JavaLogo werd ontwikkeld voor het vak informatica in de tweede fase. In dit artikel beschrijft **Peter Boon** wat JavaLogo is en geeft hij een beeld van de verschillende mogelijkheden die het biedt. Ook voor het vak wiskunde zou JavaLogo wellicht gebruikt kunnen worden bij praktische opdrachten.

JavaLogo en praktische opdrachten wiskunde

Inleiding

JavaLogo is in eerste instantie ontwikkeld voor het vak informatica in de tweede fase. De programmeertaal Java wordt gebruikt om tekeningen op het beeldscherm te maken. Om dit voor een beginnend programmeur wat makkelijker te maken, zijn er aan de taal Java een aantal eenvoudig te gebruiken tekenopdrachten toegevoegd. Met deze overzichtelijke set opdrachten kan een aantal basisvaardigheden van het programmeren worden aangeleerd, zonder te verdrinken in de complexiteit van een professionele taal die Java toch is.
Door het meetkundige karakter van de met JavaLogo gemaakte figuren en door de meetkundige vaardigheden die nodig zijn om dit soort figuren te maken, dringt de vraag zich op of JavaLogo ook een rol kan spelen in het wiskundeonderwijs. Door het praktische karakter van JavaLogo liggen er wellicht kansen bij de praktische opdrachten voor wiskunde B.
Met dit artikel willen we JavaLogo onder de aandacht brengen van wiskundedocenten door een beeld te geven van de mogelijkheden. Ook komen ervaringen opgedaan met leerlingen ter sprake.

JavaLogo en informatica

Het onderdeel programmeren vormt binnen het nieuwe vak informatica slechts een klein onderdeel. Door de beperkte tijd die aan dit onderdeel kan worden besteed, werd in de aanwezige methoden het programmeren aanvankelijk vrij theoretisch benaderd. De belangrijke achterliggende concepten werden uitgelegd, maar tot serieus praktisch werk voor de leerlingen kwam het eigenlijk niet. In de praktijk bleek dit niet goed te werken. Het theoretische karakter maakte de stof voor de leerlingen erg saai en de concepten bleken niet goed uit de verf te komen als ze niet praktisch handen en voeten kregen. Een grote praktische component is nodig, ook binnen de beperkte tijd die voor dit onderwerp beschikbaar is.
Wanneer je praktisch gaat programmeren zul je moeten kiezen voor een taal. De beschikbare tijd staat niet toe dat je aan de slag gaat met verschillende talen. We hebben de keuze uit vele talen, van grote algemeen bruikbare talen (Java, Pascal, Visual Basic, ...) tot talen voor een specifiek doel (besturingstalen voor Lego-robots) of specifieke onderwijstalen (Logo). De talen hebben allen voor- en nadelen.
Het liefst zouden we willen kiezen voor een taal als Java. Hierin komen alle concepten over programmastructuur en object-oriëntatie goed naar voren. Het heeft echter een groot bezwaar: de complete taal Java is voor een beginnend programmeur met weinig tijd te groot en te complex. Daardoor wordt de drempel erg hoog.

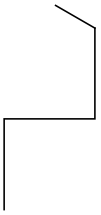
De taal Logo bevindt zich aan de andere kant van het spectrum. Het schrijven van kleine tekenprogramma's in Logo is laagdrempelig, leerlingen zien al snel resultaten. Voor kinderen is het bijzonder stimulerend, omdat de resultaten van hun programma's direct op het scherm verschijnen.
Logo is zeer geschikt om een aantal basisprincipes van het programmeren te leren, zoals variabelen, het opsplitsen van een programma in procedures, parameteroverdracht en herhalingsstructuren.
Logo heeft ook nadelen. Het is een wereld die op zichzelf staat, je moet een grote stap maken om bij andere, volwassen programmeertalen te komen. Het is geen taal waarin 'serieuze' software ontwikkeld kan worden. Er is geen link naar de principes van object-georiënteerd programmeren.
In JavaLogo is geprobeerd om de voordelen van de verschillende keuzemogelijkheden te combineren. We werken in Java, maar de drempel is verlaagd door de toevoeging van eenvoudig te gebruiken tekenopdrachten, vergelijkbaar met de tekenopdrachten van Logo. Door deze aanpak beginnen leerlingen met eenvoudige programma's, maar wel in de syntax van Java. Hiermee staat de deur naar een volwassen taal wijd open.

