CONSTRUCT 2

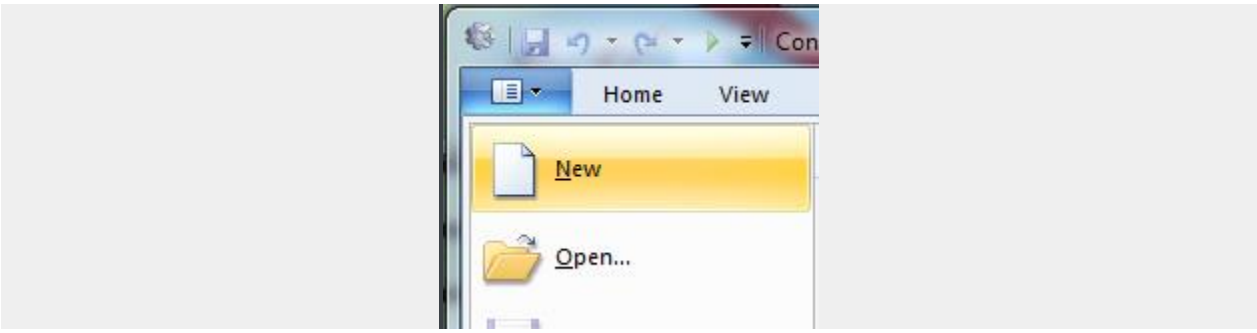
Contents

INSTALLEREN CONSTRUCT 2

De Construct 2 editor is alleen voor Windows beschikbaar, maar de games die je maakt kunnen overal op draaien, zoals Mac, Linux of iPad. Construct 2 kan ook op gelimiteerde gebruikersaccounts worden geïnstalleerd. Het is ook portable, zodat je het bijvoorbeeld ook kunt installeren op een USB-stick, zodat je het kunt meenemen!

AAN DE SLAG

Nu je alles hebt ingesteld, start Construct 2. Klik op de *File* -knop, en kies *New* .



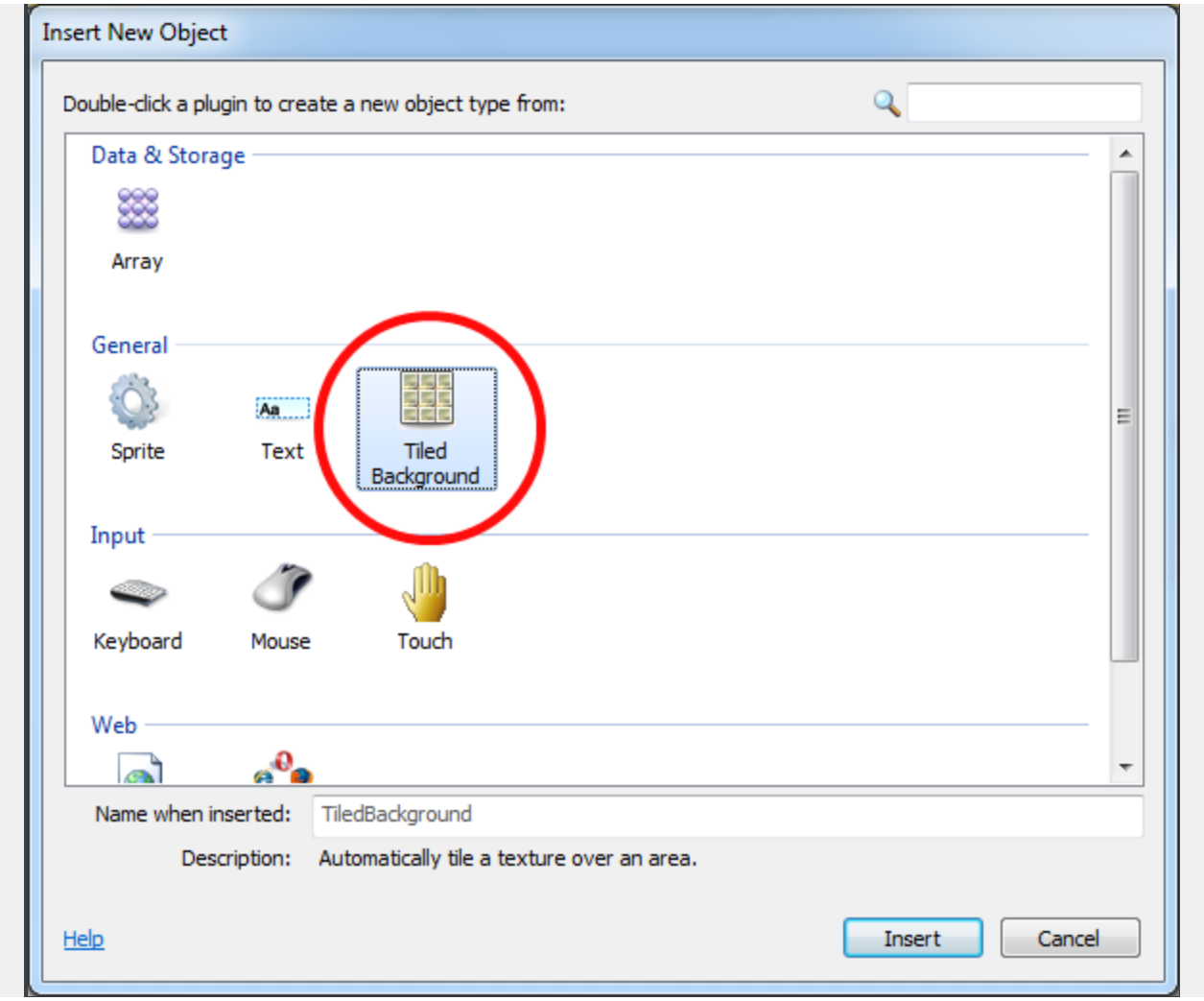
In het *New Project* dialoogvenster, hoeft je niets te veranderen. Klik op *Create project*. Construct 2 zal het gehele project in een enkele ***.capx*** file opslaan voor ons. Je zou nu een lege *lay-out* moeten zien - de lay-out is een ontwerp weergave waarin je objecten kunt maken en plaatsen. Denk aan een lay-out als aan een spelniveau of menu scherm. In andere software, kan dit ook wel een *room*, *scene* of *frame* worden genoemd.

INVOEGEN VAN OBJECTEN

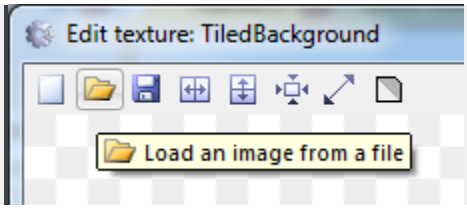
BETEGELDE ACHTERGROND

Het eerste wat we willen is een herhalende achtergrond tegel. Het *Tiled Achtergrond* object kan dit voor ons doen. Allereerst is hier je achtergrond textuur - klik met de rechtermuisknop en bewaar het ergens op je computer (wel onthouden, je hebt deze straks weer nodig):

Dubbelklik nu ergens in het lay-out venster om een nieuw object in te voegen. (Later, als het gevuld is, kunt je het ook met de rechtermuisknop openen en kiezen voor *Insert new object*.) Zodra de *Insert new object* dialoogvenster verschijnt, dubbelklik op het Tiled Background object om deze te plaatsen.

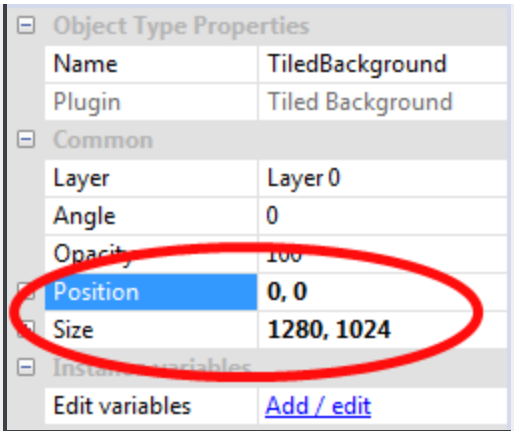


Er verschijnt een vizier om aan te geven waar je het object kunt plaatsen. Klik ergens in de buurt van het midden van de lay-out. De *textuur editor* opent om de textuur voor de tegels in te voeren. Laten we de afbeelding importeren die we eerder hebben opgeslagen. Klik op het map-pictogram om een textuur te openen, je kunt het vinden waar je het bestand hebt opgeslagen bij het downloaden, en selecteer de textuur.

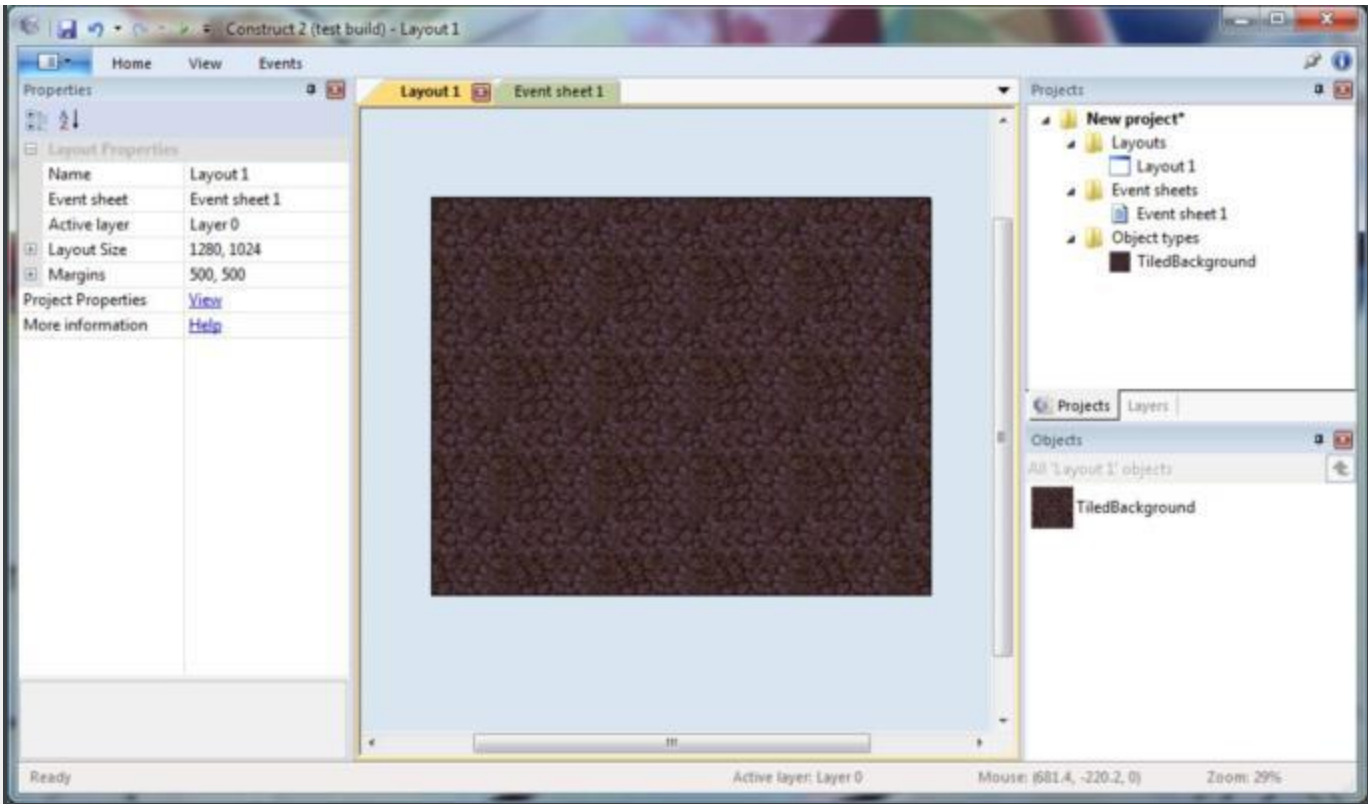


Sluit de textuureditor door te klikken op de X in de rechterbovenhoek. Als er om wordt gevraagd, zorg ervoor dat je eerst opslaat! Nu zou je je betegelde achtergrond object

