Scratch (8)

René Suiker

Een gelukkig Nieuwjaar!

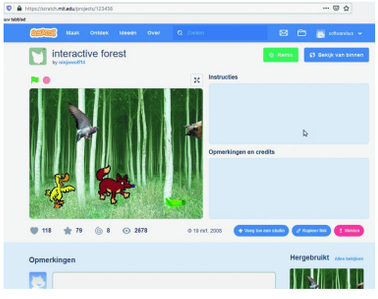
En zo is het ineens 2020. Niet terwijl ik dit schrijf, maar wel wanneer jullie dit lezen. Hoewel, misschien is het dan wel 2021 of nog later, maar in elk geval, in 2020 is onze SoftwareBus 2020-1 bezorgd. En daarvoor is dit artikel geschreven.

En wat zijn we al ver gekomen. Ik geloof niet dat er ooit in de geschiedenis van ons blad een reeks artikelen op hetzelfde thema zolang heeft aangehouden. We hebben ooit wat artikelen gehad over Excel, over WordPress en over GIMP, maar ik denk niet dat we acht artikelen in een reeks ooit eerder haalden. Maar misschien gaat iemand mij wel corrigeren. Ik ben pas bij de SoftwareBus betrokken sinds 2004, denk ik, dus er is nog veel historie van voor die tijd *(Helaas voor René: de serie GigaHits heeft veel langer gelopen: van 2005 tot eind 2016: 12x6 = 72 afleveringen, Red.)*.

In elk geval, ik hoop dat jullie het nog steeds leuk vinden, maar het is de bedoeling dat de frequentie van Scratch wat omlaag gaat. Althans, voor zover het mijn artikelen betreft. Want ik nodig natuurlijk iedereen uit, om voor de even nummers van de SoftwareBus ook een artikel over Scratch te schrijven. Ik nodig eigenlijk iedereen uit voor elke SoftwareBus om een artikel te schrijven, over de computerhobby of de hobbycomputer, want we zouden het helemaal niet vervelend vinden als we de basis van auteurs wat konden verbreden.

Traditiegetrouw vat ik nog even samen wat we vorig jaar allemaal deden. Ik merk dat er niet heel veel huiswerk wordt ingestuurd, maar misschien doet menigeen toch serieuze pogingen. Ik wilde er in elk geval mee doorgaan.
Het leuke van programmeren is natuurlijk, dat je problemen kunt laten oplossen. En geloof me, een goede opgave verzinnen is nog moeilijker dan hem oplossen. En zeker op het gebied van beginners, waarmee je een beroep doet op aanwezige kennis en niet al te veel nieuwe inzichten verwacht.

Vorige keer begonnen we met een willekeurig project uit Scratch. Het project was niet heel bijzonder, maar het was wel voor iedereen te volgen.
Voor wie het niet meer weet: je kunt altijd surfen naar iets als **https://scratch.mit.edu/projects/xxx** waar je voor xxx een getal kunt invullen. Soms bestaan de projecten niet, vooral als je een te groot getal invult. Ook kan het voorkomen dat een project wel bestaat, maar dat je er niet kunt komen omdat het niet is gedeeld.



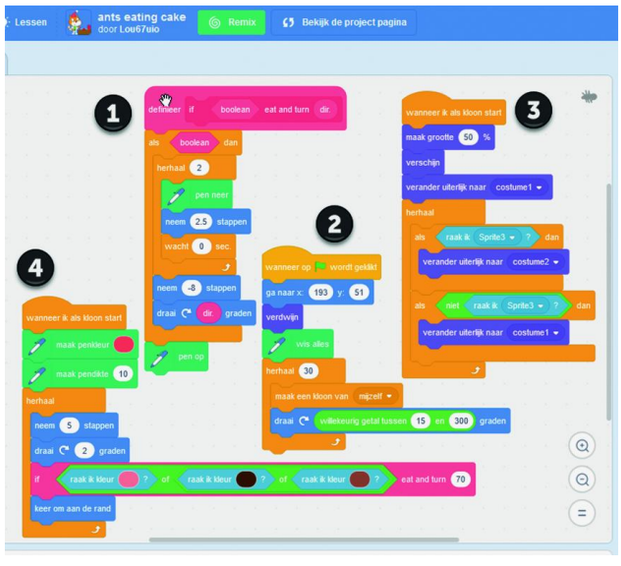
Zoals ik vorige keer al aangaf zijn we een groot voorstander van het delen van je projecten, maar liefst pas als je het min of meer hebt afgerond.
Ik heb de vorige keer ook uitgelegd hoe je meer over Scratch kon 'ontdekken', gewoon binnen de omgeving. Ik hoop dat jullie dat ook hebben gedaan, dan kunnen jullie het straks (of nu al) beter dan ik. Ook heb ik uitgelegd dat je aan projecten werkt, maar dat je projecten kunt groeperen in studio's. En verder kun je andere Scratchers volgen. En je kunt projecten van anderen kopiëren en er zelf verder aan werken.
We hebben het project 'ants eating cake' in detail bekeken. Dit hebben we gebruikt als basis voor het huiswerk. Na de opgaven nog even het codeblok ter herinnering.