Hoe werkt JavaLogo?

Hoe schrijven leerlingen een programma in JavaLogo? Als voorbeeld bekijken we een serie eenvoudige logo-instructies die een zigzag tekenen.

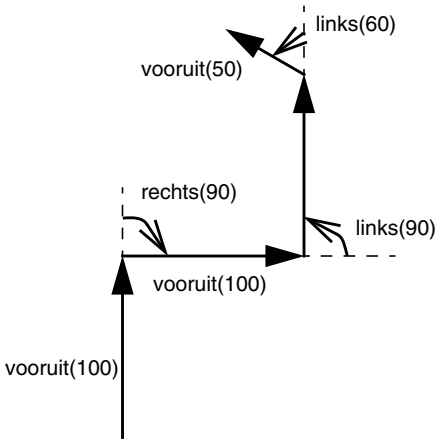
De LOGO-tekenopdrachten

De tekenopdrachten die in JavaLogo gebruikt worden, nemen een centrale plaats in. De belangrijkste tekenopdrachten zijn hetzelfde als de tekenopdrachten van het programma 'LOGO'. Het zijn opdrachten die een 'pen' besturen waardoor er een tekening gemaakt wordt.
Een voorbeeld:
Wanneer we de figuur hiernaast willen tekenen, dan gebruiken we de volgende opdrachten (we beginnen de tekening onderaan):

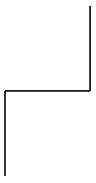


- vooruit(100) Hiermee maken we een lijn van 100 eenheden lang.
- rechts(90) We slaan rechtsaf onder een hoek van 90 graden.
  (We kiezen dus een nieuwe richting voor de pen.)
- vooruit(100) Weer een lijn van 100 eenheden.
- links(90) We slaan linksaf onder een hoek van 90 graden.
- vooruit(100) Weer een lijn van 100 eenheden.
- links(60) We slaan linksaf onder een hoek van 60 graden.
- vooruit(50) We maken een lijn van 50 eenheden.

In de figuur hieronder zien we nog eens stap voor stap de uitvoering van deze serie opdrachten.




```
VOORUIT(100)
RECHTS(90)
VOORUIT(100)
LINKS(90)
VOORUIT(100)
```



In voorbeeld 1 is te zien wat dit in JavaLogo wordt.
We zien een programma (public class Zigzag). Binnen dit programma staat een procedure (public void tekenprogramma()) met daarin de tekenopdrachten die de gewenste figuur tekenen. Voor de volledigheid merken we op dat het programma Zigzag een uitbreiding is van het programma TekenApplet (extends TekenApplet). Dit geldt voor alle JavaLogo programma's (in het programma TekenApplet zijn de tekenopdrachten gedefinieerd; daar staat wat ze moeten doen).

```
import logotekenap.*;

public class Zigzag extends TekenApplet
{
    public void tekenprogramma()
    {
        vooruit(100);
        rechts(90);
        vooruit(100);
        links(90);
        vooruit(100);
    }
}
```

Voorbeeld 1

In het leerlingenmateriaal bij JavaLogo worden stapsgewijs verschillende programmeertechnieken besproken en geoefend. Aandacht wordt besteed aan het werken met variabelen, het maken van subprocedures (in Java heten

```
import logotekenap.*;

public class Krans extends TekenApplet
{
    double zijde, afstand;

    public void tekenprogramma()
    {   zijde = 50; afstand = 100;
        penUit();stap(-50,-150);penAan();
        rechts(90);
        for(int i=0 ; i<9 ; i++)
        {   rozet(zijde);
            vooruit(afstand);
            links(40);
        }
    }
    void rozet(double z)
    {   for(int i=0 ; i<45 ; i++)
        {   vierkant(z);
            rechts(8);
        }
    }
    void vierkant(double z)
    {   for(int i=0 ; i<4 ; i++)
        {   vooruit(z);
            rechts(90);
        }
    }
}
```