moeten zien in de lay-out. Laten we het formaat wijzigen om de gehele lay-out te bedekken. Zorg ervoor dat het geselecteerd is, dan zou de ***Properties*** *Bar* aan de linker kant alle instellingen voor het object moeten laten zien, waaronder de grootte en positie. Stel de positie in op 0, 0 (linksboven in de lay-out), en de grootte 1280, 1024 (de grootte van de lay-out).



Laten we ons werk eens bekijken. Houdt control vast en scroll het muiswiel naar beneden om uit te zoomen. Of klik *view - zoom out* een paar keer. Je kunt ook de middelste muisknop, of spatie ingedrukt houden, om rond te kijken. Netjes hè? Je betegelde achtergrond bedekt de gehele lay-out nu:



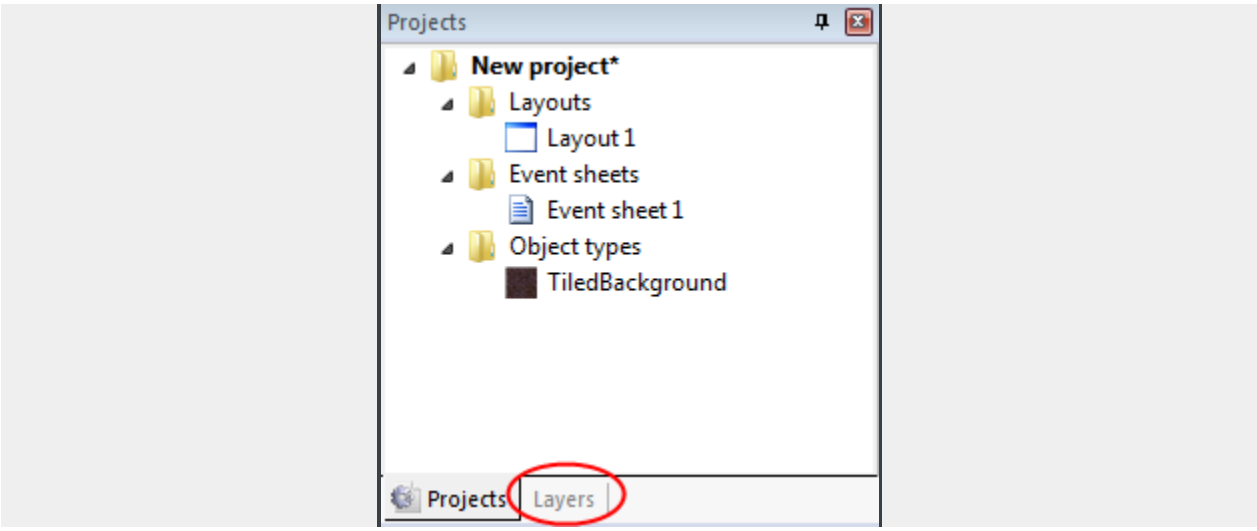
Via control + 0 of klik op *view* - *zoom tot 100%* om terug te keren naar de 1:1 view.

EEN LAAG TOEVOEGEN

Oké, nu willen we wat meer objecten toevoegen. Echter, gaan we zullen per ongeluk steeds de betegelde achtergrond selecteren, tenzij we achtergrond *op slot* doen, waardoor deze niet meer selecteerbaar is. Laten we gebruik maken van het lagen systeem om dit te doen.

Lay-outs kunnen bestaan uit meerdere *lagen*, die je kunt gebruiken voor een groep objecten. Stel lagen voor als glasplaten gestapeld bovenop elkaar met voorwerpen geschilderd op elke glasplaat. Hiermee kun je eenvoudig regelen welke voorwerpen op anderen lijken, en lagen kunnen worden verborgen, vergrendeld, hebben parallax effecten toegepast, en nog veel meer. Bijvoorbeeld, in dit spel willen we alles boven de betegelde achtergrond hebben, dus wij kunnen een nieuwe laag bovenop maken voor onze andere objecten.

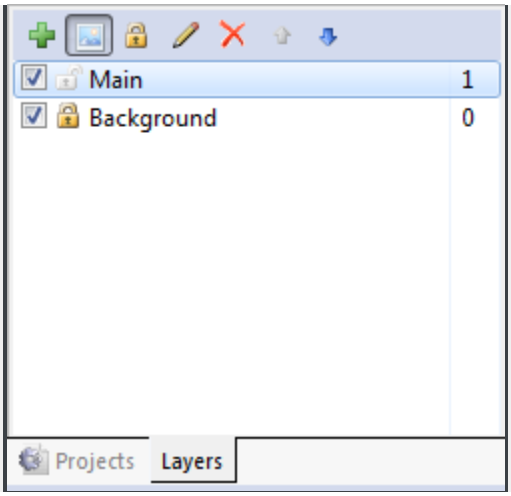
Om lagen te beheren, klikt je op het tabblad Lagen, die meestal naast de *Project bar* staat:



Je zou nu *Layer 0* in de lijst moeten zien (Construct 2 telt vanaf nul, omdat dat bij programmeren beter werkt). Klik op het potlood pictogram en hernoem het naar *Achtergrond*, want dit is onze achtergrondlaag. Klik nu op de groene 'add' icoon om een nieuwe laag toe te voegen voor onze andere objecten. Laten we deze laag *Main* noemen. Tot slot, als je de sloticon klikt naast *Background*, wordt

het *gesloten*. Dat betekent dat je niet in staat bent om iets te selecteren op die layer. Dat is heel handig voor onze betegelde achtergrond, omdat je het gemakkelijk per ongeluk kunt selecteren en dat is nu niet meer nodig. Echter, als je nodig hebt om wijzigingen aan te brengen, kunt je gewoon op het hangslot te klikken om hem weer te ontgrendelen.

Met de selectievakjes kunt je ook de lagen in de editor verbergen, maar wij hebben dat niet nodig op dit moment. Je lagen bar ziet er nu als volgt uit:



Zorg ervoor dat de 'Main' laag is geselecteerd in de lagen bar. Dit is belangrijk - de geselecteerde laag is de actieve laag*. Alle nieuwe ingevoegde objecten worden ingevoegd in de actieve* laag, dus als het niet geselecteerd is, zullen we het per ongeluk in de verkeerde laag invoegen. De actieve laag wordt weergegeven in de statusbalk, en verschijnt ook in een tooltip bij het plaatsen van een nieuw object - het is de moeite waard om dat in de gaten te houden.

VOEG DE INPUT OBJECTEN TOE

Richt je weer op de lay-out. Dubbelklik om een nieuw object in te voegen. Dit keer selecteert je het Mouse object, omdat we de muis-input nodig hebben. Doet hetzelfde voor het Keyboard object.

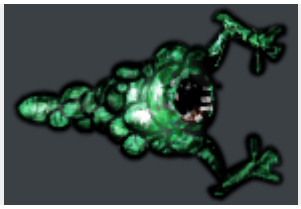
Opmerking: deze objecten hoeven niet in een lay-out geplaatst te worden. Ze zijn verborgen, en werken automatisch door je gehele project heen. Nu zijn alle lay-outs in ons project voorzien van muis en toetsenbord acceptatie.

DE GAME OBJECTEN

Het is tijd om aan ons spel objecten toe te voegen! Hier zijn je textures - sla ze allemaal op de harde schijf, zoals eerder.

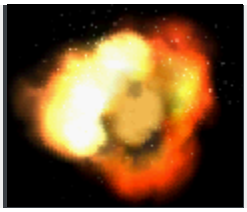
Speler:

Monster:



Kogel:

and Explosie:



Voor elk van deze voorwerpen, zullen we gebruik maken van een *Sprite* object. Het is gewoon een textuur, die je kunt bewegen, draaien en vergroten of verkleinen. Games zijn over het algemeen samengesteld uit sprite objecten. Laten elk van de vier

bovengenoemde objecten toevoegen als sprite object. Het proces is vergelijkbaar met het plaatsen van de getegelde Achtergrond:

1. Dubbelklik om een nieuw object in te voegen

2. Dubbelklik het 'Sprite' object.

3. Wanneer de muis verandert in een crosshair, klik je ergens in de lay-out. De tooltip moet 'Main' zijn. (Onthoudt dat dit de actieve layout is.)

4 De textuur editor verschijnt. Klik op het pictogram Openen en laadt een van de vier patronen.)

5. Sluit de textuur editor, waarmee je je wijzigingen opslaat. Je ziet nu het object in de lay-out!

Let op: een andere snelle manier om sprite objecten te voegen is met slepen en neerzetten van het afbeelding-bestand vanuit Windows naar het lay-out gebied. Construct 2 zal een Sprite met die textuur voor je maken. Zorg ervoor dat je elke afbeelding een voor een naar het layout scherm sleept. Als je alle vier de afbeeldingen in een keer sleept, zal Construct 2 een Sprite maken met vier animatie frames.

Beweeg de *kogel* en *explosie* sprites ergens buiten de rand van de layout – we willen ze nog niet zien als het spel begint.

