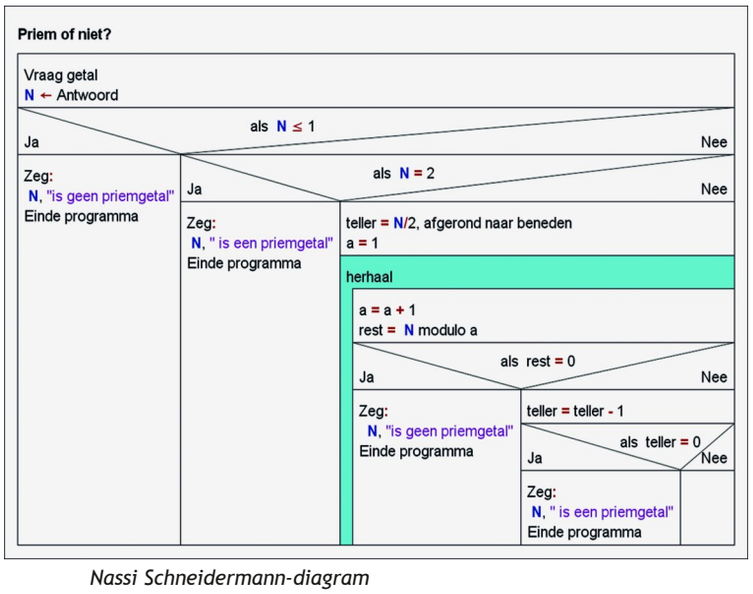
Scratch (11)

René Suiker

Toch nog een Scratch, ondanks de verwachtingen. Het idee was dit jaar drie afleveringen Scratch. Hopelijk vindt u het niet vervelend, maar het worden er toch vier. U weet misschien nog: de activiteiten rondom Scratch werden ‘gesponsord’ door het Platform WebOntwerp. Dat was een win-win-situatie, want zo kon het Platform elk nummer iets schrijven én kreeg Scratch aandacht. Maar eigenlijk is het een gekunstelde constructie, want al draait Scratch op het world wide web, het heeft niets met webontwerp zelf te maken. En daarom in dit nummer een bijdrage rondom webontwerp én een aflevering Scratch. Eerlijk gezegd ook, omdat we tussen de nummers 5 en 6 van de SoftwareBus maar weinig tijd hebben om kopij te verzamelen, en over Scratch kan je gauw makkelijk én uitgebreid schrijven. Vandaar.

Er is in elk geval één lezer die mijn werk beziet, want ik kreeg naar aanleiding van mijn artikel over priemgetallen in Scratch zowaar weer eens een reactie. En leuke reactie, van de heer Theo Hupkens, die een programmeeropleiding geeft. Hij had daarbij voor een examen een opgave had gemaakt over Nassi Schneidermann-diagrammen, waar ik ook uren over zou kunnen vertellen, maar dat voert in dit verband misschien wat ver. Maar in dat kader had hij een voorbeeld gemaakt rondom priemgetallen. Dat voorbeeld had hij ook getest met feitelijke code, in dit geval Python, maar hij wilde het ook in Scratch proberen.

