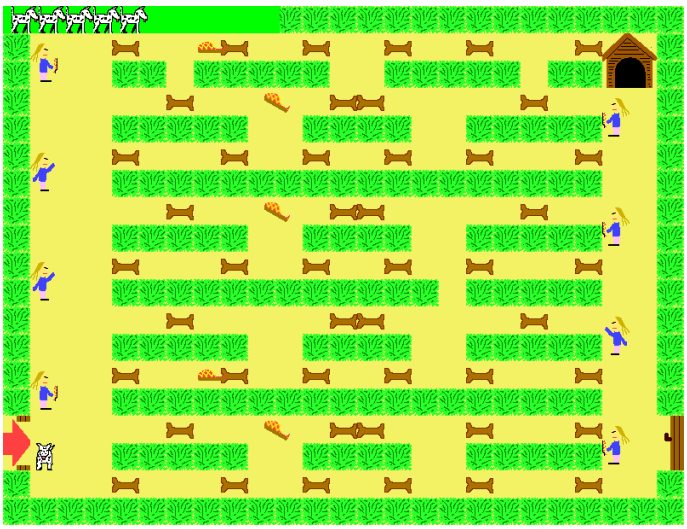
Deel 2



Het project Gamemaker Kids is mede mogelijk gemaakt door:

VSBfonds

Auteur: Klasien Postma

Eindredactie: Sacha de Ruiter, Technika 10 Nederland

Copyright: Creative Commons

www.gamemakerkids.nl/index.pl/licentie

Uitgave: September 2007

Inhoudsopgave

Hoofdstuk 1: Inleiding 5
Hoofdstuk 2: Wat is een goed spel 6
Verschillende soorten spellen 6
Het besturingsobject 7
Het einddoel 8
Beloningen 8
Hindernissen 9
Opbouw van makkelijk naar moeilijk 10
Een 2D of 3D spel 11
Hoofdstuk 3: De basis van je spel - muur, IK en doel 12
Wat heb je minimaal nodig voor een doolhofspel, voorbeeld bekijken 12
Samenvatting 13
Sprites ontwerpen 14
Zelf sprites maken 14
Objecten maken 18
Een speelveld maken 18
Een tweede level maken 20
Hoofdstuk 4: Basisobjecten voor een doolhofspel 21
Basisobjecten, voorbeeld bekijken 21
Hoe zijn de basisobjecten geprogrammeerd? 21
Als volgend level bestaat, ga naar volgend level, anders 23
Samenvatting 25
Zelf basisobjecten ontwerpen 25
Zelf je basisobjecten programmeren 26
Hoofdstuk 5: Beloningen, punten scoren 28
Beloningen ontwerpen, voorbeeld bekijken 28
Hoe zijn de beloningen geprogrammeerd? 29
Tekenen in je scherm met actie Draw rectangle 30
Samenvatting 32
Zelf beloningen ontwerpen 32
Zelf je beloningen programmeren 34
Hoofdstuk 6: Monsters 36
Monsters ontwerpen, voorbeeld bekijken 36
Hoe zijn de monsters geprogrammeerd? 38
Onzichtbare muurtjes 39

Samenvatting ... 40
Zelf monsters ontwerpen ... 41
Zelf je monsters programmeren ... 42
Hoofdstuk 7: Niet zo maar naar het volgende level ... 44
Hindernis om naar het volgende level te gaan, voorbeeld bekijken ... 44
Hoe zijn de hindernissen geprogrammeerd? ... 46
De actie 'Een object verandert in een ander object' ... 46
Event 'Als de animatie is afgelopen' ... 47
De actie 'Als het aantal objecten gelijk is aan nul' ... 48
De actie 'Als de score gelijk is aan...' ... 49
Samenvatting ... 50
Zelf nog meer hindernissen ontwerpen ... 50
Zelf je hindernissen programmeren ... 51
Hoofdstuk 8: Opbouw van makkelijk naar moeilijk ... 53
Levels bijwerken en testen ... 53
Hoofdstuk 9: Je spel afmaken ... 54
Geluiden ... 54
Achtergrondmuziek ... 54
Bleeps ... 55
Help ... 56
Startscherm, tussenschermen, eindscherm ... 56
Een uitvoerbaar bestand maken ... 59
Hoofdstuk 10: Extra trucs ... 60
Belangrijk ... 60
Een animatie maken ... 60
Een sprite maken die andere kant op kijkt ... 63
Sprite wisselen wanneer de ik-figuur van richting verandert ... 64
Kopiëren van objecten ... 65
Sprite of object wisselen wanneer ander object van richting verandert ... 65
Soms wel, soms niet: gebruik van dobbelsteen ... 68
Objecten willekeurig rond laten lopen ... 72
De ik-figuur laten gooien of schieten ... 76
Een wekker gebruiken ... 80
Een ander object laten gooien of schieten ... 80
De zandloper ... 84
Hoofdstuk 11: Tenslotte ... 87
Overzicht van cursusboeken Game Maker ... 87
Hoofdstuk 12: Bijlage - testschema voor de eindtest ... 88

Hoofdstuk 1: Inleiding

In deel 1 van de cursus Game Maker heb je geleerd hoe je met het programma Game Maker een spel kunt maken. De nadruk lag vooral op hoe het programma Game Maker werkt.
In deel 2 van de cursus ga je zelf een spel ontwerpen, bouwen en testen. Je leert hoe je dit aanpakt. Je ontwerpt stap-voor-stap zelf een game.
Wat maakt een spel tot een goed spel? Waar moet je allemaal op letten?
Daarna ga je programmeren wat je zelf ontworpen hebt.
Tijdens het bouwen leer je nog veel meer mogelijkheden van Game Maker.
Tenslotte ga je jouw spel testen met het testschema wat je hebt gemaakt.

We gaan er van uit dat op jouw computer Game Maker versie 7.0 al is geïnstalleerd. Verder is er een map GameMaker geïnstalleerd. Daarin staan de mappen GamemakerBestanden, MijnGames en Voorbeeldgames. Is dat niet zo, download dan de installatiehandleiding en lees daar hoe dat moet. Kom je er niet uit, vraag om hulp. En we adviseren dat je eerst deel 1 doorwerkt, voordat je aan deel 2 begint.

Veel plezier!!

Hoofdstuk 2: Wat is een goed spel

Verschillende soorten spellen

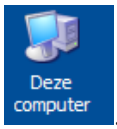
Op internet zijn veel sites te vinden waar je spellen on-line kunt spelen. Bijvoorbeeld op funnygames.nl, mijnspeeleiland.nl, nuspele.nl , spele.nl, spelletjes.nl enz. enz.
Je ziet op die sites al dat de spellen zijn onderverdeeld in soorten spellen.
Voorbeelden van soorten spellen zijn:

- behendigheidsspellen
- avonturenspellen ook wel platformgames genoemd
- racespellen
- doolhofspellen

1) Geef van elke soort spel een voorbeeld.

Soort spel	Voorbeeld
behendigheidsspel	Beertje
avonturenspel, platformgame	
racespel	
doolhofspel	

In dit boek ga je een doolhofspel ontwerpen en programmeren. In de map GameMaker in de map Voorbeeldspellen staat een voorbeeld van een doolhofspel: Snuf.

2) **Dubbelklik** op het bureaublad op **Deze computer** 

.

3) Ga naar de map **GameMaker** en dan naar de map **Voorbeeldgames**. In deze map zie je vier bestanden met een geel icoon. Dit zijn games die je zonder Game Maker kunt spelen. **Dubbelklik** op **Snuf** om het spel te starten. Probeer het spel uit te **spelen**. Je kunt het spel afsluiten door op **<ESC>** te klikken.

Het spel is expres makkelijk gemaakt. Het zal je niet veel moeite kosten om alle levels te halen. Het is dan ook vooral bedoeld als voorbeeld bij dit boek. Bij de opdrachten wordt steeds verwezen naar onderdelen van dit spel.

Wat maakt een spel nou tot een goed spel?
Ook al lijken spellen totaal verschillend, toch zijn er onderdelen die altijd in elk spel voorkomen. In de komende paragrafen leer je wat je nodig hebt om een goed spel te maken.

Het besturingsobject

Een onderdeel dat in elk spel voorkomt is een iets of iemand die je met de toetsen of muis kunt laten bewegen. Dat heet besturen. Wij noemen de persoon of het ding dat je kunt besturen het besturingsobject.

4) Ga naar de map Voorbeeldgames en dubbelklik op een spel om het spel te starten voor het beantwoorden van onderstaande vragen.
Wat kun je besturen in:
   a) Breakout? .....
   b) Beertje? .....
   c) Snuf? .....

5) Wat kun je besturen in:
   a) een racespel? ……
   b) een flipperspel? ……..
   c) een voetbalspel? ……

Het einddoel

Een ander onderdeel is het einddoel. Elk spel heeft een einddoel. Er komt een moment dat je het spel hebt uitgespeeld, je hebt het einddoel gehaald. Soms moet je net zolang doorgaan totdat je af bent. Het spel eindigt dan omdat je af bent, omdat de tijd om is of omdat je levens op zijn. Het einddoel is dan meestal om een zo hoog mogelijke score te halen.
Bij andere spellen, zoals bij doolhofspellen gaat het vaak om het uitspelen van een spel. Je hebt de hele route afgelegd. Alle hindernissen die je onderweg tegenkwam heb je overwonnen.

6) Wat is het einddoel in:

a) Breakout? .....

b) Beertje? .....

c) Snuf? .....

7) Wat kan een einddoel zijn in:

a) flipperspel? .....

b) een avonturenspel? .....

Beloningen

Soms kan het best lang duren voordat je een level haalt. Dan is het leuk als de speler ook tussendoor wordt beloond. Er zijn tijdelijke beloningen, waar je even van profiteert.
Een voorbeeld van een tijdelijke beloning is een geluidje, zoals een applaus. Een berichtje: 'Goed zo'. Of tijdelijk extra kracht of snelheid.
Een beloning kan ook blijvend zijn, bijvoorbeeld extra punten of je komt in een nieuw level.

8) Geef 2 voorbeelden van tijdelijke beloningen in het spel Snuf. Geef daarbij aan wanneer je die beloning krijgt.

1. .....

2. .....

9) Geef 2 voorbeelden van blijvende beloningen in het spel Snuf. Geef daarbij aan wanneer je die beloning krijgt.

1. .....

2. .....

Hindernissen

Het einddoel moet je niet te makkelijk kunnen halen. Daarom is er altijd iets dat het spel moeilijker moet maken, waardoor je af gaat of levens kwijt raakt. Dat noem je hindernissen. Alles waardoor een spel moeilijker wordt, noemen we hindernissen. Soms zijn hindernissen dingen, maar niet altijd. Het kan ook zijn dat je iets binnen bepaalde tijd moet halen. Of dat je een puzzel moet oplossen.

Vaak zijn er meerdere hindernissen die een spel moeilijker maken.
In een avonturenspel zijn er bijvoorbeeld monsters of moet je oppassen dat je niet ergens in valt. Dat zijn allemaal zaken die het spel moeilijker maken.

10) Noem 4 hindernissen in het spel Snuf?

1. .....3. .....

2. .....4. .....

Meestal ben je, als je een hindernis niet haalt, niet meteen af. Als je in het spel Beertje tegen een rode bal botst, dan verlies je een leven. Pas na 5 levens ben je af.

