Scratch (6)

René Suiker

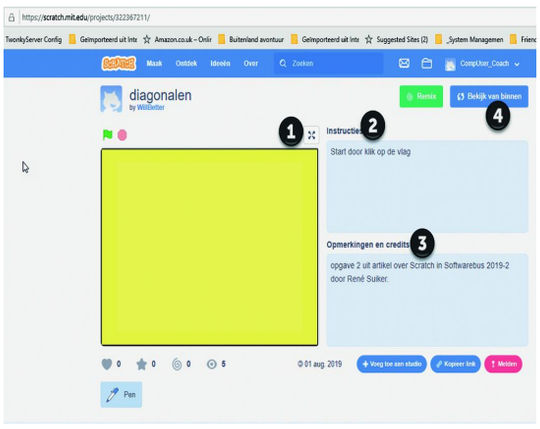
Dit is deel 6 in een reeks artikelen over de programmeer-app Scratch, geschreven door onze hoofdredacteur René Suiker. Zonder enige voorkennis duikt hij in de materie, en tijdens die duik verwerkt hij de opgedane feiten en weet die samen te vatten in een zeer leesbaar en leerzaam verhaal. Zelf vindt hij dat niks bijzonders. Maar het mag hier wel eens gezegd worden: de redactie heeft daar heel veel bewondering voor.(Red.)

Vorige keer, SoftwareBus 4, heb ik overgeslagen. Deels, omdat Scratch al in een artikel van onze voorzitter, Ton Valkenburgh, onder de aandacht kwam. En twee artikelen over Scratch, hoe leuk ook, leek me wat te veel van het goede. En een keer Scratch onder Linux was natuurlijk leerzaam én twee vliegen in één klap, want we hebben een leuk Platform Linux, waar heel veel kennis zit, maar het komt niet altijd tot uiting in artikelen. Vandaar dat ik even rust had. En dat was fijn, dat had ik ook even nodig.

Maar nu weer lekker scratchen. Tijdens de CompUfair, waarover elders in dit blad meer, mocht ik een workshop geven. Dat gaf me de gelegenheid om met een aantal mensen met Scratch bezig te zijn, plus dat ik natuurlijk een en ander moest voorbereiden. Na de rustweek lekker aan de slag.

Tijdens de voorbereiding ging ik eerst maar eens kijken wat ik allemaal al behandeld had. Dat is best al veel, als je er eens naar terugkijkt. Vijf artikelen, elk zo'n vier pagina's, het is nog geen boek, maar een begin is er. Daarnaast nog even gekeken naar het huiswerk dat ik had opgegeven, plus de binnenkomende reacties.
Dan nog even organiseren dat ik een docentenaccount op de site van Scratch kreeg, zodat ik tijdens de workshop op de CompU*fair* even snel accounts bij kan maken. Al met al moet je bij een workshop niet alleen een goed verhaal hebben, je moet ook een doel hebben voor je deelnemers én je moet je logistiek op orde hebben.

Het huiswerk van de afgelopen keren was misschien voor sommigen wat te hoog gegrepen, maar ik heb ook goede oplossingen binnen gekregen. Het leuke van Scratch is dat je me alleen maar het projectnummer hoeft door te geven en ik kan zelf gaan kijken.
Hoe kijk je dat nu na? Wel, allereerst ga ik gewoon naar zo'n project pagina, zoals:
scratch.mit.edu/projects/322367211/.
Dit is de inzending van een van onze lezers, Wim Boon. Hij heeft meerdere opgaven gemaakt en hij geeft er blijk van, goed te snappen wat we aan het doen zijn. En hij heeft er geen bezwaar tegen dat we zijn inzending gebruiken voor dit artikel. Dat brengt me, als klein zijstapje, ook nogmaals bij het hele idee van Scratch: we delen alles met elkaar. Maar uiteraard vind ik het wel netjes om het te vragen, alvorens ik iemands inzending ga gebruiken.