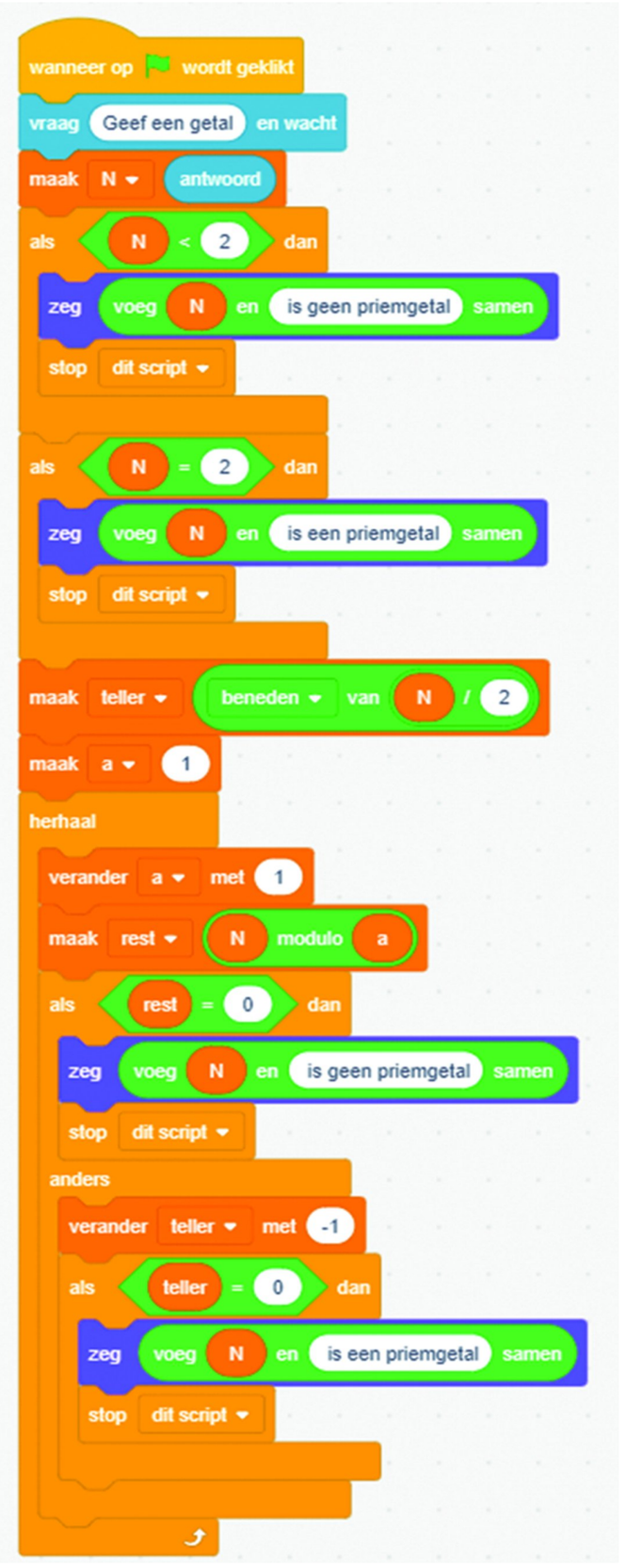
Een Nassi Schneidermann-diagram helpt bij het gestructureerd programmeren. Het voorbeeld dat de heer Hupkens meestuurde is als volgt. Ik heb het opgenomen, omdat het ook leerzaam is.



Nassi Schneidermann-diagram

In dit schema wordt grafisch weergegeven hoe de logica van de opname rondom het priemgetal in elkaar steekt. Het begint met het vragen van een getal. Als dit getal kleiner of gelijk is aan 1, dan is het geen priemgetal. Als het getal 2 is, dan is het een priemgetal. Voor getallen groter dan 2 kijkt het programma in de lus, die N/2 keer doorlopen wordt, of het getal een keer deelbaar is door de teller, die vanaf 2 telkens met 1 verhoogd wordt. Als er een keer sprake is van een deelbaarheid, dan is het geen priemgetal. Als er geen getal gevonden kan worden waardoor het deelbaar is, dan is het dus wel een priemgetal. Op zich netjes uitgewerkt.

De heer Hupkens had het schema ook in Scratch uitgewerkt, dat ziet er dan als volgt uit:



In Scratch vertaald

Zoals u ziet wordt het schema bijna woord voor woord gevolgd. Dit programma is een stuk efficiënter dan het voorbeeld dat ik had gegeven, dus heel leuk dat iemand de moeite neemt om dat ook uit te proberen. Chapeau!

Maar een aantal ‘problemen’ die ik in de opgave had opgenomen zijn ook in dit programma nog aan de orde. Wel gaat het voor een getal een stuk sneller dan het voorbeeld dat ik heb gegeven.

