

Game Maker Tutorial

Doolhof Spellen Maken

Geschreven door Mark Overmars

Copyright © 2007 YoYo Games Ltd

Vertaling: Sion Oppeneer

Nederlandse Game Maker Community (www.game-maker.nl)

Laatst veranderd: 21 februari 2007

Gebruikt: Game Maker 7.0, Lite of Pro versie, Advanced Mode

Niveau: Beginner

Doolhof spellen zijn erg populaire spel types en zij zijn gemakkelijk te maken met *Game Maker*. Deze tutorial laat in een aantal stappen zien hoe je een dergelijk spel kunt maken. Het leuke deel is dat we vanaf de eerste stap een speelbaar spel hebben die, in verdere stappen, uitgebreider en aantrekkelijker wordt. Alle voorbeelden zijn toegevoegd in de map **Examples** en kunnen geopend worden in *Game Maker*. Ook alle resources zijn beschikbaar in de map **Resources**.

Het spel idee

Voordat we beginnen met een spel te maken, moeten we eerst bedenken hoe we het spel willen hebben. Dit is de meest belangrijke (en in bepaalde zin de moeilijkste) stap in het bedenken van een spel. Een goed spel is leuk, verrassend en verslavend. Er moeten duidelijke doelen voor de speler zijn en de gebruikersinterface moet handig zijn.

Het spel dat we gaan maken is een doolhof spel. Elke room bevat een doolhof. Om uit dit doolhof te ontsnappen, moet de speler alle diamanten verzamelen en naar de uitgang gaan. Om dit te doen, moet de speler puzzels oplossen en monsters ontwijken. Er kunnen veel puzzels worden gemaakt: blokken moeten in gaten geduwd worden; delen van de room kunnen weggeblazen worden met bommen, enz. Het is belangrijk om al deze dingen niet meteen in de eerste room te laten zien. Om het spel interessant te houden verschijnen de verschillende onderdelen geleidelijk aan.

Het belangrijkste object is een persoon die wordt bestuurd door de speler. Er zijn muren (misschien meerdere types om het spel aantrekkelijker te maken). Er zijn diamanten om te verzamelen. Er liggen voorwerpen verspreid die iets doen als ze opgepakt of aangeraakt worden door de speler. Een bepaald object zal de uitgang van de room zijn. En er zijn monsters die uit zichzelf bewegen. Maar laten we die dingen maar één voor één behandelen.

Een eenvoudig begin

In het begin laten we de diamanten even achterwege. We willen een spel waarin je de uitgang moet bereiken. Er zijn drie belangrijke objecten in het spel: de speler, de muur en de uitgang. We zullen een sprite en een object nodig hebben voor deze objecten. Je kunt het eerste simpele voorbeeld vinden onder de naam **doolhof_1.gmk**. Open het en probeer het eens.

De objecten

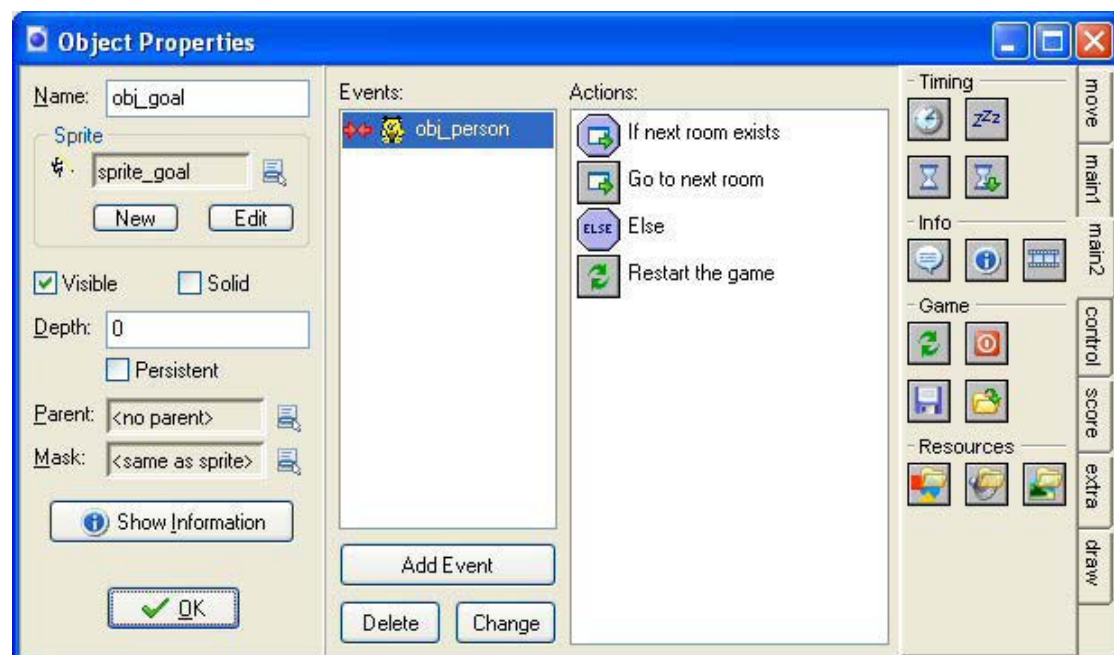
Laat ons eerst de objecten eens maken. Voor elk object gebruiken we een simpele 32x32 sprite:



Maak eerst drie nieuwe sprites en noem ze `spr_person`, `spr_wall` en `spr_goal`. Het beertje en het goal moeten transparant zijn. De muur moet niet transparant zijn.