Voorbeeld 2

die ‘methoden’) voor deeltaken, parameters, het gebruik van herhalingsopdrachten, enzovoort.
Een voorbeeld van een groter programma in JavaLogo waarin wat meer programmeertechniek gebruikt wordt, staat in voorbeeld 2. In dit programma wordt gebruik gemaakt van variabelen, herhalingsopdrachten en zelfgedefinieerde extra methoden (subprocedures), allemaal in de syntaxis van Java. De methode *vierkant(double z)* tekent een vierkant met zijde *z*. De methode *rozet(double z)* tekent 45 vierkanten en roept hiervoor de methode *vierkant(double z)* aan. Elk vierkant is steeds 8 graden ten opzichte van de vorige gedraaid, *tekenprogramma()* roept negen keer de methode *rozet(double z)* aan. Na elke rozet volgt *vooruit(afstand)* en *links(40)*. Het resultaat is een negenhoek, met op elk hoekpunt een rozet. Zie figuur 1.

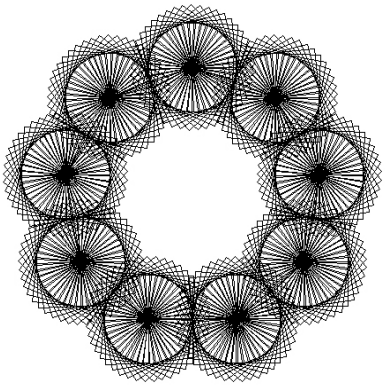


fig. 1 Resultaat van het programma Krans

Behalve het tekenen van lijnen kunnen in JavaLogo ook vlakken worden ingekleurd. Door handig gebruik te maken van subprocedures en herhalingen kunnen vlakvullingen als figuur 2 gemaakt worden.

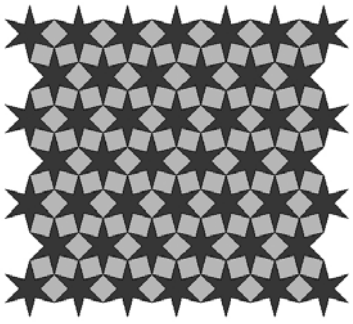


fig. 2 Vlakvulling gemaakt met JavaLogo

Verschillende leerlingen hebben deze vlakvulling gemaakt als praktische opdracht voor informatica.

Interactieve programma’s

Een van de eigenschappen van computerprogramma’s is interactie met de gebruiker. Een gebruiker kan gegevens invoeren waarop het programma kan reageren. De programma’s die we hiervoor hebben bekeken, hadden die eigenschap niet. Er werd een tekening gemaakt, dat was alles. Om invoer door de gebruiker mogelijk te maken kunnen in JavaLogo zogenaamde invoervariabelen gebruikt worden. We geven als voorbeeld een programma dat een vierkant tekent waarvan de zijde door de gebruiker kan worden veranderd. Zie figuur 3.