Ik had zelf een stuk code achtergelaten, waarbinnen nog een functie ‘priemgetal’ gedefinieerd moest worden. Ik heb dus niet alles in één loop gedaan, maar had een ‘hoofdprogramma’ en een ‘subroutine’ gedefinieerd. Dat is met name voor grote, complexe zaken aan te bevelen, omdat dit het onderhoud ten goede komt. Het maakt het zoeken van fouten (debuggen) ook eenvoudiger, maar voor iets zo rechttoe-rechtaan als een priemgetallencontrole is het wat overdreven. In het kader van de opdracht was het trouwens wel verantwoord.

Mijn functie, nog steeds niet helemaal optimaal:

De functie ‘priemgetal’

Bij nummer (1) zie je een gekleurd blok ‘definieer’ met de naam van de functie erin. Dat vind je terug onder ‘mijn blokken’ binnen de codegroepen. In dit geval is het een routine die verder geen input nodig heeft om zijn werk te doen, omdat hij de globale variable ‘input’ meeneemt.

Feitelijk is het netter om routines zo te schrijven dat ze deze waarde meenemen, dus als argument in de functie. Daar kan ik wel een keer wat uitgebreider op ingaan, maar niet in dit kader. Hier heb ik er voor gekozen om de aanroep eenvoudig te houden, zie verderop bij figuur 4.

Bij nummer (2) worden twee variabele gedefinieerd binnen de functie. In sommige talen zouden deze buiten de routine niet beschikbaar zijn, maar in Scratch kan je deze waarde wel teruggeven naar het hoofdprogramma. De functie zelf geeft namelijk geen waarde terug. Verder kan je alle variabelen in het speelveld zichtbaar maken. Weet u nog hoe dat moet?

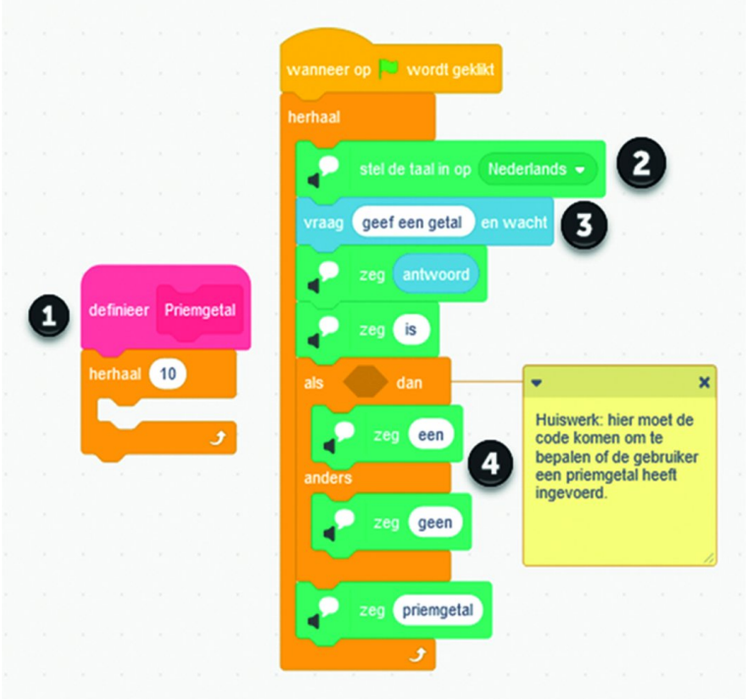
Bij blokje 3 wordt de herhaling in gang gezet. In principe ga ik, door te herhalen, zo vaak als het opgegeven getal ruim te veel keren de lus door. Daar heb ik in een later stadium wat aan gedaan, want in feite is het genoeg om de lus te doorlopen vanaf de waarde 2 tot en met de wortel van de input. Maar Scratch kent de functie ‘wortel’ niet. Die zou ik ook wel kunnen definiëren, maar dat voert wat erg ver in dit verband.

De heer Hupkens maakte gebruik van de modulo-functie; dat is feitelijk netter (en efficiënter) dan de code die ik bij blok (4) gebruik, dus dat is een verbetering van mijn code.