Nu maken we drie objecten. Laten we eerst het muur object maken. We geven het de `spr_wall` als afbeelding, noemen het `obj_wall` en maken het solide door het vakje bij **Solid** aan te vinken. Zo kunnen andere objecten, zoals de speler, niet meer door de muur. De muur doet verder niets. Daarom hoeven er verder geen events in.

Laten we als tweede het doel object maken. Dit is het object dat de speler moet bereiken. We hebben besloten om deze een plaatje te geven van een finish vlag. Zo is het duidelijk voor de speler dat hij hier moet zijn. Als de speler hier komt, moet hij naar de volgende room. Daarom zetten we deze actie in het **Collision** event (deze kan je vinden onder de **main1** tab). Dit geeft een probleem als de speler de laatste room heeft bereikt. Daarom moeten we iets meer doen. Eerst kijken we of er een volgende room is. Zo ja, dan gaan we daar naar toe. Anders herstarten we het spel. Dus de event komt er zo uit te zien:

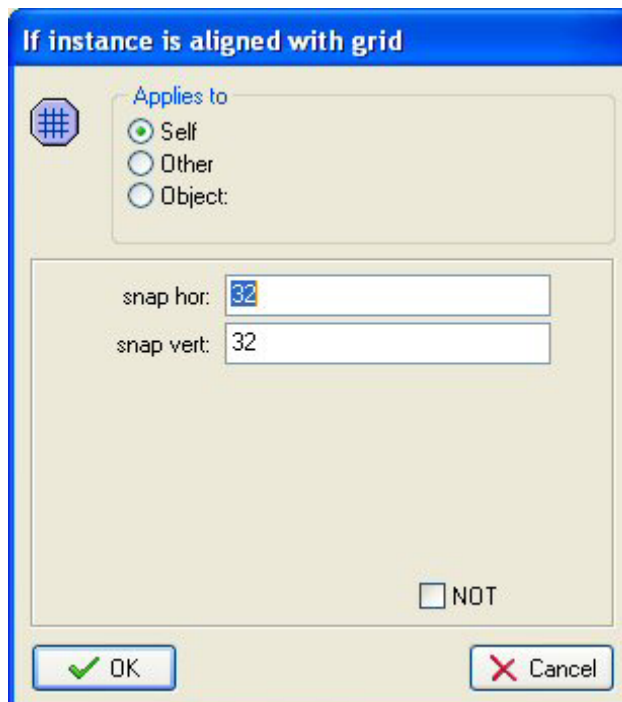


Natuurlijk moeten we in het uiteindelijke spel iets meer doen als de speler het laatste level klaarspeelt, zoals het laten zien van een leuk plaatje, of hem een positie geven in de lijst van de beste spelers. Dit beslissen we later.

Uiteindelijk hebben we een personage nodig die door de speler wordt bestuurd. Hier is wat meer werk nodig. Het moet reageren op de invoer van de speler en het mag niet botsen met de muur. We zullen de pijltjes gebruiken voor de besturing. (Dit is het

meest begrijpelijk, dus handig voor de speler). Er zijn verschillende manieren om een speler te laten bewegen. De makkelijkste manier is om de speler één kader in de aangegeven richting te laten bewegen als de speler op een pijltjestoets drukt. Een tweede manier, deze zullen we gebruiken, is dat de speler beweegt in de richting zolang de toets is ingedrukt. Een andere manier is om de speler te laten bewegen totdat een andere toets wordt ingedrukt (zoals bij Pacman).

We hebben voor alle vier pijltjestoetsen acties nodig. De acties zijn best onbenullig. Ze zetten gewoon de goede richting om te bewegen. (Als snelheid gebruiken we 4). Om te stoppen wanneer de speler de toets loslaat gebruiken we het **Keyboard** event voor <no key>. Hier stoppen we de beweging. Er is toch een probleem. We willen de speler wel in het raster laten lopen van het doolhof. Anders wordt het bewegen lastig. Je zou bijvoorbeeld precies goed moeten stoppen om een afslag te nemen. Dit kan als volgt gedaan worden. In de **control** tab is er een actie om te kijken of een object in een raster zit. Alleen als dit zo is, gebeurt de volgende actie. We zetten dit in elke pijltjestoets event en zetten de waarden op 32, omdat dit de rastergrootte is van het doolhof:



We moeten natuurlijk stoppen als we een muur raken. Daarom zetten we in de **Collision** event van de speler met de muur een actie die de beweging stopt. Voor één ding moet je hier oppassen. Als de sprite van de speler niet volledig in het kader past, zoals gewoonlijk, kan het gebeuren dat je speler niet in het raster zit als het met de muur botst. (Of preciezer, dit gebeurt als er een grotere rand dan de snelheid rond de sprite is). Dan komt de speler vast te zitten, omdat het niet meer reageert op de toetsen (want het zit niet meer in het raster). De oplossing is om de sprite groter te maken, of om precise collision checking aan te zetten en de bounding box op full image te zetten.