Deze objecten zullen *Sprite, Sprite2, Sprite3* en *Sprite4* worden genoemd. Dat is niet erg handig - dit wordt snel verwarrend. Hernoemen ze
naar *Speler, Monster, Kogel* en *Explosie* afhankelijk van de sprite. Je kunt dit doen door het object te selecteren, dan kun je de naam in de eigenschappen bar veranderen:

HET TOEVOEGEN VAN GEDRAG

Gedraging is voorverpakte functionaliteit in Construct 2. Bijvoorbeeld, je kunt een *Platform* gedrag toe voegen aan een object, en het *Solid* gedrag op de grond, en je kunt direct rondspringen als in een platformer. Je kunt hetzelfde in de gebeurtenissen doen, maar dat duurt langer, en het is niet echt nodig als het gedrag al goed genoeg is! Dus laten we eens kijken naar welk gedrag we kunnen gebruiken. Construct 2 heeft onder anderen deze gedragingen;

8 Direction movement . Hiermee kun je een object bewegen met de pijltjestoetsen. Dit zal goed zijn voor de beweging van de speler.

Bullet movement . Deze beweegt een object vooruit vanuit zijn huidige hoek. Dit gaat geweldig zijn voor de kogels van de speler. Ondanks de naam, zal het ook mooi werken om de monsters te bewegen - omdat deze beweging alleen de Objecten naar voren verplaatst met een bepaalde snelheid.

Scroll to. Dit zorgt dat het beeldscherm rond een object beweegt (ook bekend als *scrollen)*. Dit gaat nuttig zijn voor de speler.

Bound to layout. Dit stopt een object bij het verlaten van de lay-out gebied. Dit is ook nuttig voor de speler, zodat die niet kan afdwalen buiten het speelveld!

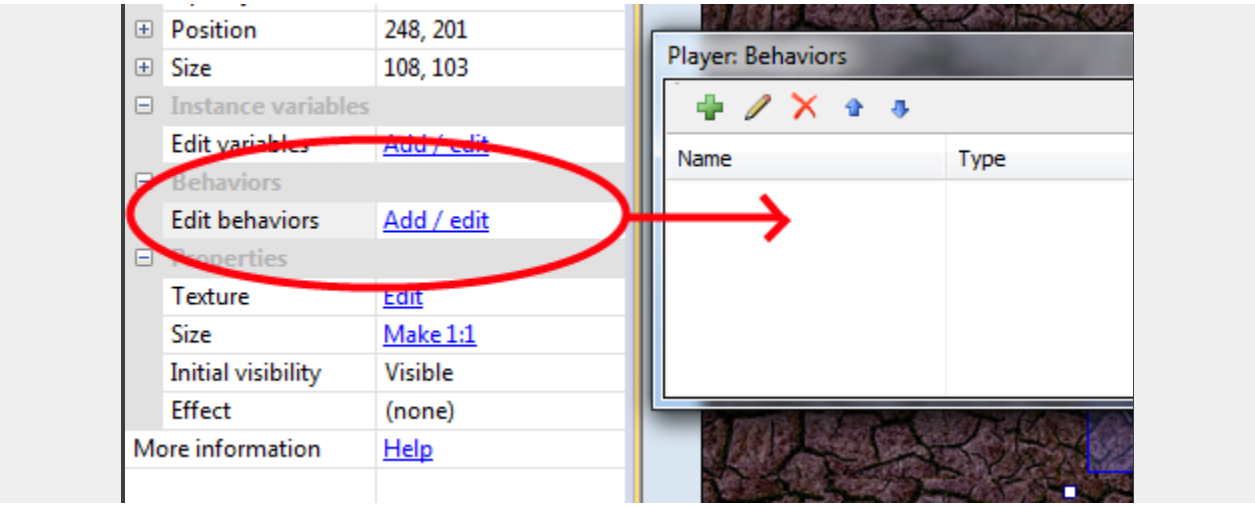
Destroy outside lay-out. In plaats van het stoppen van een object bij verlaten van de layout gebied, wordt het object vernietigd. Het is nuttig voor onze kogels. Zonder dat, zou de kogels steeds op het scherm blijven vliegen, dat kost telkens een beetje geheugen en rekenkracht. In plaats daarvan moeten we de kogels vernietigen zodra ze de layout verlaten.

Fade. Dit maakt dat een object geleidelijk uitdooft, wat we zullen gebruiken voor de explosies.

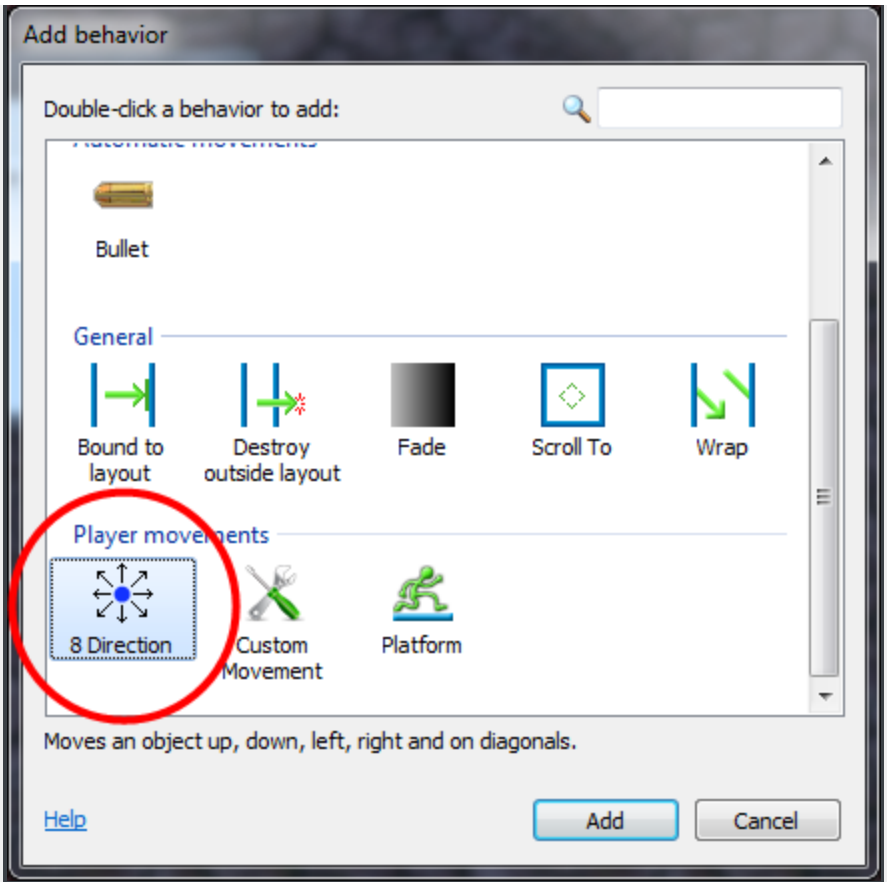
Voeg deze gedragingen toe aan de objecten die ze nodig hebben.

HOE VOEG JE GEDRAG TOE

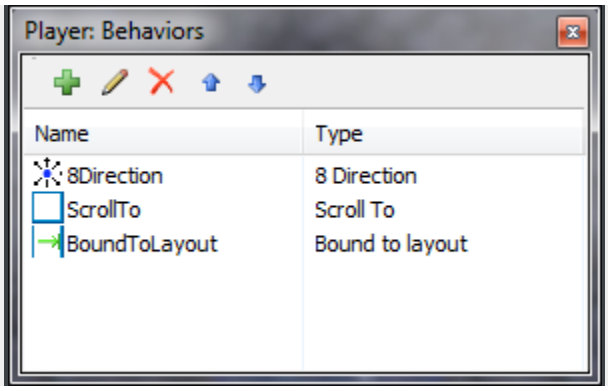
We voegen het 8 direction movement gedrag aan de speler toe. Klik op de speler om deze te selecteren. In de eigenschappen bar, let op de *Behaviors* categorie. Klik daar op *Add / Edit* . Het gedrag dialoogvenster voor de speler zal openen.



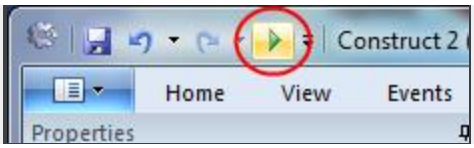
Klik op de groene icoon 'add behavior' in het behaviors dialoogvenster. Dubbelklik op 8 direction movements om het toe te voegen.



Doe hetzelfde nogmaals en voeg ditmaal het Scroll To gedrag toe om het beeldscherm de speler te laten volgen, en ook het Bound to layout gedrag om de speler in de layout te houden. Het dialoogvenster gedragingen ziet er nu als volgt uit:



Sluit het dialoogvenster gedrag. Klik op Run om het spel te proberen! *Sla het programma eerst op*



Hopelijk heb je een HTML5 compatibele browser geïnstalleerd. Anders moet je de laatste versie van Firefox of Chrome of Internet Explorer installeren als je op Vista en hoger werkt. Zodra u het spel loopt, merk je dat je rond kunt bewegen met de pijltjestoetsen, en het scherm volgt de speler! Je kunt ook niet buiten de layout ruimte, dankzij het Bound to lay-gedrag. Dit is waar het gedrag goed voor is. Het snel toevoegen van stukjes gedrags functionaliteit. We zullen het evenement systeem binnenkort gebruiken om aangepaste functionaliteit toe te voegen.

HET TOEVOEGEN VAN DE ANDERE GEDRAGINGEN

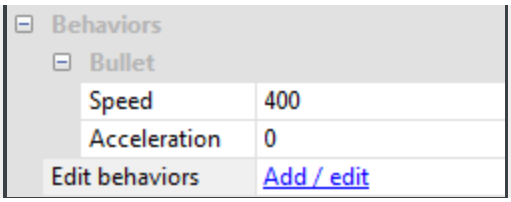
We kunnen het gedrag aan de andere objecten op dezelfde manier toevoegen - selecteer het, klik op *Add / Edit* om het dialoogvenster gedrag te openen, en voeg het gedrag toe. Laten we die andere gedragingen toevoegen:

Voeg Bullet movement en Destroy outside layout aan het Kogel object toe (geen verrassingen hier)

Voeg Bullet movement toe aan het Monster object (omdat het monster een kant op beweegt net als de bullet)

Voeg het Fade gedrag aan het Explosie object toe (dus het verdwijnt geleidelijk na het verschijnen). Standaard vernietigd het Fade gedrag ook het object nadat het is vervaagd, dat bespaart ons zorgen over onzichtbare Explosie voorwerpen waardoor het spel vastloopt.

Als je het spel start, kun je merken dat het enige wat anders is, dat elk monster dat je kunt zien plotseling vrij snel weg schiet. Laten we het vertragen naar een lagere snelheid. Selecteer het Monster object. Omdat we een gedrag hebben toegevoegd, krijgen we extra eigenschappen die zijn verschenen in de eigenschappen bar:



Dit stelt ons in staat om te kijken hoe gedragingen werken. Verander de snelheid van 400 naar 80 (dit is in afgelegde pixels per seconde).