Figuur 1 - Ingezonden huiswerk

Figuur 1 - Ingezonden huiswerk

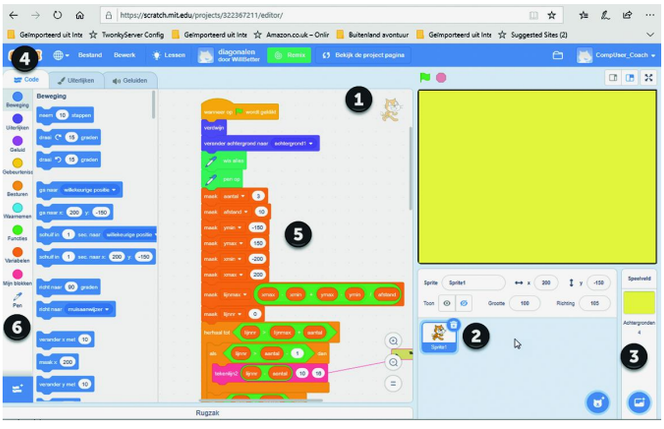
Onder (1) zien we de knop om het speelveld te maximaliseren. Voor de test doe ik dat ook. Bij (2) zie je de instructies, die de ontwerper (Wim Boon dus), voor de gebruiker achterlaat. In dit geval dus de instructie om het programma te starten met de groene vlag. Bij (3) zie je opmerkingen en credits, zeg maar wat achtergrondinformatie over dit programma. In dit geval dus kort en bondig, maar je kunt hier ook uitgebreidere informatie kwijt.

Testen dus, ik klik op maximaliseren (1) en dan op de groene vlag. Vervolgens zie ik drie diagonalen van linksboven naar rechtsonder lopen en dat is wat ik gevraagd heb. Als ik op de rode vlag druk terwijl het programma loopt, dan stopt de uitvoering. Dat is ook correct, zo hoort het te werken met Scratch.
Omdat de opgave vrij rechttoe rechtaan was, valt er verder niet heel veel te testen. De lijnen worden getekend en ze lijken te bewegen, dat wil zeggen, als de vierde lijn getekend wordt, dan wordt de eerste weggehaald, en zo voort. Het lijkt dus dat de lijnen bewegen. Helemaal geslaagd en als beste inzender willen we hem dan ook vooral aanmoedigen om door te gaan.
Overigens, ik heb niet bij elk mogelijk vakje een nummer gezet hierboven, maar boven het vlaggetje vind je de naam van het programma en daaronder wie het gemaakt heeft. Als je op die naam klikt, dan kun je andere projecten van deze Scratcher vinden.

Goed, hoe heeft hij dit nu gedaan? Daar zijn we natuurlijk allemaal benieuwd naar. En zoals in figuur 1 bij (4) is aangegeven, kunnen we de zaak 'van binnen' bekijken. Dat doen we uiteraard ook, dus klik maar gerust op 'bekijk van binnen'. We zien dan figuur 2.

Figuur 2 - Uitwerking van het huiswerk

We zien hier een aantal zaken die de moeite van het vermelden waard zijn. Onder (1) zien we het werkveld waar het programma geschreven wordt. In de afbeelding hierboven zie je, zowel rechts van dit blok als eronder, de mogelijkheid om te scrollen.
Die mogelijkheid is er altijd, maar we hebben hem nu ook feitelijk nodig om alle code te kunnen bekijken. Er is meer dan op de pagina past, tenzij je alles verkleint, maar dan kan ik het in elk geval niet meer lezen.
Bij (2) zie je de sprites. In dit geval is het er maar één, want we maken niet echt gebruik van verschillende figuren hier. Je ziet dat Sprite1 geselecteerd is, en bij (1) zie je dat in het werkveld deze sprite ook geselecteerd is. Dat wil dus zeggen, dat we nu naar de code kijken van Sprite1. Ik had het al eens eerder gemeld: Scratch is echt object-georiënteerd en dat zie je ook in het werkveld.
Bij (3) kun je de achtergronden bekijken die we gebruiken. In dit geval een effen achtergrond. Tijdens de test blijkt



