Scratch (21)

René Suiker

De CompU*fair*

Zoals ik in het vorige nummer reeds schreef, de belangstelling voor Scratch viel de laatste tijd wat tegen. En dat verbaast me een beetje, als ik heel eerlijk ben. Want het is een heel laagdrempelige manier om de jeugd op een actieve manier bij de computerhobby te betrekken en ook om ze dus de manier van denken bij te brengen waar ze in de toekomst profijt van kunnen hebben. Op een speelse manier leren ze de computer te laten doen wat ze hem opdragen.

Ik was dan ook behoorlijk teleurgesteld door de minimale opkomst, vooral ook, omdat de mensen die wel komen altijd enthousiast worden over de mogelijkheden. En op verschillende plaatsen wordt ook met Scratch gewerkt door de jeugd. Dus, neem volgende keer eens (klein)kinderen mee naar de CompUfair en stuur ze langs om met Scratch kennis te maken.

Programmeren

Zoals de vaste lezers reeds (kunnen) weten onderhoud ik een prettige samenwerking met de IG Programmeren, een zeer actieve interessegroep binnen de HCC. Op vrijdagavond organiseert deze club altijd een sessie over games programmeren. Ik ben daarbij aangesloten, omdat ik meer over Unity wilde weten, maar intussen draag ik ook bij aan deze avond met Scratch. Unity en Scratch zijn totaal verschillend, maar er zijn ook overeenkomsten. En als je daar eens kennis mee wilt maken, de bijeenkomsten zijn online en bij mijn weten voor alle HCC leden toegankelijk. Als je meer wilt weten, kijk dan eens op de website van onze zustergroep: **https://-programmeren.hcc.nl/**

Tijdens één van deze sessies spraken we ook over een project ‘Tech-lokaal Maas & Waal’, waarbinnen met Scratch een Space Invaders-spel wordt gebouwd. Ook dat is natuurlijk een klassieker en ik ben van plan daar ook aandacht aan te besteden in deze artikelenreeks. Ik had eigenlijk eerst iets als ‘Life’ willen doen, een simulatie uit de jaren 70-80 van de vorige eeuw, maar ik kan nergens meer de exacte beschrijving en regels terugvinden. Het ging om een simulatie van levende units, die reproduceerden als ze nog een levend wezen naast zich hadden, maar die stierven als er vier wezens om hen heen zaten, door verstikking. Als iemand de exacte regels weer kan vinden houd ik me aanbevolen. Maar anders ga ik ook eens naar Space Invaders kijken. Ik heb uiteraard toestemming om dit te gebruiken en daarmee mijn lezers ook.

Bricks

Maar eerst nog even terug naar het project dat we aan het ontwikkelen waren. Ik heb dus geen oplossingen gezien van mijn lezers en dat vind ik wel jammer, want het is toch leuk zelf zo’n spel voor en vooral ook mét je (klein)kinderen in elkaar te zetten. Samen aan iets werken, dat geeft toch een binding en uiteindelijk ook nog een resultaat om trots op te zijn.

Als basis voor je creatieve werk kun je gewoon gebruik maken van het vrij beschikbare basisspel, dat je hier kunt vinden: **https://scratch.mit.edu/projects/740481798/** Maak gerust een eigen ‘Remix, sloop eruit wat je niet nodig hebt en bouw je eigen aanpassingen, op basis van je eigen creativiteit of op basis van de aanwijzingen en opgaven. Ik hoop volgende keer toch een aantal remixen van dit programma te zien.

Ik heb zelf ook nog even naar het programma gekeken, maar zoals bekend doe ik het allemaal nog wel een beetje rustig aan. En een deel van de problemen waarvoor ik de lezer om hulp vroeg heb ik intussen zelf opgelost. En dat was allemaal nog veel makkelijker dan ik dacht. Gewoon even kijken wat er allemaal bij de instructieset staat en je komt vanzelf op ideeën.

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

Uitwerking:
Oorspronkelijk stond er ‘rood’ in de opgave, maar omdat het blokje al twee uiterlijken had - een blauw blokje en een oranje blokje - heb ik het simpel gehouden. Met de diverse uiterlijken kun je het jezelf natuurlijk iets moeilijker maken en voor mooiere uiterlijken zorgen, maar ik ben niet zo’n talentvolle artiest en ik beperkt me tot het programmeren. En niet dat ik daar nou veel talent voor heb, maar ik kan er redelijk mee uit de voeten. Ik heb dus niets gesleuteld aan de blokjes, maar ik heb gewoon gebruik gemaakt van de functionaliteit die standaard bij Scratch zit. En dus kunnen jullie dit ook.