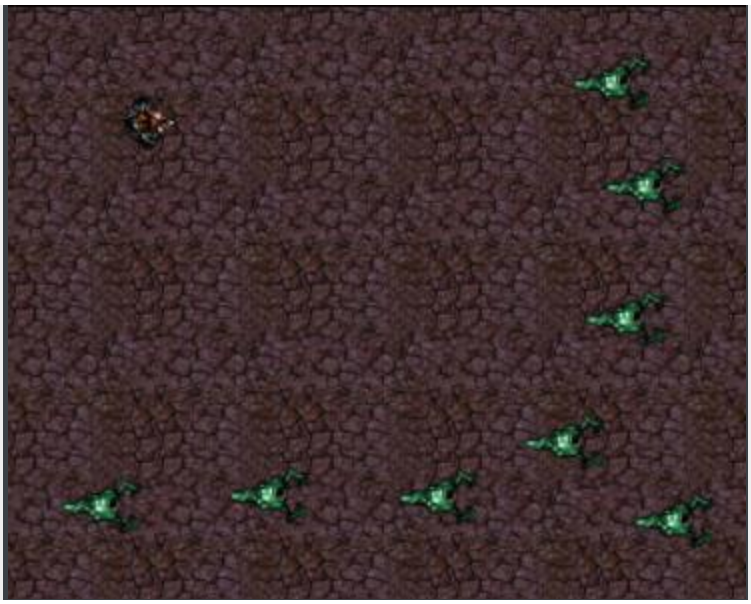
Verander ook de Bullet object snelheid naar 600, en de Explosie *Fade out tijd* naar 0.5 (dat is een halve seconde).

MAAK WAT MEER MONSTERS

Terwijl je control vasthoudt, klik en sleep gelijktijdig het Monster object. Je zult zien dat er andere monster object *instanties* bijkomen. Dit is gewoon een ander object van het Monster *type object.*

Objecttypen zijn in wezen 'klassen' van objecten. In het geval van het event systeem, hou je je vooral bezig met objecttypen. Je kunt bijvoorbeeld een event maken dat zegt: "Kogel botst met Monster". Dit betekent eigenlijk " *Elke instantie van het* Kogel *object type* botst met *een instantie van de* Monster *object type* " - in tegenstelling tot het hebben van een apart event voor elk monster. Met Sprites, delen alle exemplaren van een objecttype ook dezelfde textuur. Dit is geweldig voor de efficiëntie - als spelers je spel online spelen, hoeven ze in plaats van 8 monster texturen voor 8 monsters, slechts een enkel monster textuur te downloaden en Construct 2 herhaalt het 8 keer. We kijken later naar *objecttypen* VS *instanties*. Voor nu, een goede manier om erover na te denken is: verschillende soorten vijanden zijn verschillende typen objecten, de daadwerkelijke vijand (waarvan er meerdere van kunnen zijn) is een instantie van dit type object.

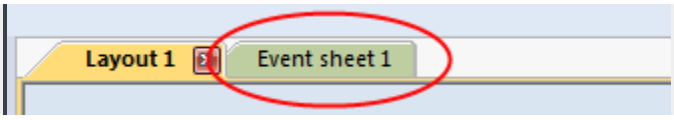
Met behulp van control + slepen, maak 7 of 8 nieuwe monsters. Plaats ze niet te dicht bij de speler, want dan kan die meteen sterven! Als het helpt kun je uitzoomen met control + muiswiel naar beneden, en verdeel ze over de hele layout. Je moet eindigen met iets wat een beetje lijkt op dit.



Nu is de tijd om onze aangepaste functionaliteit via de visuele manier van programmeren van Construct 2 toe te voegen - het *event systeem*.

EVENTS

Klik eerst op de *Event sheet 1* tab aan de bovenkant om over te schakelen naar de *Event sheet editor*. Een lijst van de gebeurtenissen noemen we een *Event sheet*, en je kunt verschillende event sheets hebben voor de verschillende onderdelen van je spel of voor de organisatie. Event sheets kunnen ook andere event sheets bevatten, zodat je gebeurtenissen op meerdere niveaus kunt hergebruiken, maar we zullen dat niet nodig hebben op dit moment.



OVER EVENTS

Zoals de tekst in het lege blad aangeeft, doorloopt Construct 2 alles in het sheet vel eenmaal per **tick**. De meeste schermen werken hun scherm 60 keer per seconde bij, daarom probeert Construct 2 dat bij te houden voor de meest vloeiende weergave. Dit betekent dat het event sheet meestal 60 keer per seconde wordt doorlopen, telkens gevolgd door opnieuw op het scherm te tekenen. Dat is wat een **tick** is - *een eenheid van "doorloop de events en teken vervolgens op het scherm"*.

Events lopen van boven naar beneden, zodat gebeurtenissen bovenaan de event pagina eerst worden uitgevoerd.

VOORWAARDEN, ACTIES EN SUB-EVENTS

Gebeurtenissen (**events**) bestaan uit condities, die testen of er wordt voldaan aan bepaalde criteria, bijvoorbeeld "Is spatiebalk ingedrukt?". Als aan al deze condities is voldaan, worden de acties van het event doorlopen, bijvoorbeeld "Maak een kogel object". Nadat de acties zijn uitgevoerd, worden ook sub-events doorlopen - deze kunnen dan meer condities testen, en meer acties doorlopen, dan meer sub-events, en ga zo maar door. Met behulp van dit systeem kunnen we de geavanceerde functionaliteit opbouwen voor onze games en apps. We hebben de sub-events niet nodig in deze tutorial.

Laten we nog een keertje kijken. In het kort, een evenement loopt in principe zoals dit:

Zijn alle condities voldaan?

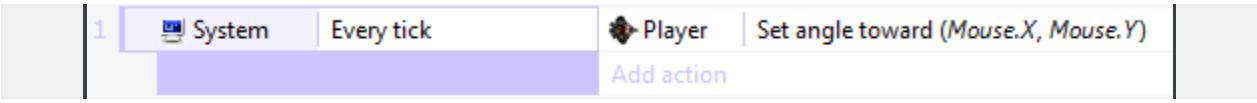
---> Ja: alle acties van het evenement doorlopen.

---> Nee: ga naar de volgende event (met uitzondering van eventuele sub-events).

Dat is een beetje een oversimplificatie. Construct 2 biedt veel evenement functionaliteit om veel verschillende dingen te kunnen die je zou willen doen. Maar voor nu is dit een goede manier om over na te denken.

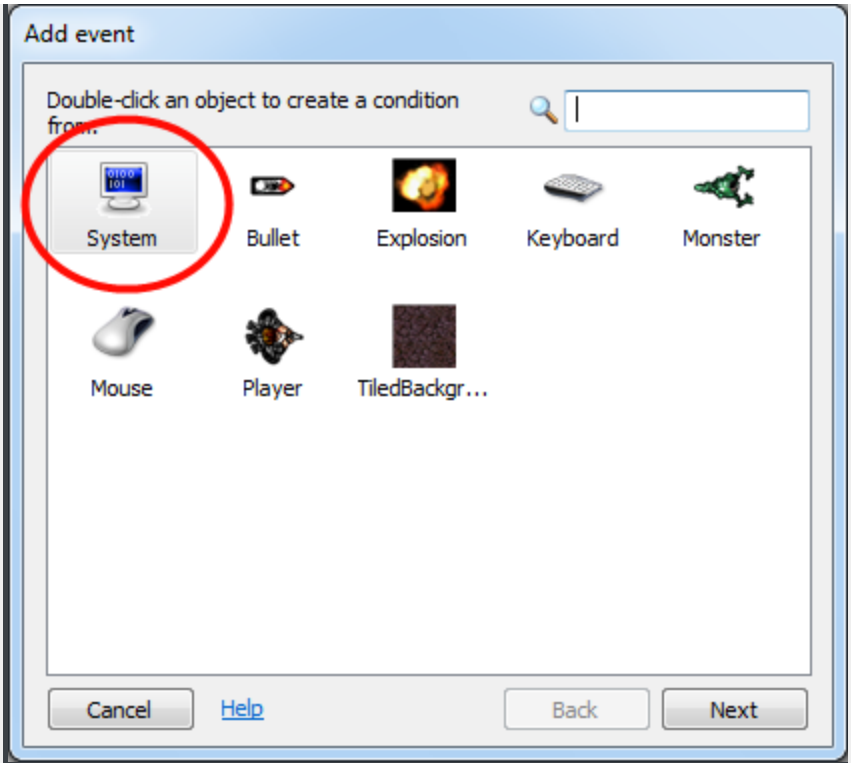
MIJN EERSTE EVENT

We willen een speler maken die altijd naar de muis kijkt. Het ziet er zo uit als we klaar zijn:

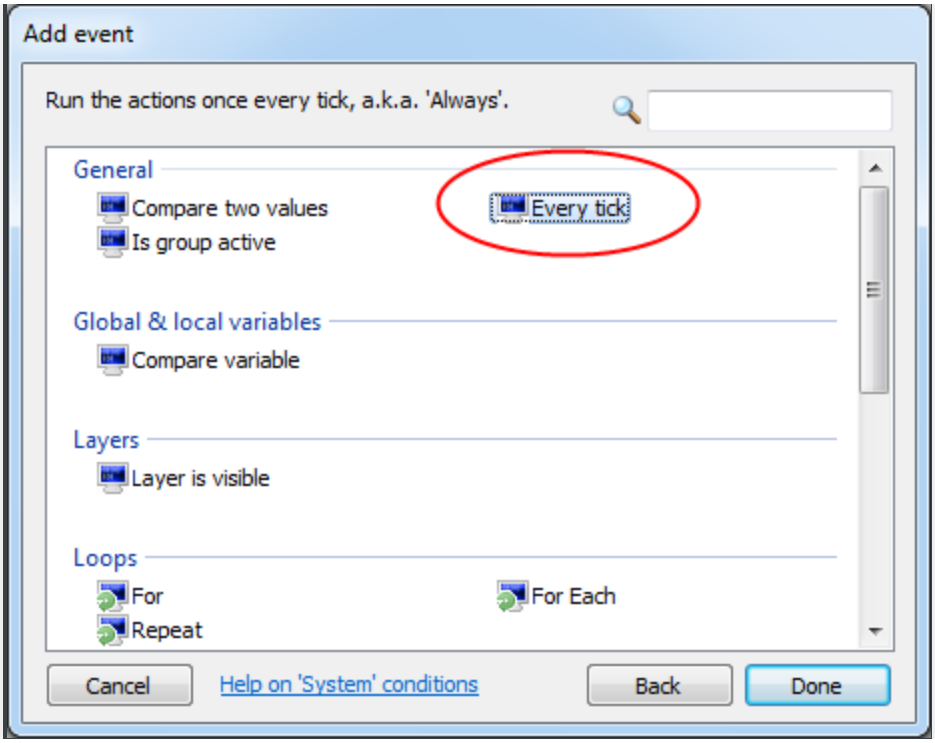


Onthoud dat een tick elke keer loopt wanneer er op het scherm wordt getekend, dus als we de speler elke tick naar de muis laten kijken, zal het lijken alsof deze altijd in richting van de muis kijkt.

