Scratch (13)

René Suiker

Zoals beloofd kom ik af en toe nog even terug op Scratch, maar het is niet meer de bedoeling dat ik dit in elke uitgave ga doen. Nog eens terugkijkend naar wat ik zoal geschreven heb kom ik tot de ontdekking, dat ik wel veel over Scratch heb uitgelegd, maar dat ik misschien in de basis wel sommige aspecten te gemakkelijk als bekend heb verondersteld.

We zijn natuurlijk CompUsers, we richten ons op toepassingen, we zijn niet de IG Programmeren. Veel van ons hebben dat in het verleden misschien wel eens gedaan, maar mogelijk heel veel van u ook nog niet. Dus misschien had ik meer mensen onderweg mee kunnen nemen, als ik de basis van programmeren had uitgelegd voordat we de diepte in gingen in Scratch.
Overigens is het niet de IG Programmeren, die zich op Scratch richt, je komt buiten CompUsers meer Scratch-kennis tegen bij de IG Artificiële Intelligentie.

Programmeren

Het voert te ver om nu nog even een hele cursus programmeren in te passen in deze reeks, maar ik ga toch even een stapje terug doen en iets meer van de basis theorie uitleggen. Als we de definitie die we op Wikipedia vinden er eens bij pakken, dan zien we het volgende:

> *Programmeren is het schrijven van een computerprogramma, een concrete reeks instructies die een computer kan uitvoeren. Dit is de taak van een softwareontwikkelaar of programmeur. Programmeren wordt in het algemeen niet direct in machinetaal gedaan, maar in een programmeertaal. De programmacode die wordt geschreven heet broncode en wordt door een assembler, compiler of interpreter omgezet in machinecode. Met name in het verleden werd voor programmeren ook coderen gebruikt.*

Door te programmeren kun je een computer dingen laten doen die jij wilt laten gebeuren. Computers kunnen bepaalde zaken beter dan mensen en vaak ook veel sneller. En een computer kan repeterend werk foutloos en probleemloos doen, terwijl wij mensen nog wel eens een steekje willen laten vallen als we hetzelfde werk keer op keer moeten herhalen.
Nu zijn er heel veel programmeertalen in omloop, elk met eigen kenmerken en sterktes en zwaktes. Scratch is ook een programmeertaal en deze wordt uitgevoerd door een interpreter. Een interpreter zet tijdens de executie van een programma het programma stap voor stap om in stukjes code die de computer uit kan voeren. Een compiler zet eerst het hele programma om in machinecode, alvorens het uitgevoerd wordt. Daarom heeft een compiler iets meer tijd nodig voordat een programma kan starten, maar loopt het programma doorgaans sneller. En een compiler bekijkt eerst het hele programma en ziet doorgaans fouten al eerder en start het programma dan helemaal niet. Een interpreter ziet de fout vaak pas als de specifieke fout aangeroepen wordt en stopt dan pas met de uitvoering.
Dit even naar aanleiding van compiler of interpreter. Assembler wordt ook genoemd, maar dat laten we in dit kader even voor wat het is.

Programmeerconcepten

Ik heb in eerdere artikelen ooit wel eens verwezen naar sites van onze zuiderburen en dat ga ik vandaag ook weer eens doen. Op de site https://www.leer-scratch.be/menu/ krijg je een cursus Scratch voorgeschoteld waarbij ze Scratch gebruiken om uitleg te geven. Heel leuk gedaan. Ik kwam op deze site terecht toen ik zeker wilde stellen, dat ik de juiste set programmeerconcepten bij elkaar had en het voelde prettig om daar de bevestiging te vinden. Maar daarnaast was ik onder de indruk van de eenvoudige wijze waarop het daar uitgelegd wordt. De volgende concepten kom je in de meeste programmeertalen tegen, ook in Scratch:
1. Sequentie
2. Input & Output
3. Variabelen
4. Herhalingen
5. Beslissingen
6. Eigen blokken

Op al deze aspecten ga ik iets verder in. Overigens is de site bedoeld voor docenten van basisschoolleerlingen in de bovenbouw. Dit sluit aan bij onze doelgroep (groot)ouders met (klein)kinderen, dus op en top bruikbaar.