Maken van rooms

Dat was alles wat moest gebeuren in de acties. Laten we nu wat rooms maken. Maak één of twee rooms die lijken op een doolhof. Zet in elke room een doel object op de plaats van bestemming en zet het speler object op de startplaats.

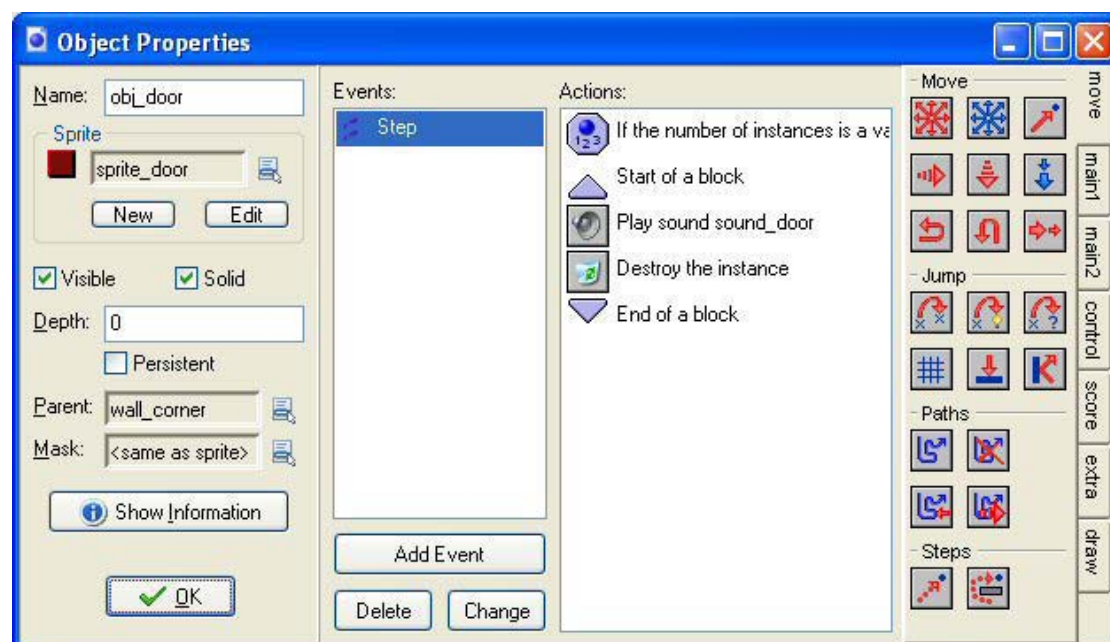
Klaar

Dat was alles. Het eerste spel is klaar. Speel er maar eens wat mee. Je kunt nog de snelheid van de speler veranderen in zijn **Create** event, wat meer levels maken, plaatjes veranderen, enz.

Diamanten verzamelen

Maar het doel van het spel was om diamanten te verzamelen. De diamanten zelf zijn gemakkelijk. Maar hoe zorgen we dat de speler niet de room uit kan als nog niet alle diamanten verzameld zijn? Hiervoor maken we een deur object. Het deur object zal zich gedragen als een muur als er nog diamanten zijn, en verdwijnen als alle diamanten weg zijn. Je kunt het tweede voorbeeld vinden onder de naam `doolhof_2.gmk`. Open het en probeer het eens.

Naast de muur, uitgang en speler hebben we twee nieuwe objecten nodig met passende sprites: de diamant en de deur. De diamant is enorm simpel. De enige actie die het nodig heeft is dat het verdwijnt als de speler het raakt. Daarom zetten we in het **Collision** event een actie om het te verwijderen. Het deur object zal geplaatst worden op een plaats die de uitgang blokkeert. Het moet solide zijn (om te voorkomen dat de speler het passeert). In het **Collision** event van de speler met de deur moeten we de beweging stoppen. In het **Step** event van de deur kijken we of het aantal diamanten 0 is, en zo ja, dat het zichzelf verwijdert. Er is een actie hiervoor. We zullen ook een geluidje laten horen, zodat de speler zal horen dat de deur open is. Het **Step** event ziet er als volgt uit:



Het iets leuker maken

Nu de basis van het spel klaar is, maken we het iets leuker.