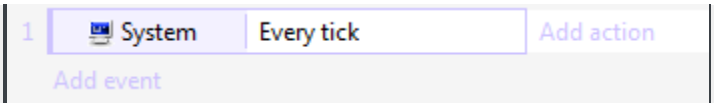
Laten we beginnen met het maken van dit event. Dubbelklik op een ruimte in het event sheet. Dit zal ons vragen om een conditie toe te voegen voor het nieuwe event.



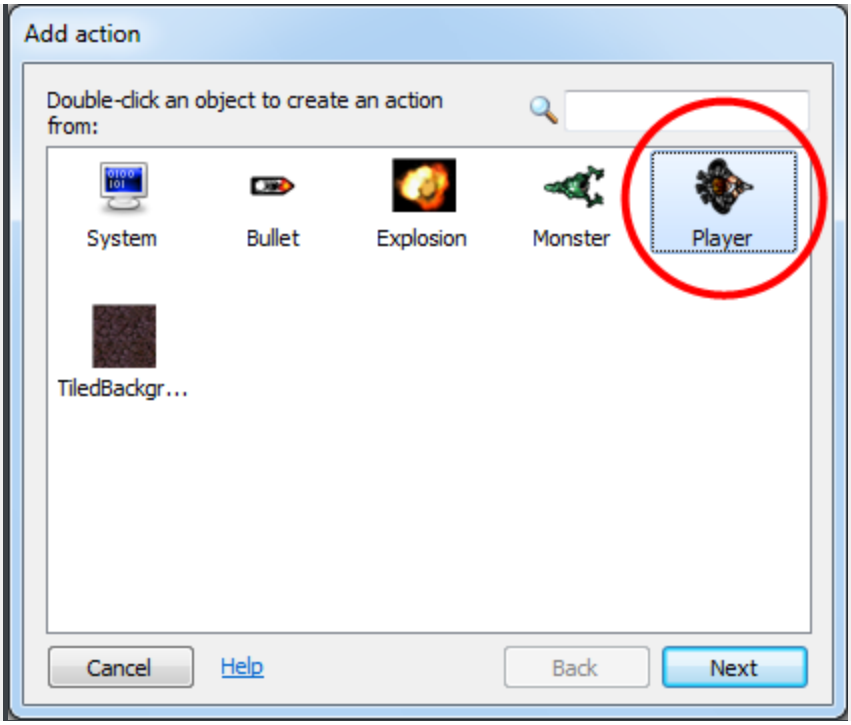
Verschillende objecten hebben verschillende condities en acties, afhankelijk van wat ze kunnen doen. Er is ook het System object, wat op de ingebouwde functionaliteit van Construct 2 vertegenwoordigd. Dubbelklik op het System object zoals afgebeeld. Het dialoogvenster zal dan een lijst van alle condities van het system object laten zien:



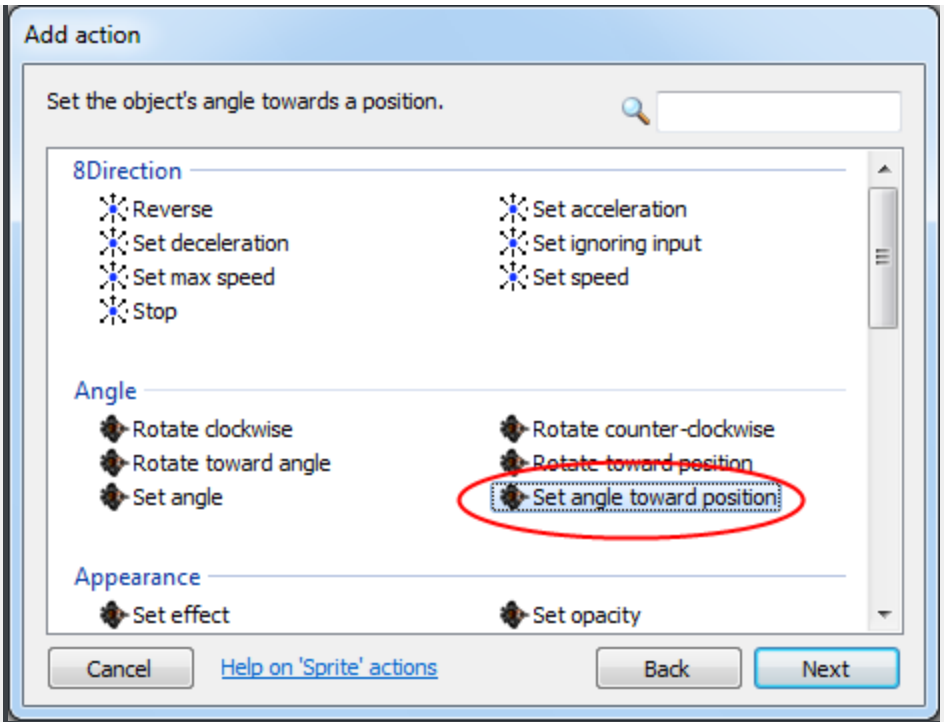
Dubbelklik op de *Every tick* conditie om deze in te voegen. Het dialoogvenster wordt gesloten en de gebeurtenis wordt gemaakt, zonder acties. Het ziet er nu als volgt uit:



Nu willen we een actie toevoegen zodat de speler naar de muis kijkt. Klik op de *Add action* link aan de rechterkant van het evenement. (Zorg ervoor dat je de Add *action* link kiest, niet Add *event* link eronder die een heel ander evenement zal toevoegen.) Het dialoogvenster Add Action zal verschijnen:



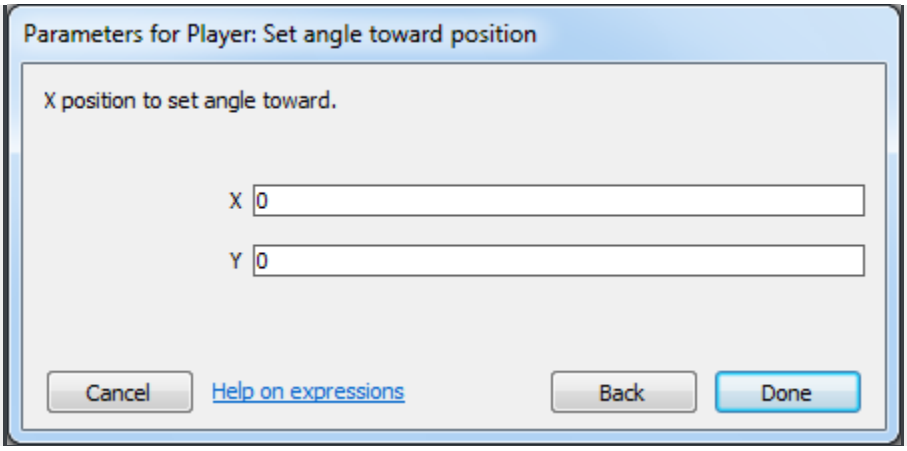
Net als bij het toevoegen van een event, we hebben dezelfde lijst van objecten om uit te kiezen, maar dit keer voor het toevoegen van een *actie* . Probeer niet in de war raken met het toevoegen van condities en het toevoegen van acties! Zoals afgebeeld, dubbelklik het *-speler* object, want het is de speler die we naar de muis willen laten kijken. De lijst met acties die beschikbaar zijn in de Speler-object verschijnt:



Merk op dat de 8-direction beweging van de speler zijn eigen actie heeft. We hoeven ons daarvoor nu geen zorgen over te maken.

In plaats van de hoek van de speler in graden te bepalen, is het handig om de Set angle toward position actie te gebruiken. Deze berekent automatisch de hoek van de speler naar het opgegeven X en Y-coördinaat en stelt de hoek van het object in. Dubbelklik op de *Set angle toward position* actie.

Construct 2 moet nu de X-en Y-coördinaat weten voor de richting van de speler:

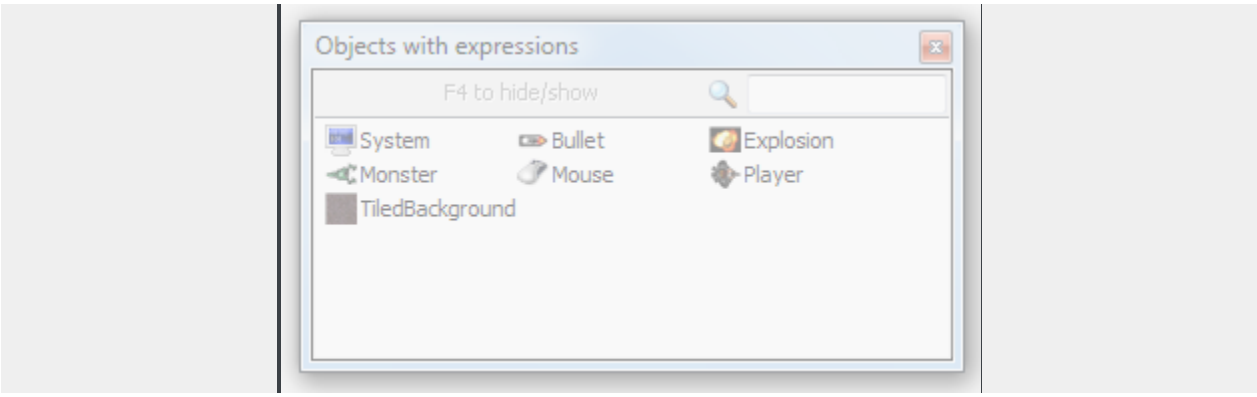


Dit zijn de zogenaamde parameters van de actie. Condities kunnen ook parameters hebben, maar *Every tick* heeft er geen nodig.

We willen de hoek naar positie van de muis in stellen. Het Mouse object kan hierin voorzien. Voer Mouse.X voor *X* en Mouse.Y voor *Y* in. Dit zijn de zogenaamde ***expressies***. Ze zijn als sommen die worden berekend. Bijvoorbeeld, je kon ook invoeren *Mouse.X + 100* of *sin (Mouse.Y)* (hoewel deze specifieke voorbeelden niet erg nuttig zou zijn!). Op deze manier kun je alle gegevens van een voorwerp, of een berekening gebruiken, om parameters te bepalen in acties en condities. Het is zeer krachtig, en een soort van verborgen bron van veel van Construct 2's flexibiliteit.