11) Na hoeveel keer tegenslag ben je bij spel Snuf in level 3 af? .....

Het is niet leuk wanneer je afgaat door domme pech. Als je afgaat moet dat je eigen schuld zijn.

12) In het spel Beertje kun je wel afgaan door domme pech. Hoe?

.....

Dat is een minpuntje in het spel Beertje.

Opbouw van makkelijk naar moeilijk

Goede spellen beginnen gemakkelijk, zodat de speler een beetje kan wennen. Als de speler beter wordt, wordt het spel meestal ook steeds moeilijker. Als dat niet zo is, dan is de lol er snel af.

Je kunt op verschillende manieren een spel steeds moeilijker maken. Meestal gebeurt dat doordat je een nieuw level in gaat. Je kunt bijvoorbeeld in een avonturenspel in een nieuw level de sterkte van monsters vergroten.

13)

a) Hoe kun je Breakout moeilijker maken? .....

b) Noem twee manieren waarop je het spel Beertje moeilijker kunt maken.

1. .....2. .....

c) Noem twee manieren waarop je een doolhofspel moeilijker kunt maken.

1. .....2. .....

Een 2D of 3D spel

Voor een goed spel is de vormgeving, hoe een spel eruit ziet, niet belangrijk. Een avonturenspel van 10 jaar geleden is in feite net zo spannend als een avonturenspel van nu. En toch geven mensen ook geld uit aan oude spellen in een nieuw jasje. Dus vormgeving is niet helemaal onbelangrijk.

In sommige spellen kan de hoofdpersoon alleen in twee richtingen bewegen. Naar links of naar rechts (horizontaal) en naar boven of naar beneden (verticaal). Dat noem je 2D (2 dimensionaal). In nieuwere spellen kun je meestal in drie richtingen bewegen. Je kunt dan ook nog naar voor en naar achteren bewegen. Dan heb je het gevoel alsof je echt in een ruimte beweegt. Dat noem je 3D spellen (3 dimensionaal).

Het ontwerpen van 3D spellen is heel erg moeilijk, omdat je dan ook nog moet letten op schaduwen en vanuit welke richting je kijkt. Zo'n spel maak je niet zomaar even in je eentje. Daar heb je een heel team van programmeurs voor nodig. Omdat het voor een goed spel niet uitmaakt en 2D veel minder moeilijk is, maken wij een 2D spel.

14) Noem twee 2D spellen en twee 3D spellen.

2D-spellen: ……en ……

3D-spellen: ……en……..

Hoofdstuk 3: De basis van je spel - muur, IK en doel

Wat heb je minimaal nodig voor een doolhofspel, voorbeeld bekijken

Je hebt in het vorige hoofdstuk geleerd waar je bij een spel allemaal op kunt letten. Wat een spel tot een goed spel maakt. In dit hoofdstuk pas je dit toe bij het ontwerp van je eigen spel, een doolhofspel. We kiezen voor een doolhofspel, omdat je dan al snel een compleet spel hebt, maar het ook heel ingewikkeld kan maken.

Je begint met het ontwerpen en bouwen van twee levels in de meest eenvoudige vorm. Later ga je het spel leuker en mooier maken met een geluiden, een leuk startscherm, meer uitdagingen, bewegende sprites enzovoort.

Wat heb je minimaal nodig voor een doolhofspel?
Wat maakt een doolhof tot een doolhof?
Juist ja, blokken of muren.

1. **Start Game maker**.
2. Ga naar de map Voorbeeldgames en **open** het bestand **Voorbeeld-1**.
3. Klik op **Run the game**
4. Hoe zien de blokken of muren er uit in Voorbeeld-1?

   .....

Wat heb je nog meer nodig? Uiteraard een object wat je met de pijltjes kan sturen, een besturingsobject, we noemen dat in dit boek de ik-figuur.

5. Wat is de ik-figuur in Voorbeeld-1? .....

En als derde - en dat had je misschien niet bedacht - heb je als object ook een doel nodig, een uitgang, een finish, waardoor je naar het volgende level gaat. Als de ik-figuur er tegen aan botst dan ga je naar het volgende level.

6. Hoe ziet het doel er uit in Voorbeeld-1?

   .....

Voor de drie objecten heb je dus drie sprites nodig.
Je hebt in het eerste deel van de cursus plaatjes gebruikt die bij het programma Game Maker zijn bijgeleverd, zoals Beertje. Het is natuurlijk veel leuker om zelf sprites te maken. In het volgende hoofdstuk leer stap voor stap hoe je dat in Game Maker kan doen.

Samenvatting

- voor een doolhof heb je minimaal drie objecten nodig:
  - een Ik-figuur,
  - muren en
  - een doel.
- voor elk van deze objecten moet een sprite worden gemaakt.

Sprites ontwerpen

Eerst ontwerp je de sprites voor je eigen spel. Een sprite is een klein plaatje. En tekenen met de muis is best moeilijk. Maak dus een eenvoudig plaatje. Maak hieronder van elke sprite een tekening.

spr_muur

spr_IK

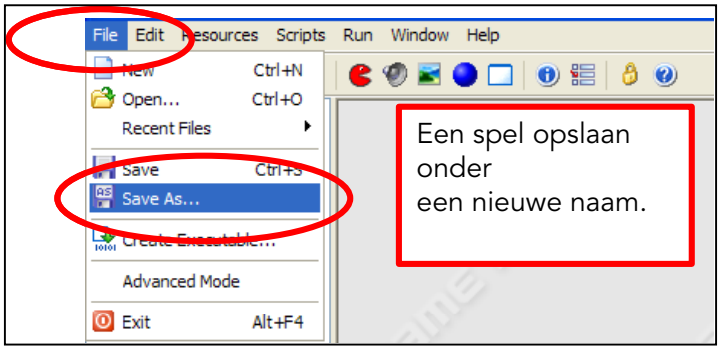
spr_doel

Zelf sprites maken

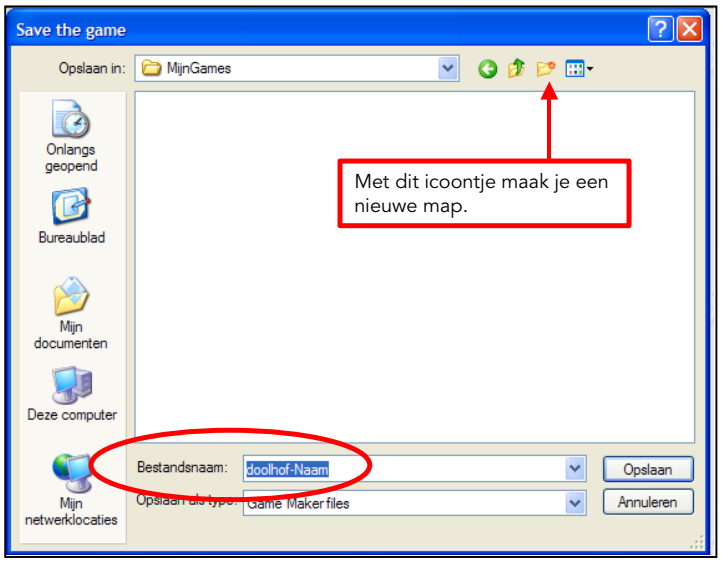
Nu je de sprites voor je eigen spel hebt ontworpen, kun je ze gaan maken.

Maar voordat je de sprites gaat maken, sla je eerst je spel op. Hoe ging dat ook al weer?

7. Start Game Maker op. Klik op **Create a new game** (=*maak een nieuw spel*) in de knoppenbalk.
8. Klik op **File** (= *Bestand*) en daarna op **Save As...** (=*Sla op als...*)

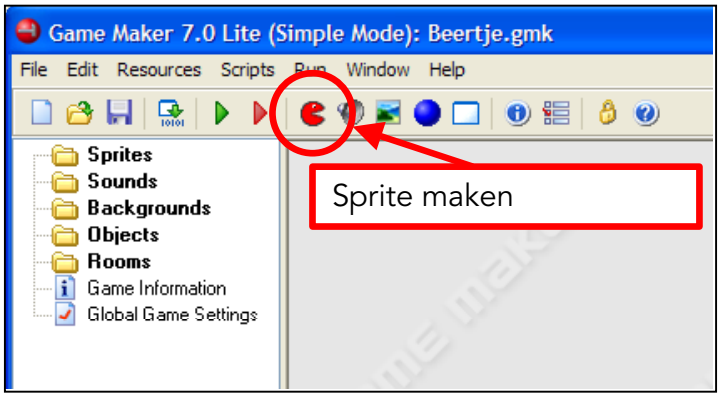


9. Ga in het vak Opslaan naar de map GameMaker en dan naar de map **MijnGames.**
10. Vul bij **Bestandsnaam doolhof-Naam-1** in, waarbij je Naam vervangt door je eigen naam, bijvoorbeeld doolhof-Laura-1. *Let op: geen spaties of rare tekens in de bestandsnaam gebruiken!!)*
11. Klik op **Opslaan**.



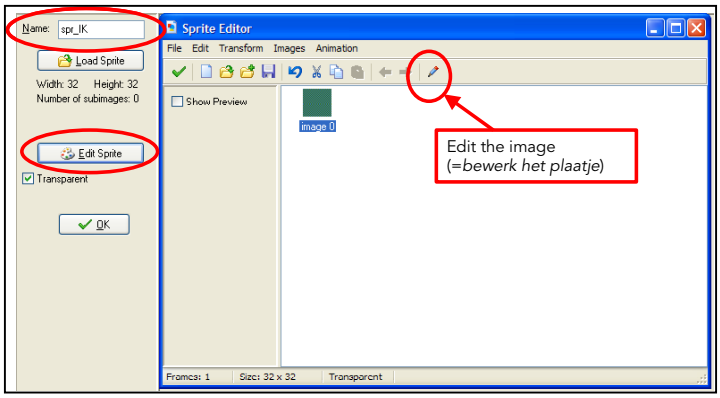
Hieronder volgt uitleg hoe je zelf een sprite kan maken.

12. Klik op **Create a Sprite** (= *maak een sprite*) op de knoppenbalk.



13. Geef de sprite de **naam spr_IK** en klik dan op **Edit sprite** (*Bewerk de sprite).*

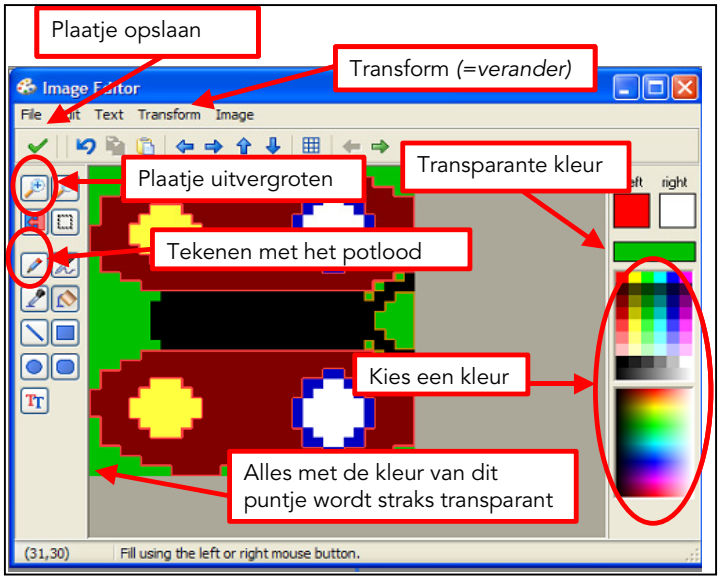
14. Klik op **Edit the image** (=*bewerk het plaatje*), het pennetje.



