Scratch (14)

Rene Suiker

De vorige keer, in SoftwareBus 3 van dit jaar, deden we even een stapje terug met betrekking tot Scratch, we keken even naar de basis van het programmeren. Niet dat er een volledige cursus programmeren werd gegeven, maar we stonden ruim stil bij de diverse concepten. En ik heb geen verzoek gekregen om er dieper op in te gaan, maar mochten jullie me tegenkomen op een CompUfair of ander CompUsers-event, dan neem ik graag de tijd om er iets dieper op in te gaan. Maar nogmaals, we zijn niet de IG Programmeren.

Huiswerk

Op basis van die concepten had ik ook wat huiswerk opgegeven. Ik roep in herinnering:
Het eerste huiswerk voor deze aflevering is als volgt:

Opgave 13.1: Laat Scratch dit plaatje tekenen
a. Maak een blok dat een willekeurige veelhoek tekent
b. Gebruik dat blok om een 10-hoek te tekenen
c. Gebruik dat blok om een negenhoek te tekenen
d. Enzovoort tot een driehoek
e. Teken een cirkel (mag je zien als een 60-hoek)
f. Probeer dit zo compact mogelijk te programmeren

Uitwerking

We weten nog, dat een volledige cirkel 360 graden omvat. Een vierkant of een rechthoek neemt ook totaal 360 graden in beslag, er zijn vier hoeken van ieder 90 graden. In feite kan je stellen, dat je voor een x-hoek je dus die 360 door x moet delen om de scherpte van de hoek vast te stellen. Dat gaat goed voor een 3-hoek, een 4-hoek, enz. Maar voor een 7-hoek wordt het iets lastiger, omdat 360 niet goed door 7 valt te delen. Ook bij 11 wordt het iets lastiger. Maar het gaat nu niet om wiskundige nauwkeurigheid, als we telkens op dezelfde plek beginnen en eerst naar rechts richten zal de afwijking in de praktijk wel meevallen.
In de instructie geef ik aan dat we een eigen blok moeten maken om een veelhoek te tekenen. Dat blok gebruiken we dan om een 10-hoek, een 9-hoek, enz., tot en met een 3-hoek te tekenen. Een tweehoek is niets anders dan een lijn, die zouden we nog kunnen tekenen, maar dat voegt niets toe. Uiteraard werken we hier verder met gehele getallen, een 3,6-hoek bestaat niet.

De opdracht is ook nog om compact te programmeren. Dat is een extra uitdaging, maar in het algemeen willen we eerst dat het programma gewoon werkt.
Scratch biedt ons wel de gelegenheid programma's overzichtelijk en compact op te stellen. Laten we maar eens van start gaan.

De hoofdlus

Figuur 1 - De hoofdlus

De kleuren zeggen veel in Scratch, ik ga het toch even expliciet benoemen, als een soort opfrisser. We beginnen dus bij (1) met de welbekende gebeurtenis, dat er op de vlag wordt geklikt. Als dat gebeurt, voeren we de initialisaties uit. Die breng ik het liefst onder in een eigen blok, dat we dus aanmaken onder 'Mijn blokken'. Dan (2) starten we met de 10-hoek en tellen terug. In de initialisatie heb ik 'hoeken' op 10 gezet, dat laat ik straks zien. Vervolgens ga ik de instructies binnen in de herhaal-lus uitvoeren totdat hoeken minder is dan 3. Elke keer ga ik dan naar een startpositie en een startrichting, om te voorkomen dat we ergens een verschuiving krijgen als ge-volg van afrondingsverschillen. Vervol-gens (4) roep ik de x-hoek functie (ook een eigen blok) aan, met het aantal hoeken en de staplengte als parameters. Dan verminder ik 'hoeken' met 1 en dat is de herhaal-lus.
Ten slotte (5) teken ik een cirkel.

De initialisatie

Figuur 2 - Initialisatie

De initialisatie is hier erg kort en eigenlijk wel vanzelfspre-kend. We willen de sprite die tekent niet zien, we tillen de pen op en we wissen het scherm. Vervolgens stellen we de variabelen in met hun startwaarden.