Sequentie

In feite de volgorde van het programma. In Scratch wordt dit grotendeels bepaald door de volgorde waarin de instructieblokken aan elkaar gekoppeld zijn. De volgorde kan soms afwijken; daar kom ik bij beslissingen en herhalingen op terug. Sommige programmeertalen gaan anders om met de volgorde. Voorheen, toen we nog minder met de grafische omgevingen werkten, begon een programma in principe bovenaan en liep door naar beneden tot het klaar was. Met herhalingen en beslissingen kon van deze volgorde worden afgeweken. Maar binnen een blok wordt toch vaak de volgorde van boven naar onder doorlopen.
In Scratch kun je meerdere blokken hebben die parallel worden uitgevoerd. Elke sprite kan blokken hebben, elke sprite kan ook nog eens gekloond worden, dus er kan veel tegelijkertijd. Maar elk van die blokken wordt in een logische volgorde afgewikkeld.

Input & Output

Een programma dat niet met de omgeving communiceert kan ook nuttig zijn, maar de meeste programma's hebben interactie met de omgeving. Een gebruiker kan input geven, bijvoorbeeld met de muis of het toetsenbord, of tegenwoordig ook met behulp van gesproken tekst of beweging van het lichaam. En output kan ook in alle mogelijke vormen, van een getal op het scherm tot een waar kunststuk uit de luidsprekers.
In Scratch kom je niet één blok 'Input & Output' tegen, maar vind je input en output op verschillende plaatsen terug. Zo zijn in feite alle 'beweging'blokken wel output, maar vind je input terug in zowel 'gebeurtenis' als 'waarnemen'. Daarnaast heb je nog extra blokken, zoals 'muziek' en 'pen' (output) en 'video' (input).

Variabelen

Variabelen staan je toe bepaalde waarden op te slaan, er bewerkingen op uit te voeren, ze herhaaldelijk te gebruiken, of dit allemaal tegelijk. Programmeertalen zijn behoorlijk onderscheidend in de manier waarop ze met variabelen omgaan. We hebben ze in feite al vanaf het eerste artikel

(meen ik) gebruikt. In Scratch kunnen we zowel enkelvoudige variabelen maken als een lijst. Lijsten hebben we nog niet behandeld, variabelen wel.

In Scratch kom je variabelen en lijsten tegen in het blok ‘variabelen’. Verder is Scratch heel ruimdenkend ten aanzien van het type variabelen. In een compileertaal is het gebruik van variabelen aan veel meer regels gebonden, maar Scratch vindt het prima als je eerst een geheel getal in een variabele stopt, vervolgens een tekst en vervolgens een gebroken getal. Alleen, bewerkingen zijn vaak wel typegebonden. Zo kun je niet ‘1’ bij een tekst optellen en samenvoegen doe je weer bij strings (dit heeft niets met ondergoed te maken, maar zijn reeksen van letters en/of cijfers).

Herhalingen

Herhalingen, ook wel lussen genoemd, zorgen ervoor, dat instructies meerdere keren uitgevoerd worden zonder dat je de instructie steeds weer moet uitschrijven. In Scratch vind je herhalingen onder ‘besturen’. Je hebt hier de opties om iets continu te herhalen, om iets een vastgesteld aantal keren te herhalen (maar dat vastgesteld kan door middel van een variabele wel variabel zijn) of om iets te herhalen zolang een bepaalde conditie van kracht is.