Je komt in een programma wat erg lijkt op Paint, wat je misschien kent.

15. Sprites moeten straks in een room precies in een hokje passen. Klik daarom op **Transform**, en dan op **Stretch**. Kies bij **Width 32 pixels**. Als er een vinkje bij 'Keep aspect ratio' staat wordt automatisch ook height 32 pixels.
16. Klik op **OK.**
17. Klik een paar keer op **Zoom in**, zodat je het plaatje vergroot weergeeft. Dan kun je preciezer tekenen.
18. Klik op **Draw on the image** (=*teken op het plaatje*)
19. Kies in het **rechtervak** een **kleur**.
20. **Teken** een plaatje. Lees voor het tekenen de tips op de volgende pagina.

Het plaatje zal er blokkerig uitzien, maar dat komt omdat je het plaatje hebt vergroot.

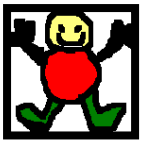


Een paar tips voor het maken van sprites

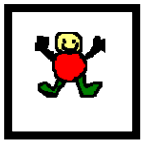
'Slecht ontworpen' sprites kunnen tot veel narigheid leiden. Hieronder tips:

- de sprites moeten precies in de hokjes van het speelveld passen. Een hokje is 32 bij 32. Dus de lengte en de breedte van een sprite moet in de tafel van 32 zitten. Lengte 64 mag (2x32) maar lengte 70 mag niet.
- de tekening van de sprite moet het vierkantje zoveel mogelijk opvullen, in ieder geval **meer dan de helft**.

Dus zo:

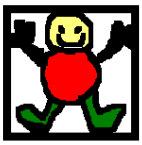


en niet zo:

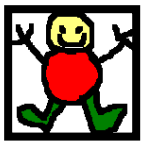


- De sprite mag **geen scherpe punten** hebben, zoals scherpe vingertjes of naaldhakken.

Dus zo:



en niet zo:



- De tekening van **spr_muur** moet het **hele vierkantje** opvullen.
- Het **puntje linksonder** in een sprite is de kleur die **doorzichtig** wordt als je bij de sprite **Transparent** een **vinkje** zet.
  Dus in de sprite hierboven zal alles wat wit is doorzichtig zijn. In jouw plaatje is dat puntje waarschijnlijk groen. Alles wat dan hetzelfde groen is in de tekening wordt dan straks doorzichtig. In het brede vak rechts zie je ook welke kleur de transparante kleur is. Wil je dat groen niet doorzichtig wordt, geef dan het puntje linksonder een andere kleur, een kleur die je niet in je tekening gaat gebruiken.
- Bij **obj-IK** en **obj-doel** zet je **wel** een vinkje, bij **obj_muur** zet je **geen vinkje** bij **Transparent**.
- Je kunt ook vormen gebruiken, zoals **cirkel**, **rechthoek** of **lijn**.
- Met de **verfbus** kun je een vorm opvullen met kleur.

21. Klik het groene vinkje om het tekeningetje. op te slaan als je tevreden bent. Klik dan op het groene vinkje om het plaatje op te slaan.
22. Vink bij **spr_IK transparent aan** en klik op **OK** om spr_ik op te slaan.
23. Maak zoals in opdracht 12 tot en met 20 ook sprite **spr_doel**.
24. Vink bij **spr_doel transparent aan** en klik op **OK**.
25. En maak zoals in opdracht 12 tot en met 20 ook nog sprite **spr_muur**.
26. Let op: vink bij **spr_muur transparent uit**. Klik op **OK**

Objecten maken

27. Maak **3 objecten** via **Create an object** . Noem ze **obj_muur**, **obj_IK** en **obj_doel**.
    a. Kies bij **elk object** de bijbehorende **sprite**.
    b. Vink bij **alledrie visible aan**.

Visible betekent zichtbaar. Als je visible niet aanvinkt is de sprite onzichtbaar wanneer je het spel speelt.

    c. Vink bij **obj_muur** en **obj_ik** ook **Solid** aan. Je moet namelijk niet door een muur kunnen lopen.
    d. Bij **obj_doel** moet **Solid niet aangevinkt** zijn, want je moet wel door de uitgang kunnen.

Je hoeft bij de drie objecten nog geen events toe te voegen.

    e. Klik bij elk van de 3 objecten op **OK** om het object op te slaan.

Een speelveld maken

We hebben nu objecten, ook al doen ze nog niks. Want je hebt nog geen events geprogrammeerd. Maar je kunt al wel speelvelden, rooms, gaan inrichten.

Een tweede level is niets anders dan een tweede speelveld of room. Als je het spel speelt zal eerst de bovenste in de lijst

worden geopend. Ga je naar een volgend level, dan wordt de volgende room in je lijst geopend.

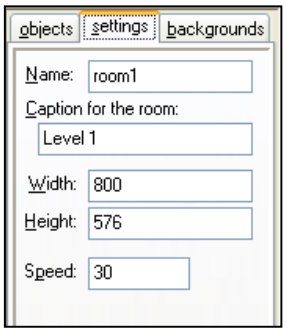
28. Maak een room, via **Create a room** .

Een paar tips voor het maken en inrichten van rooms

- zorg dat de **hokjes 32** bij **32** zijn, zodat de sprites er precies in passen
- zorg dat de lengte en breedte van het speelveld in de tafel van 32 zit, zodat je geen halve hokjes hebt. Bijvoorbeeld width (=*lengte*) **800** (=25x32) en height(=*hoogte*) **576** (=18x32).
- zorg dat de **paadjes** overal minimaal **2 hokjes** breed zijn

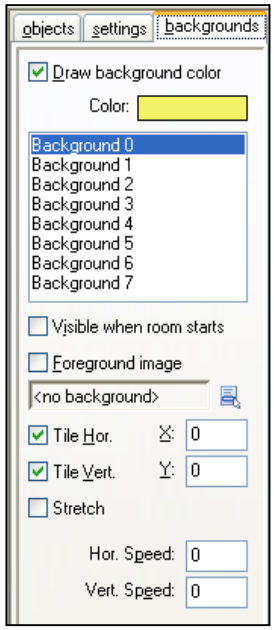
29. Spr_muur was 32 hoog en 32 breed. Kies daarom bij **Snap X 32** en bij **Snap Y 32** .

30. Klik op **tabblad Settings**.
    Vul bij **Caption for the room** een **naam** in, deze verschijnt straks boven in de blauwe titelbalk.
    Vul bij **Width 800** en bij **Height 576** in.



Je krijgt zo geen halve hokjes, want 800 en 576 zitten allebei in de tafel van 32.

31. Klik op **tabblad Backgrounds**.
    Kies bij **Color** een leuke **achtergrondkleur** voor je speelveld. Kies een kleur waartegen je sprites goed opvallen. (Dus geen rode achtergrond als je een rode sprite hebt gemaakt.)



32. Klik op het groene vinkje om de room op te slaan.

Een tweede level maken

De tweede room die je gaat maken, zal erg lijken op de room die je al gemaakt hebt. De instellingen zullen hetzelfde moeten zijn. Daarom is het handiger de eerste room te kopiëren.

33. Maak een **tweede room, een tweede level** door met je **rechtermuisknop** op **room1** te klikken en te kiezen voor **Duplicate**.
34. **Verander** in **room2** alleen op het tabblad **settings** de **naam** bij **Caption for the room**.

Nu ga je room1 inrichten met de objecten die je hebt.

35. Dubbelklik op **room1**. Kies object **obj_muur** en maak een rand met dit object om het speelveld heen. Laat ergens **één blokje open**, als **uitgang**.
36. **Maak** daarna een niet te moeilijk **doolhof**.
    Zorg dat de paadjes minimaal 2 hokjes breed zijn.
37. Plaats **obj_IK** in het speelveld, zo ver mogelijk van de uitgang en plaats **obj_doel** in de uitgang.
38. Klik op het groene vinkje om **Room1** op te slaan.
39. Richt op dezelfde manier **Room2** in, maar maak het doolhof iets moeilijker.
40. Klik op **Opslaan** .

Hoofdstuk 4: Basisobjecten voor een doolhofspel

Basisobjecten, voorbeeld bekijken

Je hebt al 3 objecten gemaakt. Je hebt de objecten ook al in de rooms geplaatst. Maar ze doen nog niets. De events en actions heb je nog niet geprogrammeerd. Dat ga je in dit hoofdstuk doen.

1. Als je je eigen spel nog niet had afgesloten, klik dan voor de zekerheid op **Opslaan** .
2. Ga naar de map Voorbeeldgames en **open** het bestand **Voorbeeld-1** .
3. Klik op **Run the game**

Beweeg met de **pijltjestoetsen**.

4. Wat gebeurt als het **hondje, obj_IK**, tegen een **muur, obj_muur**, botst?
   Als het obj_IK en de muur obj_muur botsen,
   dan.....
5. Wat gebeurt als het **hondje** in het eerste level **obj_doel, de pijl**, raakt?
   Als obj_IK en obj_doel botsen, **dan** .....
6. Wat gebeurt als het **hondje** in het tweede **level obj_doel** raakt?
   Als obj_IK in het tweede level obj_doel raakt
   1. **dan**.....
   2. en **dan** .....

7. **Sluit** met **<ESC>** het spel en keer weer terug in Game Maker.

Hoe zijn de basisobjecten geprogrammeerd?

8. **Dubbelklik** op object **obj_IK.** (zie je geen obj_IK klik dan op het plusje voor objecten)

9. Je ziet nu de events van obj_IK.

   Eén event die je waarschijnlijk nog niet kent is **No key**. (Key betekent toets)

   a. Wat zal 'No key' betekenen? .....

   b. Schrijf alle events en alle acties bij obj_IK in eigen woorden op.

      Gebruik in je omschrijving 'als' en 'dan.' Bijvoorbeeld:

      Als er op pijltje links wordt geklikt, **dan** gaat obj_IK naar links, met snelheid 6.

      Als.....

      dan.....

      Als.....

      dan.....

      Als.....

      dan.....

      Als.....

      dan.....

Als Snuf tegen een muur botst kun je dat op twee manieren programmeren:

- Bij object Snuf: als er een botsing is tussen Snuf en de muur.....
- Bij object Muur: als er een botsing is tussen de muur en Snuf.....

Bij het object IK, bijvoorbeeld Snuf, moeten toch al veel events worden geprogrammeerd. Daarom kiezen we er voor om de events tussen Snuf en een ander object vooral bij het andere object toe te voegen. Voordeel daarvan is dat je makkelijker kan zien welke event bij welk level hoort. Maar het is niet fout om het andersom te doen.

10. **Dubbelklik** in voorbeeld-1 op object **obj_muur**.

    Er staan bij de event 'Als obj_muur botst en obj_IK botsen' twee acties.

    a. Welke acties?

       1. .....

       2. .....

De tweede actie, Align to a grid (=*spring terug in een hokje*) is heel belangrijk in een doolhofspel. Als je deze actie vergeet, kan een object blijven 'kleven' en kan je niet verder spelen.

b. **Dubbelklik** op de actie **Align to a grid of 32 by 32**.
c. Waar zal '32 by 32' op slaan? .....
d. Welk object springt terug in een hokje na de botsing? Hoe zie je dat?
.....
e. Klik op **Cancel** om de actie 'Align to a grid' te sluiten.

