur te beschrijven. De 'R' staat voor 'rood', de 'G' staat voor 'groen' en de 'B' staat voor 'blauw'. Een voorbeeld van een RGB-combinatie is **255,0,0**: de eerste waarde ('rood') is 255, en de andere waarden zijn 0. Dus geeft deze combinatie helder rood. Elke waarde van een RGB-combinatie moet tussen de 0 en 255 liggen. Hier is een kleine lijst van veelgebruikte kleuren:

0,0,0	zwart
255,255,255	wit
255,0,0	rood
150,0,0	donkerrood
0,255,0	groen
0,0,255	blauw
0,255,255	lichtblauw
255,0,255	roze
255,255,0	geel

Tabel 5.2: Veelgebruikte RGB-combinaties

Er zijn twee commando's die om een RGB-combinatie als invoer vragen. Deze commando's zijn **canvaskleur** en **penkleur**.

sprite

Een sprite (zeg maar ˝sprait˝) is een klein plaatje dat over het scherm kan worden verplaatst. De schildpad in dit programma is zo'n sprite.

OPMERKING

Let op: In deze versie van KTurtle kun je de sprite niet in iets anders veranderen. In toekomstige versies zal dit wel mogelijk zijn.

Hoofdstuk 6

Vertaalgids voor KTurtle

Zoals u waarschijnlijk al bekend is, kan de programmeertaal TurtleScript van KTurtle worden vertaald. Hierdoor wordt een barrière weggenomen, vooral voor jonge kinderen, bij hun pogingen de eerste beginselen te begrijpen van het programmeren.

Bij het vertalen van KTurtle naar een andere taal, vindt u, naast de GUI-strings, de programmeercommando's, de voorbeelden en de foutmeldingen, in de standaard .pot-bestanden die bij het vertalen voor KDE worden gebruikt. Alles wordt op de voor KDE gebruikelijke wijze vertaald, niettemin wordt u sterk aangeraden u op de hoogte te stellen van hoe deze moeten worden vertaald (zoals u ook kunt lezen in het commentaar voor de vertaler).

Zie ook https://edu.kde.org/kturtle/translation.php voor meer informatie over het vertalen. Hartelijk dank voor uw werk! KTurtle hangt sterk af van de vertalingen.

Hoofdstuk 7

Dankbetuigingen en licenties

KTurtle

- Cies Breijs cies AT kde DOT nl
- Anne-Marie Mahfouf annma@kde.org
- Proefgelezen en verbeterd door Philip Rodriguesphil@kde.org
- Bijgewerkte vertalingshandleiding en enkele verbeteringen door Andrew Coles andrew_coles AT yahoo DOT co DOT uk

Jaap Woldringhjjh punt woldringh op planet punt nl

Hoofdstuk 8

Index