Scratch (20)

René Suiker

Nummer 20

Als ik niet al zo dik was, zou ik vinden dat ik wel een gebakje van de club heb verdiend, met zo veel artikelen over Scratch. Scratch is een heel dankbaar onderwerp gebleken, waar ik nog niet over uitgepraat ben. En ik ga dus ook nog even verder. Niet alleen in de SoftwareBus, maar soms dus ook tijdens bijeenkomsten.

Terugblik

Elders in dit nummer blikken we terug op de HCC!kennisdag van 17 juni jl. Ik gaf daar een korte lezing over Scratch, terwijl ik me voorbereid had op een workshop van 90 minuten. Het vergde wat creativiteit om dat even gauw om te gooien, maar dankzij een klein maar enthousiast publiek lukte dat toch nog.

Tijdens de komende CompU*fair* (zie elders in dit nummer, 16 september aanstaande) ga ik proberen wat uitgebreider stil te staan bij Scratch en wil ik, al dan niet in de vorm van een workshop, het spel Bricks uitleggen. En dat dan gericht op beginners, dus een ambitieus doel. Ik heb dan ook nog geen idee of ik voldoende tijd krijg en hoe ver ik in de beschikbare tijd kan komen. Zoals de vaste lezers intussen gezien hebben kun je al vrij snel een werkend spel opleveren, dat gaan we vast en zeker halen. Maar er is natuurlijk de mogelijkheid om het spel te verfraaien en daar kun je lang mee doorgaan, maar dat hoeft niet.

Vorige keer ging ik wat dieper in op het objectgeoriënteerd programmeren en ik ging in op het huiswerk vanuit die gedachtegang. Ik hoop dat het allemaal een beetje te volgen was. En anders moet je het je (klein)kinderen maar vragen, die zijn intussen helemaal in deze stijl groot geworden en snappen misschien wel helemaal niets van het sequentieel programmeren.

De vorige keer behandelden we maar drie van de vier opgaven, waarvoor ik trouwens geen inzendingen heb gezien. De laatste opgave uit aflevering 18 hebben we maar herbenoemd naar opgave 19.1, zodat ik in dit artikel niet vele artikelen hoef terug te lezen, maar ik kan alles in het voorgaande artikel lezen. Daar ga ik maar mee aan de slag en dan zien of we ruimte over hebben voor wat nieuwe theorie.

Overigens, ik had niet alle uitwerkingen letterlijk gegeven, omdat ik het wat interactiever wil maken, maar ik heb dus geen oplossingen gezien. Wel jammer, want het is toch leuk zelf zo'n spel voor je (klein)kinderen in elkaar te zetten en te vertellen dat je het zelf gemaakt hebt. Hopelijk hebben jullie dat toch gedaan, alleen mij niets verteld.

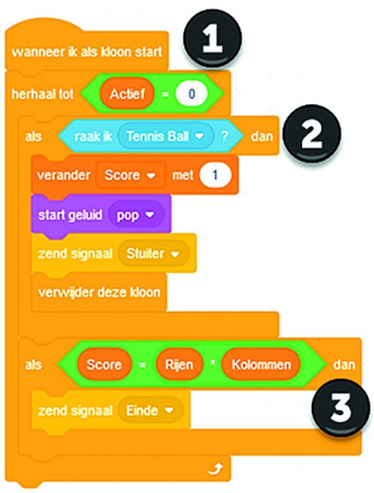
Het huiswerk, de uitwerking

Voor de wat gevorderde programmeurs had ik wat wijzigingen in gedachten, verwoord in opgave 19.1. De basis vind je hier: **https://scratch.mit.edu/projects/740481798/**
Maak gerust een eigen 'Remix'.

Opgave 19.1 (voorheen 18.4)
1) *Maak een paar blokjes rood. Als deze geraakt worden kleuren ze blauw. Als ze nog eens geraakt worden, verdwijnen ze.*

Uitwerking:
Voor het gemak lezen we in de opgave 'Oranje' voor 'Rood' want dan kunnen we min of meer gebruik maken van de 'Uiterlijken' die met de blokjes meegeleverd worden. Daar moet dan waarschijnlijk nog wel handmatig aan gesleuteld worden, maar dat laat ik aan jullie creativiteit over, hierover straks meer.