*Hebt je een **foutmelding** die zegt: "mouse is not an object name" Zorg er dan voor dat je de muis object toevoegt! Ga terug naar pagina 2 en controleer dit bij "Add the input objects".*

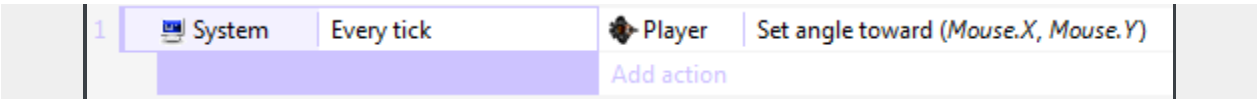
Je vraagt je misschien af hoe je alle mogelijke expressies zou moeten herinneren die je zou kunnen gaan gebruiken. Gelukkig is er het "object panel" dat je erboven zou moeten zien zweven. Standaard is het vervaagd, zodat het je niet afleidt.



Ga er met de muis overheen, of klik erop, en het zal volledig zichtbaar worden. Dit dient als een soort woordenboek van alle uitdrukkingen die je kunt gebruiken, met beschrijvingen, en om je te helpen herinneren. Als je op een object dubbelklikt, zie je al haar expressies vermeld. Als je dubbelklikt op een expressie, zal het ook

automatisch voor je ingevoegd worden, dat bespaart je het uittypen van de expressies.

Hoe dan ook, klik Done in het parameters dialoogvenster. De actie wordt toegevoegd! Zoals je al eerder zag, zou het er zo uit moeten zien:



Daar heb je je eerste actie! Probeer het spel *wel eerst even opslaan*, de speler moet nu in staat zijn om te bewegen zoals voorheen, maar nu met het gezicht altijd gericht naar de muis. Dit is onze eerste stukje functionaliteit op maat.

SPEL FUNCTIONALITEIT TOEVOEGEN

Als elke gebeurtenis net zo gedetailleerd wordt beschreven als hiervoor, wordt dit een heel lang tutorial. Laten we de beschrijving een beetje inkorten voor de volgende gebeurtenissen. Vergeet niet, de stappen om een conditie of actie toe te voegen zijn:

1. Dubbelklik om een nieuwe gebeurtenis in te voegen, of klik een *Add action* link om een actie toe te voegen.

2. Dubbelklik op het object waarin de conditie/gebeurtenis zich bevindt.

3. Dubbelklik op de conditie/gebeurtenis die je wilt.

4. Voer parameters in, als dat nodig is.

Vanaf nu aan, zullen gebeurtenissen worden beschreven als het object, gevolgd door de conditie / actie, gevolgd door de parameters. Zo zou de gebeurtenis die we zojuist hebben ingevoerd worden geschreven:

Voeg de conditie toe: *System -> Every tick*

Voeg de actie toe: *Player -> Set angle towards position* -> X: *Mouse.X*, Y: *Mouse.Y*

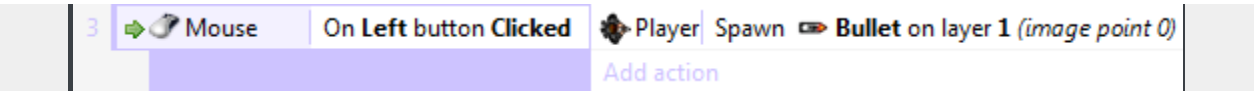
ZORG DAT DE SPELER GAAT SCHIETEN

Wanneer de speler klikt, moet er een kogel schieten. Dit kan gedaan worden met de *Spawn a object* actie in Speler, die een nieuwe instantie van een object op dezelfde positie en in de hoek creëert. De *Bullet movement* die we eerder toegevoegd hebbem zorgt dat deze naar voren vliegt. Maak de volgende gebeurtenis:

Conditie: *Mouse -> On click* -> Left clicked (default)

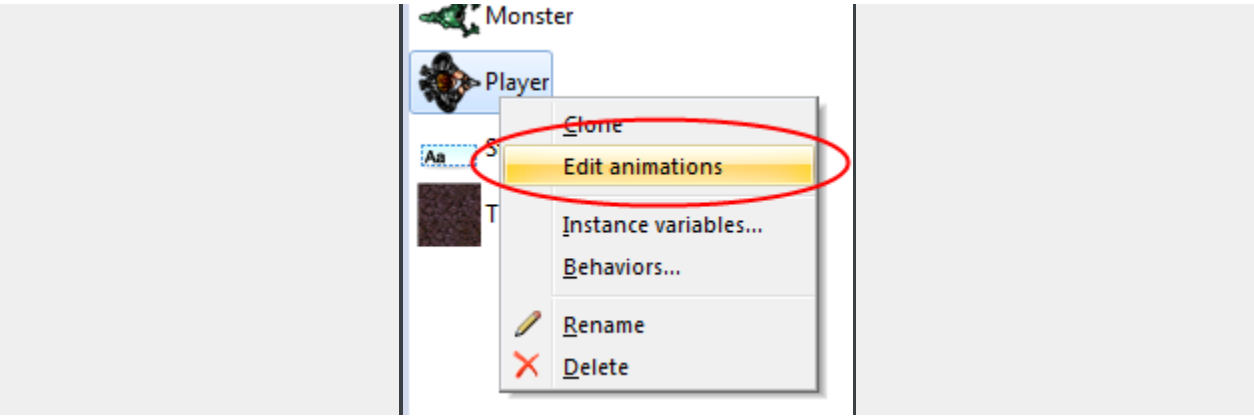
Actie: *Player -> Spawn another object* -> Kies voor *Object*, kies het *Kogel* object. Bij *Layer*, voeg 1 in (de "Main" laag is laag 1 - onthoudt Construct 2 telt vanaf 0). Laat *Image point* 0.

Je gebeurtenis ziet er nu als volgt uit:

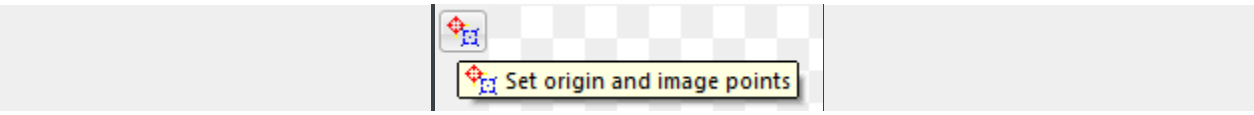


Als je het spel draait, zult je merken dat de kogels vanuit het midden van de speler schieten, in plaats van het einde van het pistool. Laten we dat aanpassen door het plaatsen van een Image point aan het eind van het pistool. (Een image point is gewoon een plek op een plaatje waaruit je objecten kunt spawnen.)

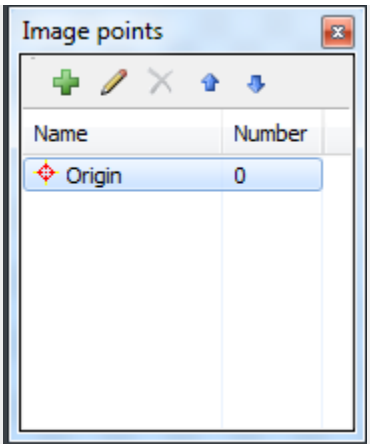
Klik met de rechtermuisknop op de speler in het project of object bar en kies Edit animations.



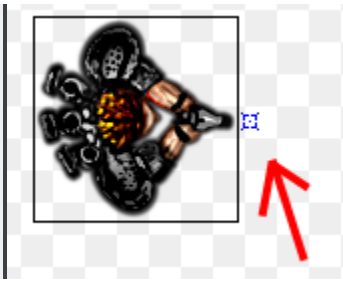
De image-editor voor de speler verschijnt weer. Klik op de origin and image points tool:



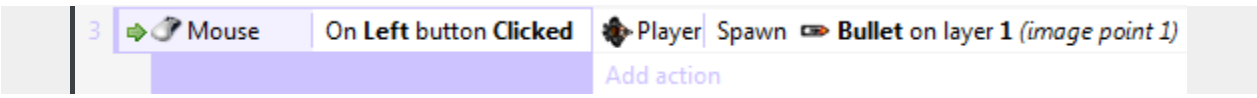
... en het dialoogvenster image points opent:



De oorsprong van het verschijnt als een rood punt. Dat is de "hotspot" of het "scharnierpunt" van het object. Als je het object draait, draait het rond de oorsprong. Wij willen een ander image point toevoegen voor het pistool, dus klik op de groene *Add* knop. Een blauw punt verschijnt - dat is ons nieuwe image point. Klik met de linkermuisknop op het einde van het pistool van de speler om het image point daar te plaatsen:



Sluit de image editor. Dubbelklik op de *Spawn an object* actie die we eerder hebben toegevoegd, en verander het *image point* naar 1 . (De oorsprong is altijd het eerste beeldpunt, en onthoud Construct 2 telt vanaf nul.) De gebeurtenis ziet er nu uit als hieronder - stel vast dat er nu *image point 1* staat:



Start het spel. *Eerst even opslaan* De kogels schieten nu uit het einde van je wapen! De kogels doen nog niets hoewel je nu zult beginnen te beseffen dat als je het eenmaal te pakken hebt met het event-systeem, dat je zeer snel functionaliteit in elkaar kunt zetten.

Laten we zorgen dat de kogels de monsters doden. Voeg de volgende gebeurtenis toe:

Conditie: *Kogel -> On collision with another object ->* kies *Monster*.

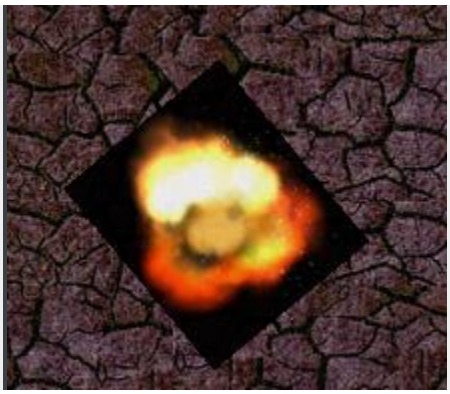
Actie: *Monster -> Destroy*

Actie: *Kogel -> Spawn another object -> Explosion*, layer 1

Actie: *Kogel -> Destroy*

HET EXPLOSIE EFFECT

Start het spel en probeer het opnemen tegen een van het monster. Oeps, de explosie heeft een grote zwarte rand!



Je had vanaf het begin misschien kunnen raden dat het er zo zou uit zou gaan zien, en kunnen vragen of ons spel echt zo zou eindigen! Maak je geen zorgen, het zal niet gebeuren. Klik op de Explosie object in de object bar in de rechteronderkant, of de Project bar (in de tabbladen bij de layers bar). De eigenschappen worden weergegeven in de properties bar aan de linkerkant. Zet aan de onderkant de Blendmode property naar Additive. Probeer het spel nu opnieuw.