De veelhoek

Figuur 3 - Veelhoek

Dit is de functie die het feitelijke werk doet. Op zich ook nog steeds lekker compact. In de hoofdlus werd deze functie een aantal keren aangeroepen, hier (1) wordt deze gedefinieerd. Als je een eigen blok aanmaakt, dan geef je deze een naam en kun je allerhande argumenten (parameters) meegeven. In dit geval hebben we gekozen voor twee parameters, name-lijk het aantal hoeken en de staplengte. Het aantal hoeken geeft uiteraard aan hoeveel hoeken je figuur moet hebben, dus een vierkant is een 4 hoek. In regel (2) maken we nog een lokale parameter aan, die we 'draaien' noemen. Deze geeft aan hoe veel we moeten draaien. Zoals hiervoor gemeld, is dat 360/(aantal hoeken). Omdat we steeds dezelfde startpositie nemen, dezelfde uitgangsrichting en dezelfde stapgrootte, kunnen we de afrondingsfouten redelijk opvangen.
Omdat we een X-hoek maken, moeten we de acties X keer herhalen (3). Wat we dan doen (4) is een aantal stappen zetten en dan draaien, zoveel graden als we eerder (2) gedefinieerd hadden. Ten slotte sluiten we de functie netjes af, door de pen weer op te tillen. Binnen de functie hadden we "m uiteraard geactiveerd.

De cirkel

Ik begon mijn programma door de cirkel te definiëren als een 60 hoek en dan de x-hoek met waarde 60 op te roepen. Dat zou goed moeten gaan, maar door een foutje liep die helemaal niet. Intussen heb ik het foutje wel gevonden, maar dat doet er nu nog even niet toe (straks wel). Daarom heb ik de cirkelfunctie iets anders gedefinieerd, dus geen gebruik makend van de veelhoek functie:

Figuur 4 - De cirkel - eerste poging

We definiëren deze functie (1) zonder parameters. We gaan naar een iets andere startpositie, want we willen gelijk draaien, we gaan dus halverwege de staplengte staan. De y-positie is uiteraard wel gelijk. Vervolgens richten we wel eerst naar 90 graden (naar rechts dus) en zetten de pen neer. Vervolgens gaan we 60 keer (3) twee stappen zetten en 360/60 = 6 graden draaien. Als dit allemaal gebeurd is zetten we de pen weer omhoog. Dit was telkens niet nodig, maar is wel zo netjes. Als de functie ooit eens in een andere context aangeroepen wordt, wil je niet dat de pen een spoor trekt als hij naar een nieuwe startpositie gaat.

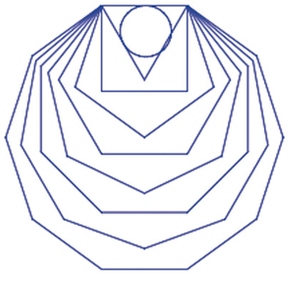
De staplengte heb ik empirisch vastgesteld op 64, zodat de cirkel mooi aansluit.

De verbeterde cirkel

Figuur 5 - Verbeterde cirkel

Deze cirkel is wat ik voor ogen had en ook lekker compact. We gaan inderdaad naar een startpositie iets verder naar rechts, zoals bij de andere cirkel. Maar daarna roepen we de veelhoek aan met 60-hoeken en stapgrootte 2.

En dan is dit het resultaat:



Figuur 6 - Het resultaat

Dit lijkt toch wel vrij goed op wat ik voor ogen had. In de oorspronkelijke run deed het programma er vrij lang over om dit figuur te tekenen en werd de cirkel niet getekend, maar verscheen er een krom lijntje vanaf de startpositie van de cirkel.

Er zat dus iets fout, waarna ik er voor koos om de cirkel apart te tekenen. Tenslotte moet de tekst wel op tijd bij de redactie liggen, anders komt de SoftwareBus niet op tijd uit.

Maar goed, ik had het probleem gevonden en omdat je van fouten kunt leren, nu de volgende vragen voor de oplettende lezer:

Opgave 14.1:

a. In welk blok zat de fout:
   1. In de hoofdlus
   2. In de initialisatie
   3. In de definitie van de veelhoek
   4. In de definitie van de cirkel

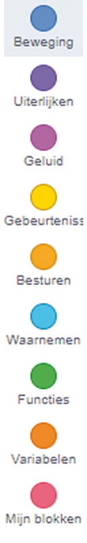
b. Gegeven dat het programma eerst de cirkel niet geheel tekende, maar ook veel trager draaide, heeft u enig idee wat de fout geweest kan zijn?