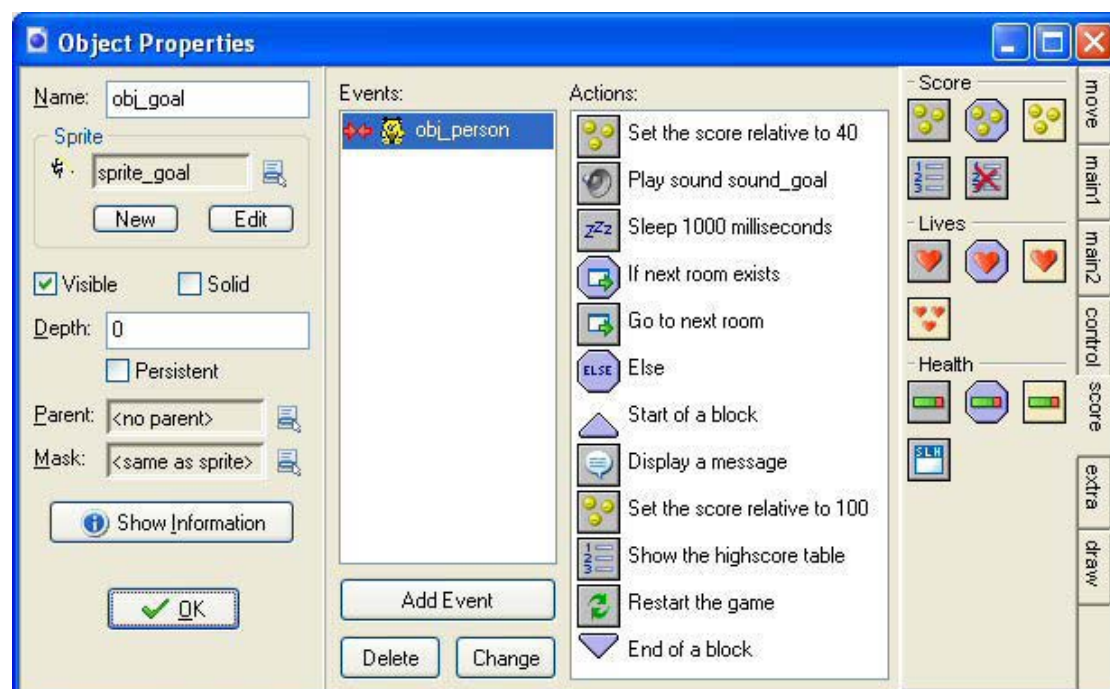
De muren zijn best lelijk. Laten we daarom drie muur objecten maken, één voor de hoeken, één voor de verticale muren en één voor de horizontale muren. Geef het de goede sprites en maak ze solide. Met een beetje aanpassing van de rooms ziet het er een stuk leuker uit. Het geven van een achtergrond afbeelding zal ook helpen.

Om te voorkomen dat we steeds het **Collision** event moeten doen voor al die muren (en later voor monsters), gebruiken we een belangrijke techniek in *Game Maker*. We maken de hoekmuur object de parent (ouder) van de andere muur objecten. Dit betekent dat de muur objecten zich zullen gedragen als varianten van de hoekmuur object. Zo gedragen ze zich hetzelfde (tenzij we ze een ander gedrag geven). Ook, voor andere instanties, zijn ze hetzelfde. Dan hoeven we alleen de botsing met de hoekmuur object te maken. Deze actie zal dan automatisch gebruikt worden voor andere muur objecten. Ook de deur object kunnen we als parent het hoekmuur object geven.

Score

Laten we de speler een score geven, zodat hij zijn vooruitgang kan zien. Dit is niet moeilijk. Voor elke verwijderde diamant geven we 5 punten. Dus voegen we in de **Destroy** event van de diamant 5 punten toe. Het uitspelen van een level geeft 40 punten, dus voegen we 40 punten toe in het **Collision** event van de uitgang met de speler.

Als de speler de laatste room uitspeelt, moet er een highscore tabel komen. Dit is gemakkelijk in *Game Maker*, want er is een actie voor. Het uitgang object wordt hierdoor iets moeilijker. Als het botst met de speler wordt de volgende event uitgevoerd:



Het voegt iets toe aan de score, speelt een geluid, wacht een tijdje en gaat dan naar de volgende room of, als het de laatste room is, laat het een bericht zien, daarna de highscore tabel en daarna herstart het spel.

Zoals je kunt zien, verschijnt de score automatisch in de bovenste balk. Dit is best lelijk. We kunnen beter een besturing object maken. Deze heeft geen sprite nodig. Dit object zal in elke room geplaatst worden. Het geeft algemene besturing in het spel. Voorlopig hebben we het nodig om de score te laten zien. In zijn **Draw** event zetten we het lettertype en de kleur en dan gebruiken we de actie om de score te laten zien.

Begin scherm

Het is leuk om te beginnen met een scherm waar de naam van het spel staat. Hiervoor gebruiken we de eerste room. We geven het een mooie achtergrond met een leuke afbeelding. (Je kunt het beste zorgen dat er geen video geheugen nodig is, omdat het alleen in de eerste room wordt gebruikt). Deze achtergrond gebruiken we voor het beginscherm (je kunt beter de achtergrondkleur en het betegelen van de achtergrond uitzetten). Een start besturing object (onzichtbaar natuurlijk) is er om te wachten tot de speler een toets indrukt en dan naar de volgende room gaat. (Het start besturing object zet de score op 0 en zorgt dat de score niet zichtbaar is in de bovenste balk).

Geluiden

Een spel zonder geluid is behoorlijk saai. Daarom hebben we een paar geluiden nodig. Eerst hebben we wat muziek nodig. Hiervoor kunnen we een leuk midi-bestand gebruiken. We laten die muziek beginnen in `start_controller`. Zet het op loop, zodat het opnieuw begint als het is afgelopen. Daarna hebben we wat geluiden nodig voor het oppakken van een diamant, het openen van een deur, en voor het bereiken van de uitgang. Deze geluiden zijn genoemd naar de hiervoor genoemde acties. Na het bereiken van de uitgang, na het geluid, is het handig om nog een sleep actie neer te zetten, zodat het even duurt voordat we naar de volgende room gaan.