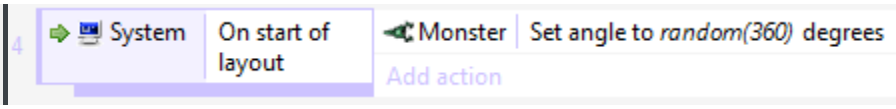
Waarom werkt dit? Zonder in te gaan op de details, zijn gewone beelden *geplakt op* het scherm. Met het additive effect, wordt elke pixel in plaats van *toegevoegd* (als in, opgeteld) bij de achtergrond pixel erachter. Zwart zijn pixels met waarde nul, zodat er niets wordt toegevoegd - je ziet de zwarte achtergrond niet. Helderdere kleuren voegen meer toe, dus die lijken sterker. Dit is geweldig voor ontploffingen en lichteffecten.

MAAK DE MONSTERS EEN BEETJE SLIMMER

Nu dwalen de monsters af naar buiten de lay-out aan de rechterkant. Laten we ze een beetje interessanter maken. Allereerst, laten we ze beginnen onder een willekeurige hoek.

Conditie: *System -> On start of Layout*

Actie: *Monster -> Set angle* -> random(360)



Ze zullen nog steeds voor altijd afdwalen als ze de lay-out verlaten, om nooit meer terug te zien. Laten we ze daar binnen houden. Wat we doen is we wijzen ze terug naar de speler als ze de lay-out te verlaten. Dit zorgt voor twee dingen: ze blijven altijd binnen de lay-out, en als de speler stilstaat, komen de monsters recht op hem af!

Conditie: *Monster -> Is outside layout*

Actie: *Monster -> Set angle toward position* -> For X, Player.X - for Y, Player.Y.

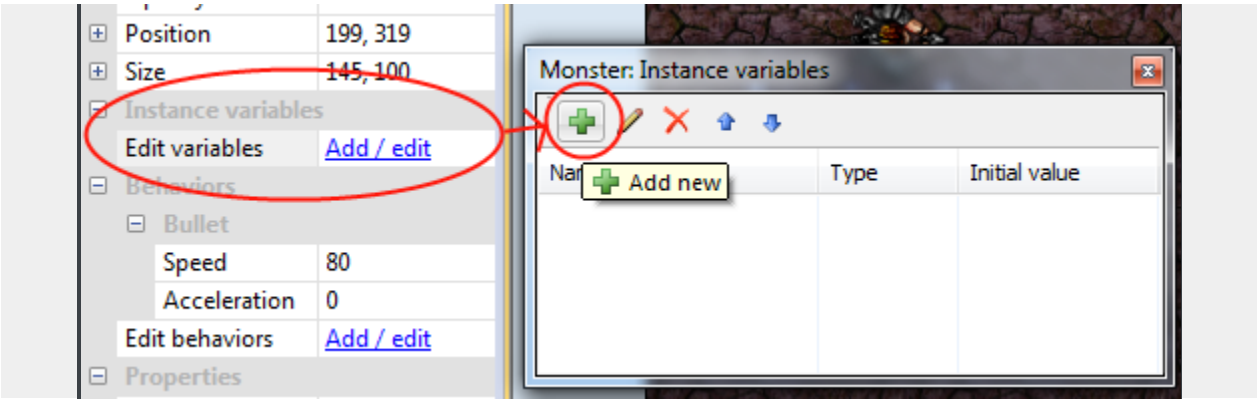
Start het spel. Als je een tijdje rond blijft hangen, zul je merken dat de monsters ook rond de lay-out blijven hangen, en ze gaan in allerlei richtingen. Het is nauwelijks AI, maar dit voldoet!

Nu, stel dat we vijf keer moeten schieten voor dat het monster sterft, in plaats van onmiddellijk dood te gaan als het wordt geraakt. Hoe gaan we dat doen? Als we maar één "Health" teller hebben en we een monster vijf keer hebben geraakt, zullen *alle* monsters sterven. In plaats daarvan moeten we voor *elk* monster zijn *eigen* gezondheid onthouden. We kunnen dat doen met instance variabelen

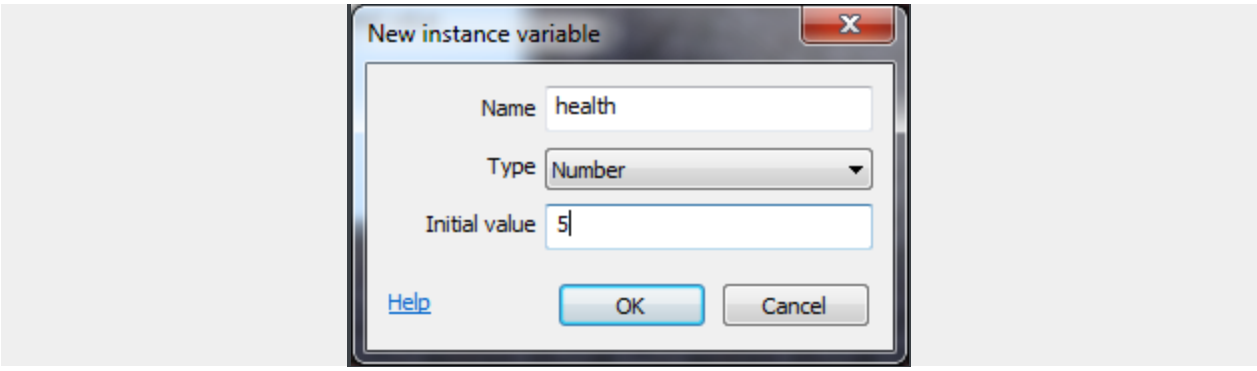
INSTANCE VARIABLEN

Instance variabelen laten je voor elk monster een eigen gezondheidswaarde opslaan. Een variabele is gewoon een waarde die kan veranderen (of *variabel zijn*), en ze worden voor elke instantie van een object afzonderlijk opgeslagen, vandaar de naam *instance variabele* .

Laten we een *health* instance variabele maken voor ons monster. Klik op het monster in de project bar of de object bar. Je kunt ook terug naar de lay-out schakelen en een monster object selecteren. Dit zal de eigenschappen van het monster in de properties bar tonen. Klik Add / edit bij Edit variables .

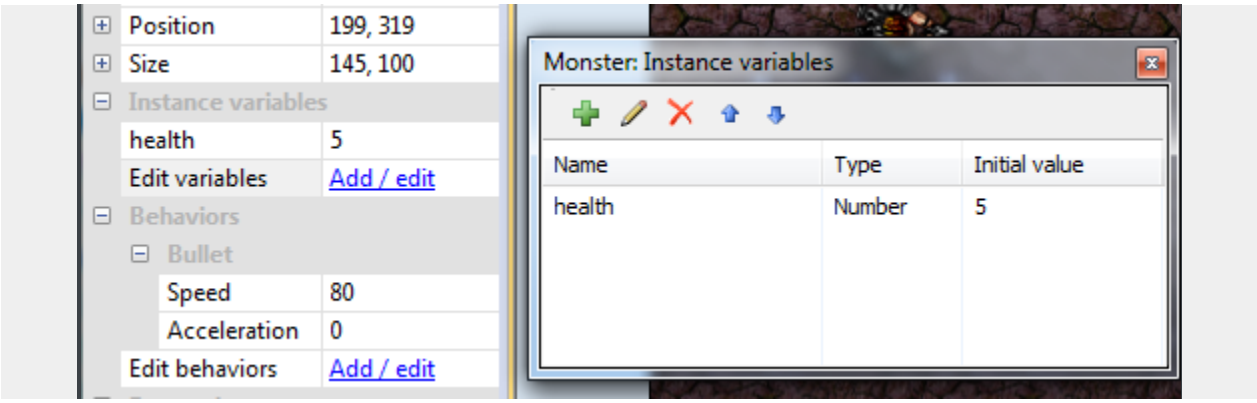


Het dialoogvenster instance Variabeles verschijnt. Het lijkt op het Behaviors dialoogvenster dat we eerder zagen, maar in plaats daarvan kun je instance variabelen toevoegen en wijzigen voor het object. Klik op de groene Add knop om een nieuwe toe te voegen.



Type in het dialoogvenster dat verschijnt health als naam, laat *Type* als Number , en voor *Initial value* vul 5 in (zoals afgebeeld). Hierdoor begint elk monster op 5 gezondheid. Wanneer ze geraakt worden zullen we 1 van de gezondheid aftrekken, en wanneer de gezondheid 0 is zullen we het object vernietigen.

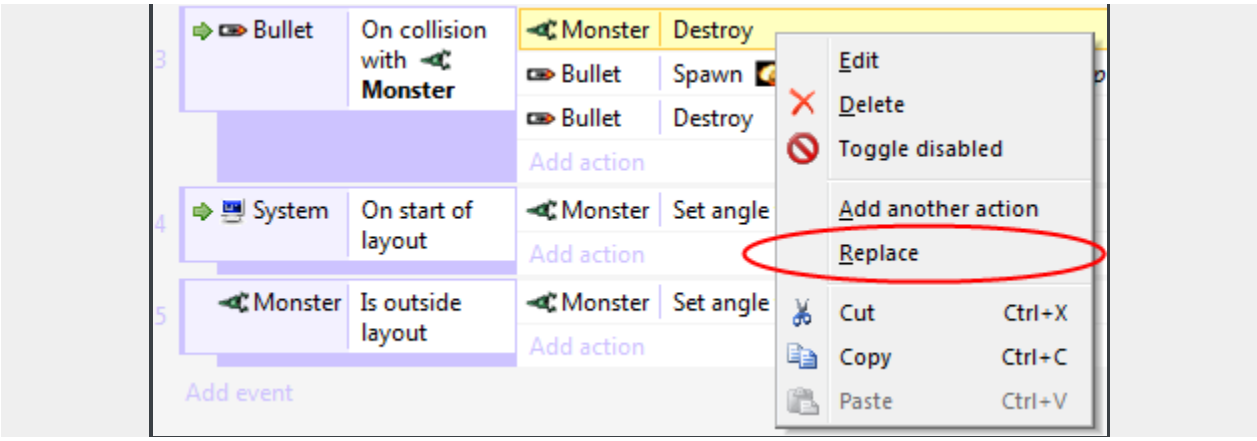
Als je klaar bent klik op OK. Merk op dat de variabele nu wordt weergegeven in het dialoogvenster instance variabeles en ook in de properties van het monster. (Je kunt beginwaarden snel veranderen in de properties bar, maar voor het toevoegen of verwijderen van variabelen moet je op de *Add / Edit* link klikken.)



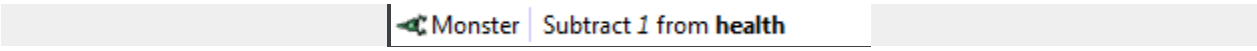
EVENTS (GEBEURTENISSEN) VERANDEREN

Ga terug naar de event pagina. Op dit moment vernietigen we monsters, zodra de kogel hen raakt. Laten we dat veranderen naar 1 aftrekken van zijn gezondheid.