Als volgend level bestaat, ga naar volgend level, anders...

11. **Dubbelklik** in voorbeeld-1 op object **obj_doel**.
    a. In opdracht 5 en 6 heb je beschreven wat er gebeurt als je in level 1 tegen obj_doel botst. Wat gebeurt er als je in level 2 tegen obj_doel botst? Beschrijf dat nog eens.

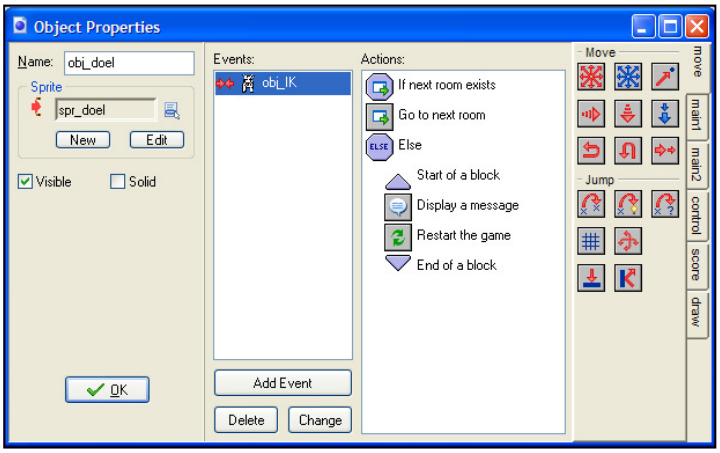
Als obj-doel en obj_IK botsen in level 1
dan .....

Als obj-doel en obj_IK botsen in level 2
dan .....
en dan .....

In voorbeeld-1 is dat heel slim opgelost.
Als obj_doel en obj_IK botsten dan wordt eerst gekeken of er een volgend level is. Als die er is, ga je naar het volgende level. Is er geen volgend level meer, dan heb je het spel dus uitgespeeld. Er verschijnt een berichtje en het spel start opnieuw.
Je programmeert dat als volgt (rechts staat de vertaling).

If next room exists	***Als*** *volgend speelveld bestaat dan*
Go to next room	*Ga* ***dan*** *naar het volgende speelveld*
Else	*Anders*
Start of a block	*Begin van een blok*
Display a message	Toon **dan** een berichtje
Restart the game	Start **dan** het spel opnieuw

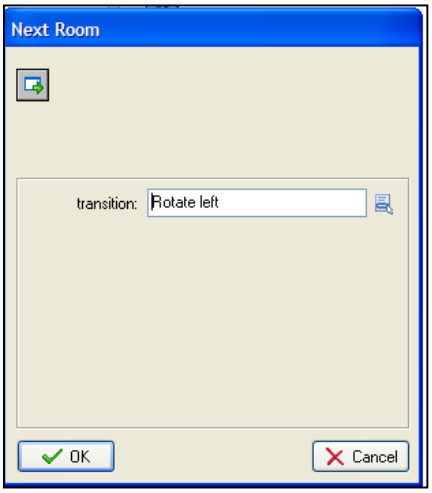
End of a block Einde van een blok



Dat je de twee regels in een blok zet is nodig, omdat anders het spel altíjd opnieuw start. Dat wil je niet. Je wilt dat dat alleen gebeurt wanneer er niet een volgend speelveld bestaat.

Bij **Go to next room**(=*ga naar het volgende speelveld*) kun je ook een leuk **effect** kiezen, bijvoorbeeld het draaien in voorbeeld-1.
Kies dan bij **Transition** (=*overgang*) **Rotate left**.

12. Dubbelklik op **Go to next room** en kijk bij de keuzelijst welke mogelijkheden er zijn.
13. Klik op **Cancel** om de actie Next Room te sluiten



Samenvatting

Belangrijk:

- Bij de ik-figuur programmeer je de events voor besturing (pijltje links, pijltje rechts enzovoorts).
- Je programmeert ook de event dat als er geen pijltje ingedrukt word de ik-figuur stilstaat
- Bij een muur moet je na een botsing met een object het object altijd naar een hokje terug laten springen. Anders blijft het object 'kleven'.
- Als je het doel hebt bereikt, ga je naar het volgende level. Als er geen volgend level meer is, start het spel bijvoorbeeld opnieuw.

Zelf basisobjecten ontwerpen

14. Maak voor je eigen objecten een ontwerp.
    Wat moet je allemaal programmeren. Schrijf hier ook zaken op als welke snelheid je ik-figuur moet krijgen, welk berichtje er moet verschijnen....

    obj_IK

    Als.....

    dan.....

    Als.....

    dan.....

    Als.....

    dan.....

    Als.....

    dan.....

    Als.....

    dan.....

    obj_muur

    Als.....

    dan.....

Als.....

dan.....

obj_doel

Als.....

dan.....

Als.....

dan.....

Anders

dan .....

.....

.....

.....

Zelf je basisobjecten programmeren

15. Ga naar de map MijnGames en **Open** Doolhof-Naam-1. Klik op **File** *(= Bestand)* en daarna op **Save As...** (=*Sla op als...)*. Sla het bestand op in de map **MijnGames** met de bestandsnaam **Doolhof-Naam-2.**
16. **Programmeer** de events bij **obj_IK**.

Belangrijk is elk stapje, dat je programmeert, te testen. Daarvoor gebruik je een testschema. In het **laatste hoofdstuk** vind je als **bijlage** een testschema dat je kunt invullen.

17. Wat kun je nu al testen? **Vul** dat in het **testschema in**. De eerste regels zijn al ingevuld.
18. Klik op **Run the game** en test aan de hand van je **testschema** of alles werkt zoals zou moeten. De kruisjes zet je pas als je de eindtest doet.
19. Doe dezelfde stapjes voor obj_muur en obj_doel. Dus:
    a. **Programmeer** de events bij het object.
    b. Werk je **testschema** bij.

c. Klik op **Run the game** en test aan de hand van je **testschema** of alles werkt zoals zou moeten.

20. Werkt alles zoals zou moeten? Dan kun je verder met het volgende hoofdstuk. Zoniet, verbeter het dan en test opnieuw.

21. Klik op **Opslaan**.

Hoofdstuk 5: Beloningen, punten scoren

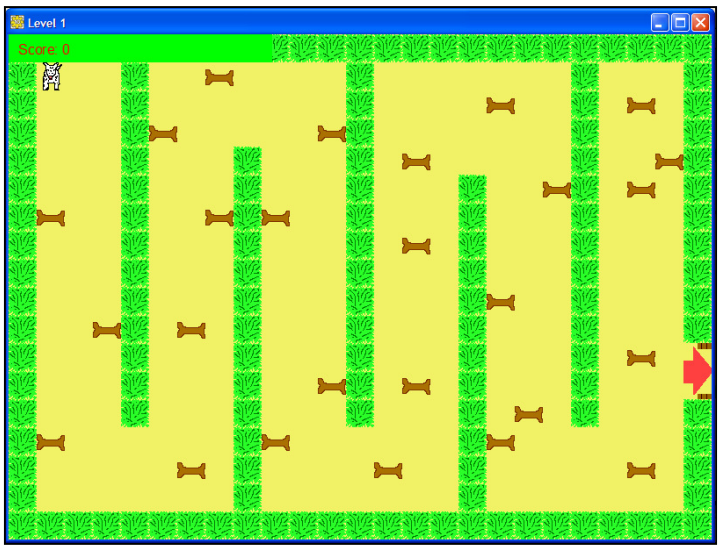
Beloningen ontwerpen, voorbeeld bekijken

Het duurt soms best lang om een level uit te spelen. Daarom is het leuk als de speler ook tussendoor wordt beloond. Belonen kan op meerdere manieren. Vaak wordt je beloond met punten, bijvoorbeeld door dingen te verzamelen. Sommige beloningen leveren meer punten op dan andere. Als je het spel uit hebt gespeeld dan wordt de hoogste-score-tabel getoond.

1. **Start Game maker of** had je nog je spel geopend, klik dan op **Opslaan** ..
2. Open uit de map Voorbeeldgames **Voorbeeld-2**.
3. Klik op **Run the game** .

In het spel uit voorbeeld-1 zijn beloningen toegevoegd. De beloningen zijn in het doolhof gezet.

4. Bekijk **level 1**.



Je ziet daar **botten**.

Wat gebeurt als je een bot aanraakt?

a. **Als** je een bot aanraakt, **dan**

.....

.....

Ga naar level 2. Je ziet daar vlinders.

b. Wat gebeurt als je een **rode vlinder** aanraakt?

Als je een rode vlinder aanraakt, **dan**

.....

.....

c. Wat gebeurt als je een **gouden vlinder** aanraakt?

Als je een gouden vlinder aanraakt, dan

.....

.....

Hoe zijn de beloningen geprogrammeerd?

5. Klik op **<ESC>** om het spel af te sluiten en terug te keren naar Game Maker.
6. **Dubbelklik** op **obj_bot** om de events te bekijken.

   Had je vraag 4a) goed? Zo nee, welke actie(s) was je vergeten?

   .....

   .....

7. Even een puntje over score.

   a. Met welke actie zorg je ervoor dat je er telkens **punten bij krijgt**?

   .....

   b. Moet je dan **relative aanvinken** of niet? .....

8. Wanneer moet de hoogste-score-lijst worden getoond? Bij welk object en event is dat geprogrammeerd? .....
9. Dubbelklik op **obj_controller**.

   Weet je nog?

   a. Bij het spel Beertje heb je geleerd hoe je er voor kon zorgen dat de hoogste-score-tabel weer leeggemaakt kan worden. Met welke event gebeurt dat?

      Als .....**dan**.....

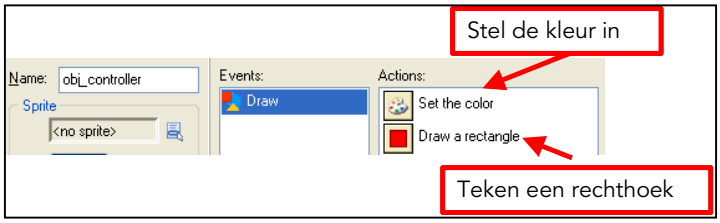
   b. Bij het spel Beertje heb je geleerd hoe je er voor kon zorgen dat de score in het scherm wordt getoond. Bekijk de acties bij event **Draw**. Door welke actie wordt de score in het scherm getekend?

      .....

Tekenen in je scherm met actie Draw rectangle

Als je de score en levens in het scherm laat tekenen, vallen ze niet altijd op tegen de achtergrond. In voorbeeld 2 zie je als je het spel speelt, dat de score in een groene rechthoek staat. Zo valt de score goed op.

Er zijn bij event **Draw** twee extra acties toegevoegd. Namelijk: **Set color** (=*stel de kleur in*) en **Draw rectangle** (=*teken een rechthoek*). De **kleur** wordt ingesteld op **groen**. Met **Draw rectangle** wordt er een groene **rechthoek** getekend.



Daarna wordt de kleur met **Set color** op **rood** ingesteld en wordt de **score** in rood weergegeven. Die rechthoek is niet perse nodig, maar hier gedaan zodat de score beter opvalt.