Opgave 7.1 Kijk wat blok 3 doet en kijk wat er gebeurt als je dit blok weglaat. Wat is het effect op het programma?

Opgave 7.2 Stel dat je de mier tijdens het bewegen telkens van uiterlijk wilt laten veranderen. In welk blok en waar in het programma moet je dan ingrijpen? Hint: je hebt de instructie 'volgend uiterlijk' nodig, dan wisselt Scratch automatisch tussen de beschikbare uiterlijken.

Opgave 7.3 We hebben nu dertig klonen rondlopen die de taart opsnoepen. Die starten ook allemaal op dezelfde plek. Pas het programma zo aan dat er tussen de 25 en 35 mieren rond gaan lopen, die ook nog eens niet op precies dezelfde plek starten, maar zeg, ergens in het blokje met een spreiding van evenveel punten als er mieren zijn ten opzichte van het originele startpunt Dus, als er dertig mieren zijn, ergens starten met een x-waarde tussen 193-15 en 193+15 en een y-waarde tussen 51-15 en 51+15.

Opgave 7.4 Voor de echte cracks: pas de beweging van de mieren dusdanig aan dat ze 'ruiken' waar de cake is en dus wat doelgerichter richting de cake lopen en hem dus effectiever opeten. Dan maken we echt een stap op weg richting kunstmatige intelligentie, dus van Gerard Vriends verwacht ik wel een goede oplossing.

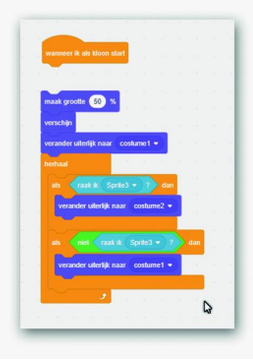


Figuur 1 - Mieren eten cake - codeblok

De makkelijkste manier om opgave 7.1 te beantwoorden is door te kijken wat er gebeurt. Als je van blokje 3 de trigger weghaalt, dan weet je dat de rest van de code niet doorlopen wordt. Je hoeft dus niet alles weg te halen, het eerste

blokje is genoeg. Je kunt het zelfs even wegschuiven, zoals hierna weergegeven in figuur 2.
Wat je ziet als je het programma start, is, dat de mieren op ongeveer (of precies?) dezelfde wijze de cake opeten, alleen zie je de mieren niet meer. Het is alsof een vochtvlek langzaam uitbreidt. Dat komt omdat met het indrukken van de groene vlag de sprite is verdwenen en in blokje 3 moet hij verschijnen, maar blokje 3 wordt niet uitgevoerd. Het programma doet het nog steeds, de mieren lopen nog steeds en eten de cake op, alleen: ze zijn onzichtbaar.

Als je opgave 7.2 wilt gaan maken moet je een paar dingen doen. Om te zien of het goed werkt is het misschien verstandig om het aantal mieren te beperken. Je kunt dit doen door in blokje 2 het aantal klonen te beperken.

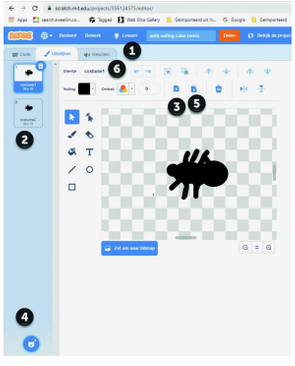


Figuur 2 - blokje 3 gesplitst

Er staat nu 30, maar je kunt daar natuurlijk 1 of 2 van maken.