De aanpassing moet plaatsvinden in het object 'Brick' en wel in het blok dat geactiveerd wordt door het starten als een kloon:



Figuur 1 - Originele code voor Bricks

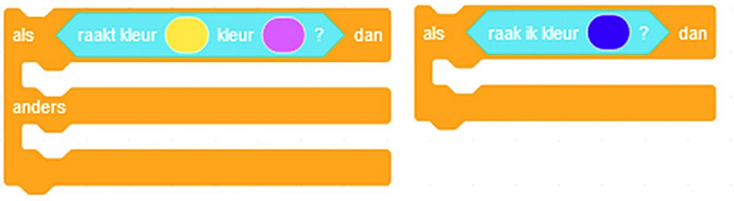
Bij (1) staat dus 'wanneer ik als kloon start'. Daarmee wordt dit stuk code geïnitialiseerd. En direct daarna moet er nog iets gebeuren, iets wat nog niet in dit blok staat. Alleen, ik vraag me af of het mogelijk is wat ik in gedachten had. Hier moeten we voor een aantal blokjes de kleur veranderen en als die kleur anders is, dan is hij nog niet gelijk weg als die geraakt wordt, maar verandert de kleur weer terug.

Ik denk dat je dan een variabele nodig hebt, die bijhoudt welke status het blokje heeft. Bij de bijzondere status hoort een alternatieve kleur. Als hij geraakt wordt met de bijzondere status, dan raakt hij zijn bijzondere status kwijt, inclusief de bijzondere kleur. Wordt hij dan weer geraakt, verdwijnt hij, net als een gewoon blokje.

Maar bij het aanmaken van de variabele kun je aangeven of het een globale variabele moet zijn of een sprite-specifieke. Ik zag niet de optie voor een kloon-specifieke. Ik moet dus testen of een sprite-specifieke ook een kloon-specifieke is, of dat alle klonen dezelfde variabele delen. En dat wil ik wel doen, maar daar heb ik nu geen tijd voor. Misschien een vrijwilliger onder de lezers? Eeuwige roem valt u ten deel als ik in de volgende uitgave uw naam noem als degene die dit probleem heeft uitgezocht en opgelost.

Alternatieve aanpak

Een alternatieve aanpak is om het met meerdere uiterlijken te doen. En afhankelijk van welke kleur de tennisbal raakt heeft dat gevolg voor de verdere afwikkeling. Want we kunnen in Scratch wel kijken of een object een bepaalde kleur raakt:

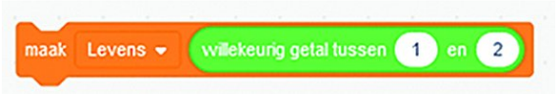


Figuur 2 - Waarnemen van kleur

Je kunt hier kijken of 'ik', dat is het object zelf, een bepaalde kleur raakt. Dit zoek je dus uit vanuit de bal en je moet natuurlijk wel zorgen dat je de kleur van de blokjes kent. Dat kun je doen door ze zelf te tekenen of door de kleur op te nemen via het pipetje. Dat beschreef ik lang geleden al, in het begin van deze serie.
Deze waarneming kun je als conditie gebruiken in een 'Als-Dan-' of 'Als-Dan-Anders'-constructie.

De waarnemingen van kleur vind je in de lichtblauwe categorie 'Waarnemen'. De vergelijkingen vind je in de oranje categorie 'Besturen'.

In beide gevallen moeten we natuurlijk wel zorgen dat sommige blokjes de speciale kleur hebben. Ook daarvoor gebruiken we weer de 'Als-Dan'-constructie. Werken we met de eerst voorgestelde oplossing, dan bepaal je een variabele die je op 1 of 2 initialiseert, op basis van willekeur. Daarvoor hebben we in de groene categorie 'Functies' de functie voor een willekeurig getal:



Figuur 3 - Willekeurig getal

In dit voorbeeld heb ik de variabele 'Levens' genoemd en een waarde 1 of 2 laten krijgen. Als je minder bijzondere blokjes wilt, kun je natuurlijk een ander bereik nemen, bijvoorbeeld tussen 1 en 4 en alleen bij 4 treedt de speciale variant op. De kans dat een bepaalde waarde voorkomt is dan 25% en je kunt natuurlijk ook een andere waarde gebruiken.
Als je dan weer terug wilt naar 2 waarden dan kun je bijvoorbeeld gebruik maken van de volgende constructie:

Figuur 4 - Verschillende kansen

In eerste instantie wordt de variabele levens ingesteld op een waarde 1, 2, 3 of 4. Elk van deze waarden heeft 25% kans om op te treden. Bij één specifieke waarde maken we de waarde 1, dus 25% kans. Bij alle andere waarden wordt hij 2, dus 75% kans. In het programma is het dan logischer om juist de waarde 2 te geven in 25% van de gevallen, maar dat kun je zelf ook verzinnen, hopelijk is het principe helder.