Opgave 14.2:

a. De opdracht was om zo compact mogelijk te programmeren. Zie je nog een manier om het programma, met behoud van functionaliteit, compacter te maken?

b. Zie je een manier om het programma efficiënter te laten lopen, dus eventueel met meer code, maar snellere uitvoering?

Nog even...
Ik was dit programma begonnen vanuit een leeg scherm, dus gewoon door binnen Scratch op ‘Maak’ te drukken. Weten we het dan, wat we allemaal moeten doen? Een ‘leeg’ project bevat één sprite, de kat. Verder nog geen code, maar de sprite wordt geleverd met twee uiterlijken en één geluid, ‘miauw’. En de volgende codegroepen zijn beschikbaar:



Figuur 7 - Codeblokken

Om met de pen te kunnen werken, moeten we eerst een uitbreiding toevoegen, de pen.

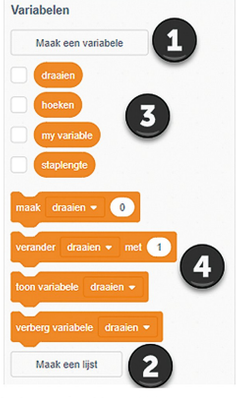
Onderaan bij deze blokken vind je een knopje waarmee je uitbreidingen kunt toevoegen.

Je hebt daarvoor nu de keuze uit elf uitbreidingen, tenminste, nu ik dit schrijf.

Misschien later nog meer.

Variabelen

Figuur 8 - Variabelen



Vervolgens nog even terugkomen op de eigen variabelen. Daarvoor druk je op het oranje cirkeltje ‘Variabelen’ en dan krijg je de volgende mogelijkheden ter beschikking:
Bij (1) maak je een nieuwe variabele. Bij (2) maak je een bijzondere variabele, namelijk een lijst. Hierin kun je verschillende waarden opslaan, daar kom ik nog wel een keer op terug, maar nu even niet. Voor nu is het genoeg om te weten dat, zodra je een lijst aanmaakt, er ook extra functies ter beschikking komen (om de lijst te bewerken).
Bij (3) zie je de variabelen die al gedefinieerd zijn. ‘My variable’ is standaard beschikbaar, die kun je ook aanpassen, maar dat doet verder niet zo ter zake. Voor het programma heb ik drie nieuwe variabelen geïntroduceerd. Je kunt de variabelen aanvinken, dan wordt de waarde op het speelveld getoond. Doe je dat voor heel veel variabelen, dan wordt het beeld onrustig, maar zolang je nog aan het ontwikkelen

bent, en zeker tijdens het foutzoeken, kan het handig zijn de variabelen in beeld te brengen.
Bij (4) staan de functies opgesomd die je ter beschikking hebt om de variabelen te manipuleren. Je kunt ze een waarde meegeven, je kunt de waarde veranderen, je kunt ze tonen en verbergen.

Als je een variabele 'per ongeluk' hebt aangemaakt, dan kun je met rechts op de variabele klikken en dan aangeven dat je hem wilt verwijderen. Pas daarmee op, want als je een variabele verwijdert die in gebruik is, dan loopt je programma niet meer. Daarom krijg je ook een waarschuwing als je een in gebruik zijnde variabele wilt verwijderen. Je kunt ook de naam van een variabele veranderen, als je achteraf gezien niet tevreden bent met de naamgeving. Dit gaat gewoon goed, want hij wordt overal aangepast waar die op dat moment gebruikt wordt.