Ook kun je in blokje 3 de grootte even op 100% zetten, dan zie je beter wat er gebeurt.

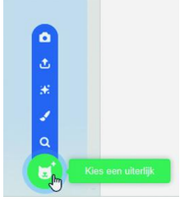
Je moet wel blokje 2 houden, want in feite gaan alleen de klonen lopen, het originele miertje, de nog niet gekloonde versie, verschijnt niet en die loopt niet, want daar is geen instructie voor.



Figuur 3 - Uiterlijken

Om de mieren wat realistischer te laten lopen moet je een tweede ‘uiterlijk’ maken. In het origineel van onze ontwerper is daar niet in voorzien, dus als je naar ‘uiterlijken’ gaat, zie je er maar één.
In figuur 3 zie je een aantal cijfers staan die voor de uitleg behulpzaam kunnen zijn. In het codeblok van sprite1, de enige sprite, kun je bij ‘1’ klikken op 'uiterlijken'.
Je krijgt dan zoiets te zien als figuur 3, maar bij ‘2’ zie je alleen ‘costume1’ staan. Die gaan we kopiëren door te drukken op de knop ‘kopie maken’ bij ‘3’. Vervolgens druk je op het poes-icoontje bij ‘4’ en dan verschijnt een keuzemenu, zoals aangegeven in figuur 4.

Dan klik je op de pen en er verschijnt een tweede uiterlijk, zichtbaar in het vakje bij ‘2’. De pen staat hier voor tekenen. Je kunt ook op een andere manier een nieuw uiterlijk maken. Als je de muis boven elk van de opties houdt, dan verschijnt een uitleg van de betekenis. Maar wij tekenen er dus één. Alleen, dat doen we niet echt.



Figuur 4 - Kies een uiterlijk

We hebben de kopie nog in het geheugen, dus als we het tweede uiterlijk geselecteerd hebben, drukken we op ‘Plakken’, knopje ‘5’ in figuur 3. Je ziet dan dezelfde figuur weer terug.
Dat is niet helemaal wat we zoeken, maar rechts van de knop ‘plakken’ zien we nog drie knopjes. We drukken éénmaal op de meest rechtse knop in het rijtje en ons uiterlijk wordt horizontaal gespiegeld.

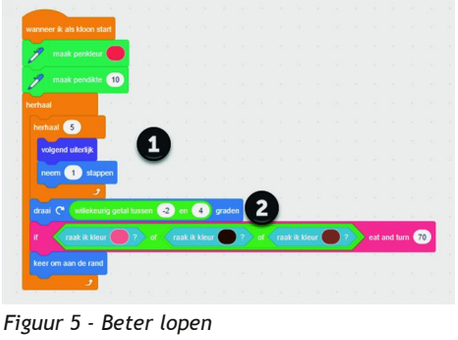
Het is de bedoeling dat we het ‘lopen’ simuleren door deze twee uiterlijken afwisselend te gebruiken. Dat is uiteraard wat houterig en niet heel echt, maar voor dit moment doen we het ermee. Uiteraard mag u dit veel mooier doen en met meerdere uiterlijken ‘echt lopen’ simuleren.

Je kunt bij ‘6’ de naam van het uiterlijk nog veranderen, maar dat is niet echt nodig. Het is wel handig dat je, als je meerdere uiterlijken hebt die je na elkaar wilt laten afspelen, een logische naamgeving hanteert. Maar wij gaan, overeenkomstig de instructie, gewoon de uiterlijken afwisselen met behulp van ‘volgend uiterlijk’ en dan doet de naam er niet toe.

Als je het tweede uiterlijk hebt gemaakt, moet je er nog voor zorgen dat het uiterlijk tijdens het lopen ook wordt gewijzigd . Het beste effect lijk je te krijgen wanneer je dat telkens per stap doet.

We moeten blokje 4 aanpassen, want daar wordt de feitelijke beweging geregeld. Dat blijkt ook wel, want dit is de eeuwige lus. Je ziet ook aan de onderkant van de lus dat er niets meer onder geplakt kan worden. Deze lus eindigt nooit, dus er is ook geen enkele aanleiding om er nog iets onder te plakken.