In blok (5) gebruik ik een manier om het programma efficiënter te maken. Als vaststaat dat het getal geen priemgetal is, bijvoorbeeld omdat het deelbaar is door 2, dan hoef je niet nog duizend keer de lus door, met alle tijdverlies van dien. Verder dacht ik dat, als ik de teller kwadrateer en ik daarmee over de input heen ga, ik ook kan ophouden, want als het getal niet deelbaar is door getallen kleiner dan de wortel, dan toch ook niet door getallen groter dan de wortel.

Controleer maar voor uzelf. Helaas, ik ben er nog niet achter waarom niet, maar het programma stopt niet op basis van ‘teller*teller> input’. Ik ben benieuwd of iemand hier een verklaring voor heeft.

De opgave zoals die in het project stond:
https://scratch.mit.edu/projects/388895306

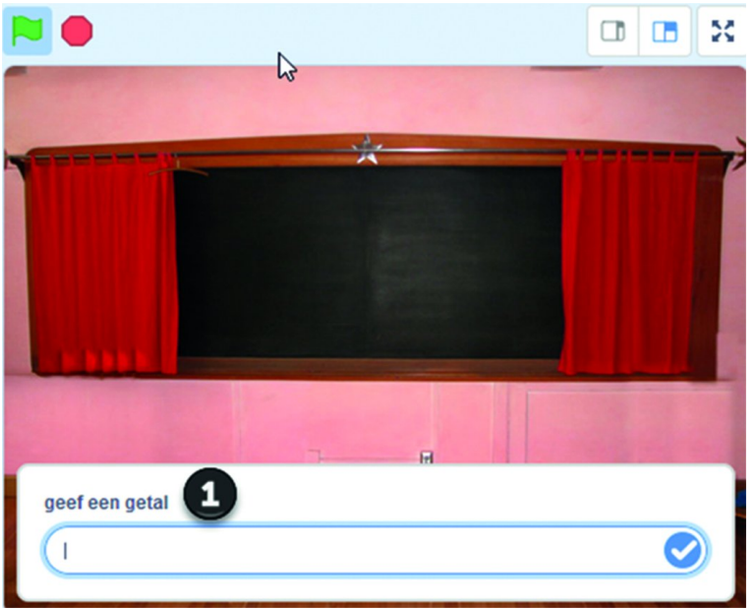


De originele opgave

Ik had al opgegeven dat je een functie moest definiëren, zoals je ziet bij blokje (1). Zie voor een voorbeeld dus figuur 3, maar deze kan nog steeds geoptimaliseerd worden. Onder andere met de lessen van de heer Hupkens, maar er zijn meer verbeteringen mogelijk.

Bij blokje (2) heb ik de taal ingesteld, want we laten hem praten. Zoals ik de vorige keer al aangaf kan je ook de stem nog instellen, maar dat is voor deze opgave niet zo belangrijk.