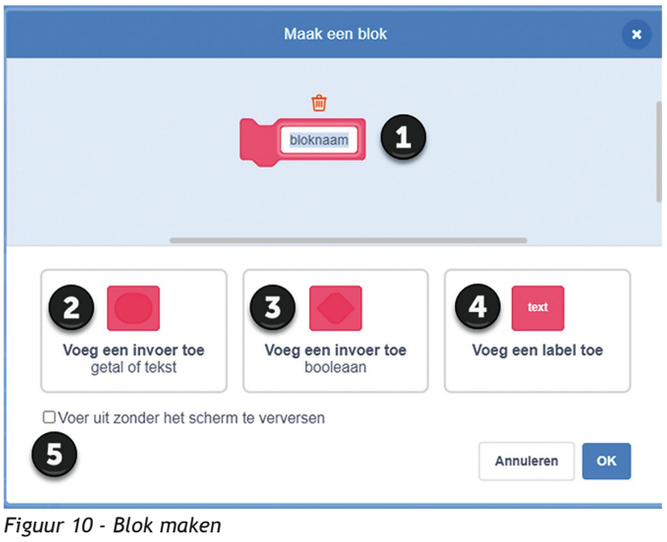
Mijn blokken

Figuur 9 - Mijn blokken

Onderaan figuur 7 zie je 'Mijn blokken' staan. Als je op dat blokje klikt, dan krijg je de opties in beeld om met je eigen blokken te werken.

Niet bijster spannend, je ziet hier een blokje 'Maak een blok' en vervolgens zie je de blokken die ik in het kader van dit projectje gedefinieerd heb. De blokjes zelf hebben de vorm die je voor veel instructies terugziet, dus met de inham boven en het bolletje onder, zodat ze goed kunnen aansluiten. En voor de parameters zijn de ovaaltjes beschikbaar, zodat je waarden, variabelen of functies kunt invoegen.

Klik je op 'Maak een blok' dan komt onderstaand scherm in beeld:



Figuur 10 - Blok maken

Figuur 11 - Definieer blok.

Bij (1) vul je de naam van je blok in. Ik zou dit wel betekenisvol doen, dat maakt later onderhoud van je project een stuk eenvoudiger. Klik je vervolgens op 'Ok' dan wordt je blok aangemaakt, kom dus ook beschikbaar in de lijst van figuur 9, en op je programmascherm zie je één extra blok beschikbaar om de functie te definiëren. In dezelfde kleur roze, en daaronder kun je alle code toevoegen die je beschikbaar hebt, inclusief andere eigen blokken.

We hebben dit blok dus benoemd, we hebben geen parameters meegegeven en nog geen inhoud gedefinieerd. Maar we kunnen desondanks 'Demo' nu wel in onze code opnemen. Hij doet dan nog niets, maar je kunt dit wel gebruiken om bijvoorbeeld je hoofdlus te testen.

Het blok staat dus links bij de blokken en in het programmaveld heb je de mogelijkheid om het blok te definiëren. Als je het blok wilt verwijderen, dan kun je rechts klikken op het 'definieer'-blok, dus niet in het blokkenoverzicht. Op beide plaatsen kun je met rechts klikken en dan 'Bewerken' kiezen. Kies je daarvoor, dan krijg je weer figuur 10 in beeld.

Als je nu twee keer op blokje (2) klikt, dan voeg je 2 parameters toe, die je bij de aanroep dus mee moet geven. Als je vindt dat je te veel parameters hebt meegegeven, dan kun je die parameter selecteren en dan met het prullenbakje erboven deze verwijderen.

Doe dit niet met de naam, want ik zou niet weten hoe je dan het blokje weer een naam moet geven. En blokjes zonder naam zijn niet handig. De parameters die je meegeeft zijn invoer-parameters; deze kun je als waarde in je onderliggende definitie gebruiken. Het is (bij mijn weten) niet mogelijk om ook een output-parameter te definiëren. Als je code een waarde terug moet geven, moet je daar een variabele voor definiëren.

Als je een blok wilt verwijderen moet je er eerst voor zorgen, dat het nergens meer wordt gebruikt. Scratch behoedt je dus voor al te onvoorzichtig handelen.

Hiermee hebben we dus de opgave 13.1 in zijn geheel behandeld. De totale code is m.i. redelijk compact, in één oogopslag:

Figuur 12 - Complete code opgave 13.1

Opgave 14.3:

a. Pas het programma zo aan, dat elke veelhoek een andere kleur krijgt
b. Pas het programma zo aan, dat de cirkel rond het midden wordt getekend en dan de veelhoeken er omheen.

Dan gaan we nog even terug naar ons priemgetallenproject: **https://scratch.mit.edu/projects/438325757/**

Ik heb daar iets aan geoptimaliseerd, zien jullie ook wát? Maar met de kennis die we vandaag hernieuwd hebben opgedaan over eigen blokken, wat zou je in de aanpak veranderen als je nu dit programma opnieuw zou schrijven?