Scratch (16)

René Suiker

Het feuilleton krijgt weer een vervolg. Niet omdat er zoveel nieuwe ontwikkelingen zijn, maar gewoon omdat ik het niet kan laten. Ik heb zelfs het idee opgevat om mijn beslommeringen rondom Scratch ook in boekvorm uit te gaan geven, maar zo ver is het nog niet. Misschien dat er dan een keer een boekreview in de SoftwareBus komt over Scratch. Je weet het niet.
Ik ben trouwens wel benieuwd of deze verhalen nog gelezen worden, want het wordt wel steeds lastiger om nieuwe onderwerpen te verzinnen. Spelen met Scratch is nog steeds wel leuk, maar de ideeën voor verhalen drogen een beetje op. Dus ik sta open voor nieuwe suggesties, onderwerpen die we kunnen bespreken.

Een toepassing van Scratch waar ik laatst op kwam was iets heel anders. In mijn vrije tijd geef ik ook nog wel eens bijles aan scholieren. Dat is een initiatief van mijn werkgever, om scholieren die een steuntje in de rug nodig hebben, maar dat niet kunnen betalen, van hulp te voorzien. En bij het uitleggen van assenkruizen, X- en Y-coördinaten en positieve en negatieve getallen, kon ik dat allemaal heel leuk visualiseren met Scratch. En het leuke is, dat ik alles wat daarvoor nodig heb al in deze reeks van artikelen heb beschreven. Dus mijn lezers kunnen hun (klein)kinderen nu niet alleen helpen programmeren, maar ook eventueel begeleiden bij hun wiskunde en natuurkunde. En dat is toch een mooi effect van deze reeks

Huiswerk

Maar eerst blikken we traditiegetrouw terug op het huiswerk van aflevering 15.

Opgave 15.1:
a. Waarom is dit het niet helemaal?
b. Wat kan je daar aan doen?

Als je het artikel niet bij de hand hebt is dit natuurlijk aardig cryptisch, maar het ging erom dat je elk figuur in een eigen kleur tekende.
De betere manier om dit effect te bereiken is om in de functie ‘maak veelhoek’ de kleur aan te passen, maar uiteraard niet binnen de lus.

Figuur 1 - Huiswerk opgave 15.1

Het was het net niet helemaal toen het nog in de hoofdlus werd gebruikt, omdat de kleuren te vaak wisselden. De beste manier is het om het te doen in de functie die de veelhoek tekent.

Opgave 15.2:
a. Probeer de hiervoor genoemde opgave 14.3.b nu zelf, met de gegeven aanwijzingen
b. Welke handige tips kan je delen, om te slagen?

Daar kan ik vrij kort over zijn: het komt erop neer de afbeeldingen goed uit te lijnen. Het kan handig zijn om een assenkruis te tekenen (die kan in de definitieve versie wel weer verdwijnen) en dan goed alle startpunten te berekenen. We weten dat het midden van het scherm positie (0, 0) is en we weten ook hoeveel stappen we zetten, met welke lengte en welke hoek.
Alleen, zoals bij zovelen zal de geometrische kennis wel een beetje weggezakt zijn, dus een handige tip is om het assenkruis wel te tekenen, maar dan vervolgens in het scherm gewoon posities uit te lezen. Besef dus wel dat mensen die professioneel met computerspellen bezig zijn, best wel hun wiskunde bij moeten houden. Want als de bewegingen en verhoudingen niet kloppen, dan oogt een spel meteen beduidend minder realistisch.