Figuur 2 - Uitwerking van het huiswerk

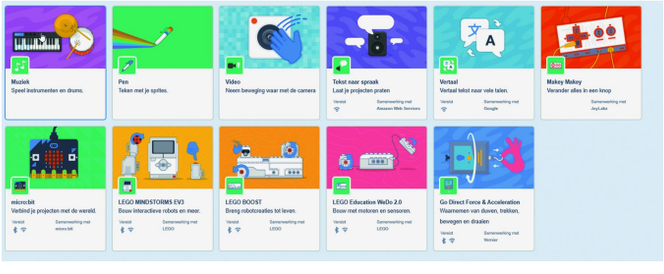
trouwens, kleinigheidje, dat de achtergrond een net iets andere kleur heeft dan wat we tekenen als we de lijnen weghalen. Een miniem kleurverschil, maar dit is een detail.

Als je de achtergrond wilt bewerken, laten we daar nu maar eens aandacht aan besteden, dan selecteer je die achtergrond. Je ziet dan dat het werkveld leeggemaakt wordt. Ook zie je dat rechtsboven Sprite1 verdwijnt en een lichte versie van de achtergrond tevoorschijn komt. Als je nu bij (4) op 'Uiterlijken' klikt, dan kun je de achtergrond bewerken. Overigens staat er dan 'Achtergronden' in plaats van 'Uiterlijken' als een achtergrond geselecteerd is. Hierover straks meer.

Bij (5) zie je dat Wim een aantal variabelen gedefinieerd heeft. Hiermee heeft hij het programma wat flexibeler gemaakt dan de feitelijke opgave, maar dat is natuurlijk helemaal prima. Je ziet bijvoorbeeld, dat hij de variabele 'aantal' gedefinieerd heeft, met een waarde 3. Als je die verandert in bijvoorbeeld 5, dan worden er geen drie lijnen getoond, maar vijf. Zo kun je ook met de andere variabelen spelen, dan heb je gelijk een leuk beeld van wat je zoal met variabelen kunt doen.
Er zijn nog twee zaken opvallend hier die enige aandacht behoeven. Toen ik de eerste keer over Scratch schreef waren de peninstructies beschikbaar in het codeblok.

In de nieuwe versie moet je deze apart activeren. Dat doe je door op het knopje 'voeg uitbreiding toe' te klikken, dat blauwe knopje links onderin.

Je krijgt dan een scherm te zien met de optionele codeblokken. Op dit moment zijn er elf uitbreidingen beschikbaar.



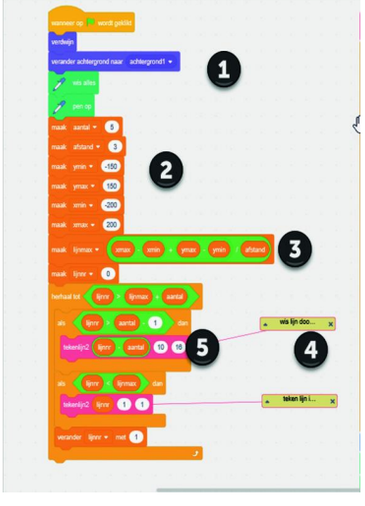
Figuur 3 - Uitbreidingen

Ik denk dat er in de loop van de tijd wel uitbreidingen bij komen, maar hier zie je dus als tweede de optie 'pen'. Als je de pen wilt gebruiken in je code, dan moet je deze hier dus activeren.
Je vindt hier ook andere uitbreidingen waarvan we in de toekomst misschien ook gebruik gaan maken. Je ziet o.a. Muziek en Video, maar ook blokken om met Lego te werken. Scratch is dus een programmeertaal met ruime uitbreidingsmogelijkheden en ik kan me zo voorstellen dat ik nog niet uitgepraat ben. Maar eerst nog even ons huiswerk.
Het tweede aspect bij (6) waar ik op wilde wijzen, dat ik nog niet behandeld had, maar dat in deze uitwerking wel aan de orde komt, is de groep 'Mijn blokken'. Hier kun je zelf routines bedenken die door je programma weer opgeroepen kunnen worden. In het geval van deze opgave heeft Wim een eigen instructie 'tekenlijn2' gedefinieerd en deze roept hij vervolgens telkens aan vanuit het hoofdprogramma.
Zij die vroeger ook geprogrammeerd hebben herkennen dit mogelijk wel. In C had je functies, in Pascal had je functies en procedures en in Basic had je subroutines. Het idee is hetzelfde, een stukje programma dat je steeds weer gebruikt zet je apart en roep je steeds weer aan, in plaats van steeds alle code te herhalen. In deze uitwerking is dat goed gegaan. Laten we eerst eens in detail naar het hoofdprogramma kijken:

Figuur 4 - Hoofdprogramma

Bij (1) zien we de start van het programma. Als er op het vlaggetje wordt geklikt gaat het programma van start en wordt allereerst de uitgangssituatie zekergesteld. Dat houdt in, dat de sprite uit beeld gaat ('verdwijn'), dat de pen omhoog gaat, dat alles wat getekend is gewist wordt en dat de achtergrond wordt ingesteld volgens achtergrond1, in dit geval een effen gele achtergrond. De volgorde doet er hier niet zo heel veel toe, want dit gebeurt allemaal redelijk tegelijkertijd.
Vervolgens worden bij (2) de variabelen gevuld met de startwaarden. We zien dat er best wel wat waarden gevuld zijn, maar daar is over nagedacht. Bij (3) wordt getoond hoe we een variabele kunnen vullen met een 'functie'.

Worden daarvoor de variabelen nog initieel gevuld met een constante waarde, de variabele 'lijnmax' wordt gevuld op basis van een berekening met behulp van de reeds gedefinieerde variabelen. Hier is de volgorde wel van belang, want je kunt niet beginnen met lijnmax, voordat je de waarden die nodig zijn hebt bepaald.



Voor de mensen die goed kleur kunnen onderscheiden zien we dus hier de oranje blokjes, die bepalen de beginwaarde van de variabelen. De groene blokjes zijn functies (die hebben ook een andere vorm, die zijn ovaal, of later een afgepunte ovaal) en daarbinnen worden dan weer de oranje variabelen gebruikt.

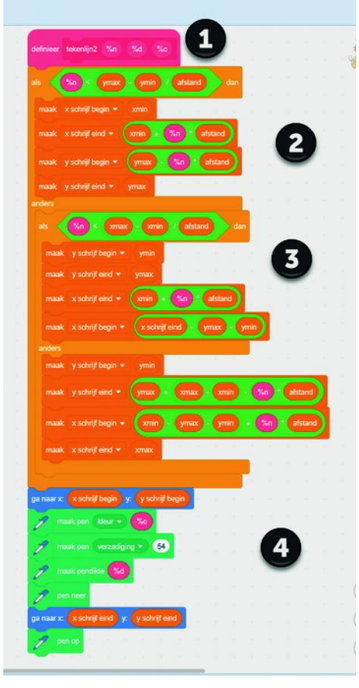
Bij (4) zien we nog een handige extra, die we ook nog niet behandeld hebben. Dat is dat grauw gele blokje, met een rood lijntje naar een ander blokje. Hierin staat commentaar. Dit is heel nuttig, als je een programma maakt dat wat gecompliceerder is, om de structuur uit te leggen. Door je programma op deze manier te documenteren kun je later, als je nog iets aan wilt passen, eenvoudig terugzien wat je gedachten waren toen je deze code schreef. Want ja, al is het blokjes aan elkaar klikken, uiteindelijk zijn we dus code aan het schrijven.

Hoe maakt je commentaar? Door op een blokje in je code te staan en dan met de rechtermuisknop te klikken. Je krijgt dan een menu te zien, dat afhankelijk is van de context, maar er staat meestal wel de optie bij 'commentaar toevoegen'. Als je daarop klikt, komt er een commentaarblokje tevoorschijn. Daar kun je dan je tekst in kwijt. Sluit je dit af, dan wordt het tekstblokje verkleind, maar er blijft in beeld dat er commentaar is in dat blokje.
Door op het driehoekje links te klikken wordt het blokje opengeklapt en kun je het commentaar lezen. Het commentaar heeft geen invloed op het programmaverloop, het is gewoon een stukje documentatie bij je programma.