En als je dan met verschillende uiterlijken wilt werken, want dat doen we nu, wordt het totaal iets als:

Figuur 5 - Verschillende uiterlijken

De bijgeleverde uiterlijken voldoen waarschijnlijk niet, omdat deze een kleurverloop in hun uiterlijk hebben, dus dan wordt het lastig om te beoordelen of de juiste kleur geraakt is. Maar dat kun je dus zelf binnen de editor oplossen. En als je heel creatief bent, laat je het kleurverloop in het midden van het blok zoals het is, als je maar aan de randen, waar het 'raken' plaatsvindt, een consistente kleur aanbrengt. Die rand hoeft op zich niet erg dik te zijn.

Dan gaan we nu naar het volgende deel van het huiswerk, nog steeds opgave 19.1 maar nu deel:
2) *Geef de speler dan drie ballen voordat hij af is.*

Ik moet zeggen, de concepten uitleggen zonder het in detail al uit te werken bevalt me wel, want op die manier zijn jullie gedwongen zelf ook iets uit te proberen. En hoewel ik direct bereid ben toe te geven dat ik zelf wel wat lui ben, is het toch vooral ook nuttig als ik jullie iets laat doen, want daar leer je veel meer van. Dus, ik ga hier ook niet alles oplossen, maar handvatten geven om het zelf te bereiken.

Ik zou een variabele 'Ballen' introduceren en die op 3 zetten. Ik zou die ook in het scherm zichtbaar maken, zoals je in een ouderwetse flipperkast ook kunt zien hoeveel ballen je nog ter beschikking hebt. Alleen, je moet zodra je een bal in het spel brengt natuurlijk de teller gelijk verlagen, dus de teller geeft aan hoeveel ballen je nog hebt als je de huidige bal verspeelt.

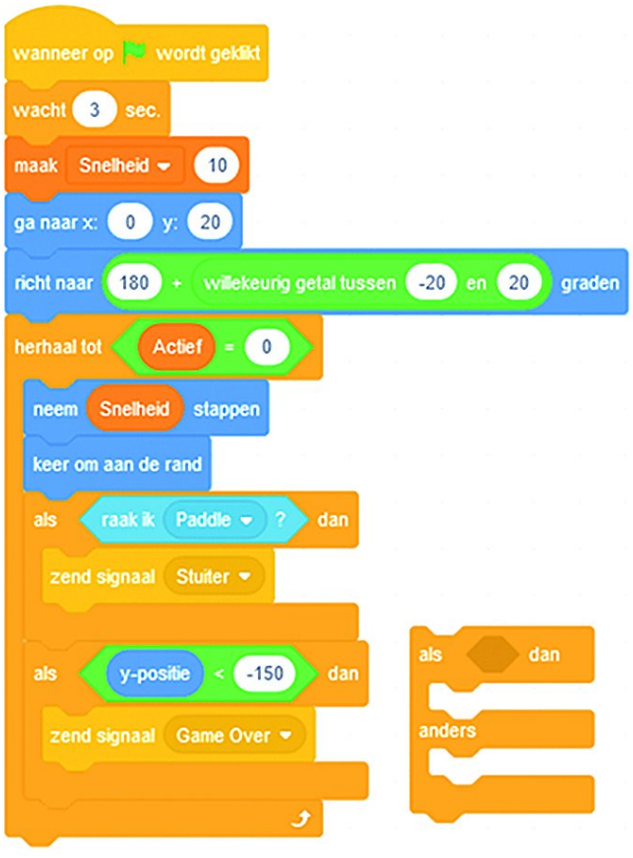
In de loop van de uitbreidingen zou je nog iets kunnen verzinnen waarmee spelers nog een extra bal zouden kunnen verdienen, maar dit terzijde.

Je had het vast al zelf verzonnen, als je nog even terugdenkt: we willen als de bal voorbij het batje komt het spel niet gelijk beëindigen, maar de speler een extra bal geven. Als de speler alle blokjes opgeruimd heeft, dan is het spel ook over, maar daar heeft hij geen extra bal voor nodig. Dus, we moeten de aanpassing maken in de code voor de bal.