Ik had intussen getest, maar kloonspecifieke variabelen bestaan niet. Althans, ik heb ze niet kunnen ontdekken. Dat maakt het lastig, want als je klonen met een globale variabele laat werken, kun je natuurlijk onverwachte dingen tegenkomen. Maar, een aantal systeemparameters zijn wel per kloon beschikbaar en daar heb ik gebruik van gemaakt.

Ik lees de opdracht in een aantal aspecten, die ik stuk voor stuk heb aangepakt. Grofweg:
1. Maak een aantal blokjes van een andere kleur
2. Als een blauw blokje wordt geraakt verdwijnt het
3. Als een oranje blokje wordt geraakt wordt het blauw

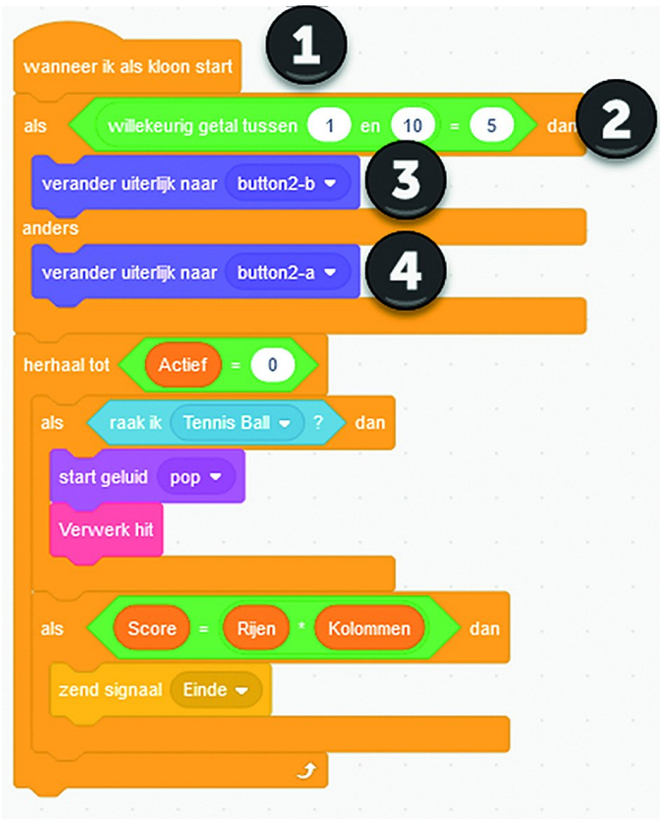
Verder heb ik alleen maar de score opgehoogd als een blokje verdwijnt, anders moet je ook even tellen hoeveel andere kleuren blokjes er zijn en moet de score dus dat aantal nog bij het aantal rijen*kolommen optellen om te kijken of je alles hebt gehad.

Deel 1: aantal blokjes met een andere kleur
Ik dacht dus dat je een variabele nodig had, maar gelukkig houd Scratch zelf genoeg bij, dus maak ik daar gebruik van.

Zie: Figuur 1 - Blokjes in verschillende kleuren op de volgende pagina

Gelukkig hoef ik niet bij te houden welke kleur de blokjes hebben, het blokje weet zelf welk uiterlijk het heeft. Dus ook een gekloond blokje weet dit. Bij (1) zie je dus welk deel van de code ik ga aanpassen: het deel van de code waarmee een blokje als kloon start.