In level 2 zijn er twee soorten vlinders, rode en gouden. Gouden vlinders leveren meer punten op. Het aantal rode en gouden vlinders blijft steeds hetzelfde. Na een botsing verdwijnt een vlinder niet, maar springt weg.

10. Ga naar **level 2** van **voorbeeld-2**.
11. Dubbelklik op **obj_vlinder.** Je ziet vier events.
    a. Klik op event **Create** (= *ontstaan*). En **dubbelklik** daarna op **Start moving in a direction.** Wat is hier geprogrammeerd?

       Als obj_vlinder ontstaat

       dan .....
    b. Klik op **Collision with obj_muur** (=*botsing met obj-muur*).

       Wat is hier geprogrammeerd?

       Als obj_vlinder botst met obj_muur

       dan .....
    c. Klik op **Collision with obj_IK** (=*botsing met obj-IK*).
    d. Wat is bij de eerste actie geprogrammeerd?

       Als obj_vlinder botst met obj_IK

       dan .....
    e. Wat is bij de tweede actie geprogrammeerd?

       Als obj_vlinder botst met obj_IK

       dan .....

Samenvatting

- Bij ontwerpen van beloningen, objecten die punten opleveren, kun je denken aan:
    - **Objecten** die **stilstaan** of die **heen en weer** of **op en neer** bewegen
    - Verschillende objecten die een **verschillend** aantal **punten** opleveren

- Na een **botsing** moeten een van beide altijd **wegspringen** of **verdwijnen** (opgegeten worden), anders blijven ze kleven en krijg je wel heel veel punten
- Over de score:
    - als de ik-figuur tegen een 'beloning' botst krijg je **punten**: actie Set Score, **relative** aangevinkt
    - De **score** in het scherm laten **tekenen**: maak bij obj_controller een event Draw en voeg van tabblad score de actie 'Draw score' toe. Valt de score niet op tegen de achtergrond, zet hem dan in een rechthoek met de actie 'Draw rectangle'.
    - Een **hoge-score-tabel** krijg je door bij obj_doel bij event botsing met obj_ik na de actie Else de actie Show the higscore table toe te voegen
    - De **hoge score tabel** weer **leeg** kunnen maken programmeer je door bij obj_controller bij event <CTRL> de actie Empty highscore (*hoge-score-tabel leegmaken*).

Zelf beloningen ontwerpen

Je gaat nu zelf 2 'beloningen' ontwerpen.

12. Ontwerp twee ' beloningen ' die jij wilt gaan maken.
    Wat moeten ze allemaal doen. Schrijf hieronder bij elk object op welke events je moet programmeren. (Als.....dan.....). Weet je niet meer precies hoe het moet, kijk dan bij de objecten in voorbeeld-2.

a. Beloning 1:

Sprite: spr_..... Events:

Als

.....

.....

.....

.....

.....

b. **Beloning 2**:

Sprite: spr_..... Events:

Als

.....

.....

.....

....

13. Schrijf hieronder ook op welke events en acties je in het **obj_controller** moet opnemen.

Als .....

.....

14. En bij **obj_doel**:

Als .....

.....

Zelf je beloningen programmeren

15. Ga naar de map MijnGAmes en **open** Doolhof-Naam-2. Klik op **File** *(= Bestand)* en daarna op **Save As...** (=*Sla op als...)*. Sla het bestand op in de map **MijnGames** met de bestandsnaam **Doolhof-Naam-3.**

Eerst ga je één beloning programmeren en testen. Als die werkt, begin je pas aan je tweede beloning.

16. Begin met **obj_controller** te programmeren
    a. Maak een object
    b. Plaats het object in elke room
    c. Programmeer de events.
    d. Wat kun je testen? **Vul** dat in het **testschema in** achter in je boek.
    e. Klik op **Run the game** en test aan de hand van je **testschema** of alles werkt zoals zou moeten.
17. Maak dan de eerste beloning, **Beloning 1**, helemaal af.
    a. Maak een **sprite** voor **Beloning 1**.
    b. Maak er een **object** van.
    c. Kies bij het object de **sprite** van de beloning.
    d. Vink **visible** en **solid aan**.
    e. Plaats de beloning bijvoorbeeld twee keer in **room 1**.

Je hoeft nu nog niet na te denken hoe vaak en waar het object in de room moet komen. Het gaat er nu om dat je gaat testen of het object goed werkt. Later, als alle objecten klaar zijn ga je de rooms wel beter inrichten en erop letten of het niet te makkelijk of te moeilijk is.

f. **Programmeer** de **events** bij je eerste beloning.

Test zoveel mogelijk elke event of het werkt. Dus als je bijvoorbeeld de event hebt botsen met obj_IK, dan kun je dat testen en kijken of het doet wat het moet doen. Dan krijg je als testregel bijvoorbeeld: **Als** de beloning en de ik-figuur botsten **dan** komen er bij de score 5 punten bij.

g. Wat kun je dus allemaal testen? **Vul** dat in het **testschema in** achter in je boek.

h. Klik op **Run the game** en test aan de hand van je **testschema** of alles werkt zoals zou moeten.

18. Doe hetzelfde voor **beloning 2.**

19. Werkt alles zoals zou moeten? Dan kun je verder met het volgende hoofdstuk. Zoniet, verbeter het dan en test opnieuw.

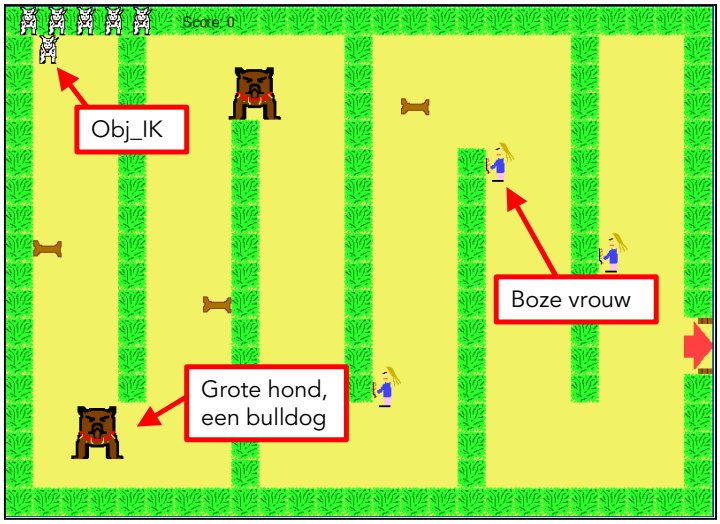
20. Klik op **Opslaan**.

Hoofdstuk 6: Monsters

Monsters ontwerpen, voorbeeld bekijken

De basis van je spel is klaar en er zijn beloningen toegevoegd. Maar het is nogal makkelijk, want zonder problemen haal je je doel. Daarom wordt het tijd wat **hindernissen** in te bouwen. Veelgebruikte hindernissen zijn **'monsters'**. Het hoeven niet echt monsters te zijn, maar we noemen ze even zo voor het gemak. Het zijn dingen of personen die je tegenwerken, die een hindernis vormen. Als je er tegenaan loopt verlies je een **leven**. Zijn al je **levens op**, **dan** begin je **opnieuw**.

1. **Start Game maker of** had je nog je spel geopend, klik dan op **Opslaan** ..
2. Open uit de map Voorbeeldgames **Voorbeeld-3**.
3. Klik op **Run the game** .



In het spel uit zijn nu hindernissen toegevoegd. De hindernissen, 'monsters' bewegen door het doolhof. In level 1 zijn de grote honden en de boze buurvrouwen de 'monsters'.

4. Bekijk **level 1**.

5. Je ziet dat de **grote honden** heen en weer lopen.

   Bij de honden zijn **3 events** geprogrammeerd. Kun jij zonder spieken bedenken wat de bijbehorende **acties** zijn? Let op, bij sommige events is er meer dan één actie.

   a. **Als** het spel begint, **dan**

      .....

      .....

      ....

   b. **Als** de hond tegen obj_muur botst, **dan**

      ....

      .....

      ....

   c. **Als** de hond tegen obj_IK botst,

      .....

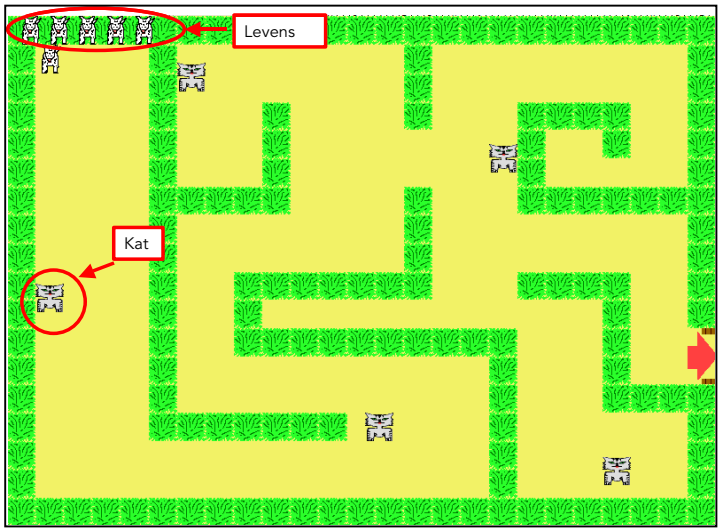
      .....

      .....

6. Je ziet ook **boze vrouwen**. Die bewegen op en neer. Moeilijke vraag misschien, maar kun jij bedenken hoe het komt dat een **boze vrouw** niet helemaal naar de overkant gaat maar **ergens onderweg** weer **omkeert**?

   .....

7. Probeer het **volgende level, level 2** te halen.



In level 2 zie je katten, die lopen overal zomaar door het doolhof.

Hoe zijn de monsters geprogrammeerd?

8. Klik op **<ESC>** om het spel af te sluiten en **terug** te keren naar **Game Maker**.

9. **Dubbelklik** op **obj_bulldog** om de events te bekijken.

   Had je vraag 5 goed? Zo nee, welke actie(s) was je vergeten bij welk event(s)?

   .....

   .....

   .....

   .....

   .....

10. Even een puntje over **levens**.
    a. Met welke actie zorg je ervoor dat er telkens **een leven af gaat**?

       .....

    b. Moet je dan **relative aanvinken** of niet? .....

Onzichtbare muurtjes

11. **Dubbelklik** op **Room1**. Je ziet in het doolhof een paar grijze blokken. Deze blokken zag je niet toen je het spel speelde. Hoe kun je de blokken wel zichtbaar maken?

    .....

12. Dubbelklik op **obj_bozeVrouw**. Door welke event keert de **boze vrouw** onderweg zo maar weer om? .....

In level-2 lopen katten door het hele doolhof? Dit programmeren is best lastig. Daarom is dit uitgelegd in het hoofdstuk: 'Extra trucs', paragraaf 'objecten willekeurig rond laten lopen'. Wil je toch heel graag dit soort 'monsters', neem dit hoofdstuk dan door.

13. In level 1 en ook in level 2 zie je dat je **begint met 5 levens**. Die zijn in het scherm weergegeven.
    a. Weet je nog hoe je dat kon programmeren?

       Vul in: **Object** .....

       Events:

       Als .....**dan** .....

       Als .....**dan** .....
    b. Bij ditzelfde object is ook geprogrammeerd wat er gebeurt wanneer je **levens op** zijn. Welke acties gebeuren er dan?

       Als je geen levens meer hebt,

       dan .....

       en **dan** .....