In het originele blok werden steeds vijf stappen genomen, dan twee graden gedraaid en dan werd gekeken of er iets te eten viel. Omdat er telkens twee graden werd gedraaid liepen de mieren min of meer rond. In het volgende codeblokje heb ik dat een beetje aangepast. Kijk maar eens mee:



Figuur 5 - Beter lopen

Het lopen van vijf stappen uit het origineel heb ik aangepast met het lusje bij ‘1’. Hier wordt vijf keer een stap gelopen, nadat het uiterlijk is veranderd. Hierdoor lijkt het meer of de mier echt loopt. Nog steeds redelijk houterig, zoals ik al eerder zei, maar als je kijkt op het grote speelveld, dan oogt het toch al aardig.
Met de draai bij ‘2’ heb ik de draai gemiddeld iets kleiner gemaakt, ongeveer 1 graad, maar ik heb de mieren ook minder gericht door wat willekeur aan te brengen. Dit oogt ook wat natuurlijker, want mieren wijken nogal eens van het rechte pad af. Je kunt natuurlijk met deze waarden spelen om het in jouw ogen meest natuurlijke gedrag te verkrijgen.

Voor opgave 7.3 moeten we aanpassingen maken in blokje 2, want hier worden de klonen aangemaakt. Er zat nog een klein addertje onder het gras, want het is niet zo moeilijk het aantal mieren te laten variëren en om een startpunt te laten variëren, maar als je het aantal mieren wilt gebruiken voor de variatie in het startpunt, dan moet je dus wel weten hoeveel mieren je gaat oproepen. Daarvoor heb je dus een variabele nodig. Laten we die maar ‘aantal mieren’ noemen.

Zoals je vast nog wel weet, kunnen we gewoon spaties gebruiken in de naam van de variabele, want we kunnen toch geen typfouten maken. Intern regelt Scratch wel een identificatie en wij hangen er een naam aan die we later kunnen selecteren.

Je kunt vervolgens wel met de opdracht ‘ga naar’ de formule compleet invoeren, maar het is misschien makkelijker om eerst de minimale en maximale startwaarden uit te rekenen, dan kun je de ‘ga naar’-opdracht wat korter maken en dus leesbaarder houden. Bovendien hoef je dan niet steeds die waarden uit te rekenen, dus in de uitvoering is het ook efficiënter. We maken dus nog vier variabelen aan.

Als we dat gedaan hebben, ziet het blokje variabelen er ongeveer zo uit als in onderstaand schema. Je ziet ook nog een ‘my variable’ staan, maar dat is de standaard variabele.

Figuur 6 - Variabelen

Je kunt hem natuurlijk ook gebruiken, maar het heeft meer zin om betekenisvolle namen te gebruiken. Je ziet links van elk van deze variabelen een vakje dat je aan kunt vinken.
Weet je nog waar dat voor was? Als je het niet meer weet moet je er maar eens een aanvinken en kijken wat er gebeurt.

Dan hebben we de code als volgt aangepast:

Figuur 7 - Klonen

We zijn niet geïnteresseerd in waar onze originele sprite heen gaat, want we gaan al onze klonen zelf positioneren. Bij het indrukken van de groene vlag zorgen we dat de primaire kloon verdwijnt en dat alles wordt gewist.

Bij ‘2’ vullen we onze variabelen, waarbij we eerst het aantal mieren berekenen, want die waarde hebben we nodig voor de volgende berekeningen. Vervolgens doen we letterlijk wat in de opdracht stond, dus je moet een paar functies binnen functies gebruiken, waarbij je dus goed moet ‘mikken’ bij het vullen van de blokjes. Maar als het mis gaat, kun je het weer eenvoudig herstellen, dus geen zorgen.

Bij ‘3’ maken we het aantal klonen en bij ‘4’ sturen we ze naar hun startpositie. Hiermee hebben we deze opgave ook opgelost. Misschien is het leuk om nu eens naar het resultaat te kijken. De mieren lopen wat rond, met een kleine neiging een beetje rechts aan te houden: je ziet een simulatie van stapjes nemen en het eetgedrag is verder niet gewijzigd.
Als alles is opgegeten, blijven de mieren rondlopen, maar het kan wel een tijdje duren, voordat het programma uitgedraaid is, omdat de mieren steeds teruglopen na een hapje. Links wordt het allemaal wel laat voordat het op is. Dat brengt ons bij de volgende opgave:

Opgave 8.1: laat het programma stoppen als alle taart op is. Voor opgave 7.4 heb ik geen antwoorden binnen gekregen. Ik weet ook niet, of het echt mogelijk is, want ik heb de benodigde functie nog niet gevonden in Scratch. Je zou willen, dat een mier kon ‘ruiken’ waar nog taart was, ofwel waar nog een restje in het donkerroze over was.