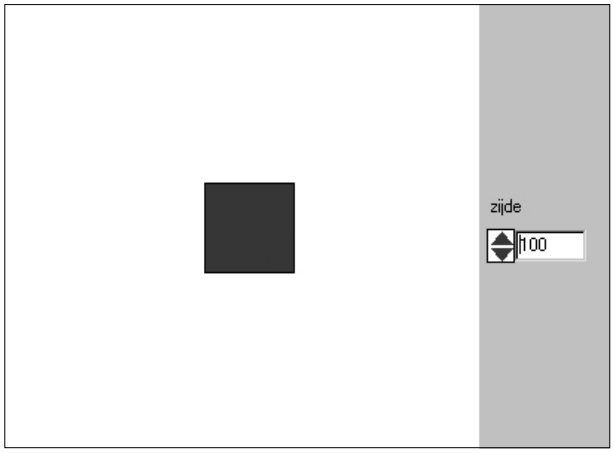


fig. 3 Vierkant met variabele zijde

De JavaLogo-programmacode zien we in voorbeeld 3.

```
import logotekenap.*;

public class Vierkant extends TekenApplet
{
1   InvoerVariabele zijdeInv;
    double zijde;

    public void initialiseer()
2   {   zijdeInv = new
        InvoerVariabele(“zijde”,0,400,100);
3       maakZichtbaar(zijdeInv);
        zijde = 100;
    }

    public void tekenprogramma()
    {   vierkant(zijde);
    }

    void vierkant(double z)
    {   for(int i=0 ; i<4 ; i++)
        {   vooruit(z);
            rechts(90);
        }
    }

4   public void invoerVarActie(invoerVariabele iv)
    {   zijde = zijdeInv.geefwaarde();
        tekenOpnieuw();
    }
}
```

Voorbeeld 3

Zonder in detail op de code in te gaan kunnen we globaal zien wat er gebeurt:

1. Er wordt een InvoerVariabele gedeclareerd.
2. Er wordt een InvoerVariabele gemaakt en
3. zichtbaar gemaakt, rechts op het scherm.
4. Opdrachten in de methode invoerVarActie worden uitgevoerd wanneer de gebruiker de waarde in het invoerveld verandert. In dit geval wordt de waarde van de variabele zijde aangepast, en het vierkant opnieuw getekend.

Ook via handelingen met de muis is interactie met een programma mogelijk gemaakt in JavaLogo. Zo is het bijvoorbeeld mogelijk om door het slepen van de muis een variabele binnen het programma te veranderen, waarna de tekening wordt aangepast. Op deze wijze kunnen figuren met de muis versleept of verdraaid worden.

Bewegende beelden zijn ook eenvoudig te maken door animatiemogelijkheden binnen JavaLogo. Een programmavariabele wordt dan continu veranderd, waarna de tekening telkens weer opnieuw wordt gemaakt.

```
import logotekenap.*;

public class DraaiVierkant extends TekenApplet
{
    double hoek;

    public void initialiseer()
    {   maakAnimatieMogelijk();
        hoek = 0;
    }

    public void tekenprogramma()
    {   links(hoek);
        vierkant(100);
    }

    void vierkant(double z)
    {   vulAan("rood");
        for(int i=0 ; i<4 ; i++)
        {   vooruit(z);rechts(90);
        }
        vulUit();
    }

    public void animatie()
    {   while(animatieLopend())
        {   hoek = hoek + 1;
            tekenOpnieuw();
        }
    }
}
```

Voorbeeld 4

Voorbeeld 4 toont een programma dat een vierkant tekent dat om een hoekpunt draait. In de code staat bij ‘public void tekenprogramma()’ *links(hoek)*, waarmee de tekenrichting een aantal graden naar links wordt gedraaid. Daarna wordt met de opdracht: *vierkant(100)* een vierkant met een zijde van 100 getekend. Omdat de variabele *hoek* in ‘public void animatie()’ voortdurend wordt opgehoogd, draait het vierkant linksom, totdat de animatie wordt uitgezet.

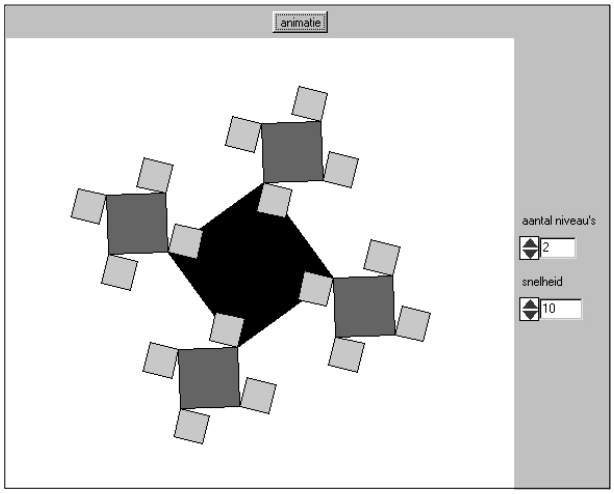


fig. 4 Schermafdruk van een animatie

Van een wat ingewikkeldere animatie staat een schermafdruk in figuur 4. Behalve van de animatiemogelijkheid is hier ook gebruik gemaakt van invoervariabelen.