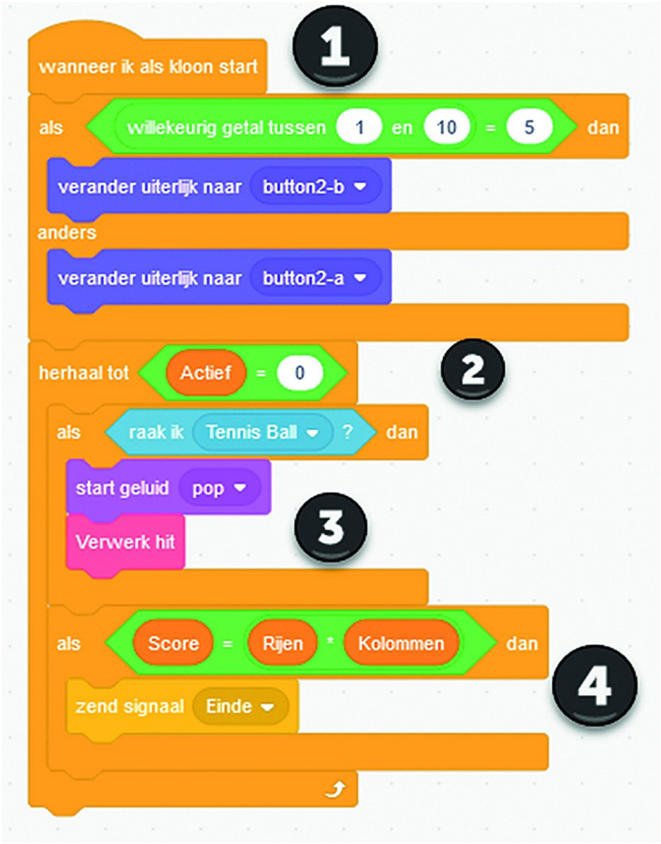
Daar heb ik bij (2) als eerste instructie een ‘Als … dan … anders’-instructie toegevoegd. Ik wil ongeveer één op de tien blokjes, maar wel willekeurig, oranje maken. Dus ik neem per kloon een willekeurig getal tussen 1 en 10. En als dat toevallig 5 is, dan krijgt dit blokje de kleur oranje, dat is gekoppeld aan het uiterlijk bij (3). In de negen andere gevallen wordt het blokje blauw (4).



Figuur 1 - Blokjes in verschillende kleuren

Vervolgens heb ik de code daaronder ook aangepast, maar daar ga ik het zo over hebben.

Deel 2 en 3: als een blokje wordt geraakt, verdwijnt het
Hier heb ik twee aanpassingen voor gedaan. Allereerst in de lus:



Figuur 2 - Aanpassing in the lus

Bij (1) zie je dat ik in hetzelfde deel van het programma blijf als hiervoor, maar ik heb nog wat aanpassingen. In (2) zie je de lus die blijft lopen totdat het programma eindigt. Aan dat signaal gaan we later ook nog iets doen, maar dit terzijde. En misschien niet in dit artikel, dat kan ook zomaar een volgend artikel worden, want ik wil en kan het niet te uitgebreid maken.

Bij (3) zie je de aanpassing. Het blokje kijkt consequent of de tennisbal aangeraakt wordt. Feitelijk gebeurt het andersom, de tennisbal raakt het blokje, maar de tennisbal doet zo veel meer en het blokje hoeft alleen maar op de tennisbal te reageren, dus het is praktischer om hier de actie te leggen. Bovendien heeft de actie ook voornamelijk betrekking op het blokje, dus vandaar dat het waarnemen op het blokje gebeurt. Maar nogmaals, je kunt het ook bij de bal leggen, alleen dan wordt het allemaal wat onoverzichtelijk, denk ik.

Je ziet bij (3) dat er een geluid gespeeld wordt Daar wordt verder niet mee gewacht, maar het is in elk geval een kort geluidje. Dit geluidje wordt bij elke stuiter met een blokje uitgevoerd, dus ongeacht of het nu een blauw of oranje blokje is. Als je dat wilt laten verschillen, dan moet je het geluidje onderdeel laten uitmaken van de ‘subroutine’ ‘Verwerk hit’.

Bij (4) is verder niets veranderd, maar ik laat nog even zien dat als alle steentjes weg zijn, het spel eindigt. We gaan nu eens kijken naar de genoemde subroutine:

Figuur 3 - Verwerk hit

Bij (1) zie je dat we hier de routine ‘Verwerk hit’ definiëren, en als je er naar kijkt zie je dat het toch geen rocket science is. Bij (2) zie je dat we altijd het signaal stuiter versturen, dat zorgt ervoor dat de bal weer weg beweegt, min of meer volgens hoek van inval is hoek van uitval.