Tot hier het huiswerk van de vorige keer. Op zich kon iedereen dit wel oplossen op basis van hetgeen we al behandeld hadden. Misschien was het gelukt, misschien wel op een heel andere manier dan ik deed, maar de aanpassingen moesten toch wel plaatsvinden in de blokjes volgens mijn uitwerking. Als je een echt andere manier hebt gevonden ben ik daar natuurlijk erg benieuwd naar. Overigens, als je een antwoord hebt dat afwijkt van mijn antwoord, maar het net zo goed werkt, dan is dat natuurlijk uitstekend.

Er is niet maar één antwoord goed: het is belangrijk dat je uitwerking werkt. En als het eenmaal werkt, dan kun je het misschien nog wel verbeteren, maar voor ons geldt: goed is goed. Maar, als we het niet echt slim kunnen maken, kunnen we natuurlijk wel doen alsof. Hoe gaat het in het echt: de mieren gaan op de zoetigheid af, pakken er iets vanaf en lopen naar het nest terug. Daar laten ze het eten achter en ze gaan weer op pad. Natuurlijk eten ze zelf ook iets, anders gaan ze dood, maar er is een stroom van mieren naar de taart en een stroom naar het nest.

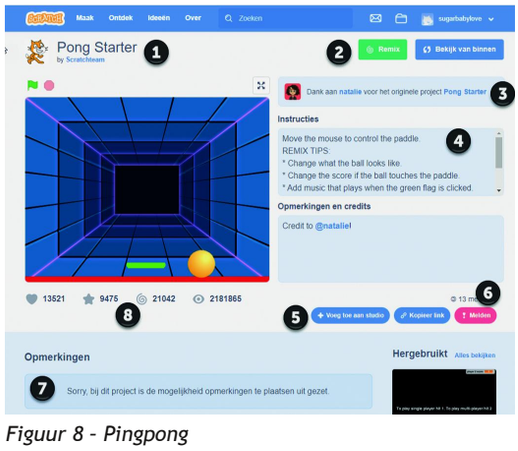
Opgave 8.2: Definieer een nest en een taart en laat de mieren de taart opeten. Hierbij mag je uiteraard gebruik maken van hetgeen we allemaal gezien hebben, maar je mag het ook van begin af aan opbouwen.
De vorige keren schreef ik al over de vele uitleg die beschikbaar is, niet in de laatste plaats op de site van Scratch zelf. Om niet alleen maar over huiswerk te praten wil ik dus ook wat nieuwe theorie behandelen.

We gaan aan de slag met ons eerste spel. Het begint natuurlijk eenvoudig en we gaan niet helemaal vanuit niets beginnen, want dan red ik het niet meer met zo’n klein artikel. De meeste lezers zijn van ongeveer mijn generatie of nog iets ouder, dus jullie herinneren je vast nog wel het computerspel ‘pingpong’. Je had normaal gesproken twee spelers, maar je kon ook tegen de ‘computer’ spelen.

In eerste instantie was het een kastje dat je aansloot op de tv en je had twee controllers waarmee je het batje omhoog en omlaag kon bewegen. Later werd dat een joystick, waarmee je meer vrijheidsgraden had. Maar we beginnen even heel eenvoudig. Met het volgende project:
https://scratch.mit.edu/projects/10128515/

Dit is een spel dat door het Scratch-team ter beschikking is gesteld. Ze geven gelijk wat tips om het te verbeteren, maar laten we eerst maar eens gaan kijken. Het is een project uit 2013, Scratch zelf is al weer iets ouder (ca. 2006) dus het is zo’n beetje uit het midden van het Scratch-tijdperk. Ik heb dit project opgepikt uit de ideeënpagina van Scratch, waar je dus nog veel meer ideeën kunt opdoen om zelf aan de slag te gaan.

Het is echt geweldig hoeveel informatie gewoon binnen de Scratch-omgeving beschikbaar is om jezelf verder te ontwikkelen. Je kunt ook buiten de Scratch-omgeving terecht, maar voorlopig zijn we hier nog lang niet uitgekeken. En we willen wat meer leven in de brouwerij, dus we gaan aan de slag met dit project.