Bij (5) komen we dan bij de feitelijke uitvoering van het programma, waarbij er twee besturingselementen worden toegepast, in een iets lichter oranje dan de variabelen. Dat is allereerst de lus, ‘herhaal tot’. Je ziet: dat blok omvat de rest van de code in het hoofdprogramma. Als de situatie bij ‘tot’ bereikt is, dan stopt het programma. Dat is de betekenis van deze lus. Hiervoor hebben we ook al lussen gezien, als eerste de lus die gewoon een vast aantal keren herhaalt (hoewel, vast, ook daar kan je een functie gebruiken die weer variabelen gebruikt, enz.). Verder hebben we ook de oneindige lus gezien, dus gewoon ‘herhaal’. Deze lus blijft doorlopen worden zolang het programma loopt.
In dit geval dus de ‘conditionele’ lus, zoals dat heet. Hij wordt doorlopen totdat een eindconditie wordt bereikt. In dit geval heeft Wim besloten elke lijn een nummer te geven. In essentie tekent Wim een aantal lijnen, op basis van een begin- en een eindpunt. Als hij drie lijnen moet tekenen, dan wist hij eerst lijn 1, alvorens lijn 4 te tekenen. Zo wordt de beweging gesuggereerd. En dat doet hij netjes, door de grenswaarden nauwkeurig te bepalen. Het feitelijke tekenen vindt plaats in de eigen functie ‘tekenlijn2’.

Figuur 5 - Eigen routine - tekenlijn2

Als je op het hoofdscherm, zie figuur 2 bij (6), op ‘mijn blokken’ klikt, dan scrolt het gebied rechts daarvan, met de feitelijke instructies, naar die positie, dat het eerste deel van ‘mijn blokken’ bovenaan staat.
En onder ‘mijn blokken’ staan alle blokken die je voor dit programma gedefinieerd hebt en daarboven nog een blokje ‘Maak een blok’. Hiermee kun je een eigen blokje maken, waarover straks meer. Ik moet straks nog even kijken of ik me in dit artikel aan alle beloftes van daarnet heb gehouden, maar ik ga het wel proberen. Maar nu dan eerst dit huiswerk, al bijna een artikel op zich.



Onder (1) dus de definitie. Hier zie je de naam van de functie, of procedure, ofwel het blok. Met deze naam roep je straks de functie aan. Je ziet ook dat er drie parameters worden meegegeven. Dat is zo leuk aan Scratch, je roept de functie aan met een blok en je ziet in dat blok ook gelijk wat je moet invullen. In de oude programmeertalen definieerde je een functie met de parameters, maar kon het nog wel eens voorkomen dat je een functie aanriep met te veel of te weinig parameters.
Je compiler kon je programma niet compileren als je een interpreter gebruikte (zoals vaak bij Basic, maar hier dus ook bij Scratch), dan liep het programma op dat moment vast. Hier dus niet, in de definitie bepaal je zelf hoeveel parameters je nodig hebt en bij ‘mijn blokken’ zie je de gedefinieerde functie staan, met daarin de blokjes voor het aantal parameters dat je nodig hebt.
Kijken we even naar de code hier, dan zien we dat er verkorte namen voor de parameters worden gebruikt, maar ik denk dat we redelijkerwijs kunnen veronderstellen, dat ‘%n’ voor het lijnnummer geldt, ‘%d’ voor de dikte van de lijn en ‘%c’ voor de kleur van de lijn. In het hoofdprogramma kun je zien dat Wim, bij het wissen van de lijn, deze iets dikker maakte dan de getekende lijn. Waarschijnlijk omdat hij had ervaren dat als hij precies dezelfde waarde koos, de lijnen niet geheel gewist werden.
Dat brengt mij op de eerst opgave voor de volgende keer:

Opgave 6.1

Waarom worden de lijnen niet goed gewist als je de dikte van wissen gelijk maakt aan de dikte van de getekende lijn?