We gaan in feite binnen een 'Als-Dan'-constructie nog een nieuwe 'Als-Dan'-constructie opnemen, die ervoor moet zorgen dat bij het kwijtraken van de bal wordt gekeken of er nog ballen te vergeven zijn voordat het spel beëindigd wordt. Dat ziet er dan dus ongeveer zo uit:

(Zie figuur 6 op de volgende pagina)

Okay, ik heb 'm nog niet ingevoegd, maar hier zie je het codeblok voor de bal. Het linkerdeel is de code zoals hij nu is, rechts zie je dus die 'Als-Dan-anders'-constructie die ingevoegd moet worden. En dan moet je eens nadenken over wat er allemaal moet gebeuren. Het blok staat al min of meer op de plaats waar het moet komen in bovenstaand schema. Maar wat moet er nu in?



Figuur 6 - 'Als-Dan-anders' binnen 'Als-Dan'

De conditie y-positie < -150 geldt dus voor de bal. Dat wil zeggen dat hij onder het batje terecht is gekomen, dus de bal is verloren. DAN gaan we kijken of de speler nog ballen over heeft. We bekijken dan of de variabelen 'Ballen' een waarde heeft die groter is dan 0. Als dat het geval is, moeten we een nieuwe bal in het spel brengen EN we moeten de variabele 'Ballen' met één verlagen. Want als we binnen deze conditie terecht komen en de variabele 'Ballen' is wel 0 (kleiner kan niet), dan treedt de 'anders'-code in werking en daarmee zenden we dan het signaal 'Game Over' alsnog.

Dit klinkt allemaal misschien wel redelijk rechttoe-rechtaan, maar hoe brengen we een nieuwe bal in het spel? Je kunt natuurlijk een signaal 'New balls please' versturen, maar wie pikt dat signaal dan op? Je kunt hier natuurlijk een stukje code voor maken die heel precies lijkt op de code die we nu hebben, dus gewoon kopiëren van het blok en er een ander begin voor zetten (een andere startconditie dus). Duidelijk is wel, dat wat er ook gebeurt, de code-aanpassing wel in het object van de bal moet plaatsvinden. Zelf houd ik niet zo van gekopieerde blokken code die naast elkaar staan, maar als je dat wilt vermijden, dan zul je toch eens het hele ontwerp weer ter harte moeten nemen en dan kijken of je de gewenste specificaties op een efficiëntere manier kunt invullen.

Dit lijkt veel extra werk en dat is het misschien ook, maar dat is de consequentie van de werkwijze van beginnen zonder alle specificaties te hebben. Nadeel van deze werkwijze is dus dat je niet altijd optimaal efficiënt programmeert, het voordeel is wel dat je snel een werkend product oplevert. Dit is op zich dus flexibel werken en je geeft je klant telkens een product waarmee de klant ervaring kan opdoen en kan zien wat hij anders wil. Dit is wat ze tegenwoordig 'agile' noemen. Het grote voordeel is dat de klant erg betrokken is én blijft bij de voortgang en dat het product steeds meer aan de wensen van de klant gaat voldoen.

Bij de traditionele ontwikkelmethoden (ook wel waterval genoemd) worden eerst alle specificaties opgehaald en dan wordt er stap voor stap ontwikkeld en als het product af is wordt de klant erbij gehaald. Nadeel is dus, dat er nooit tussentijds bijgestuurd kan worden. En vaak kreeg je dus producten die wel precies volgens de specificaties waren opgeleverd, maar die toch niet helemaal waren wat de klant bedacht had. Het uitschrijven van specificaties is een vak op zich en de traditionele softwareontwikkeling kost ook vaak zoveel tijd, dat de gewenste specificaties in feite tijdens het traject al achterhaald zijn. Tot zover deze kleine zijstap in mijn vakgebied, niet echt de hobby, maar hopelijk toch interessant.

Het totale herontwerp van de bal zou er grofweg zo uit kunnen zien:

- Wanneer er op de vlag wordt geklikt
- Wacht drie seconden
- Vul de variabele 'Ballen' met waarde '2'
- En zend dan signaal 'Schiet de bal af'.

De rest van het huidige codeblok wordt dan min of meer opgenomen in een ander codeblok, dat start met 'wanneer ik signaal 'Schiet de bal af' ontvang. ….'