Bij (3) kijken we naar het blokje. Uiterlijk nummer 2 is het oranje blokje, uiterlijk met ‘button2-a’ is het blauwe blokje. Ik had die uiterlijken ook kunnen herbenoemen, maar het idee is simpel. Als het een oranje blokje betreft, dan wordt die blauw. Als het al een blauw blokje betreft (4) (het ‘anders’ deel van de vergelijking) dan wordt de score opgehoogd en verdwijnt het blokje. Hiermee hebben we dus bereikt, dat sommige blokjes twee keer geraakt moeten worden voordat ze verdwijnen.

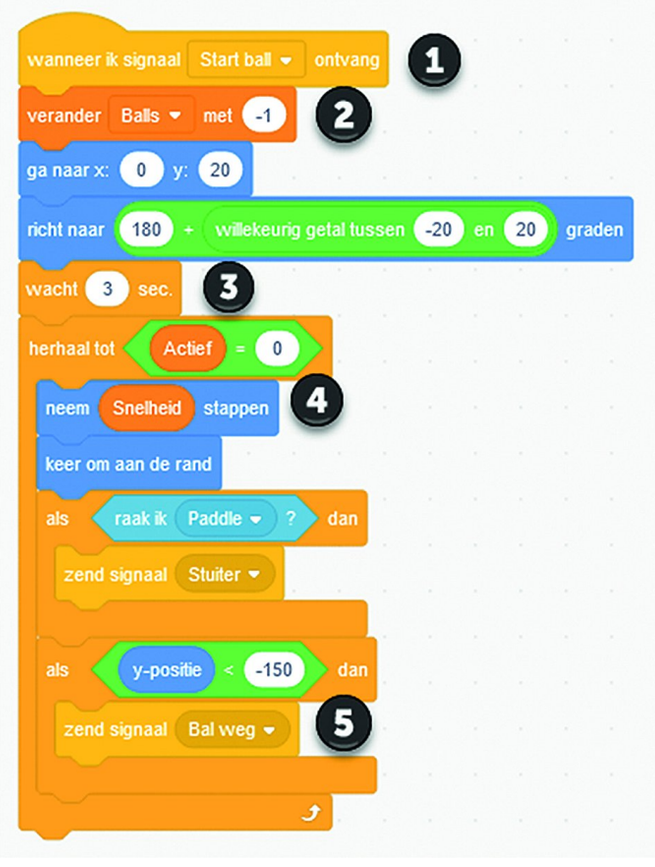
Ik heb dus afgezien van het controleren op kleur: ik heb gewoon gekeken of objecten elkaar raken, ongeacht de kleur.

Opgave 19.1, deel 2:
Geef de speler dan drie ballen voordat hij ‘af’ is.

De vorige keer gaf ik uitleg hoe je dit aan kan pakken, maar ik heb geen correcte inzendingen gekregen, dus ging ik het

weer zelf uitwerken. Hopelijk zien jullie wat ik doe en ben je met me eens dat het allemaal best behapbaar is. Dat is toch het mooie van redeneren vanuit de objecten.

Nog even terugkomen op variabelen. De vorige keer heb ik uitgelegd dat je 'globale' variabelen hebt, die dus voor alle objecten toegankelijk zijn, en je hebt lokale variabelen, die binnen het object toegankelijk zijn. Binnen Scratch heb je dan de varianten wel benoemd; er zijn programmeertalen (zoals C++) waar je nog wat specifieker onderscheid kunt maken, tot en met variabelen die alleen binnen een lus leven. Het voordeel van lokale variabelen is dat je vrij stevige controle hebt over wie er iets met die variabelen doet. En met globale variabelen kan in principe elk object iets doen. Er is in Scratch wel een klein praktisch nadeel aan een lokale variabele. Als je een variabele op je scherm wilt tonen, dan komt het label er altijd bij en dat label bevat ook de naam van het object waar de variabele bij hoort. Dat wil ik nu niet, dus daarom ga ik hier gebruik maken van een globale variabele, maar ik zie er op toe, dat die alleen maar binnen het object 'Tennis Ball' gebruikt wordt.
Overigens, misschien wil ik 'm toch nog ergens anders gebruiken, omdat ik de initialisatie op één plek wil laten gebeuren en de activiteiten dan gecontroleerd na elkaar wil laten plaatsvinden. Dat kan binnen de achtergrond, maar 't kan ook binnen een willekeurig object. Om deze synchronisatie netjes te laten plaatsvinden ga ik dus een aantal aanpassingen in de code maken. Het komt er hier op neer dat ik de initialisaties per object ga veranderen.
Ik vervang de instructie 'Wanneer op ⚑ wordt geklikt' door de instructie 'Wanneer ik signaal … ontvang'. En het groene vlaggetje wordt gebruikt om alle initialisaties op gang te brengen. Of alleen de eerste, en die brengt als hij klaar is de tweede op gang, enz.
Dit dus alvast vooruit, maar we gaan hier al op vooruitlopen bij het oplossen van het voorliggende vraagstuk, want we willen dat de bal naar de startpositie brengen en die dan laten vallen, telkens laten plaatsvinden als een nieuwe bal in het spel wordt gebracht. Ook de eerste keer.