3d-figuren

Met de tekenopdrachten links(..), rechts(..) en vooruit(..) is het mogelijk om door een tweedimensionale ruimte te wandelen. Met een andere set tekenopdrachten is het in JavaLogo ook mogelijk om door een driedimensionale ruimte te wandelen. (Een driedimensionale ruimte die natuurlijk weer netjes wordt geprojecteerd op ons tweedimensionale beeldscherm.)
Figuur 5 geeft een beeld van het gebruik van enkele van deze 3d-tekenopdrachten.

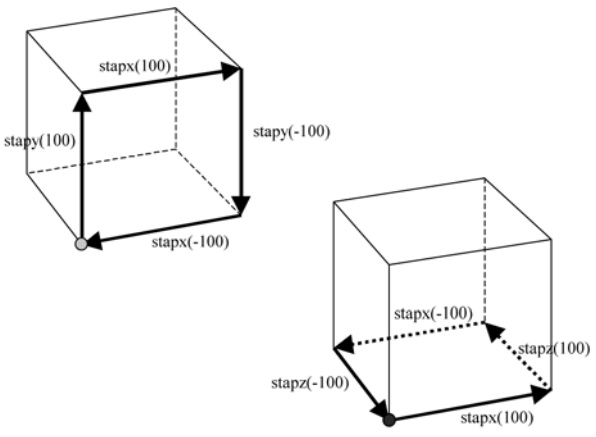


fig. 5 De werking van 3d-tekenopdrachten

In combinatie met de muisinteractie- en animatiemogelijkheden zijn allerlei leuke objecten te maken, die kunnen worden rondgedraaid en zelf worden opengevouwen met de muis, zoals het doosje in figuur 6. Dit object werd door enkele leerlingen als praktische opdracht informatica gemaakt.

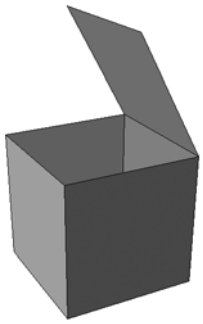


fig. 6 Ronddraaiend doosje met deksel

Maar het kan natuurlijk ook nog ingewikkelder, zoals figuur 7 laat zien.

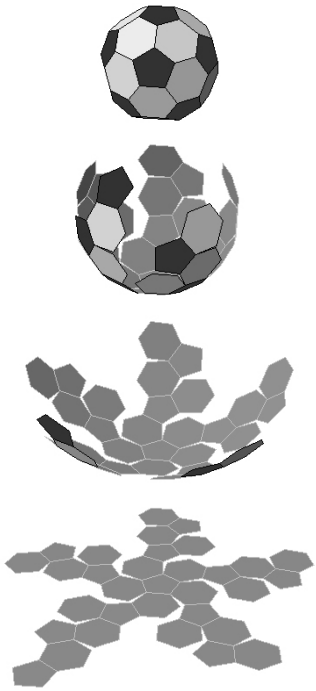


fig. 7 Uitslag van een voetbal

Recursie: fractals

Door gebruik te maken van procedures die zichzelf aanroepen kunnen fractals getekend worden. De programmacode bij voorbeeld 5 hoort bij de zeef van Sierpinsky. Het is verrassend om te zien hoe kort de code is wanneer recursie gebruikt wordt.

De fractal van figuur 9 is een variatie op de bekende pythagorasboom. Het aantal recursieniveaus is instelbaar met behulp van een invoervariabele.

```
import logotekenap.*;

public class Sierp0 extends TekenApplet
{
    public void tekenprogramma()
    {   penUit();stap(-200,-200);rechts(30);penAan();
        driehoekFractal(8, 500);
    }
    void driehoekFractal(double niv, double z)
    {   if(niv >0)
        {   for(int i=0 ; i<3 ; i++)
            {   vooruit(z);
                rechts(120);
                driehoekFractal(niv-1, z/2);
            }
        }
    }
}
```