Figuur 8 - Pingpong

Bij het openen van de projectpagina ziet het beeld er ongeveer uit zoals hierboven. Bij ‘1’ zie je weer de naam van het programma en de auteur, in dit geval een team.

Bij ‘2’ de bekende knoppen ‘Remix’ en ‘Bekijk van binnen’. We gaan natuurlijk remixen, want we gaan er wat aan verbouwen, maar voorlopig kijken we nog even rond. Bij ‘3’ zie je de basis van dit programma, dus Natalie (uit het Scratch-team) heeft de basis gelegd.
Als wij nu een remix maken, zie je hier twee regels staan, want wij danken weer het Scratch-team voor het originele project, maar daaronder komt de dank aan Natalie dan te staan.
Ik weet niet hoeveel lagen het verder gaat, maar zo houd je dus wel de credits als iemand jouw werk gebruikt. En, zoals eerder opgemerkt, binnen Scratch word je aangemoedigd om werk van elkaar te gebruiken.

Bij ‘4’ zie je wat instructies over de werking van het programma, maar in dit geval ook wat tips om het verder te bewerken. Dit gebied is niet zo heel groot, dus zit er een

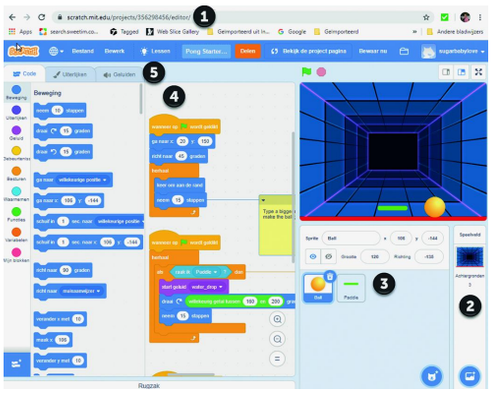
scrollbar naast. Je kunt dus meer tekst toevoegen dan er in het schermpje past. Bij ‘5’ kun je het project toevoegen aan een studio; uiteraard moet je daarvoor wel rechten op die studio hebben.

Bij ‘6’ zie je nog een knopje ‘! Melden’. Het is de bedoeling dat Scratch een veilige omgeving is en blijft. Als mensen een project plaatsen dat tegen de principes van Scratch ingaat, dan kun je dat melden.

Het team achter Scratch zal dan het project bekijken en eventueel maatregelen nemen. Ik ga er niet al te veel woorden aan vuilmaken, maar aangezien het een kindvriendelijke omgeving is, verwacht ik dat projecten met veel geweld, met seks en met racisme allemaal niet mogen. En reclame voor roken of alcoholgebruik zal ook wel niet de bedoeling zijn, om over drugs maar te zwijgen.

Enfin, Scratch doet er dus alles aan om een veilige omgeving te zijn en het team kan niet alles bekijken wat gepubliceerd wordt, maar met behulp van tips kan er gerichter gekeken worden. Ten slotte valt nog op bij ‘7’, dat het plaatsen van opmerkingen bij dit project niet (meer) mogelijk is. Je hebt bij je projecten een schakelaartje waarmee je commentaar aan en uit kunt zetten. Het is natuurlijk handig om dit op ‘uit’ te zetten, zolang je wel iets gepubliceerd hebt, maar nog niet helemaal klaar bent.

Maar als je project al een paar jaar bestaat, dan heeft nieuw commentaar niet zoveel zin als je toch niet van plan bent het te lezen. Laten we eens van binnen gaan kijken wat we hiervan kunnen leren. Als het goed is, oogt dit al vertrouwd.



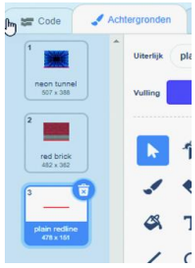
Figuur 9 - Pingpong achter de schermen

Als je de URL bekijkt, dan zie je dat ik intussen een remix gemaakt heb, want het nummer wijkt af. Je ziet ook dat het nummer intussen iets boven de 356 miljoen zit, dus het gaat hard.