Beslissen

Soms moet een programma een beslissing nemen. De meeste beslissingen bevinden zich ook in de categorie ‘besturen’. Je hebt het blok ‘als’ met ‘dan’, waarbij de instructie bij ‘dan’ wordt uitgevoerd als de conditie ‘waar’ is. Je hebt ook de constructie ‘als’ met ‘dan’ en ‘anders’. Hierbij wordt de instructie (of reeks instructies, want dat kan in Scratch ook heel eenvoudig) bij ‘dan’ uitgevoerd als de conditie ‘waar’ is en als dat niet het geval is, dan wordt de instructie bij ‘anders’ uitgevoerd. Deze beslissingen kunnen ook nog eens in elkaar genesteld worden. Het leuke van Scratch is dat door de blokstructuur je lussen en ‘als-dan’-constructies niet op de verkeerde manier in elkaar kunt schuiven. Het past of het past niet.
Dan kun je nog steeds wel in je logica fouten maken, maar syntactisch klopt het in elk geval altijd.
Ik zei dat beslissingen bijna altijd onder ‘besturen’ stonden, maar we hebben ook elders met beslissingen te maken. Bij ‘beweging’ hebben we het blok ‘keer om aan de rand’ hetgeen natuurlijk ook een impliciete beslissing is.
Verder vind je bij het blok ‘waarnemen’ een hoop interessante opties die je kunt gebruiken bij het vaststellen van de condities in de ‘als-dan’-constructies

Eigen blokken

Bij eigen blokken denk je aan functies of procedures die we zelf een naam geven. Deze manier van werken hebben we ook al toegepast. Door stukjes code zo te benoemen kun je de code overzichtelijker houden dan wanneer je telkens hele blokken herhaalt in je programma. Het is fijn als je de werking van je programma in één oogopslag kunt doorgronden. Met heel ingewikkelde programma’s lukt dat niet, maar het streven is altijd nuttig. En de vorige keer hebben jullie gezien dat je een compleet doolhofspel toch op één pagina kon krijgen.
Tot zo ver de basis over programmeren. Op de hiervoor genoemde website kom je ook een leuke uitleg tegen over algoritmes en decompositie. Beide heel nuttig bij het zelf programmeren en misschien kan ik daar een volgende keer dieper op in gaan, bij gebleken belangstelling.

De oplossing wordt op die site ook gegeven, maar het eerste huiswerk voor deze aflevering is als volgt:

Opgave 13.1: Laat Scratch het hiernaast staande plaatje tekenen

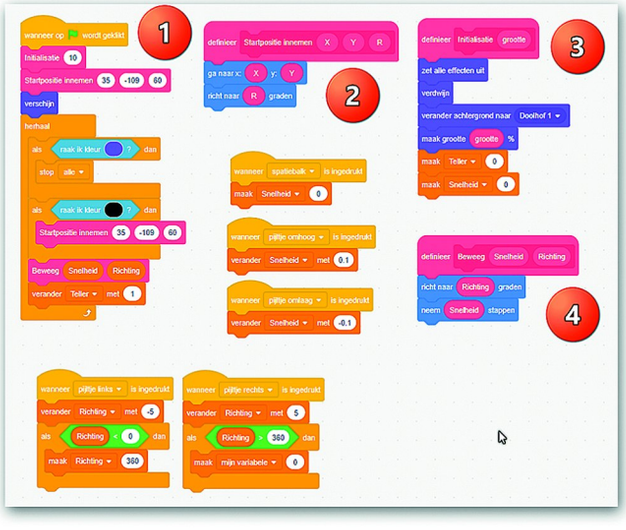
a. Maak een blok dat een willekeurige veelhoek tekent
b. Gebruik dat blok om een 10-hoek te tekenen
c. Gebruik dat blok om een negenhoek te tekenen
d. Enzovoort tot een driehoek
e. Teken een cirkel (mag je zien als een 60-hoek)
f. Probeer dit zo compact mogelijk te programmeren

Het doolhof

Het doolhof, het huiswerk uit aflevering 11, hadden we in aflevering 12 behoorlijk uitgewerkt, maar er stonden nog een paar puntjes open.