Voorbeeld 5

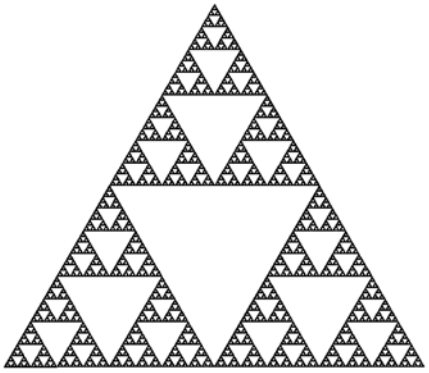


fig. 8 Zeef van Sierpinsky

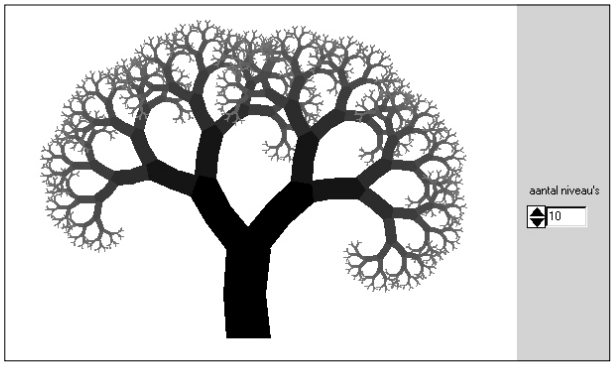


fig. 9 Pythagorasboom

Ervaringen in de klas

Zowel in 5 HAVO als in 5 atheneum heb ik vorig schooljaar twee blokken van elk twaalf lessen besteed aan programmeren met JavaLogo. Het ging hier om lessen voor

het vak informatica. In het eerste blok moesten de leerlingen allerlei figuren programmeren, waarbij dan gaandeweg enkele basistechnieken van het programmeren werden behandeld en geoefend. Dit blok werd afgesloten met een praktische opdracht waarin de leerlingen een vlakvulling, vergelijkbaar met die van figuur 2, moesten maken. Het tweede blok werd besteed aan interactieve programma's. De praktische opdracht bij dit blok was vrijer van opzet. Leerlingen moesten zelf een ontwerp maken voor een programma waarin een aantal technieken moest worden gebruikt.
In de eerste les leerden de leerlingen omgaan met Visual J++, de ontwikkelomgeving waarin de programma's worden ingetypt, gecompileerd, uitgevoerd en opgeslagen. Dat ging vrij vlot, zodat de meeste leerlingen al in de eerste les een klein werkend programma hadden gemaakt.
De nauwkeurigheid die nodig was om een werkend programma te maken was voor een aantal leerlingen schokkend. Een puntkomma vergeten aan het eind van een regel bleek al funest. Het viel op dat meisjes, meer dan jongens, de rust konden opbrengen om de opdrachten goed te lezen en nauwkeurig te werken, waardoor ze sneller tot het beoogde resultaat kwamen.
De volgende lessen werden besteed aan het maken van steeds complexer wordende figuren, waarbij geleidelijk een aantal programmeertechnieken werd aangeleerd.
Het manipuleren met hoeken en stappen was met name voor een aantal HAVO-leerlingen met wiskunde A niet altijd makkelijk. Wanneer in de serie tekenopdrachten één keer een verkeerde hoek werd opgegeven, dan had dat natuurlijk gevolgen voor alles wat na die verkeerde opdracht getekend werd. Een bijna goed programma kon daardoor toch een volkomen verkeerde tekening maken. Dat leed bij sommige leerlingen tot een groeiende radeloosheid. Gelukkig is het in JavaLogo ook mogelijk de tekeningen stap voor stap te laten maken, zodat de verkeerde stap wat makkelijker op te sporen was.
Toch bleef vrijwel iedereen zeer gemotiveerd. Als de juiste figuur na veel geworstel op het scherm verscheen, was dit voor de meesten een grote beloning. Het was wel van belang dat een leerling die het spoor echt bijster was op tijd hulp kreeg.