Samenvatting

- voorbeelden van 'monsters' zijn:
  - een monster wat tussen twee muren heen en weer of op en neer loopt.
  - een monster wat tussen onzichtbare muren heen en weer of op en neer loopt.
  - achter in je boek in het hoofdstuk 'trucs', paragraaf 'objecten willekeurig rond laten lopen' staat hoe je monsters willekeurig door een doolhof kunt laten lopen
- na een botsing moet een van beide altijd wegspringen of verdwijnen, anders blijven ze kleven en ben je in een keer al je levens kwijt
- over levens:

- je begint altijd met een aantal levens: maak bij obj_controller event create, actie Set lives.
- je kunt de levens als plaatjes in je scherm laten tekenen: maak bij obj_controller event Draw, actie Draw life images.
- als de ik-figuur tegen een 'monster' botst verliest hij een leven: actie Set lives, new lives -1, relative aangevinkt
- als je geen levens meer hebt is het Game over en begint het spel opnieuw: maak bij obj_controller event No more lives, actie Message (Game over!) en actie Restart the game.

- ook heen en weer lopende monsters moeten bij botsing met een muur naar een hokje terugspringen. Actie 'Align to a grid'

Zelf monsters ontwerpen

Je gaat nu zelf 2 'monsters' ontwerpen.

14. Bedenk twee 'monsters' die jij wilt gaan maken. Wat moeten ze allemaal doen.
    a. Schrijf hieronder bij elk object op welke events je moet programmeren. (**Als.....dan.....**). Weet je niet meer precies hoe het moet, kijk dan bij de objecten in voorbeeld-3.

    Monster 1:

    Sprite: spr_..... Events:

    Als

    .....

    .....

    .....

    .....

    .....

    .....

    .....

.....

b. **Monster 2**:

Sprite: spr_..... Events:

Als

.....

.....

.....

.....

.....

.....

.....

.....

15. Schrijf hieronder ook op welke events en acties je in het obj_controller moet opnemen.

Als .....

.....

.....

.....

Zelf je monsters programmeren

16. **Open** Doolhof-Naam-3. Klik op **File** *(= Bestand)* en daarna op **Save As...** *(=Sla op als...)*. Sla het bestand op in de map **MijnGames** met de bestandsnaam **Doolhof-Naam-4.**

Eerst ga je een monster programmeren en testen. Als die werkt, begin je pas aan je tweede monster.

17. Probrammeer de events en actions bij **obj_controller**.
    a. Wat kun je testen? **Vul** dat in het **testschema in** achter in je boek.

b. Klik op **Run the game** en test aan de hand van je **testschema** of alles werkt zoals zou moeten.

22. Maak dan het eerste monster, **Monster 1**, helemaal af.
    a. Maak een **sprite** voor **monster 1**.
    b. Maak een **object**. voor **monster1**.
    c. Kies bij het object de **sprite** van het monster.
    d. Vink **visible** en **solid aan**.
    e. Plaats het monster in **room 1**.
    f. **Programmeer** de **events** bij je eerste monster.

Test zoveel mogelijk elk stapje of het werkt.

g. Wat kun je testen? **Vul** dat in het **testschema in** achter in je boek.

h. Klik op **Run the game** en test aan de hand van je **testschema** of alles werkt zoals zou moeten.

23. Doe hetzelfde voor **monster 2.**
24. Wil je ook iets met **onzichtbare muren**?
    a. Maak dan een object obj_onzmuur, plaats die in de room en programmeer de bijbehorende events. Vergeet niet bij het object Visible uit en Solid aan te vinken.
    b. Werk het testschema bij met testregels.
    c. Test of het goed werkt.
25. Werkt alles zoals zou moeten? Dan kun je verder met het volgende hoofdstuk. Zoniet, verbeter het dan en test opnieuw.
26. Klik op **Opslaan**.

Hoofdstuk 7: Niet zo maar naar het volgende level

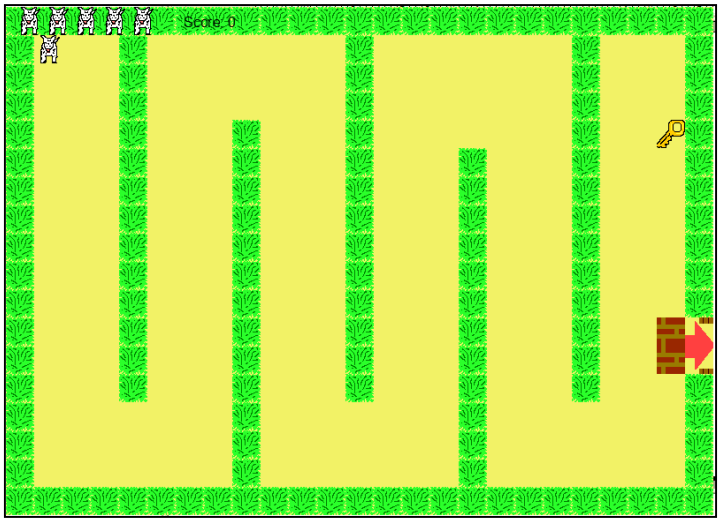
Hindernis om naar het volgende level te gaan, voorbeeld bekijken

In het vorige hoofdstuk heb je 'monsters' als hindernissen geprogrammeerd. Maar er zijn nog veel meer mogelijke hindernissen.
Bijvoorbeeld een extra hindernis om naar het volgende level te komen.
Voorbeelden van zo'n extra hindernis voordat je naar het volgende level mag:

- je moet eerst een sleutel pakken of
- je moet eerst een muurtje opblazen of
- eerst moeten alle botten op zijn of
- je moet eerst 50 punten hebben gehaald

1. **Start Game maker of** had je nog je spel geopend, klik dan op **Opslaan** ..
2. Open uit de map Voorbeeldgames **Voorbeeld-4**.
3. Klik op **Run the game** .

In het spel uit zijn de beloningen en hindernissen uit de Rooms weggehaald, zodat we alleen hoeven te kijken naar de extra hindernis om naar het volgende level te komen.



4. Bekijk **level 1**.

   Je ziet dat je niet door de uitgang kan, want er staat een muurtje voor.
   Pak de **gouden** sleutel. Let op wat met de sleutel en wat met het muurtje gebeurt.
   Dus **Als** de **gouden sleutel** en de ik-figuur botsen,
   dan.....
   en dan.....
5. Ga naar **level 2**. Je ziet nu niet het doel, maar een deur.
   Pak de **rode** sleutel. Let op wat met de sleutel en wat met de uitgang gebeurt.
   Dus **Als** de **rode sleutel** en de ik-figuur botsen,
   dan.....
   en dan.....
6. Ga naar **level 3**. Je ziet dat er weer een muurtje voor de uitgang staat. Nu is er geen sleutel, maar is er een zogenaamde **trigger**, waarmee je iets mee kunt opblazen.
   Pak de **trigger**. Let op wat met de trigger en wat met het muurtje gebeurt.
   Dus **Als** de trigger en de ik-figuur botsen,
   dan.....
   en dan.....
7. Ga naar **level 4**. Je ziet nu weer een deur in plaats van een uitgang. Verder liggen er botten. Pak alle botten. Let op wat er gebeurt als je het laatste bot oppakt. Vul in:
   Als .....
   dan.....

8. Ga naar **level 5**. Je ziet ook nu weer een deur in plaats van een uitgang. Verder vliegen er vlinders. Pak de vlinders. Let bij elke vlinder die je pakt of er iets gebeurt met de deur. Zodra er iets gebeurt, kijk dan wat de score is.
   Vul in:

Als .....

dan.....

Hoe zijn de hindernissen geprogrammeerd?

9. Klik op **<ESC>** om het spel af te sluiten en **terug** te keren naar **Game Maker**
10. Dubbelklik op obj_goudenSleutel. Welke acties zijn geprogrammeerd bij botsing met obj_IK? Vul in:

    Als obj_goudenSleutel en obj_IK botsten,

    dan .....

    en dan .....

De actie 'Een object verandert in een ander object'

In het tweede level veranderde de deur in een pijl, zodra je de rode sleutel pakte. De deur heeft ander gedrag dan de pijl. Je kunt niet door de deur, maar wel door de pijl. En als je daar doorheen gaat, ga je naar het volgende level. Dit betekent dat je voor deur een apart object moet maken met events die anders zijn dan die van de pijl.

11. Dubbelklik op **obj_deur**. Bekijk de event **Collision with obj_IK**. Wat is hier geprogrammeerd?

    Als obj_deur en obj_IK botsen,

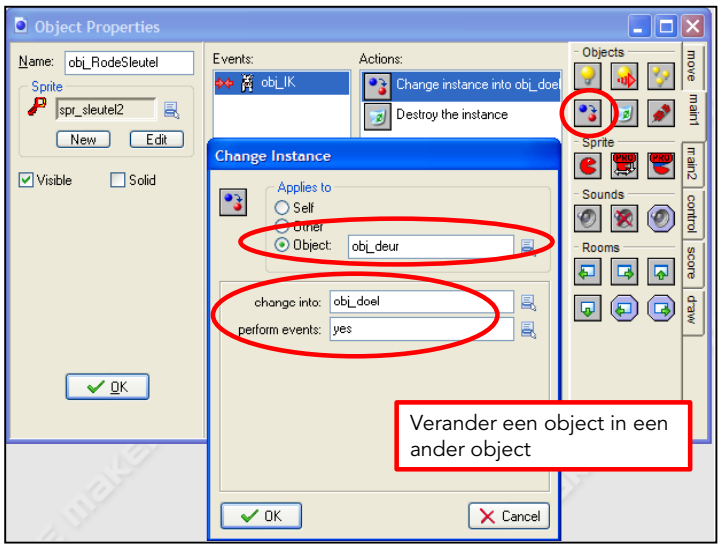
    dan .....

    en dan .....

    Merk ook op dat **Solid** is aangevinkt.
12. Dubbelklik op **obj_rodeSleutel**.

Daar zie je bij event **Collision with obj_IK** de actie **Change instance in obj_doel**.

13. Dubbelklik op deze actie.



'Change instance in obj_doel' betekent 'verander de instantie in obj_doel'. Een instantie is eigenlijk niet hetzelfde als een object. Maar voor nu mag je dat wel zo opvatten. Dit betekent dus dat het object wordt veranderd in een ander object.

a. Welk object wordt veranderd in welk ander object?

.....

b. Dus: **Als** obj_rode sleutel en obj_IK botsen,

dan .....

Let op dat **perform events** (=*events uitvoeren*) op **Yes** staat. Want je wil wel dat obj_doel de events uitvoert zoals bij botsing met obj_IK naar volgende level.

c. Klik op **Cancel** om de actie weer te sluiten.

Event 'Als de animatie is afgelopen'

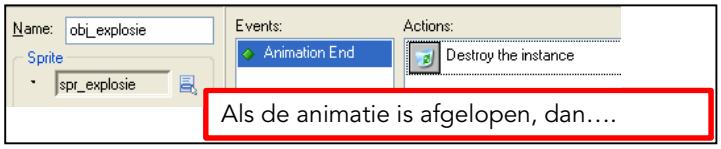
Dit principe wordt ook gebruikt in level 3. Daar verandert het muurtje in een explosie als de 'trigger' en de IK-figuur botsen.

De sprite van obj_explosie is een animatie, een filmpje.