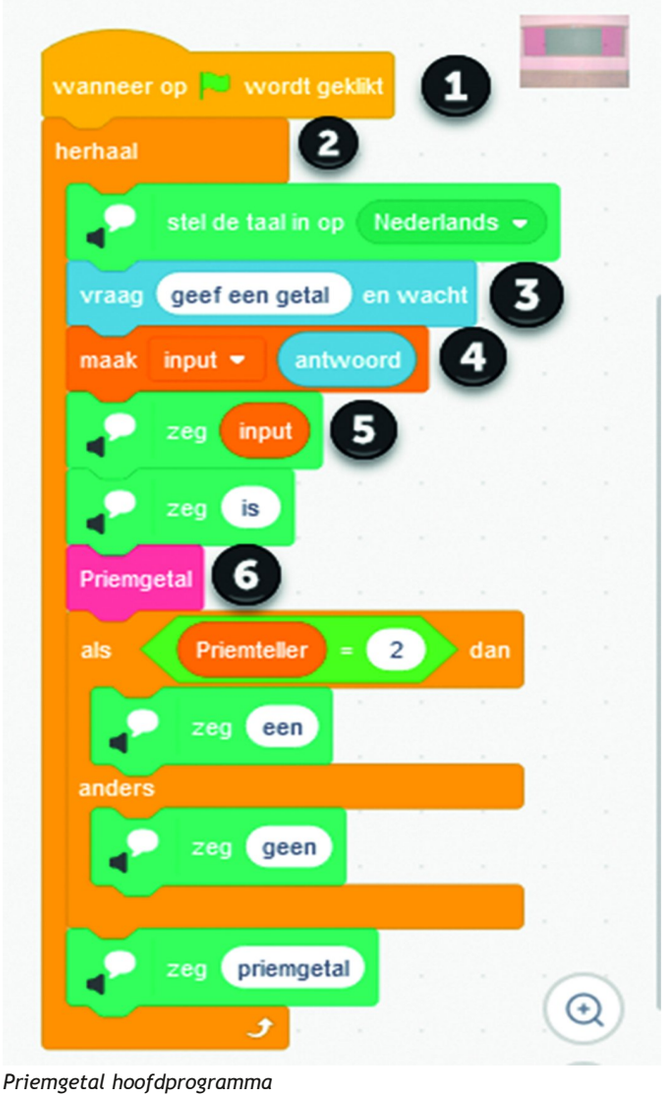
Bij blokje (3) vraag je de gebruiker om input. Op het scherm ziet dit er als volgt uit:
(Zie de afbeelding op de volgende pagina)



Input op het scherm

In blokje (4) van figuur 4 wordt vervolgens aangegeven of het een priemgetal is of niet.

In mijn uitwerking onder account 'softwarebus':
https://scratch.mit.edu/projects/438325757
heb ik het hoofdprogramma een klein beetje aangepast en dit ziet er nu als volgt uit:
Bij blokje (1) zie je dat het programma start met het in-



Priemgetal hoofdprogramma

drukken van de groene vlag, de gebruikelijke manier van het starten van een Scratch-programma (maar het hoeft niet per se zo; er zijn alternatieven).

Bij blokje (2) zie je dat we dit programma met een eeuwige lus hebben gebouwd. Dat betekent dat het programma alleen stopt met de rode knop of door de browserwindow af te sluiten. Bij blokje (3) vraag je weer de input van de gebruiker en bij blokje (4) sla je het gegeven antwoord op in de variabele 'input'.

Bij blokje (5) begin je te praten, waarbij Scratch het opgegeven getal noemt en alvast 'is' zegt, dus bijvoorbeeld 'vijftien is'.

Bij blokje (6) wordt dan de functie 'priemgetal' aangeroepen, die een waarde 'priemteller' teruggeeft. Als die de waarde 2 heeft, dan is het een priemgetal, als die een andere waarde heeft, dan is het geen priemgetal.

Achteraf is de code die de heer Hupkens aanlevert om het antwoord te geven veel netter. Weet u ook waarom? Ik kom er straks op terug; eerst maar eens even naar de opgaven van de vorige keer:

Opgave 10.1:
Hoe is het programma te foppen? En wat kun je er aan doen?
Er is geen controle of er een geheel getal wordt ingevoerd. Als je een tekst invoert dan vindt het programma nooit een priemgetal, dus zal hij zeggen: elf is geen priemgetal.
Je kunt dat oplossen door de input te controleren voordat je ermee aan de slag gaat. In feite wil je dat de input een positief geheel getal is. Alle andere input is nooit een priemgetal. Dus als de input niet aan dit basiscriterium voldoet, dan kun je gewoon zeggen dat het geen priemgetal is, zonder de priemgetalfunctie aan te roepen. Maar in feite is dat ook nog niet correct, want als iemand 'elf' intypt zegt het programma nu 'elf is geen priemgetal' en dat is natuurlijk onjuist.
Beter is dus om eerst te controleren of er een getal wordt ingevuld. Als er een tekst wordt ingevuld vraag je nadrukkelijk om een getal en ga je niet verder.