Figuur 4 - De bal

Zie hier de nieuwe hoofdlus van de tennisbal. Bij (1) hebben we dus de reactie op de groene vlag weggehaald. Deels om de controle op de initialisatie te hebben, maar vooral ook om deze code voor elke nieuwe bal te kunnen doorlopen.

Bij (2) zeggen we dat de teller voor de ballen met één wordt verlaagd. Op het scherm zie je dan het resterend aantal ballen, dus los van de bal in het spel. We zouden in feite daaronder nog kunnen controleren of de teller niet onder de nul zakt, want als dat het geval is, dan is er ergens iets misgegaan, maar zou je nu 'Game Over' kunnen activeren. Ik ga ervan uit, dat er nog niets misgaat, dus ik laat het even zoals het is, maar het kan geen kwaad om op meer plaatsen rekening te houden met onverwachte gebeurtenissen. Voor zo'n beperkt programma als dit is dat misschien wat overdreven, maar in het algemeen is het verstandig om je programma robuust te bouwen, dus zoveel mogelijk rekening te houden met mogelijke verstoringen.

De wachttijd (3) stond eerst bovenin dit blok, maar ik wil dat de bal alvast naar de startpositie gaat en daar wacht. Hier kunnen we t.z.t. ook variabelen voor gebruiken, maar laten we ons nu even richten op de huidige problematiek.

Bij (4) hebben we niets veranderd, alleen voorheen werd de snelheid op 10 gedefinieerd in dit blok. Ik houd er rekening mee dat we de snelheid gedurende het spel kunnen aanpassen, dus in dit blok gebruiken we de snelheid zoals die buiten dit blok gedefinieerd is. Op die manier houd ik individuele blokjes overzichtelijk.

Bij (5) stond voorheen dat het signaal 'Game Over' moest worden verzonden. Dat heb ik nu aangepast naar het signaal 'Bal weg'. En wat er dan moet gebeuren regel ik in een separaat stukje code. Ik had al die code ook hier kunnen onderbrengen, maar ik houd van overzichtelijke code en dat bereik ik op deze manier, denk ik. Misschien dat als performance een keer een issue wordt, ik technisch moet optimaliseren, maar vooralsnog ga ik vooral voor leesbaarheid.

Dan nu dus de verwerking van 'Bal weg':

Figuur 5 - Bal weg

Zoals je bij (1) ziet wordt dit stukje code getriggerd door het signaal 'Bal weg' dat hiervoor werd gestuurd. Bij (2) wordt gekeken of er nog ballen zijn. Als dat het geval is, dan wordt het signaal 'Start ball' gegeven en treedt het vorige blok weer in werking. Als dat niet het geval is, wordt 'Actief' op 0 gezet, waarmee in principe allerhande loops ophouden. Daarnaast wordt het signaal 'Game Over' verstuurd, die de tekst 'Game Over' op het scherm zet (zoals voorheen) en ook alle activiteit stopt. Als je nu een heel ingewikkeld stukje

code hebt gemaakt en je bang bent dat je vergeet hoe het precies werkt, kun je met een rechterklik op een blok een menu oproepen dat je de gelegenheid geeft commentaar op je code te geven. Echte programmeurs weten dat het heel belangrijk is om je code van uitgebreid commentaar te voorzien. Tot nu toe, in deze kleine programma's, die op zich heel overzichtelijk zijn, is dat misschien wat overdreven, maar ik weet niet of ik het al eerder vermeld heb, en het is een nuttige toepassing binnen Scratch. Ik meen dat ik het al eens vertelde, maar dat is lang geleden.