In het hoofdstuk Extra trucs leer je hoe je zo'n animatie zelf kan maken.

14. Dubbelklik op obj_explosie.

    Je ziet daar de event **Animation end** (=*als de animatie is afgelopen*).



Als het muurtje is veranderd in een explosie, zou die explosie maar blijven herhalen. Dat wil je niet, als het filmpje één keer heeft gedraaid, moet het stoppen, moet de explosie verdwijnen. Daarom gebruiken we bij obj_explosie de event Animation end.

15. Vul in: **Als** de animatie is afgelopen,

    dan.....

De actie 'Als het aantal objecten gelijk is aan nul'

In level 4 veranderde de deur in een pijl zodra alle botten op waren. Er werd blijkbaar steeds geteld hoeveel botten er nog waren.

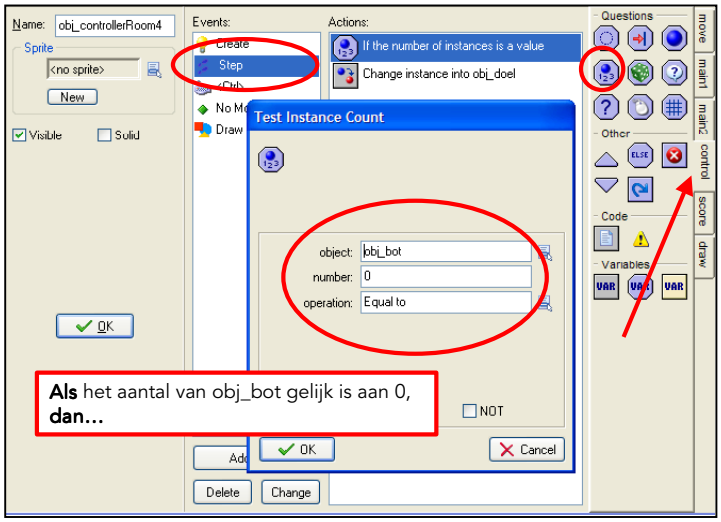
16. Kijk bij de **objecten**. Je ziet dat er nu per room een controller object is.

Dat is niet per se nodig, maar voor de uitleg nu wel even zo gedaan.

17.

    a. Dubbelklik op **obj_controllerRoom4**.

       Je ziet daar een nieuwe event **Step**.



Step wil zeggen, dat de acties die er bij horen bij elk stapje worden gedaan.
Bij de event Step zie je de actie **Test instance count**. Dit betekent zoiets als, tel het aantal objecten.

b. Dubbelklik op de actie **Test instance count**.

Bij **object** staat het object wat geteld moet worden.
Bij **number** (=*aantal*) staat het aantal waarbij er iets moet gebeuren, nu dus 0.
Bij **operation** staat nu **Equal to** (=*gelijk aan*). Maar hier kan ook Larger then (=*groter dan*) of Smaller then (=*kleiner dan*) worden gekozen. Dus in totaal staat er dus:
Als het aantal objecten obj_bot gelijk is aan 0...

c. Wat gebeurt er als het aantal objecten obj_bot gelijk is aan nul?

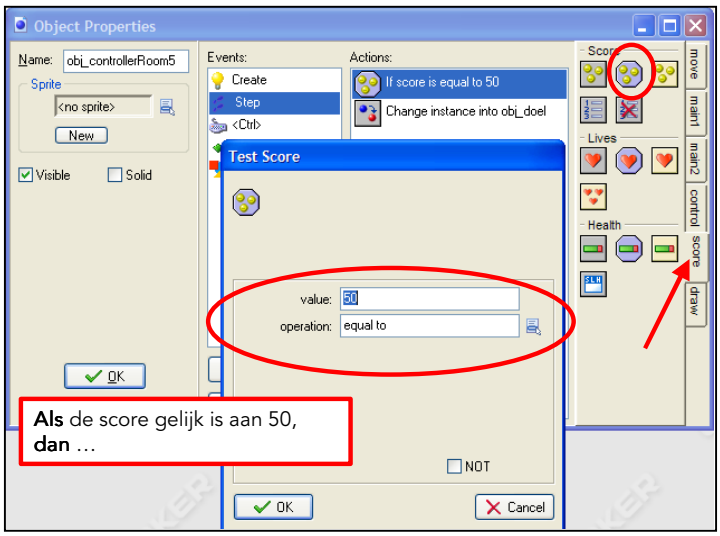
.....

d. Klik op **Cancel** om de actie te sluiten.

De actie 'Als de score gelijk is aan...'

In de vorige paragraaf heb je geleerd hoe je kunt programmeren dat je pas naar een volgend level kan als objecten op zijn.
Een andere mogelijkheid is dat je pas naar een volgend level kan als je een bepaalde score hebt gehaald. In level 5 moest je 50 punten halen om naar het volgende level te kunnen.

18. Dubbelklik op obj_controllerRoom5. Je ziet daar ook de event Step.



19. Bij deze event staat de actie **Test score**.
    a. Klik op de event Step en dubbelklik dan op de actie Test score.

Bij **value** (=*waarde*) staat de score waarbij er iets moet gebeuren, nu dus **50**.
Bij **operation** staat nu **Equal to** (=*gelijk aan*). Maar hier kan ook Larger then (=*groter dan*) of Smaller then (=*kleiner dan*) worden gekozen. Dus in totaal staat er dus:
Als de score gelijk is aan 50...

    b. Klik op Cancel om de actie te sluiten.

Samenvatting

- Voorbeelden van hindernissen om naar het volgende level te kunnen zijn:
  - je moet eerst iets pakken, zoals een **sleutel**, zodat een **muurtje** voor de uitgang **verdwijnt**
  - je moet eerst iets pakken, zoals een **sleutel**, zodat een **deur** waar je niet doorheen kunt **verandert** in een deur (pijl) waar je wel doorheen kunt.
  - je moet eerst naar een **trigger** om een muurtje voor de uitgang **op** te **blazen**
  - eerst moet je bepaalde **objecten,** zoals de botten, **allemaal pakken**
  - eerst moet je een **bepaalde score halen**
- Met de **event Step** kun je een actie bij elke stap laten uitvoeren, zoals objecten tellen of score controleren.
- Met de **event Animation end**, kun je een actie laten uitvoeren zodra de animatie is afgelopen
- Ook hier geldt, als een bewegend object tegen een stilstaand object botst, moet het bewegende object terugspringen in een hokje. (**Align to a grid**)

Zelf nog meer hindernissen ontwerpen

Je gaat nu zelf hindernissen ontwerpen om naar het volgende level te gaan.

Als je nog nieuwe objecten nodig hebt, moet je daarvoor eerst een sprite maken.

20. Welke hindernissen wil je gaan maken? Welke events zijn er nodig? Bij welke objecten. Schrijf dat hieronder allemaal op. Vergeet niet de events bij je controller-object.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Zelf je hindernissen programmeren

21. **Open** Doolhof-Naam-4. Klik op **File** *(= Bestand)* en daarna op **Save As...** *(=Sla op als...)*. Sla het bestand op in de map **MijnGames** met de bestandsnaam **Doolhof-Naam-5.**

    a. Programmeer de events bij een object.

    Test zoveel mogelijk elk stapje of het werkt.

    b. Wat kun je testen? **Vul** dat in het **testschema in** achter in je boek.

    c. Klik op **Run the game** en test aan de hand van je **testschema** of alles werkt zoals zou moeten.

22. Werkt alles zoals zou moeten? Dan kun je verder met het volgende hoofdstuk. Zoniet, verbeter het dan en test opnieuw.
23. Klik op **Opslaan** .

Hoofdstuk 8: Opbouw van makkelijk naar moeilijk

Levels bijwerken en testen

Je hebt nu de objecten geprogrammeerd en getest. Maar is je spel niet te moeilijk? Is het wel moeilijk genoeg dat het ook spannend is? Heb je er op gelet dat het eerste level makkelijk is en het tweede level moeilijker? Dat ga je nu aanpassen.

1. **Open** Doolhof-Naam-5. Klik op **File** *(= Bestand)* en daarna op **Save As...** *(=Sla op als...)*. Sla het bestand op in de map **MijnGames** met de bestandsnaam **Doolhof-Naam-6.**
2. Pas level 1 zo aan dat het niet moeilijk is om het volgende level te halen. Test het level. Ben je tevreden, ga dan verder.

Testen van level 2 wordt lastig als je steeds eerst level 1 moet halen, voordat je naar level 2 kan.
Hoe kan je dat makkelijker maken? Door het **level** waar je aan werkt, **boven aan de lijst** te zetten.

3. Klik met je muis op Room 2. Hou de muisknop ingedrukt en **sleep Room 2 naar boven aan de lijst**.

Als je nu klikt op Run begint het spel in level 2.

4. Pas level 2 zo aan dat het wat moeilijker is om het volgende level te halen. Test het level. Ben je tevreden, ga dan verder.
5. Level 2 af? Sleep het level dan weer terug naar de juiste plek.

Wil je nog meer levels maken dan kun je hiervoor de objecten gebruiken die je al hebt gemaakt. Kijk ook bij trucs voor meer mogelijkheden.

6. Klik op **Opslaan** .

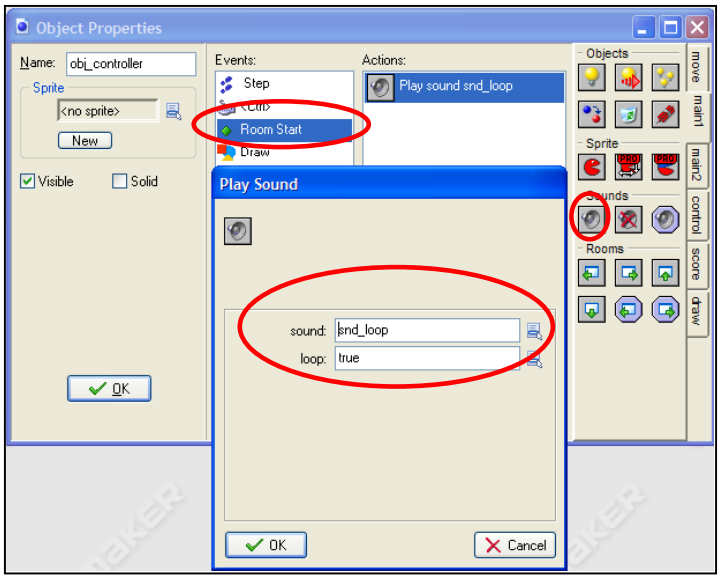
Hoofdstuk 9: Je spel afmaken

Geluiden

Achtergrondmuziek

Een achtergrondmuziekje bij je spel is ook leuk. Bij games wordt meestal een zogenaamde **loop** gebruikt De melodie houdt nooit op, maar wordt steeds herhaald.