Opgave 10.2:
Zowel de functie 'Priemgetal' als het hoofdprogramma kunnen efficiënter. Welke mogelijkheden zie je en hoe los je het op?
Je hoeft niet de hele lus door; als je ervan uitgaat dat elk getal deelbaar is door zichzelf, kun je bij 1 beginnen en maar doorgaan tot de wortel van het getal, naar boven afgerond. In feite hoef je zelfs alleen maar te bekijken of het deelbaar is door priemgetallen. Als het niet deelbaar is door priemgetallen, is het vanzelf een priemgetal. Alleen, dat bijhouden van priemgetallen is een complexe klus, maar het verminderen van het aantal keren dat de lus doorlopen wordt levert nadrukkelijk tijdwinst op. Niet meetbaar bij kleine getallen, maar zodra je boven het miljoen uitkomt ga je het wel merken.

Oorspronkelijk had ik de 'stop dit script'-instructie niet toegepast, maar dit levert veel tijdwinst op. Als je nu het getal 1234561890 invult, dan komt het programma gelijk met het antwoord.

Het hoofdprogramma kan efficiënter door niet telkens de taalkeuze te herhalen. Eén keer instellen is genoeg. Ik weet niet of het veel winst geeft, maar het is in elk geval beter om het niet onnodig te herhalen.

Waarom was de constructie voor het antwoord geven van de heer Hupkens beter? Omdat hij in één keer het antwoord geeft. Mijn constructie begint met het antwoord, maar als het een groot getal is, dat toch wel op een priemgetal lijkt, dan kan het wel even duren voordat het antwoord gegeven wordt. En dat klinkt niet erg professioneel.
Eerder vroeg ik of u nog wist hoe op het speelveld de vari-

abelen zichtbaar werden gemaakt. Dit gaat als volgt, mocht u het vergeten zijn:
Bij blokje (1) kies je ‘variabelen’. Onder blokje (2) kan je

Variabelen

voor elke variabele vastleggen of die op het speelveld getoond moet worden. Dit oogt niet erg fraai, zoals ik het hier laar zien, maar dit is wel gemakkelijk bij het fout zoeken. Ik raad dus aan, om deze aan te laten staan, totdat je zeker weet dat je programma foutloos is.

Op dit moment ben heb ik het gevoel, dat ik Scratch wel zo’n beetje heb uitgelegd. In 11 artikelen, plus nog wat artikelen van anderen, hebben we de nodige onderwerpen behandeld:

1. Uitleg over wat Scratch is
2. Een account maken
3. Programmeren met de standaard bouwstenen
4. Het belang van Sprites
5. De structuur van een programma
6. Variabelen
7. Modules
8. Eigen bouwblokken
9. Praten
10. Animaties
11. Input van de gebruiker
12. Rekenen
13. De Scratch omgeving en kennis delen

Ik verwacht nu eigenlijk, dat u samen met uw (klein)kinderen aan de slag kan en alle uitdagingen aankunt. Zaken die ik nog uit wil zoeken, maar waarvoor nu echt de tijd ontbreekt, zijn het aansturen van andere apparaten, zodat je met Scratch bijvoorbeeld je drone kunt aansturen, of een ander apparaat, zoals bijvoorbeeld Arduino, 3D printers of bijvoorbeeld Fischer Techniek.

Opgave 11.1:

a. Zoek een plaatje van een doolhof
b. Gebruik dat als achtergrond in je programma
c. Zoek een leuk plaatje van een object om door het doolhof te loodsen
d. Verzin een toetsenbordbesturing
e. Maak een programma waarmee de gebruiker het object door het doolhof loodst
f. Houd onderweg de tijd bij
g. Telkens als de rand wordt geraakt gaat het object terug naar de startpositie
h. Bij vijf keer herstart is het spel afgelopen
i. Als het de gebruiker lukt om tot het eind te komen volgt een muziekje en wordt de tijd in het groot getoond

Succes!