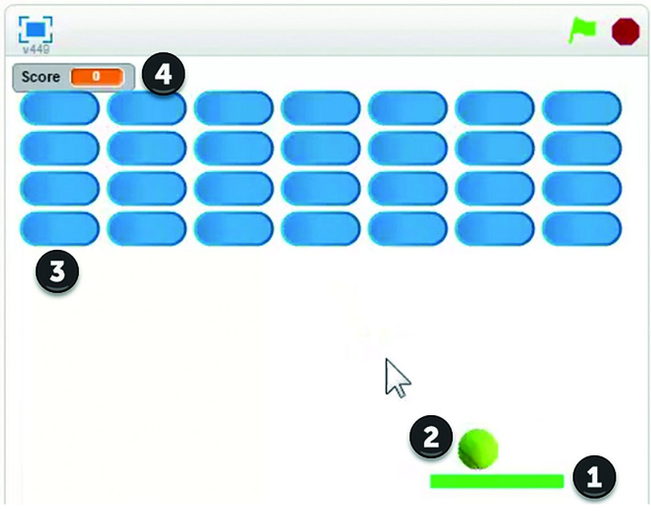
Opgave 15.2:
a. Hoe kan je voorkomen, dat de bal door de wand gaat?
b. Zit het probleem inderdaad in de hoeken? Zo ja, wat kan je daar aan doen? Zo nee, wat is dan het grote probleem.

Tja, daar is in Scratch dus geen eenvoudige oplossing voor. Dat ging in Unity toch wat eenvoudiger, hoewel we daarvoor ook wel de documentatie in moesten. Ik had vroeger een natuurkundeleraar, die af en toe met onzinnige vragen kwam als: hoe hard moesten twee treinen tegen elkaar rijden, zodat ze gewoon doorgingen, omdat het materiaal al weg was voordat het door had dat er een botsing was. En als aanvullende vraag, hoe groot is dan de kans dat ondertussen passagiers in de andere trein terecht komen. Daar is natuurlijk geen eenvoudig antwoord op en zeker niet op het niveau van de middelbare school, maar het zette mensen wel aan het denken en dat was zijn bedoeling. Waarom zeg ik dit? Nu, ik denk dat het probleem waar we mee geconfronteerd worden hier wel mee te maken heeft. Als we tegen Scratch zeggen, dat we bij de wand om willen keren, dan kijkt hij (vermoedelijk, ik heb het niet gebouwd) aan het eind van een beweging of hij bij de rand is, en zo ja, dan keert hij dus om. Zo nee, dan gaat hij verder. En daar zit de crux, hij kijkt bij het einde van de beweging. En als hij dan al de rand voorbij is, dan mist hij de rand.

Wat je eraan kunt doen is om de hele beweging niet in één commando te zetten, zoals we met onze eenvoudige code hebben gedaan, want dan treedt dit ongewenste effect op. Als je sneller wilt bewegen, moet je dat nog steeds in stapjes doen die zo klein zijn, dat je niet in één stap door de hele wand bent.
Dat kan ik hier allemaal gaan uitwerken, maar dan haken de meeste lezers wel af, vermoed ik, dus misschien moeten we maar vaststellen dat bij alles wat Scratch kan, het bouwen van een deeltjesversneller misschien iets te hoog gegrepen is.

Monty Python

Ofwel, ‘… and now something completely different’. Iets anders dus, nieuwe stof, nieuwe problemen, nieuwe oplossingen. We gaan vandaag aan de slag met een ‘echt’ spel, dat de meeste lezers misschien nog wel kennen. Ik meen dat het Bricks heette, een spel waarmee je een balletje tegen een muur aan liet stuiteren en telkens brak dan een steen. Uiteindelijk kreeg je de bal zelfs achter de muur en dan stuiterde hij op en neer zonder dat je wat moest doen. Als de bal dan weer naar beneden kwam moest je hem opvangen met een batje en dan stuiterde hij weer omhoog. Als je de bal miste was je beurt voorbij. We beginnen rechttoe rechtaan:

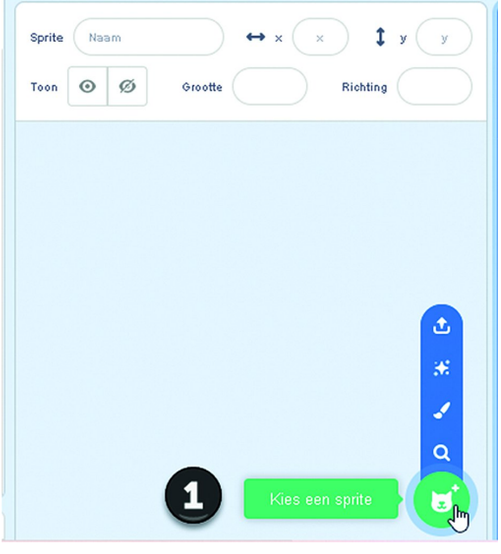


Figuur 2 - Bricks in alle eenvoud

We hebben bij (1)dus een batje, dat we bewegen om de bal op te vangen als die naar beneden komt. We hebben dus een bal (2), die naar beneden valt en dan weer omhoog stuitert als hij op het batje komt, maar als hij naast het batje komt valt hij door en is het spel uit (game over). Bij (3) hebben we de steentjes die weggespeeld moeten worden en bij (4) zie je een scoreboard.

Al met al niet zo'n heel ingewikkelde lay-out, zelfs een beetje saai, maar we gaan dat nog wel pimpen. Maar eerst maar eens zien dat we het spel werkend krijgen. Ik laat me trouwens hier inspireren door Al Sweigart, die ik in het vorige artikel ook al noemde. Wel doe ik een paar dingen op mijn manier, maar wel alle respect voor zijn werk. Voor hen die het Engels goed beheersen, kijk gerust eens op zijn website https://inventwithscratch.com/ aangezien hij daar in hoofdstuk 5 dit spel volledig beschrijft. Gratis online te lezen.

We beginnen met een nieuw project, door middel van de menukeuze 'Maak' (als je Scratch in het Nederlands gebruikt, maar ik ga er verder vanuit, dat dit het geval is). Dan ga je naar de sprite area (rechts onder) en daar gooien we de kat weg (gewoon op het prullenbakje klikken verschijnt zodra je over de kat gaat met je muiscursor). En dan maken we een nieuwe sprite aan, en selecteren 'Kies een sprite' zoals hieronder aangegeven:



Figuur 3 - Kies een sprite

Op deze manier voegen we het batje toe (kies voor 'paddle' in de lijst), het balletje (kies 'tennis ball') en de stenen (kies 'button 2'). In het overzicht staan de beschikbare sprites gewoon op alfabet.

Het batje

Er komt straks misschien nog wel meer bij, maar laten we hier maar eens mee beginnen. We beginnen met een stukje nieuwe theorie en we beginnen met de code voor het batje. Selecteer de sprite door hem aan te klikken en dan gaan we de code maken om het batje te besturen.
We laten het spel weer beginnen door op de groene vlag te klikken. We willen een besturing met de muis, maar het batje moet wel op dezelfde Y-coördinaat blijven. We gebruiken het volgende stukje code om het batje te bewegen:

Figuur 4 - Bewegen van het batje

Met (1) regelen we dat het allemaal begint met de groene vlag. En met (2) initialiseren we twee variabelen. Bij (3) zetten we het batje in het midden onderaan. We laten nog wel wat ruimte onder het batje, maar je kunt ook iets meer of iets minder ruimte kiezen. Bij (4) krijgen we wat nieuwe theorie, dan krijgen we een nieuw stukje theorie, namelijk het bepalen van de draaistijl van een sprite. Hier hebben we de keuze uit drie opties. Zoals hierboven aangegeven is de standaard-optie. Een sprite draait dan wel 180 graden langs de verticale as wanneer hij de andere kant op gaat, maar verder draait hij niet. We willen dit batje niet laten draaien, maar of hij nu helemaal niet draait of op deze wijze, maakt in dit geval niets uit. De alternatieven zijn: 'niet draaien', dan ziet de sprite er altijd hetzelfde uit, of 'helemaal rond' en dan richt de sprite zich in de richting die ingesteld is met het commando 'richt naar…'.

Misschien nu nog niet heel relevant, maar als we straks wat extra elementen aan het spel gaan toevoegen kan het wel degelijk relevant worden. Maar ik geef toe, het is pas echt relevant als je de manier van bewegen van Al overneemt, dat doen we nog niet.