Bij (2) zien we het ‘als-dan-anders’ controle-element. Hier wordt aan het begin een ‘waardevergelijking’ gemaakt. Als het ‘waar’ is wordt de code na ‘dan’ doorlopen. Als het niet waar is wordt de code na ‘anders’ doorlopen. Ook hier kun je weer binnen zo’n blok ongebreideld code invoegen.
Je ziet zelfs bij (3) dat hier binnen het blok ‘anders’ weer een ‘als-dan-anders’-blok is opgenomen. Je kunt ook weer een lus opnemen binnen een ‘als-dan-anders’-constructie en binnen die lus ook weer een lus, enzovoort.
Bij (4), ten slotte, vinden we het feitelijke tekenen van de lijn. Besef dat door de uitgangssituatie de pen omhoog is: er wordt dus geen lijn getekend als de sprite zich verplaatst. Dan wordt de sprite op het beginpunt neergezet. Vervolgens wordt de pen omlaag gezet en dan beweegt de (onzichtbare) sprite zich naar het eindpunt. Ten slotte wordt de pen weer opgetild, zodat deze actie zich kan blijven herhalen.
Al met al vind ik dat voor een mooie oplossing is gekozen. Het eigen blokje om de lijnen te tekenen liep voor de behandeling uit, en als gezegd, ik ga er zo mee verder.
Het uit elkaar trekken van het hoofdprogramma en deze functie heeft dus het voordeel dat je de code om te tekenen maar één keer hoeft te schrijven en te testen. Als dit goed is, kun je wijzigingen in het hoofdprogramma aanbrengen en erop vertrouwen, dat de subroutine het blijft doen. Dat is dan ook onze volgende opgave.

Opgave 6.2

as het programma zo aan dat de getekende lijnen in willekeurige kleuren zijn. Let op, dit doe je dus in het hoofdprogramma. Het ‘eigen blok’ hoeft hiervoor niet aangepast te worden.
Ten slotte nog, voor de liefhebbers. Je ziet onderin het blok dat de feitelijke lijn tekent. Je kunt je voorstellen dat je hier ook een eigen blok voor maakt. Op die manier roep je binnen een blok een volgend blok aan. Ook dit is weer handig, omdat je dan dat tweede blokje eventueel later ook in andere routines kunt gebruiken.

Opgave 6.3

Maak een eigen blokje dat één lijn tekent, met zes parameters: startpositie x en y, eindpositie x en y en de kleur en de dikte.

Opgave 6.4

Pas het blok ‘tekenlijn2’ zo aan, dat deze ons nieuwe blokje aanroept, in plaats van de reeks instructies bij (4).
Ten slotte, maar ik weet nog niet of dit kan, maar ik zou het leuk vinden, dus een leuke extra opgave. Nu wordt het wissen van de lijnen gedaan door deze met de achtergrondkleur te ‘overschrijven’. Hier is de achtergrond geel, dus wordt een gele lijn over de bestaande lijn getekend.

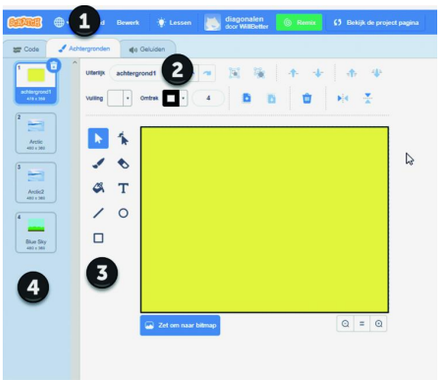
Ik zag graag een functie die een lijn tekent (die hebben we) en één die een lijn wist (die hebben we nog niet), zodat de achtergrond weer zichtbaar wordt, ongeacht hoe de achtergrond eruit ziet. De lijnen moeten dus over bijvoorbeeld een foto lopen en de foto moet weer zichtbaar zijn aan het eind.
Ik weet dat het kan met de ‘wis alles’ voordat je lijnen tekent, dus de opdracht is sowieso uitvoerbaar, maar het zou mooi zijn als ook gewoon lijntje voor lijntje gewist kan worden. Zoek maar eens uit met de kleuren of er zoiets mogelijk is.