Vind het event dat zegt: *Kogel - on collision with Monster* . We hebben eerder een "destroy monster" actie gemaakt. Laten we dat vervangen met "subtract 1 from health". Klik rechts op de "destroy monster" actie en klik Replace .



Hetzelfde dialoogvenster verschijnt alsof we een nieuwe actie invoegen, maar deze keer zal de actie die we geselecteerd hebben vervangen. Kies *Monster -> Subtract from* (in de *Instance variabeles* categorie) -> Instance variabele "health", en voer 1 voor *Value* . Klik Done . De actie moet nu als volgt worden weergegeven:

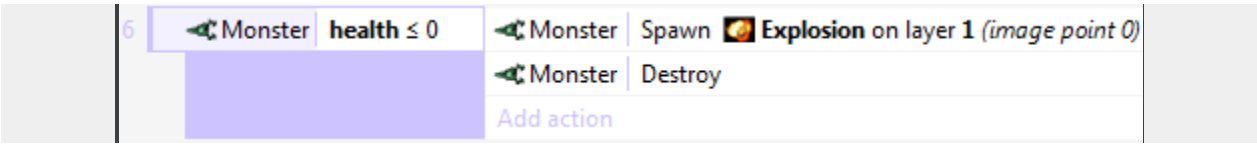


Als we nu op de monsters schieten verliezen ze 1 health en de kogel explodeert, maar we hebben geen gebeurtenis gemaakt om monsters te doden als hun gezondheid op nul staat. Voeg een event toe:

Conditie: *Monster -> Compare instance variable* -> Health, *Less or equal*, 0

Actie: *Monster -> Spawn another object* -> Explosion, layer 1

Actie: *Monster -> Destroy*



Waarom "less or equal 0" in plaats van "equals 0"? Stel dat we een krachtiger wapen hebben dat 2 zou aftrekken van health. Als je een monster neergeschiet, zou zijn gezondheid achtereenvolgens: 5 , 3 , 1 , - 1 , - 3 worden... Zie je dat op geen enkel moment de gezondheid precies *nul* was, dus het zou nooit sterven! Daarom is het een goede gewoonte om "less or equal" te gebruiken om te testen of de gezondheid op is.

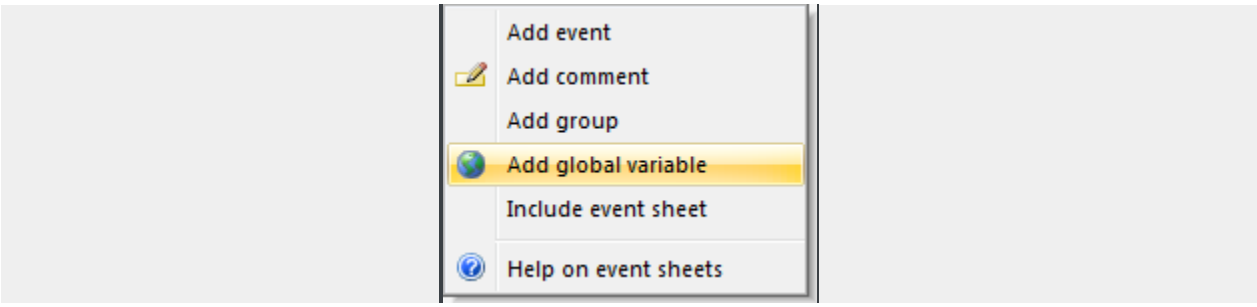
Start het spel. Je moet nu de monsters vijf keer raken om ze te doden!

BIJHOUDEN VAN DE UITSLAG

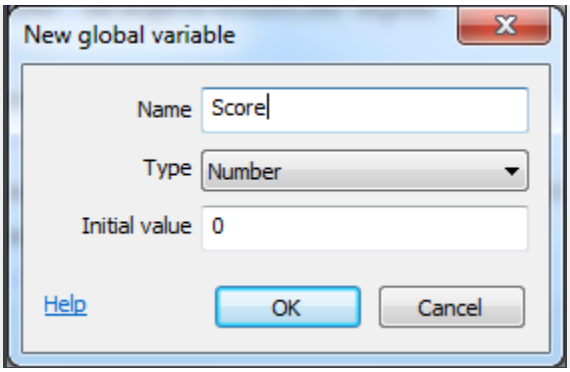
Laten we een score maken, zodat de speler weet hoe goed ze het hebben gedaan. We hebben hiervoor nog een variabele nodig. Je zou kunnen denken "laten we de score als een instance variabele van de speler bijhouden". Dat is geen slecht eerste idee, maar vergeet niet de waarde is opgeslagen "in" het object. Als er geen instanties zijn, zijn er ook geen variabelen! Dus als we de speler vernietigen, kunnen we niet meer te vertellen wat zijn uitslag was, want die werd vernietigd met de speler.

In plaats daarvan kunnen we een global variabele gebruiken. Net als een instance variabele, kan een globale variabele (of gewoon "global") tekst of een getal opslaan. Elke variabele kan een getal of een enkel stuk tekst opslaan. Globale variabelen zijn ook beschikbaar voor het hele spel in alle layouts - handig mochten we andere niveaus toe voegen.

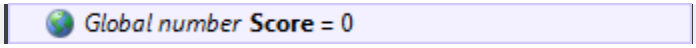
Rechts-klik in de ruimte aan de onderkant van het event sheet, en selecteer *Add global variable*.



Voer Score in als de naam. De andere velden zijn OK, het maakt een getal dat begint bij 0.

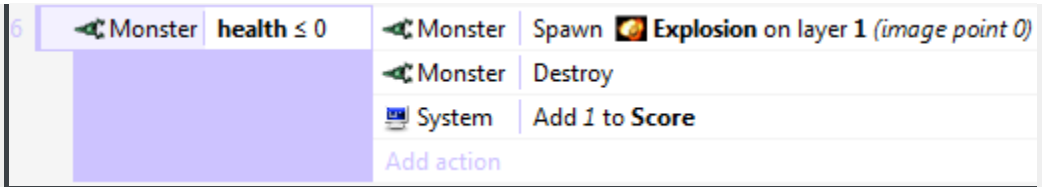


Nu verschijnt de globale variabele als regel in het event sheet. Het staat in dit ene event sheet, maar het is toegankelijk vanaf elk event blad in elke lay-out.



Opmerking: er zijn ook *local* variabelen die alleen kunnen worden geopend door een kleinere "scope" van gebeurtenissen, maar we hoeven ons daar nu geen zorgen over te maken.

Laten we de speler een punt geven voor het doden van een monster. In ons "Monster: health less or equal 0" event (wanneer een monster sterft), klik Add action , en selecteer *System -> Add to* (onder Global en local variabeles) -> Score , value 1 . Nu zou de event er zo uit moeten zien:



Nu heeft de speler een score, die met 1 toeneemt voor elk monster dat ze te doden - maar ze kunnen hun score niet zien! Dat Laten we aan de speler zien met een tekstobject.

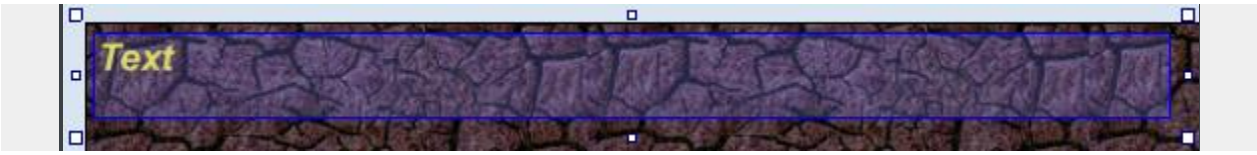
HET CREËREN VAN EEN HEADS-UP DISPLAY (HUD)

Een heads-up display (aka HUD) is de interface die de speler de gezondheid, de score en andere informatie in het spel laat zien. Laten we een heel eenvoudige HUD maken met een tekstobject.

De HUD blijft altijd op dezelfde plaats op het scherm. Als we een aantal interface-objecten hebben, willen we niet dat ze weg schuiven als de speler rondloopt - ze moeten op het scherm blijven. Standaard scrollen lagen. Om ze op het scherm te houden, kunnen we de laag Parallax instelling gebruiken. Parallax laat verschillende lagen met verschillende snelheden scrollen voor een soort semi-3D-effect. Als we de parallax echter op nul zetten, scrollt de laag helemaal niet - ideaal voor een HUD.

Ga terug naar de layers bar die we al eerder gebruikten. Voeg een nieuwe laag toe genaamd *HUD.* Zorg ervoor dat het bovenop ligt, en dat het geselecteerd is (onthoudt dit maakt het de actieve laag). De properties bar moet nu zijn eigenschappen tonen. Stel de Parallax property in op 0, 0 (dat is nul op zowel de X-en Y-as).

Dubbelklik een ruimte om een nieuw object te plaatsen. Kies deze keer het Text object. Plaats het in de linkerbovenhoek van de layout. Het zal moeilijk te zien zijn als het zwart is, dus in de propeties bar, maak het vet, cursief, geel, en kies een iets groter lettertype. Pas de grootte aan zodat het breed genoeg is om een redelijke hoeveelheid tekst te bevatten. Het moet er ongeveer zo uitzien:



Ga terug naar het event sheet. Laten we de tekst bijgewerkt houden met de score van de speler. In het Every tick event dat wij eerder hebben toegevoegd, voeg de volgende actie toe *Text -> Set text* .

Met behulp van de & operator, kunnen we een getal omzetten naar tekst en met andere tekst samenvoegen. Dus voer als de tekst in:

"Score: " & Score

Het eerste deel (*“Score:”)* betekent dat de tekst altijd zal beginnen met de zin: *Score:* .
Het tweede deel (*Score)* is de werkelijke waarde van de Score globale variabele.
De & voegt ze samen in een stuk tekst.

Start het spel, en schiet op een aantal monsters. Je score wordt weergegeven, en het blijft op dezelfde plaats op het scherm!

AFWERKING

We zijn bijna klaar. Laten we nog wat puntjes bijwerken.

Ten eerste, laten we eens wat monsters regelmatig tevoorschijn komen, anders is er niets meer te doen als eenmaal alle monsters door zijn. We zullen elke 3 seconden een nieuw monster creëren. Voeg een nieuw event toe:

Conditie: *System -> Every X seconds* -> 3

Actie: *System -> Create object* -> Monster, layer 1, 1400 (voor X), random(1024) (voor Y)

1400 is een X-coördinaat vlak bij de rechterrand van de lay-out, en *random(1024)* is een willekeurige y-coördinaat voor de hoogte van de lay-out.

Tot slot, laten we eens spoken maken die de speler kunnen doden.

Conditie: *Monster -> On collision with another object* -> Speler

Actie: *Speler -> Destroy*