Nog even in de herinnering:
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

Je kunt het hele project bekijken op:
https://scratch.mit.edu/projects/474487745/fullscreen/ en je kunt er dan ook een kopie van maken om er zelf aanpassingen op te doen.
We hadden nummer h en i nog openstaan na het voorgaande artikel, omdat we er vanwege ruimtegebrek geen tijd voor hadden.
Kijken we nog eens naar de hele doolhof, dan zouden we inzicht moeten krijgen in waar we onze zaken moeten regelen:



Figuur 1 - Het doolhofprogramma

Nog even ter herinnering, blokje ‘1’ is de hoofdlus, blokje ‘2’ is een eigen blok ‘startpositie innemen’, blokje ‘3’ is een eigen blokje ‘initialisatie’ en blokje ‘4’ is ons eigen blokje

'beweeg'. De overige blokjes zijn voor de besturing.
Opgave h vraagt natuurlijk om een variabele. Die variabele moet uiteraard geïnitialiseerd worden, dus is er een aanpassing nodig in blokje 3. Het ligt voor de hand een variabele te definiëren die 'herstarts' heet, die te initialiseren op waarde 0 (want pas na een herstart heb je de eerste herstart bereikt). Vervolgens moet je elke keer dat je kleur zwart raakt, dus net onder het midden van blokje 1, deze variabele met 1 ophogen. Vervolgens kun je binnen het blok nog een blok met een 'als-dan' toevoegen. Namelijk 'als' 'herstarts = 5' dan 'stop alle'.
We kunnen een en ander nog netjes combineren, want in feite zeg je:

- Als je de rand aanraakt
- Dan verhoog je de variabele herstarts
- Als variabele herstarts gelijk is aan 5
  - DAN stop je alles,
  - ANDERS ga je weer naar startpositie

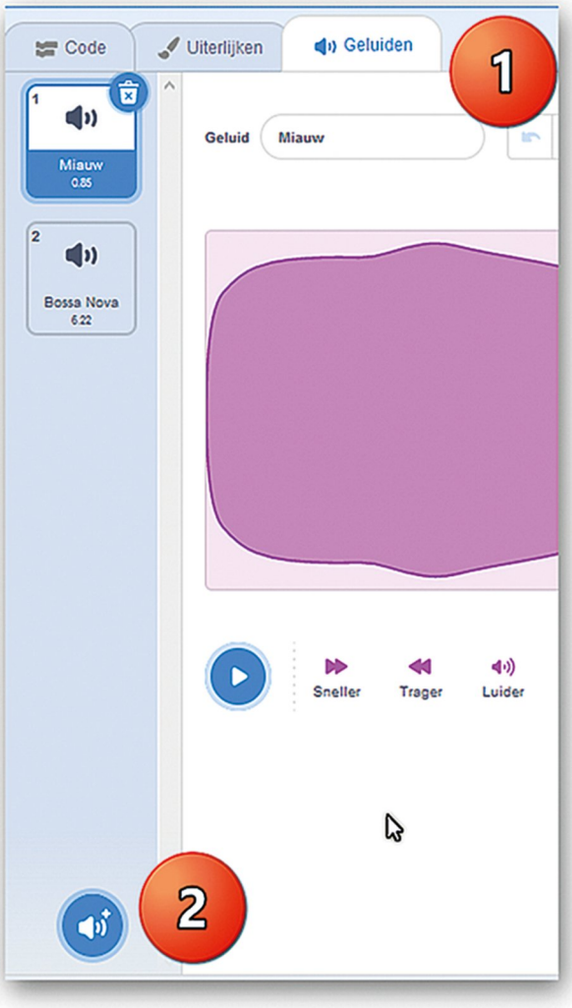
Je zou kunnen afvragen of de variabele groter is dan of gelijk is aan 5, maar volgens de geldende logica zou hij nooit groter kunnen worden dan 5.
Je zou kunnen overwegen om die 'startpositie innemen' uit te breiden met 'gelijk de snelheid terug te brengen naar 0', want nu kan het voorkomen dat bij een zekere snelheid de mot gelijk 5 keer tegen de rand opknalt en dan is het spel in feite uit. Maar dat is een uitbreiding voor de liefhebbers, voor een volgende keer.
De nieuwe hoofdlus ziet er dan als volgt uit:

Figuur 2 - Hoofdlus met oefening H

Ten aanzien van opgave I geldt, dat we niet direct een nieuwe variabele nodig hebben, maar wel een nieuw blokje. We zouden dit blokje 'vier overwinning' kunnen noemen en dat blokje moet in feite twee dingen doen. Overigens hebben we ervan afgezien de tijd te registreren, maar hebben we een stappenteller gebruikt. Je zou natuurlijk een start- en een eindtijd kunnen registreren en dan het verschil uitrekenen, maar je kunt je ook voorstellen, dat je gewoon de teller in het groot toont.
Daarnaast moet er een muziekje gespeeld worden. Op dit moment is daar helemaal niet in voorzien, dus dat moet je ook programmeren. En dat zal nog niet meevallen als je geen verstand van muziek hebt. Met de huidige sprite kun je alleen maar miauwen, dat kunnen we nauwelijks feestelijk noemen.

Wat je kunt doen, is aan de sprite geluiden toevoegen. Dat kun je doen door zelf geluiden op te nemen, maar gelukkig biedt Scratch je ook de mogelijkheid gebruik te maken van de bibliotheek. Een aantal sprites zijn beschikbaar met geluiden, zoals de kat die met zijn miauw mee komt.
Er zijn, zoals gebruikelijk, meerdere wegen die naar Rome leiden. Je moet echter wel verstandig ingrijpen in de code en mogelijk kun je nog wel een keer je neus stoten.
We gaan in eerste instantie voor de makkelijke weg, maar ook hier geldt: je kunt het zo mooi maken als je zelf wilt, maar je zult dan wel moeten zoeken naar de mogelijkheden.

Er waren dus twee aspecten te regelen, het tonen van de score en het spelen van een muziekje. Het spelen van een muziekje is eenvoudig te regelen of moeilijk te regelen. Ik heb zelf voor de makkelijke manier gekozen:



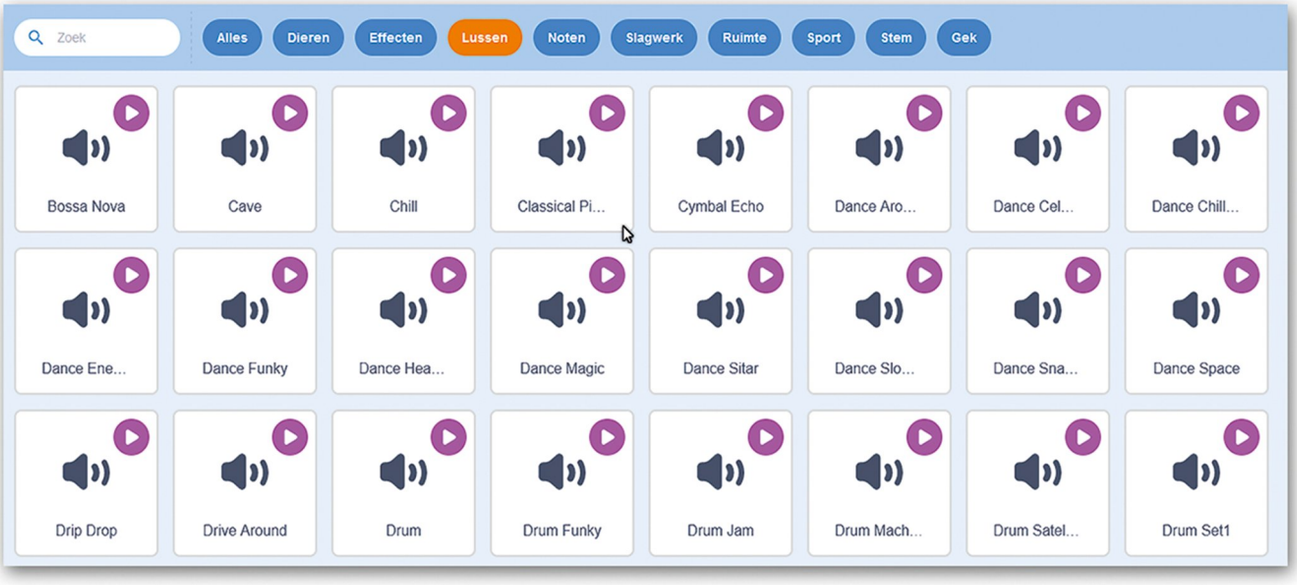
Figuur 3 - Geluiden 1

Ik kies de optie ‘geluiden’ bij ‘1’. Het codeblok verdwijnt dan even uit het zicht en ik kan zien welke geluiden allemaal ter beschikking staan van onze sprite. In dit geval ben ik nog met de mot bezig, maar blijkbaar beschikt deze ook over een miauw. De ‘Bossa Nova’ had ik reeds tijdens het experimenteren toegevoegd, maar dat zou een leuke kunnen zijn om de overwinning te vieren.

Klik bij ‘2’ op de knop om een geluid toe te voegen. Je hebt daar vier opties, de eerste (onderste) is zoeken in de bibliotheek, en die is uitgebreid:

Er is bij de lessen van Scratch zelf wel een voorbeeld hoe je een naam zou kunnen uitbeelden met grote letters en dat is niet zo moeilijk als je dat met vaste letters doet, maar nog niet heel snel geregeld met sprites en variabelen. Los daarvan zijn er wel lettersprites beschikbaar, maar nog geen cijfersprites.

Als je dat dus wilt, dan zal je die zelf moeten tekenen. Maar dan kun je het zo mooi maken als je wilt. Je zou ze ook kunnen opzoeken op internet en dan uploaden naar je spriteset. De aanpassingen, zoals ik heb voorbereid:



Figuur 5 - Geluiden 2 - bibliotheek

Je ziet hierboven een deel van de bibliotheek onder de categorie ‘lussen’. Je kunt ook een andere categorie selecteren of ze gewoon allemaal bekijken. Bij elk geluid kun je tijdens het zoeken met je muis over het paarse icoontje gaan en dan hoor je het geluid als voorbeeld. Als het je bevalt, kun je dit geluid aan de sprite toevoegen. Zoals je in figuur 4 kunt zien heb ik Bossa Nova toegevoegd, omdat mij dit wel een leuk deuntje leek om de overwinning te vieren. Maar uiteraard mag je zelf weten wat je wilt spelen.

Een complexere manier om het op te lossen is het beschikbaar maken van een extra sprite die al met meer geluiden komt. Je kunt deze dan activeren op het moment dat je over de finish gaat. Weten jullie het nog: bij gebeurtenissen kon je signalen versturen. En zoals je kunt reageren op de groene vlag (ook bij gebeurtenissen), zo kun je ook reageren op het ontvangen van een signaal. En zo kun dus met een andere sprite een muziekje gaan spelen, op basis van zo’n signaal van de ene sprite aan de andere.

Dat klinkt misschien omslachtig, maar als je op een gegeven moment een mooi spel gemaakt hebt en je wilt dat ook op een mooie manier laten eindigen, dan kun je dus bij wijze van spreken een band laten optreden en die gezamenlijk muziek laten maken. Zoals ik al zei: je kunt het zo mooi maken als je wilt.

Het valt ook nog niet mee om de score groot in beeld te krijgen. Wat ik heb gedaan, ik heb de sprite naar het midden van het scherm gestuurd, al is hij nog steeds onzichtbaar. Daar heb ik ‘m dan de score laten zeggen. Ik ben er nog niet achter, hoe ik de fonts van de gesproken tekst kan aanpassen, misschien dat iemand dat eens wil uitzoeken.

Figuur 6 - Aanpassing I

Figuur 7 - Vier overwinning

Aan jullie de uitdaging om e.e.a. te verfraaien. Ik zie of hoor graag of dit gelukt is. Veel plezier weer met Scratch.