En dan binnen de zojuist vorm gegeven 'Als-Dan-anders'-constructie, in het 'Dan'-gedeelte, zenden we het signaal 'Schiet de bal af'. Dan roept in feite het blok zichzelf weer aan, maar dat doet verder niet ter zake en gaat ongetwijfeld goed. Let dan goed op, waar/wanneer je de variabele 'Ballen' aanpast en wanneer je de inhoud van de variabele controleert. Als je meer - of juist minder - ballen krijgt om mee te spelen, is er een goede kans dat hier de fout zit.

Als iemand hier problemen mee ondervindt, vertel het me gerust en laat me dan ook de totale code zien. Dat doe je door je project te delen en me dan de link of het nummer te verschaffen. Mail me via gameontwerp@compusers.nl. Misschien kan ik je gelijk helpen, misschien zijn de problemen dusdanig, dat ik er een nieuw artikel voor moet schrijven. Maar ik waardeer elke serieuze poging.

3) *In mijn beleving ging in de Arcade-versie het batje altijd heen en weer, als je rechter pijltoets indrukt gaat hij langzaam rechtsaf tot de rand en blijft daar, idem voor linksaf. Maak deze besturing.*

Hier hoef je niet heel diep over na te denken. We besturen het batje, dat is het object 'Paddle'. Voor de aanpak kun je even terugkijken hoe we de besturing van de kever hebben aangepakt in het project 'Doolhof'. Alleen, daar hadden we zowel horizontale als verticale beweging, dat hoeft hier niet. Daar hadden we verschillende snelheden, dat hoeft hier niet. We hebben hier alleen twee richtingen, links en rechts. Als het batje tegen de muur komt verandert de bal van richting en reageert hij op de pijltoetsen, op de verwachte wijze.

We hadden hiervoor een 'Beweeg'-functie in het leven geroepen die we in een eeuwige lus bleven aanroepen. Die beweegfunctie had een snelheid en een richting. Hier is de richting beperkt tot de x-as en de snelheid is bijvoorbeeld -5 en +5. En bij het doolhof hadden we alleen maar te maken met de rand van het doolhof, en dan begon je opnieuw. Hier wil je dat het batje aan de rand omkeert.
Je kunt het trouwens ook moeilijker maken door het precies andersom te laten reageren, dus met de linker pijltoets stuur je het batje naar rechts en andersom.

4) *Als je de bal raakt in het midden van het batje, dan blijft het spel zoals het is. Als je hem helemaal rechts raakt, stuitert hij iets meer naar rechts, als je hem helemaal links raakt, stuitert hij iets meer naar links. Helemaal links en rechts is zeg maar 25% van de lengte.*

Dit is nog wel een dingetje. Hiervoor zal je moeten weten wat het begin van het batje is en ook de maat van het batje

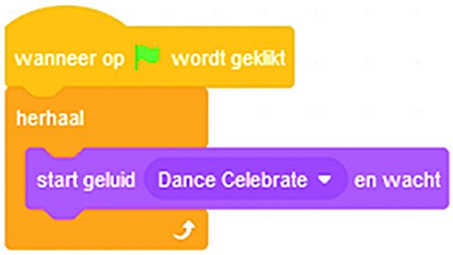
en dit willen we het liefst ook in variabelen vastleggen, want er komen nog aanpassingen aan de maat van het batje aan. En dan moet je dus tijdens de botsing in de gaten hebben wat de afstand tot de tennisbal is en op basis van die afstand t.o.v. de totale lengte kun je dan een effect toevoegen aan het stuiteren op het batje.

Dit vergt waarschijnlijk wat fine tuning om het geheel te krijgen zoals je het wilt, dus met een soort natuurlijk gedrag. Ik ben benieuwd of iemand hier in wil duiken en met een mooi algoritme komt. Het e-mailadres is intussen bekend.

Opgave 19.2

Omdat ik ook nog iets nieuws wilde inbrengen krijgen jullie ook nog een theorievraag. Als je goed naar de blokken hebt gekeken, zien jullie nergens de muziek starten. Dus, los van de benoemde objecten is er blijkbaar nog een object. Wat is dat mysterieuze object, dat tot nog toe niet benoemd is?

Deze vraag was echt heel makkelijk, als je althans een beetje hebt opgelet de afgelopen edities. De muziek wordt afgespeeld door dit stukje code:



Figuur 7 - Achtergrondmuziek

Dit stukje code kan in feite in elk object worden geplaatst, maar ik had 't bij wijze van grap in het speelveld geplaatst, dus bij de achtergronden. Daar stond verder ook geen enkele andere code en dit is op zich een nuttige plek voor 'algemene' code, die dus niet specifiek op één object betrekking heeft.