Voor de eerste praktische opdracht, het maken van een van de vlakvullingen in figuur 10, hadden de leerlingen een hele ochtend de tijd. Er werd gewerkt in groepjes van twee. Zelfstandig, en in veel gevallen zeer fanatiek. Wel waren er een paar groepjes die de hulp van de docent echt nodig hadden.
In het tweede blok van twaalf lessen begon er met name in de groep HAVO-leerlingen een tweedeling zichtbaar te worden. Deze tweedeling liep logischerwijs vrijwel parallel aan het soort profiel dat de leerlingen deden.
Veel leerlingen uit de natuurprofielen begonnen zich gretig de nieuwe interactieve mogelijkheden van JavaLogo eigen te maken en werkte fanatiek aan ontwerpen voor de

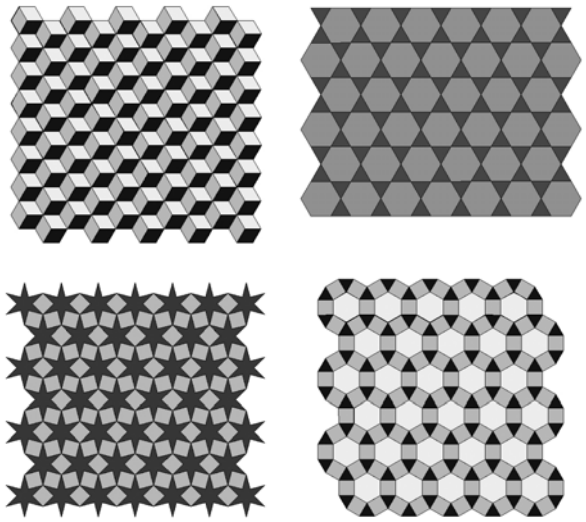


fig. 10 Praktische opdracht vlakvullingen

tweede praktische opdracht. De meeste leerlingen uit de maatschappijprofielen hadden het wel een beetje gezien en vond het allemaal maar lastig worden. Bovendien lieten zij zich soms ontmoedigen door de prachtige ontwerpen van sommige leerlingen, waaraan zij toch niet konden tippen. Het vergde nogal wat tact om deze groep op hun eigen niveau gemotiveerd bezig te laten zijn en soms lukte dat ook niet.
Dat sommige leerlingen tot meer in staat zijn dan je als docent voor mogelijk houdt, blijkt uit het ontwerp van Roan en Jochem uit 5 atheneum. Zij maakten het spel 4-op-een-rij, maar dan voor drie dimensies. Geheel muisgestuurd en inclusief applaus voor de winnaar. Het speelbord kan met de muis worden rondgedraaid. Zie figuur 11.

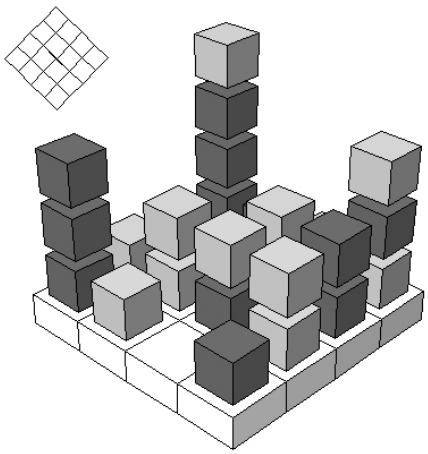


fig. 11 4-op-een-rij, gemaakt door twee leerlingen

Een praktische wiskundeopdracht

Ten slotte wil ik een van de programmeeropdrachten die de leerlingen hebben gemaakt wat nader bekijken, omdat het in feite een praktische variant is van een bekende wiskundeopdracht.

```
import logotekenap.*;

public class Opdracht15 extends TekenApplet
{
    InvoerVariabele aantalHoekenInv;
    InvoerVariabele zijdeInv;
    double aantalHoeken;
    double zijde;

    public void initialiseer()
    {   aantalHoekenInv = new InvoerVariabele("aantal",3,20,3);
        maakZichtbaar(aantalHoekenInv);
        zijdeInv = new InvoerVariabele("zijde",0,400,100);
        maakZichtbaar(zijdeInv);
        aantalHoeken = 3;
        zijde = 100;
    }
    public void tekenprogramma()
    {   penUit();stap(-zijde/2,-zijde);penAan();
        rechts(90);
        vulAan("rood");
        for(int i=0 ; i<aantalHoeken ; i++)
        {   vooruit(zijde);
            links(360/aantalHoeken);
        }
        vulUit();
    }
    public void invoerVarActie(InvoerVariabele iv)
    {   if(iv==aantalHoekenInv)
        {   aantalHoeken = aantalHoekenInv.geefWaarde();
            tekenOpnieuw();
        }
        if(iv==zijdeInv)
        {   zijde = zijdeInv.geefWaarde();
            tekenOpnieuw();
        }
    }
```