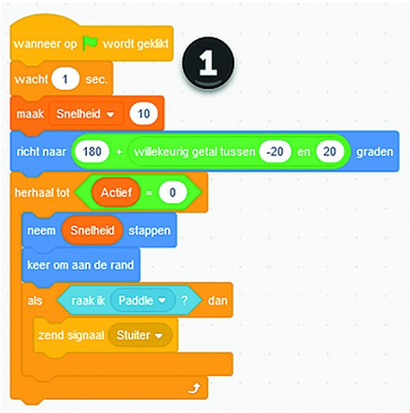
Als je bovenstaande code hebt ingevoerd, dan kun je dit deel van de code testen door op de groene vlag te klikken. Je ziet dat het batje naar zijn startpositie gaat en daarna horizontaal beweegt naar de X-positie van de muis, maar verticaal op zijn plaats blijft. De twee variabelen hebben een initiële waarde gekregen, maar in de loop van het spel kunnen we deze aanpassen om het iets moeilijker te maken. Overigens zijn er nog meer manieren waarop je het spel moeilijker kunt maken.

De variabele 'Actief' heb ik in het leven geroepen om de lus te beëindigen als het spel is afgelopen. Op deze manier hoef ik niet een 'Stop alles' te gebruiken om de beweging te stoppen. Waarom dit nuttig is, daar komen we straks op. Als je trouwens bij Al Sweigert kijkt, dan zul je zien dat hij de vertraging in de beweging op een andere manier oplost. Zoals zo vaak bij programmeren zijn er meer wegen die naar Rome leiden. En als je mijn manier niet fijn vindt spelen, dan kun je zijn manier overnemen.

Het balletje

De volgende stap is relatief eenvoudig. Selecteer in het sprite-blok de tennisbal, want we gaan daar de code voor maken. We hebben een balletje, dat valt in principe naar beneden op basis van de zwaartekracht. Het balletje kan stuiteren, dus als het op het batje valt stuitert het weer omhoog, min of meer volgens 'hoek van inval' is 'hoek van uitval' zoals we vroeger bij natuurkunde hebben geleerd. Al-

leen, om het niet al te makkelijk te maken, moeten we hier een klein beetje variatie introduceren.
We laten het balletje zijn werk doen volgens de volgende code:



Figuur 5 - Balletje

Bij (1) zie je dat het balletje ook reageert op de groene vlag, maar het wacht nog even voor het in beweging komt, zodat de speler even tijd heeft om te reageren. In de praktijk is 1 seconde wel goed, maar je kunt dit natuurlijk ook veranderen. We beginnen met een snelheid van 10, in een variabele, zodat we dit in de loop van het spel kunnen aanpassen.

Bij (2) wordt het balletje in de startpositie gebracht en schuin naar beneden gericht. Hier heb ik een random-factor ingebouwd, zodat je niet altijd precies hetzelfde spelverloop hebt.

Bij (3) hebben we de actieve beweging in een ‘eeuwige’ lus. In dit geval niet eeuwig, we hebben het afhankelijk gemaakt van de variabele ‘Actief’, zoals we ook deden bij het batje. Binnen de lus beweegt de bal volgens de aangegeven richting met de opgegeven snelheid. Dit zorgt voor beweging tegen vloer, plafond en randen. Voor de bodem en het batje moeten we straks nog maatregelen nemen.

Bij (4) nemen we al de eerste maatregel, namelijk als het batje geraakt wordt. Dan stuurt de code een signaal, dat er voor zorgt, dat de bal omkeert. Die code ziet er als volgt uit:

Figuur 6 - Omdraaien

Bij (1) reageren we op het signaal, zoals we soms ook op de vlag reageren. We hadden dit blokje code natuurlijk ook gewoon in het vorige blok kunnen inbouwen, maar we gaan dit blokje nog een keer gebruiken straks en dat kan dus op deze manier. Want het signaal ‘Stuiter’ kan ook op andere plaatsen gegeven worden. En ook door meerdere sprites ontvangen worden. Dat is één van de krachtige aspecten van Scratch.