Dan nu nog even, tot dusver nog met de vlag gestart, de initialisatie van de bal:

Figuur 6 - Initialisatie van de bal

Bij (1) dus nog de initialisatie met de vlag, maar dat kan dus via een bericht. Bij (2) zie je de initialisaties van de variabelen die betrekking hebben op de bal, hoewel 'Actief' globaal is. 'Balls' en 'Snelheid' misschien ook wel, maar die hebben alleen betrekking op de bal, dus die kun je als lokaal beschouwen.

Bij (4) zie je hoe zo'n stukje commentaar er uitziet. Je kunt met het kruisje rechtsboven je commentaar weggooien. Met een driehoekje links bovenin (waar nu de (1) staat) kun je 'm inklappen en rechts onderin kun je omvang van het blokje aanpassen. Zo hebben we dus nu ook dit deel van de opgave opgelost. Je ziet, het was allemaal niet zo heel moeilijk.

Opgave 19.1 deel 3
In mijn beleving ging in de Arcade-versie het batje altijd heen en weer, maak deze besturing.

Ook dit is allemaal niet heel moeilijk. Ik heb er een iets uitgebreide besturing bij gemaakt, maar het principe is eenvoudig. De paddle start in het midden en staat daar stil. Met de pijl naar rechts laat je 'm naar rechts bewegen, met de pijl naar links beweeg je 'm naar links. Met de snelheid die hij heeft blijft hij heen en weer bewegen. Met een pijltje verhoog je de snelheid en bepaal je de richting. Met de spatiebalk zet je 'm stil.

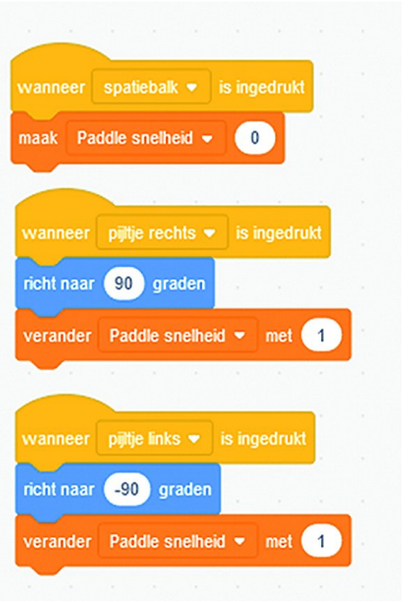
Figuur 7 - Paddle beweging

Bij (1) hebben we ook weer de vlag weggehaald en de beweging op een signaal laten reageren. Zolang de variabele 'Actief' niet gelijk is aan 0 (2) blijft de paddle bewegen, en wel met een meegegeven richting en een elders bepaalde snelheid (3). Hij keert om aan de rand (4). Dit is alles.

De initialisatie is ook recht-toe-recht-aan:

Figuur 8 - Initialisatie van de paddle

Ook (1) hier nog de vlag, eenvoudig om te zetten in een bericht. Bij (2) worden een aantal variabelen geïnitialiseerd. Bij (3) wordt de startpositie bepaald en zorgen we ervoor dat de paddle altijd horizontaal blijft, ongeacht de richting die hij beweegt. En vervolgens starten we de paddle (4).



Figuur 9 - Paddle besturing

De besturing van de paddle is ook weer supersimpel:

Hier in feite drie afzonderlijke blokjes. Het enige wat gebeurt is afhankelijk van de toets, en wordt de snelheid en eventueel de richting aangepast.

Hiermee hebben we opgave 19.1 voor de eerste drie delen uitgewerkt. En heb ik alweer 4 pagina's gevuld. Het moeilijkste stuk van het stuiteren afhankelijk van waar je raakt, is nog niet uitgewerkt en heb ik ook nog niet uitgewerkt.

Ik sta dus open voor suggesties. En ook de Invaders zijn niet aan bod gekomen. Er is dus nog genoeg te doen. Veel plezier met Scratch. En ook het huiswerk van de vorige keer staat nog, dus ik zou zeggen, doe je best.

Huiswerk

Opgave 21.1
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 21.2
Hoe zou je geluidjes kunnen toevoegen als de bal het batje raakt en als de bal een steentje raakt?

Opgave 21.3
Maak een eigen werkende versie van Bricks en verras me!