Maken van rooms

Nu kunnen we rooms maken met diamanten. Je kunt best de eerste room zonder diamanten er in laten. Dit is goed, want het introduceert de speler in het bewegen naar de vlag, voordat het diamanten moet verzamelen. Door het geven van een goede naam naar de volgende room met de diamanten zal de speler begrijpen wat te doen.

Monsters en andere uitdagingen

Het spel begint er al leuk uit te zien, maar bevat nog steeds weinig, en is best saai om te spelen. Daarom hebben we nog wat actie nodig in de vorm van monsters. Ook zullen we bommen, beweegbare blokken en gaten toevoegen. Het complete spel kan gevonden worden onder de naam `doolhof_3.gmk`.

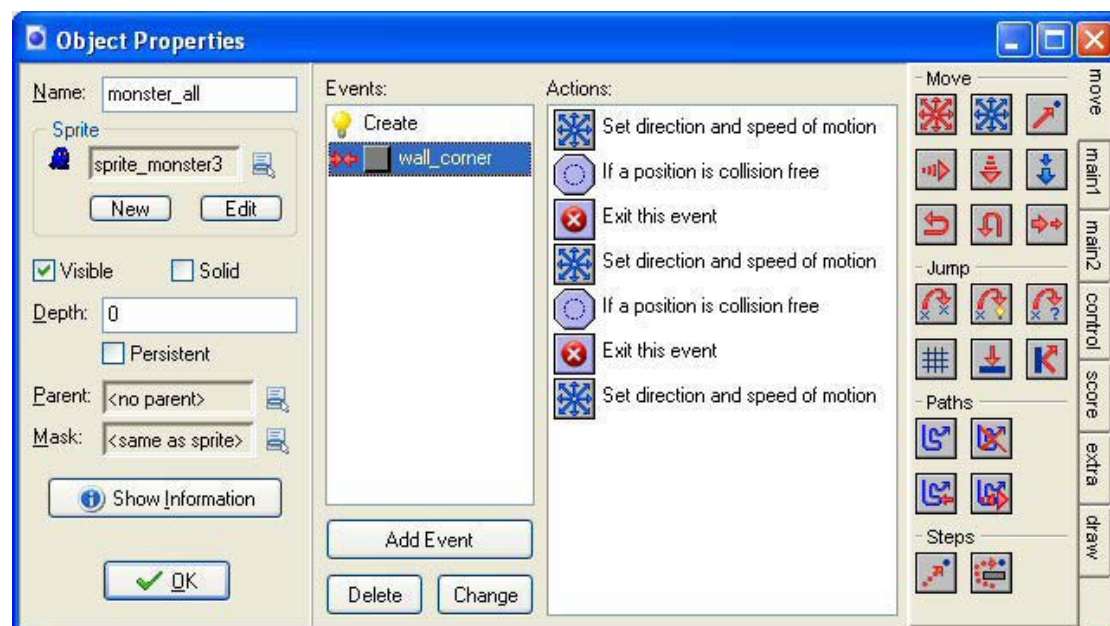
Monsters

We zullen drie monsters maken: één die links en rechts beweegt, één die op en neer beweegt en één die in vier richtingen beweegt. Een monster toevoegen is best simpel. Het is een object die in een richting begint te bewegen en verandert van richting als die een muur raakt. Als de speler een monster raakt, is hij dood, ofwel, het level wordt herstart en de speler verliest een leven. We geven de speler drie levens om mee te beginnen.

Laten we eerst een monster maken dat links en rechts beweegt. We gebruiken er een simpele sprite voor en maken een object met deze sprite. In zijn **Create** event beslist het om links of rechts te gaan. Om het leven wat moeilijker te maken, zetten we de snelheid wat hoger. Bij een botsing keert het monster horizontaal om.

Het tweede monster werkt precies hetzelfde, maar dan begint het op of naar te bewegen en, als hij botst, keert het verticaal om.

Het derde monster is iets ingewikkelder. Het begint óf horizontaal óf verticaal te bewegen. Als het een muur raakt, beslist het om naar links of rechts te gaan. Als dat niet kan keert het om. Dit ziet er als volgt uit:

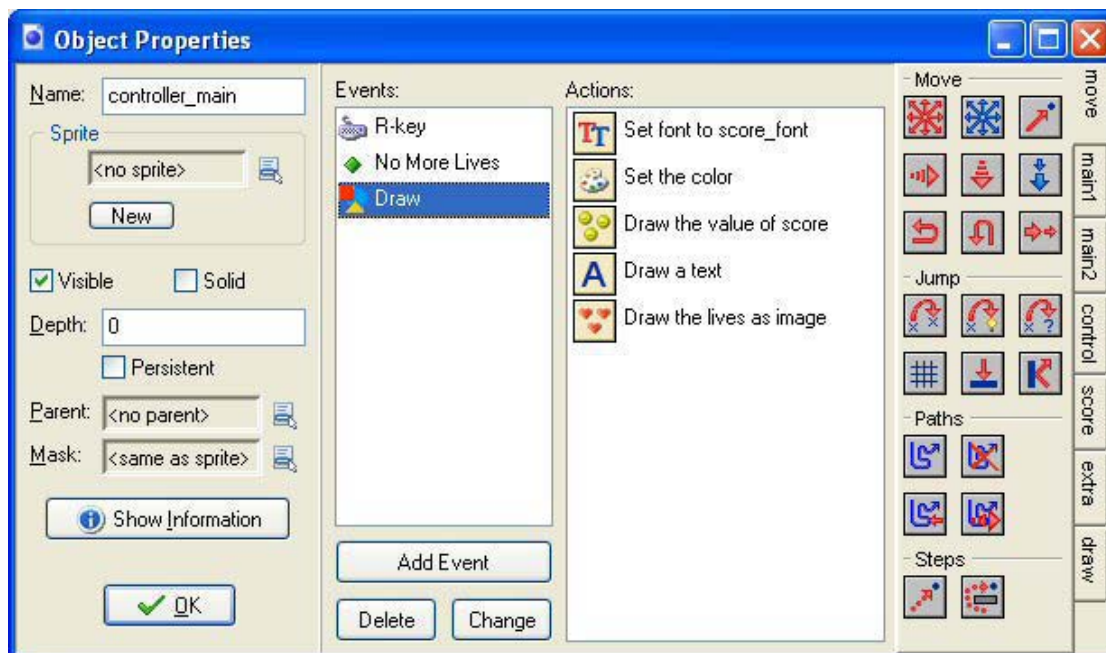


Om problemen te voorkomen dat het monster te klein is, vinken we precise collision checking uit en zetten we de bounding box op full image.

Als de speler met een monster botst, moeten we een vreselijk geluidje afspelen, een poosje wachten, de levens met één laten afnemen en dan de room herstarten. (Merk op dat deze volgorde belangrijk is. Anders worden de andere acties niet uitgevoerd als de room is herstart). Het besturing object, in het **No More Lives** event, laat de highscore tabel zien en herstart het spel.

Levens

We hebben het levens mechanisme van *Game Maker* gebruikt om de speler drie levens te geven. Het zou ook leuk zijn om het aantal levens te laten zien. Het besturing object zou dit kunnen doen net als bij de score. Maar het zou leuker zijn om kleinere afbeeldingen van de speler te laten zien als levens. In de **Score** tab is hier een actie voor. Het **Draw** event ziet er nu als volgt uit:



Je kunt zien dat we ook een speciaal lettertype hebben gebruikt voor de score.

Bommen

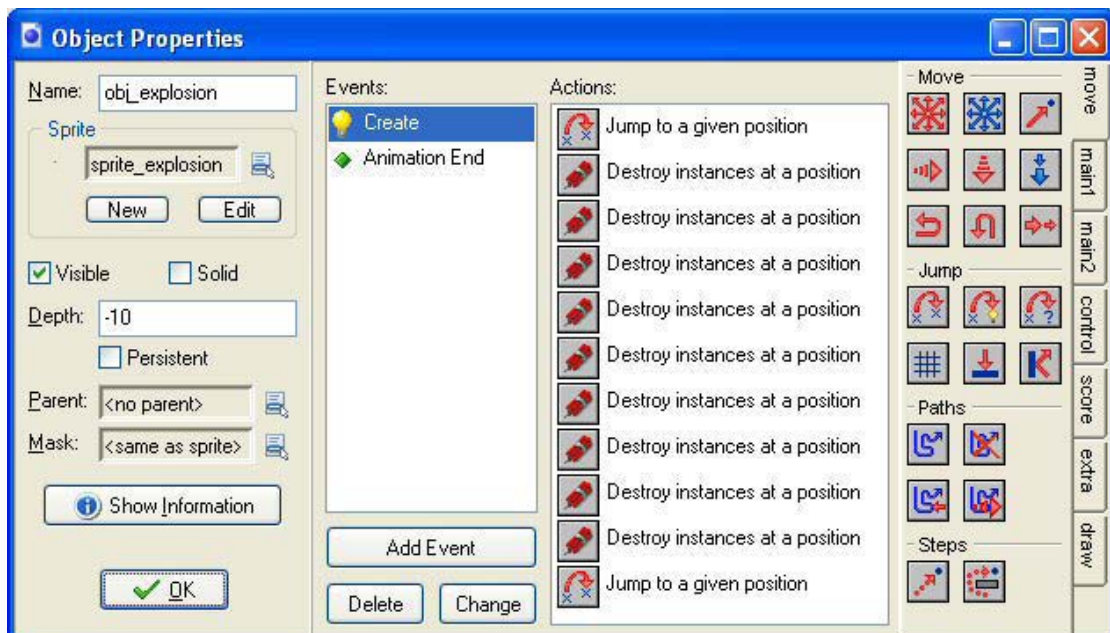
Laten we nu eens bommen en activeringen toevoegen. Het idee is dat als de speler naar de activering gaat, alle bommen exploderen, alles vernietigend in hun buurt. Dit kan gebruikt worden om gaten in de muur te blazen en monsters te doden. We hebben drie nieuwe objecten nodig: een activering, een bom en een explosie. Voor alles hebben we een passende sprite nodig.