Bij (2) keren we om, maar daar voeg ik nog een random-factor aan toe. Blokje (3) is speciaal. Omdat we in de code voor de bal kijken of we het batje raken en dan de code versturen, willen we zeker weten dat we niet heen en weer willen blijven stuiteren, omdat we het batje blijven raken. Dit is een trucje dat ik van Al Sweigart heb overgenomen.

De stenen

Om de stenen te plaatsen gebruiken we de sprite ‘button 2’, die we gelijk even hernoemen in het sprite blok. Je kunt er bijvoorbeeld ‘steen’ van maken, maar omdat we het programma ‘Bricks’ hebben gedoopt, maar ik er ‘brick’ van.

Doet er verder niet toe, want je gebruikt de naam nooit direct, maar altijd via referenties die je selecteert.
Om de blokjes te plaatsen hebben we de volgende code onder deze sprite aangebracht:

Figuur 7 - De stenen

Ook hier (1) reageren we in eerste instantie op de groene vlag. Je kunt echt een ongelimiteerd aantal blokken op de groene vlag laten reageren. We laten de sprite verschijnen, voor het geval dat nog niet het geval was. Vervolgens zetten we een aantal variabelen die we nodig hebben op een goede startwaarde. Ik houd ervan om hier variabelen voor te gebruiken, omdat je niet weet of je gedurende het programmaverloop hier nog een keer wijzigingen in wilt aanbrengen.

Bij (3) maken we onze stenen iets kleiner en dan zetten we de originele sprite op de positie van de eerste steen.

Bij (4) gaan we twee herhaallussen in. De buitenste lus zorgt ervoor dat we een aantal rijen van stenen krijgen. De binnenste lus zorgt er binnen de rij voor dat er een aantal kolommen komen. Binnen de huidige variabelen worden er straks 32 stenen geplaatst.

Bij (5) maken we de stenen door een kloon van de originele sprite te maken. Deze kloon komt op precies dezelfde plaats als de originele (maar de originele loopt straks weg) en heeft verder precies dezelfde eigenschappen en code. De kloon doet nog niets, want al heeft hij dezelfde code, de huidige code wordt al uitgevoerd en er is geen trigger om de code ook voor de sprite uit te voeren, dat komt straks.

Bij (6) zorgen we ervoor dat de originele kloon verdwijnt, zodat we niet nog een rij van één steen hebben.
De grote truc in dit spel is dat we niet de bal laten kijken of die een steen raakt, maar we laten de steen kijken of die door de bal geraakt wordt. Op die manier kun je met je code bij de steen blijven en daar de juiste acties voor in gang zetten. De juiste acties zijn in dit geval: de bal te laten stuiteren, de steen te laten verdwijnen en de score met één te verhogen. En als de laatste steen verwijderd is, dan kunnen we ook de waarde van de variabele ‘actief’ veranderen, waardoor de bewegingen stoppen. En daar kun je ook weer een aantal acties aan koppelen.

We kijken even naar de code voor elke individuele steen:

Figuur 8 - De stenen 2 - de acties

Bij (1) zie je dat dit stukje code geactiveerd wordt als een steen eenmaal als kloon is aangemaakt.

Bij (2) zitten we weer in de bijna eeuwige lus. Binnen deze lus kijken we (3) continu of de steen door de bal geraakt wordt. Is dit het geval, dan voeren we de acties uit (4) zoals ik zojuist al besprak: we verhogen de score, we laten de bal weer stuiteren (net zoals net door middel van het signaal 'Stuiter') en we laten de steen verdwijnen. Zoals gezegd, we beginnen met een eenvoudige versie van dit spel.

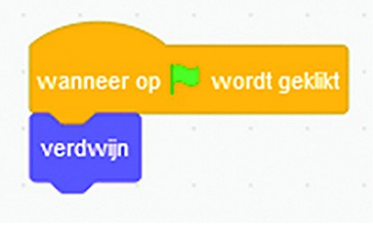
Bij (5) controleren we of alle stenen weg zijn. Door de manier van werken kunnen we vaststellen dat de score alleen verhoogd wordt als we een steen verwijderen. We weten hoeveel stenen er zijn als gevolg van de variabelen. En als alle stenen weg zijn, dan sturen we het signaal 'Einde'. Daar kunnen we dan weer iets leuks mee doen, in de code van één van de bestaande sprites, of een nieuw te introduceren sprite. En ik kies voor dat laatste.