Die opdracht 'start geluid … en wacht' is wel goed om te gebruiken. Als je alleen maar 'start geluid …' gebruikt in zo'n lus, dan hoor je heel snel achter elkaar het beginstuk, één toon of zo en dan wordt hij alweer opnieuw opgestart, dus dat werkt niet.

Opgave 19.3

Definieer de initialisatie in dit mysterieuze object en bepaal hoeveel signalen nodig zijn om het geheel in goede banen te leiden.

In feite bepaal je dit natuurlijk zelf, want wat de één mooi vindt hoeft de ander nog niet mooi of logisch te vinden. Maar zelf zou ik het mooi vinden, als je eerst de blokjes plaatst, dan de muziek opstart, dan het batje laat zien en bewegen en dan pas de ballen lanceert.

Dat zou betekenen dat de groene vlag alleen maar de blokjes plaatst en als dat gebeurd is, zendt hij een signaal 'Blokjes staan'. Dat kan dan zowel de muziek opstarten als het batje tonen. Het batje ('Paddle') heeft op zich eenvoudige code, in één blok initialiseert die zichzelf en begint te bewegen. Dat blok moet dus niet op de groene vlag reageren, maar op het signaal 'Blokjes staan'.

Alleen, wanneer schieten we die ballen nu af? Wel, de wachttijd die we bij die bal hadden ingesteld bij het afschieten was 3 seconden. Dat is een mooie tijd en als we de code voor die bal niet hadden aangepast, hadden we 'm ook kunnen laten reageren op 'Blokjes staan', maar nu hebben we hierboven gezegd dat we dat anders doen. Je kunt in de achtergrond een kleine taak opnemen die reageert op 'Blokjes staan', dan even wacht en dan het signaal voor het starten van de ballen verstuurt. Op die manier hebben we dus na de groene vlag twee signalen nodig om het spel volledig geïnitialiseerd op te starten. Maar, zoals ik al zei, dat is maar een mening. Er zijn meerdere wegen die naar Rome leiden en deze uitdaging kan ook op meerdere manieren opgepakt worden. Wie een echt briljante aanpak met mij wil delen is bij deze uitgenodigd.

Opgave 19.4

Hoe kan je bereiken, dat de timing echt correct is bij het uitvoeren van de initialisatie, ongeacht het aantal blokjes dat je maakt of de snelheid die je gebruikt?

In feite heb ik dat bij de vorige opgave al uitgelegd. Je voert een taak uit en pas als die klaar is signaleer je naar de volgende taak dat die mag beginnen. En als je echt alle taken na elkaar wilt uitvoeren, dan kun je op die manier synchroniseren.

Besef wel: codeblokken hangen aan een object, maar signalen zijn voor alle objecten te horen en alle objecten kunnen ook alle signalen versturen. Variabelen kunnen globaal zijn of bij een module horen.

CompU*fair*

Zoals ik al aangaf, tijdens de HCC!kennisdag had ik niet zo uitgebreid op Scratch kunnen ingaan, maar tijdens de volgende CompU*fair* wil ik proberen een wat ruimer tijdslot te krijgen om jullie allemaal mee te nemen in dit leuke project. En zoals ik ook al aangaf, je kunt dit samen met je (klein)kind thuis ook maken: door de objectgeoriënteerde aanpak is het allemaal redelijk te overzien. Als je dit allemaal in Basic of C moet programmeren heb je veel meer werk en moeten de (klein)kinderen al wat ouder zijn om dit allemaal te volgen.

Volgende keren

Zoals ik al eerder eens aangaf, het wordt wel een uitdaging steeds iets nieuws te verzinnen. Ik denk dat we op 'Bricks' nog wel wat kunnen uitbouwen, dus dat wordt wat voor de volgende keer, maar ik sta open voor tips voor volgende onderwerpen. Zelf denk ik al aan LIFE, een van de eerste computersimulaties. Dat moet ook in Scratch kunnen.

Huiswerk

Opgave 20.1

De blokjes staan wel erg stil bovenin het scherm. Het zou leuk zijn als ze af en toe een beetje bewegen, maar wel synchroon. Hoe pak je dat aan?

Opgave 20.2

Hoe zou je geluidjes kunnen toevoegen als de bal het batje raakt en als de bal een steentje raakt?

Opgave 20.3

Maak een eigen werkende versie van Bricks en verras me!