De bom is zeer simpel. Het zit daar maar en doet niets. Om zeker te zijn dat monsters er over heen lopen (in plaats van er onder) zetten we de depth op 10. Objecten worden neergezet in volgorde van diepte. De objecten met de hoogste diepte worden het eerst neergezet. Daarom zullen die lager liggen dan die met een lagere diepte. Door het zetten van de depth van de bom op 10, zullen de andere objecten, met een standaard diepte van 0, er boven komen.

De activering is ook behoorlijk simpel. Als het botst met de persoon, verandert het alle bommen in explosies. Dit kan door de actie te gebruiken die een object in een ander object verandert. Bovenaan zorgen we dat dit gebeurt met alle bommen.



Het explosie object laat gewoon de animatie zien. Na de animatie verwijdert het zichzelf. (Zorg er wel voor dat de origine van de explosie op de goede plaats is als de bom er in verandert). Het object moet ook alles vernietigen wat in de buurt is. Dit vraagt wat meer werk. Ten eerste willen we dat de explosie zich niet zelf vernietigt, dus zetten we het even aan de kant. Daarna gebruiken we de acties om alle objecten in de buurt te vernietigen op de oude plaats van de explosie. Als laatste zetten we de explosie terug op de oude plaats.



Weet dat het verkeerd gaat als de speler vlakbij de bom is! Zorg daarom dat de activering niet te dichtbij de bommen is. Het is belangrijk om de levels zorgvuldig te ontwerpen met bommen en activeringen, zodat er interessante uitdagingen komen.

Blokken en gaten

Laten we iets anders maken dat het mogelijk maakt om meer ingewikkelde puzzels te maken. We maken blokken die door de speler weggeduwd kunnen worden. Ook maken we gaten waar de speler niet over kan, maar gevuld kunnen worden met blokken om nieuwe paden te maken. Dit geeft veel mogelijkheden. Blokken moeten op een bepaalde manier weggeduwd worden om paden te maken. En je kunt monsters gevangen zetten met blokken.

Het blok is een solide object. Het grootste probleem is dat het moet bewegen naar de richting van de speler als het wordt geduwd. Als het botst met de speler, ondernemen we de volgende acties: we kijken of de relatieve positie `8*other.hspeed`, `8*other.vspeed` leeg is. Dit is de positie waar het blok moet komen. We doen hetzelfde als er een gat object op die positie is. Om te voorkomen dat monsters over de blokken heen lopen, maken we het hoekmuur object een parent van het blok. Dit geeft een klein probleem. Omdat er een **Collision** event bestaat tussen de speler en de hoekmuur en niet tussen de speler en het blok, wordt dat event uitgevoerd en staat de speler stil. Om dit op te lossen zetten we een comment in het collision event van de speler met het blok. Nu wordt deze event uitgevoerd en staat de speler niet stil. (Om preciezer te zijn, de nieuwe collision event neemt de collision event van de parent over. Zoals vermeld, kan je dit gebruiken om de child objecten een iets ander gedrag te geven dan de parent).

Het gat is een solide object. Als het botst met het blok, verwijdert het zichzelf én het blok. We geven ook deze het hoekmuur object als parent, zodat het zich gedraagt als een muur.

Met de blokken en gaten kan je allerlei intrigerende rooms maken. Er is alleen één probleem. Je kunt jezelf gemakkelijk opsluiten, zodat de room niet meer opgelost kan worden. Daarom moet je de speler de mogelijkheid geven om het level te herstarten, wat één leven kost. Hiervoor gebruiken we de R toets om te herstarten. In het besturings object zorgen we in het keyboard event voor deze toets voor het afnemen van één leven en het herstarten van de room.

Een paar uiteindelijke verbeteringen

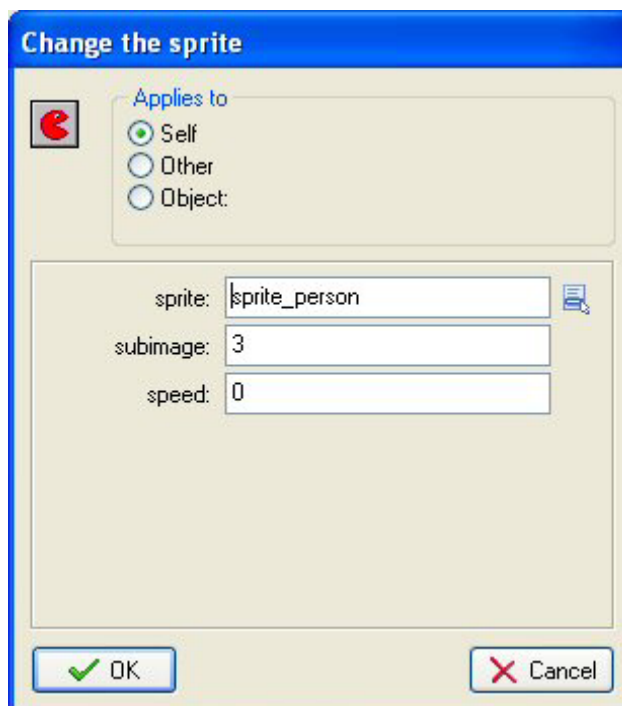
Laten we nu ons spel afmaken. We moeten in ieder geval de graphics verbeteren. Ook hebben we meer interessante levels nodig. Hiervoor maken we wat meer bonussen en toevoegingen. Het uiteindelijke spel kan gevonden worden onder de naam `doolhof_4.gmk`.