Het einde op deze manier beschouwen we als winst. We kunnen ook verliezen, namelijk als de bal de bodem raakt, dus onder het batje uitkomt.
We maken hiervoor twee sprites. Dit doen we door sprites toe te voegen, via de kat met de plus rechts onderin, zoals we in figuur (3) al zagen, maar nu klikken we op het kwastje:

Figuur 9 - Maak een sprite

We komen dan in de tekenfunctie terecht. Hier maken we een sprite die we met tekst vullen. In dit geval maken we er twee aan, één bijvoorbeeld met 'gefeliciteerd' en één met 'jammer, volgende keer beter' of iets van soortgelijke strekking. Maak de teksten goed groot, want het moet wel zichtbaar zijn.

In mijn spel heb ik de ene sprite 'Game Over' genoemd, de ander 'You win', maar zoals eerder gezegd, die naam maakt niet zo veel uit.
De sprites staan vervolgens in het scherm, waar je ze kunt verplaatsen op een plek die je leuk lijkt. Zodra op de groene vlag geklikt wordt wil je dat de sprite verdwijnt. Dat kunnen we voor beide sprites als volgt regelen:



Figuur 10 - Verdwijn

Vervolgens is het zaak dat we de code nog aanpassen, zodat de juiste tekst in de juiste situatie optreedt. Voorlopig is dat dus simpel. De 'You win' sprite komt te voorschijn als deze sprite het bericht 'Einde' krijgt, als volgt dus:

Figuur 11 - Gewonnen

Dus, bij (1) ontvangt deze tekst sprite het signaal 'Einde', dan wordt onderliggende code uitgevoerd.

Bij (2) zetten we 'actief' op 0, waarmee een aantal zaken stoppen.

Bij (3) komt de tekst van de overwinning in beeld.

En bij (4) stoppen we alles. Bij (2) deden we dat ook al, alleen iets minder definitief en misschien niet volledig. Op dit moment is het dus een beetje dubbelop, maar met regel 2 bereiden we ons voor op nieuwe levels, een uitbreiding dus. Met regel (4) is dat niet meer aan de orde.

Om het spel als verliezer te beëindigen moeten we iets anders doen. We hebben wel een vergelijkbaar stukje code nodig, maar met een andere sprite, dus een andere tekst. En dus met een ander signaal, namelijk 'Verloren'. Alleen, dat hebben we nog niet verzonden. En daar gaat het huiswerk van deze keer dus over.

Opgave 16.1:
a. Onder welke sprite moet je een code-aanpassing doen?
b. Hoe ziet deze code-aanpassing eruit?
c. Als je dat gedaan hebt, werkt het spel dan correct?

Opgave 16.2:
a. Is het spel uit te spelen?
b. Als het te moeilijk is, hoe kun je het makkelijker maken?
c. Als het te makkelijk is, hoe kun je het moeilijker maken?

Opgave 16.3:
a. Het spel werkt, maar wat mis je nog?
b. Het spel stopt toch niet nadat de stenen weg zijn, wat ging er mis en hoe kun je dat herstellen?

Voor een volgende keer kunnen we het spel dus nog (flink) uitbreiden, maar in één artikel hebben we toch een klassiek computerspel gebouwd dat zich laat besturen en zich verder correct gedraagt. En nagenoeg alle code was al bekend. Toch wel leuk dat we dit in Scratch kunnen, met uiteindelijk slechts een paar regels code.

In feite heb je hier ook de basis in handen om één van de eerste computerspellen te bouwen, het pingpongspel dat je met twee personen kunt spelen. Je kunt dit opbouwen met twee batjes, die afhankelijk van de score steeds kleiner kunnen worden. De ene speler kan sturen met het toetsenbord, de ander met de muis. Probeer het eens uit.