Je ziet bij ‘2’ dat er drie achtergronden zijn. Als je op de achtergronden klikt, dan zie je bij ‘5’ het tabblad ‘Uiterlijken’ veranderen naar ‘Achtergronden’ en dan kun je de achtergronden bekijken. In feite zijn er maar twee spelachtergronden. Er is een achtergrond genaamd ‘Plain redline’; die als basis dient. Dit is een leeg veld, met onderin de rode lijn. Alle achtergronden die je voor dit spel wilt maken moeten die lijn hebben, want straks gaan we kijken of de bal die kleur raakt:

Je kunt dus achtergronden erbij maken, maar gebruik dan deze basis, dan weet je zeker dat je die rode balk in de juiste kleur beschikbaar hebt. Je kunt er dan zelf een maken door te tekenen, maar je kunt ook iets anders gebruiken. Leef je uit.
Bij ‘4’ zie je de bekende code, in dit geval die welke behoort bij de geselecteerde sprite bij ‘3’. Er zijn slechts 2 sprites in het spel, namelijk de bal en het batje. We gaan zo wat dieper op de code in. We kijken eerst even naar de code van het batje (Paddle), want die is heel eenvoudig.
Het is heel simpel: wanneer de groene vlag wordt aangeklikt, blijft het batje in horizontale richting de muis volgen, maar in verticale richting gebeurt er niets.
De programmeur heeft het batje ergens op de bodem gelegd en de y-waarde zit daarmee vast, in dit geval op -144.

Waarschijnlijk was het idee ‘hoe minder code, des te minder vertraging’, maar als je dus je batje een keer ergens anders heen hebt geschoven - je kunt tenslotte alle sprites oppakken en verplaatsen - dan kun je hem dus niet vanzelf weer terug naar beneden brengen.



Figuur 10 - Achtergronden

Het kan lastig spelen worden als het batje te hoog zit. De code voor de bal is iets ingewikkelder, maar ook niet echt heel complex. We zouden dit moeten kunnen begrijpen.

In blokje ‘1’ wordt bij het klikken op de vlag de bal naar een startpositie gebracht, vanwaar hij gaat bewegen. Vervolgens blijft hij eeuwig bewegen, keert alleen om als hij aan de rand komt. Dat is zijn primaire taak. Maar er zijn nog wat blokken die de boel kunnen verstoren.

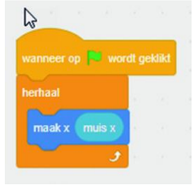
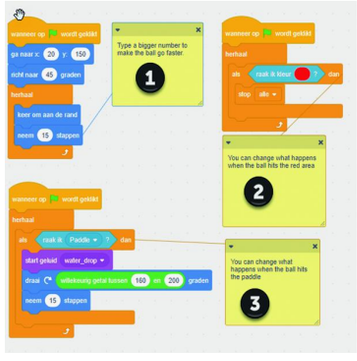


Fig 11- Paddle code

In blokje ‘2’ wordt continu gekeken of de bal de kleur rood raakt. Als dat het geval is wordt de actie ‘Stop alle’ uitgevoerd, oftewel, het programma wordt beëindigd.

In blokje ‘3’, ten slotte, wordt gekeken of het batje wordt geraakt. Als dat het geval is wordt er een geluid afgespeeld.



Figuur 12 - Code van de bal

Er zijn wat geluiden beschikbaar, ik wil daar een volgende keer nader op ingaan. Vervolgens wordt er gedraaid, gemiddeld zo’n 180 graden, dus als hij recht van boven komt, dan gaat hij ook ongeveer recht naar boven terug. En dit wordt de hele tijd herhaald.
Er zijn dus vier stukken code die parallel worden uitgevoerd zolang het programma loopt. Echt multitasking dus.

Huiswerk

En nu is het dus tijd voor eigen werkzaamheden, weer een stukje huiswerk. Ik had al twee opgaven opgeschreven, en daar komen er nu een paar bij.

Opgave 8.3: Voeg een scorebord toe dat bijhoudt hoeveel keer het batje de bal heeft geraakt.

Opgave 8.4: Zorg dat het batje bij aanvang van het spel op de goede hoogte staat en bouw ook een vertraging in.

Opgave 8.5: Maak de bal steeds iets kleiner als het batje wordt geraakt.

Opgave 8.6: Laat de bal ook sneller gaan als het batje een veelvoud van tien keer is geraakt.

Opgave 8.7: Maak het batje kleiner na elke 25e keer dat de bal wordt geraakt.

Opgave 8.8: Probeer een score van 100+ te halen.

Veel plezier met het spel en met de opgaven!