Voorbeeld 6

Maak een programma waarin je een regelmatige veelhoek tekent, waarvan de gebruiker de lengte van de zijde en het aantal hoekpunten met behulp van invoervariabelen kan variëren.

De wiskundeopdracht die moet worden opgelost om de opdracht tot een goed einde te brengen luidt:

Gegeven is een regelmatige veelhoek met *n* hoekpunten. Geef de grootte van elk van de hoeken, uitgedrukt in *n*.

Om de tekening te kunnen maken is eigenlijk niet de binnenhoek, maar het complement nodig. Zie figuur 12.
Na wat gepuzzel lukt het meestal wel om de gevraagde uitdrukking 360/n te vinden. Leerlingen vinden het echter verbazend dat je die formule ook echt in het programma kunt gebruiken als waarde voor een hoek. Het is geweldig om te zien hoe leerlingen op deze manier de kracht van het werken met variabelen op een heel directe manier ervaren, als het programma eenmaal werkt.
Maar dat is nog niet alles. ‘Maar nu wil ik dat de veelhoek

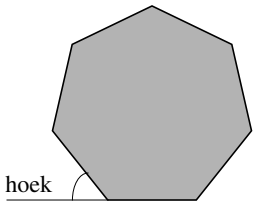


fig. 12 Hoe kun je een regelmatige vierhoek tekenen?

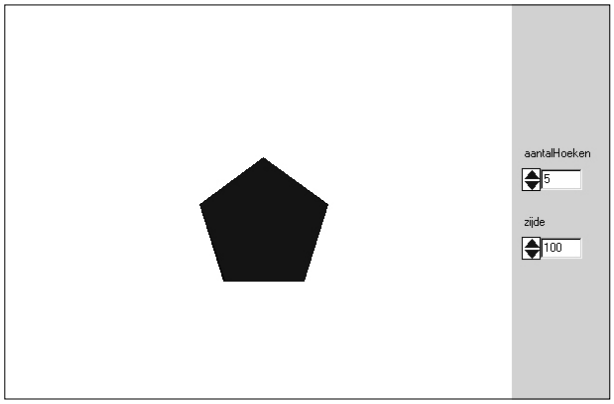


fig. 13 Veelhoek met invoervariabelen

voor elk aantal hoeken steeds in het midden van het scherm staat’, was de opmerking van een van de leerlingen. ‘Hoe los ik dat op?’

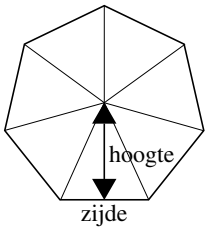


fig. 14 Hoe krijg je een veelhoek midden op het scherm?

Prachtig dat er spontaan een nieuwe onderzoeksvraag ontstaat, waarvoor ook een stukje niet-triviale wiskunde nodig is. Om de tekening in het midden neer te kunnen zetten, heb je de hoogte nodig, zoals in figuur 14 is aangegeven. (De ‘tekenpen’ begint namelijk in het midden van het scherm.) De wiskundeopdracht wordt dan:

Geef de hoogte als functie van de variabelen *zijde* en *aantal Hoekpunten*.

Peter Boon, Chr. Gymnasium Sorghvliet, Den Haag

Extra informatie
Leerlingenmateriaal bij JavaLogo is onder de titel: *JavaLogo bij Informatica deel 1* uitgegeven bij uitgeverij Edu’Actief. ISBN 90 5766 6936.
Er is ook een website met informatie over JavaLogo: www.pbjboon.demon.nl/javalogo. Daar kan de JavaLogo software worden gedownload. Ook is hier een verzameling applets, gemaakt in JavaLogo, te bekijken.