Betere graphics

De graphics van het spel zijn nu een beetje magertjes. Laten we daar wat aan doen. Het belangrijkste ding wat we willen veranderen is dat de persoon kijkt in de richting waar hij gaat. De eenvoudigste manier is om een sprite te maken die bestaat uit 4 plaatjes, één voor elke richting, als volgt:



Normaal gaat Game Maker door al die plaatjes. Dit kunnen we voorkomen door de variabele `image_speed` op 0 te zetten. Als we de richting van de speler veranderen, kunnen we het plaatje veranderen bij de actie om de sprite te veranderen:



Hetzelfde kunnen we doen voor alle monsters, maar hier zijn geen aparte events om van richting te veranderen. Het is makkelijker om een test in het **End Step** event te zetten om te zien in welke richting het monster beweegt en de sprite daarop aan te passen.

Bonussen

Laten we twee bonussen toevoegen: de één geeft je 100 punten en de ander geeft je een extra leven. Ze zijn allebei enorm simpel. Als ze met de speler in contact komen, spelen ze een geluid, verwijderen ze zichzelf en dan voegen ze óf 100 punten óf een extra leven toe. Dat is alles.

Eenrichtingsverkeer

Laten we eenrichtingspaden maken die je maar via één kant kan passeren om de levels moeilijker te maken. Hiervoor maken we vier objecten, allemaal in de vorm van een pijl wijzend in de richting die hij opgaat. Als de speler er volledig opstaat, moeten we het in de goede richting sturen. Dit doen we in de **Step** event van de speler. We kijken of de speler zich in het raster bevindt en of hij op één van de pijlen staat. Zo ja, dan bewegen we hem in de goede richting. (We zetten de snelheid op 8 om het leuker te maken).

Bange monsters

Om soort Pacman levels te maken geven we elk monsters een variabele `bang`. In hun **Create** event zetten we die op 0. Als de speler een ring oppakt, zetten we de variabele op true en veranderen we de sprite zodat de monsters inderdaad bang lijken. Als de speler een monster raakt, kijken we eerst of het bang is of niet. Zo ja, dan zetten we

het monster op zijn beginspositie. Anders verliest de speler een leven. Zie het spel voor details.

Laten we er nu een spel van maken

We hebben nu flink wat objecten gemaakt, maar we hebben nog steeds geen echt spel. Een belangrijke factor in spellen is het ontwerpen van levels. Ze moeten van gemakkelijk naar moeilijk gaan. In het begin mogen er maar een paar objecten gebruikt worden. Later verschijnen er meer. Zorg ervoor dat je geheimen hebt die pas in level 50 komen of zo. Natuurlijk moeten de levels aangepast worden op de doelgroep. Voor kinderen heb je heel andere levels nodig dan voor volwassenen.

Ook heeft een spel documentatie nodig. In *Game Maker* kan je heel gemakkelijk documentatie toevoegen door **Game Information**. Als laatste, spelers zullen het spel niet achter elkaar uitspelen. Daarom moet je eigenlijk een mechanisme maken dat je kunt bewaren en laden. Gelukkig is dit heel gemakkelijk. *Game Maker* heeft een ingebouwd bewaar- en laadmechanisme. F5 slaat het spel op en F6 laadt het laatst opgeslagen spel. Je moet dit wel in je documentatie weergeven.

Je vindt het complete spel, met dit erbij, in het bestand `doolhof_4.gmk`. Open het, speel het, bekijk het en verander het zoveel als je wilt. Het belangrijkste is om meer levels te maken (er zijn er nog maar 20). Je kunt ook andere objecten maken, zoals bijv. sleutels die bepaalde deuren openen, transporteurs die je van de ene naar de andere plaats brengen, kogels die de speler kan schieten om monsters te doden, deuren die van tijd tot tijd open en dicht gaan, ijs waarop de speler dezelfde richting blijft glijden, schietende vallen, enz.

Slot

Ik hoop dat deze tutorial je heeft geholpen om spellen te maken in *Game Maker*. Vergeet niet om eerst je spel te ontwerpen en dan het stap voor stap te maken (of beter, object voor object). Er zijn altijd meerdere manieren om iets te bereiken. Dus als iets niet lukt, probeer het anders. Succes!

Verder lezen

Om verder te lezen over het maken van games met *Game Maker* kun je ons boek kopen:

Jacob Habgood en Mark Overmars, *Leer jezelf MAKKELIJK... Games ontwerpen met Game Maker*, 2007, ISBN 978-90-5940-284-3.

Dit boek geeft stap-voor-stap een introductie van de vele aspecten van *Game Maker* en in dit proces maak je negen prachtige spellen die ook nog eens leuk zijn om te spelen.