Opgave 6.5

Maak een functie die een lijn wist, dus ongeacht de achtergrond. Maak dan de hele opgave nog een keer compleet, maar dan met een achtergrond die niet effen is, maar bijvoorbeeld de standaardachtergrond ‘Arctic’.

OK, u heeft nog een en ander van mij te goed, op basis van beloftes eerder in dit artikel. Allereerst: de uiterlijkheden. Je klikt dus op het uiterlijk (achtergrond1) en je ziet dat het codeblok leeg is. Er is namelijk nog geen code actief voor de achtergrond. Maar klik je dan niet op ‘code’ maar op ‘achtergronden’, dan zie je dit beeld:

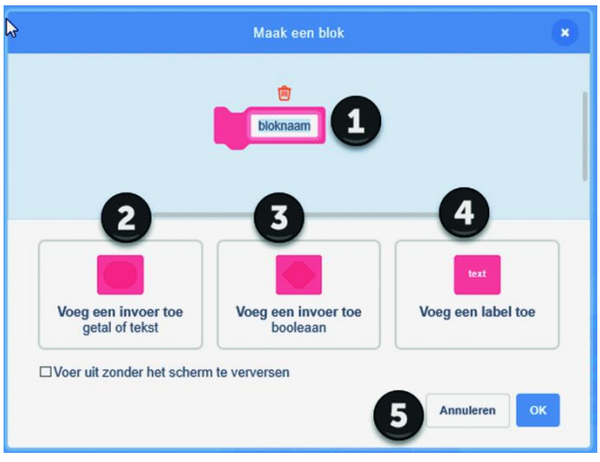
Figuur 6 - Achtergronden



Bij (1) zie je dus de knop ‘achtergronden’. Toen we nog een sprite geselecteerd hadden stond hier ‘Uiterlijken’, een en ander is dus contextgevoelig.
Bij (2) zie je een aantal mogelijke bewerkingen, maar ook de naam van deze achtergrond. Om de lading beter te dekken zou je hem ook ‘geel’ kunnen noemen. Als je dat doet, dan past Scratch dit ook aan op de codepagina waar je naar deze achtergrond verwijst.
Het object is intern eenduidig geïdentificeerd, de naam die je er zelf aan geeft wordt dus overal direct mee aangepast.

Bij (3) zie je nog wat gereedschappen om de afbeelding te bewerken. Deze komen je vast bekend voor van andere programma’s waarmee je figuren kunt bewerken, dus hier ga ik niet uitgebreid op in.
Bij (4) zie je welke achtergronden beschikbaar zijn voor dit programma. Onderin zie je nog een knopje (valt buiten beeld van figuur 6) om achtergronden toe te voegen. Er zijn achtergronden vanuit Scratch beschikbaar, maar je kunt ook zelf achtergronden definiëren of zelfs foto’s uploaden.

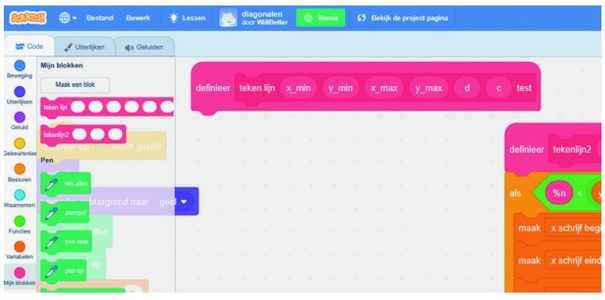
Als je met rechts op een van de afbeeldingen klikt, dan krijg je een menu met daarop de opties ‘dupliceren’, ‘exporteren’ (om eventueel in een ander programma te importeren) en ‘verwijderen’. Verwijderen kan ook door een achtergrond te selecteren en op de prullenbak te klikken.
Zoals in zoveel programma’s: als je per ongeluk iets weggooit, kun je met Ctrl+Z de actie in veel gevallen nog ongedaan maken. Experimenteer gerust met wat achtergronden, zodat je er meer vertrouwd mee raakt. Een ander onderdeel dat ik nog zou toelichten was het ‘eigen blokje’. Als je op ‘maak een blok’ klikt, komt het volgende scherm in beeld:



Figuur 7 - Eigen blok

Bij (1) vul je de naam in van de functie die je gaat maken. Bijvoorbeeld ‘Teken lijn’. In veel programmeertalen kun je in functiebenamingen geen spaties gebruiken, maar in Scratch wel, want er ontstaat geen verwarring: je kiest het blok, je hoeft de naam niet te typen. De interne representatie van het blok is uniek, maar daar heb je als programmeur niets mee te maken. Klik ik dan op het knopje onder (2), dan zie je het blok direct veranderen. Er staat dan de naam van de functie, met daarachter één invoerparameter. Als ik tevreden ben met de parameters, dan klik ik bij (5) op OK en dan zie ik het blok in het codeveld staan.

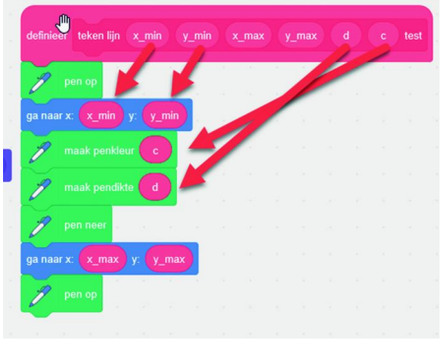
Heb ik toch nog iets vergeten in de declaratie, dan kan ik met rechts op het blok klikken en dan kiezen voor ‘aanpassen’; dan kom ik weer in het ‘eigen blok’-scherm terecht, zoals in afbeelding 7, maar dan wel met de reeds gedefinieerde parameters.



Figuur 8 - Eigen blok definitie

Je ziet nu het blok ‘definitie’ bovenin staan, maar er hangt nog niets onder. We moeten nu gaan programmeren wat dit blok doet en daarbij moeten we de parameters gebruiken die het blok bij zich heeft.
Die parameters zie je niet bij de variabelen staan: die vergen een eigen aanpak. Je kiest bijvoorbeeld bij ‘beweging’ het blokje ‘ga naar x: y:’
Daar staan nu twee waardes bij, afhankelijk van de positie van de sprite op het moment dat je het blokje pakt. Dit is heel handig, want zo heb je de beginpositie bij de hand. En je kunt hier dus ook een functie in ‘proppen’. En, in dit geval, in elk van de blokjes één van de parameters van de functiedefinitie. Je krijgt dan iets als:

Figuur 9 - Teken lijn



En hiermee is deze functie gedefinieerd. Alle parameters zijn ook gebruikt en nodig. Het woordje test was om te kijken wat een label doet, maar dat is m.i. slechts een extra toevoeging, als de naam niet helemaal duidelijk is. Als we nu nog eens kijken bij de codeblokken, onder ‘mijn blokken’, dan is er dus een functie bijgekomen, die heet ‘teken lijn’ en die heeft zes parameters. Deze kun je vervolgens gebruiken in je code als een gewoon blokje.

Ten slotte, maar dat had ik hier verder niet beloofd, kijk ik nog even terug naar de CompUfair van 28 september jl. Hier hield ik een workshop over Scratch. Zoals bekend, tijdens een workshop kunnen de deelnemers niet alleen luisteren naar wat er verteld wordt, maar wordt ook enige prestatie verwacht. Hiervoor hadden we een aantal pc’s (normaal 12, maar nu was er één stukgegaan) en men kon aan de slag met Scratch in groepjes van twee achter één pc.
De workshop begon om 11:30 uur en toen ik de zaaldeuren dicht deed was ik een beetje geschrokken van de lage opkomst, maar al snel kwamen toch wat meer mensen en we hadden toen alle beschikbare computers bemand (helaas alleen heren deze keer), en er was één koppel vader-zoon in de zaal. Daar had ik ook op gehoopt en ze deden enthousiast mee. Hopelijk volgende keer nog meer teams van verschillende generaties, want het voegt echt wat toe. Maar ook de anderen waren natuurlijk welkom en samen hebben we Scratch een beetje ontdekt. In een uurtje kun je niet heel veel zien, maar het was leuk en leerzaam.
Er komt een vervolg!