1. **Open** Doolhof-Naam-6. Klik op **File** *(= Bestand)* en daarna op **Save As...** *(=Sla op als...)*. Sla het bestand op in de map **MijnGames** met de bestandsnaam **Doolhof-Naam-7.**
2. Maak met de knop **Create a sound** een nieuw geluid.
3. Geef het geluid de naam **snd_loop.**
4. Klik op **Load sound** en ga naar de map **Gamemakerbestanden**.
5. Ga dan naar de map **Sounds**. Klik op **loop** en klik op openen.
6. Klik op **OK** om het geluid op te slaan.
7. Dubbelklik op **obj_controller**
8. Klik op **Add event** en kies **Other** en dan **Room start**.
9. Voeg bij event **Room start** de **actie Play sound** toe. (tabblad main1)
10. Kies bij **loop true.** Klik op **OK.**



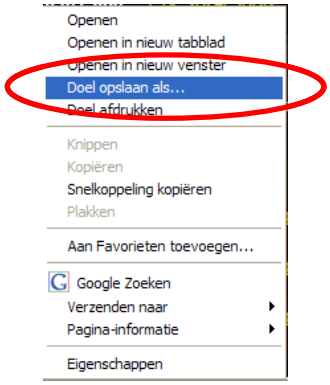
11. Klik op Run om te **testen** of het geluid goed werkt. Voeg de testregels toe aan je **testschema**.

Je zou als loop ook een mp3 kunnen gebruiken, maar dan duurt het downloaden en laden van je spel erg lang.

Je kunt dan beter loops van internet downloaden.

Een aantal sites waar je geluiden kunt downloaden vind je op www.gamemakerkids.nl. Let altijd op dat je alleen gratis loops gebruikt (free downloads). Op de site www.gamemakerkids.nl vind je bijvoorbeeld een link naar de site van net4TV met loops. Die zijn gratis te downloaden.

12. Klik **op de site www.gamemakerkids.nl** op de **link Downloads** en dan op **Loops**. Je komt nu op de site waar je melodieën kunt downloaden.
13. Klik met je **rechter muisknop** op een muziekje en klik op **Doel opslaan als**.
14. Sla de loop op in de map **MijnGames**.
15. Maak een **sound** met deze loop.
16. Kies bij **obj_controller** bij **event Game start** deze sound.



Bleeps

Een spel zonder geluid is minder leuk dan met geluid. In de map GamemakerBestanden staan in de map Sounds allerlei geluidjes. In Game Maker boek deel 1 heb je geleerd hoe je geluidjes kunt toevoegen aan een spel.

17. Maak met de knop **Create a sound** geluidjes.

    Drie geluiden is een goed begin. Kies een geschikt geluid voor als je:

    a. punten verdient (**snd_punten**)
    b. een leven verliest (**snd_leven**)
    c. naar een volgend level gaat (**snd_gehaald**)

18. Voeg de geluiden toe als:
    a. actie bij **events botsingen** waar je **punten** mee **verdient**
    b. voor **botsingen** waar je **leven verliest**.
    c. voor **botsing** met **obj_doel**, waardoor je naar een volgend level gaat.

Help

Je spel is nog niet af. Je hebt nog geen helptekst waarin wordt uitgelegd hoe je spel werkt.

19. Maak een helptekst. Dubbelklik op **Game information** in de Resource explorer (=*bronnenverkenner*), waar alle onderdelen van je spel staan. Typ hier helptekst.
20. Klik op **Opslaan** .

Startscherm, tussenschermen, eindscherm

De meeste spellen hebben ook een startscherm, waarin de belangrijkste zaken staan. Vaak wordt daarvoor als achtergrond een schermafdruk van een van de levels gebruikt.

21. Ga naar de map MijnGames en **open** Doolhof-Naam-7. Klik op **File** *(= Bestand)* en daarna op **Save As...** (=*Sla op als...)*. Sla het bestand op in de map **MijnGames** met de bestandsnaam **Doolhof-Naam-8.**
22. Klik op **Run the game** .
23. Klik op de toets <F9> en klik op <ESC> om terug te keren in Game Maker.
24. Start het programma **Paint** op (via de Windows-knop **Start**, dan **Alle programma's**, dan **Bureau-accessoires** en klik dan op **Paint**.)

Het scherm van het tekenprogramma **Paint** wordt nu geopend.

25. Klik op **Bestand**, en dan op **Openen.**
26. Ga naar de map **MijnGames** en kies bijvoorbeeld **screenshot100.** Klik op **Openen.**
27. Je ziet nu een afbeelding van je programma. Hierin kun je teksten typen.

28. Klik op menu **Bestand** en dan op **Opslaan als** ... .
29. Kies als bestandsnaam **Achtergrond** en klik op Opslaan.
30. Sluit Paint met het kruisje en ga weer terug naar **Game Maker**. Waarschijnlijk zie onderin in je **taakbalk** wel een **knop** met het Game Maker icoon. Klik daarop. Is dat niet het geval, start dan Game Maker en open **Doolhof-Naam-8**.
31. Maak nu een nieuwe achtergrond met **Create background**
32. Geef de achtergrond de **naam bg_startscherm**.
33. Klik op **Load background**.
34. Ga naar de map **MijnGames** en klik op **Achtergrond**.
35. Klik op **Openen** en klik daarna op **OK** om de achtergrond op te slaan.
36. Maak nu een nieuwe room met **Create a room** .
37. Ga naar **tabblad Settings.**
38. Vul bij **Name Startscherm**, bij **Width 800** en bij **Height 576** in.

39. Ga naar **tabblad Backgrounds**. Kies als achtergrond **bg_startscherm**.
40. Sleep deze **room Startscherm** tot **boven aan de lijst**.

Wat nu nog nodig is, is een object met als enig event, dat als je op de entertoets klikt, je naar het volgende level gaat. We geven dit object de naam obj_controllerStart. De toetsen <ESC> en F1 werken sowieso al, dat zijn in Game Maker standaardknoppen en daar hoef je dus niks voor te programmeren.

41. Klik op **Create an object**. Geef het object de naam **obj_controllerStart**. Het object hoeft geen sprite.
42. Klik op **Add event**, klik dan op **Keyboard** en kies dan **<Enter>.**
43. Voeg bij de event <Enter> voeg je de **actie Next room** toe . Deze staat op **tab main1**.
44. Dubbelklik op room Startscherm en **plaats obj_controllerStart** in deze **room**.
45. Klik op **Opslaan** .
46. Klik op Run om te **testen** of het startscherm goed werkt. Voeg de testregels toe aan je **testschema**.

Je kunt tussen twee levels ook tussenschermen maken, met korte uitleg van het volgende level. Dit gaat op dezelfde manier als het maken van het startscherm. Datzelfde geldt voor het eindscherm. Plaats die als room onderaan. Haal dan het berichtje bij obj_doel wel weg, want die is dan overbodig.

Achtergronden kun je ook downloaden van internet. Kijk op www.gamemakerkids.nl voor geschikte sites.
Of je kunt een eigen foto gebruiken als achtergrond.

Een uitvoerbaar bestand maken

In principe heb je nu een compleet spel. Je kunt het spel *nog* mooier en leuker maken. Bijvoorbeeld nog meer levels, of sprites die naar links of naar rechts kijken als je naar links of naar rechts loopt. Dit kun je allemaal leren door het hoofdstuk Trucs door te nemen.

47. Ben je tevreden, dan kun je een uitvoerbaar bestand maken door te klikken op de knop **Create a stand-alone executable for your game** .
48. **Sla** het bestand **op** in de map **MijnGames**. Geef het bestand een **naam zonder spaties en rare tekens**.
49. **Test** nu aan de hand van je **totale testschema** het hele spel door.
50. Werkt het volgens jou goed, laat dan een **vriendje** of **vriendinnetje testen**. Vind hij of zij het een leuk spel, spannend genoeg, niet te moeilijk?
51. Pas zonodig met de opmerkingen je spel nog aan.
    Is het goed? Dan kun je het spel bijvoorbeeld op internet zetten. Misschien mag het wel op de website van je school of van je club.

Hoofdstuk 10: Extra trucs

Belangrijk

Vergeet niet steeds bij elke verandering het volgende stappenplan:

1. **Ontwerp** maken
2. Spel **onder een nieuwe naam** opslaan
3. Je ontwerp **programmeren**
4. Je bestand **opslaan**
5. Je **testschema** bijwerken met nieuwe testregels
6. Aan de had van je testschema **testen** of je aanpassing goed werkt
7. Pas als je tevreden bent over je aanpassing, dit lijstje weer voor de volgende aanpassing doorlopen.

Een animatie maken

Een animatie, een soort filmpje, maak je door 2 of meer plaatjes snel af te wisselen. Het makkelijkst is om het plaatje te kopiëren en dan het gekopieerde plaatje aan te passen.

Voorbeeld bekijken

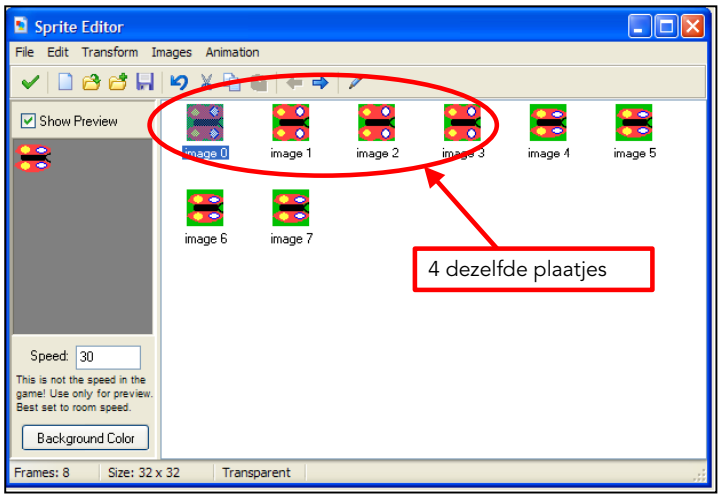
Om een voorbeeld van een animatie te bekijken, open je uit de map Voorbeeldgames het bestand voorbeeld-animatie.

1. **Start** Game maker of had je nog je spel geopend, klik dan op **Opslaan** ..
2. Open uit de map Voorbeeldgames **Voorbeeld-animatie**.
3. Klik op **Run the game** .

Je ziet een fladderende vlinder. Het maken van zo'n vlinder is niet heel ingewikkeld.

4. Klik op **<ESC>** om het spel af te sluiten en terug te keren naar **Game Maker**.
5. Dubbelklik op **spr_vlinderRechts** om de sprite te bekijken.
6. Klik op **Edit sprite.** Zet een **vinkje** bij **Show preview.**

Je ziet de animatie. Een animatie is niets anders dan een aantal plaatjes wat elkaar afwisselt. In dit geval zijn de eerste 4 plaatjes hetzelfde en de andere 4 plaatjes zijn ook hetzelfde maar net anders. Daardoor lijkt het dat de vlinder fladdert. In Game maker is de standaardinstelling dat 30 plaatjes één seconde duurt.



Zelf animatie maken.

7. **Open** de **laatste versie** van je eigen spel. Sla je spel op met een naam waarbij je het versienummer 1 hoger maakt.
8. **Dubbelklik** op een **sprite** waar je een animatie van wil maken.
9. Klik op **Edit sprite**.
10. Klik op het **plaatje** en dan op **Copy to the clipboard** (=Kopieer naar het klembord) en klik dan op **Paste** (= Plakken) .

Je hebt nu twee precies dezelfde plaatjes

11. Dubbelklik nu op een van beide plaatjes.
12. Verander iets aan het plaatje.

Je hoeft aan een plaatje maar weinig te veranderen of je hebt al een leuke animatie. In het voorbeeld hierboven hoefde je alleen maar de vleugels wat smaller te maken en het lijkt al of de vlinder fladdert.

13. Klik op het **groene vinkje** als het plaatje af is, en zet een vinkje bij **Show preview**. Je kunt dan zien of de animatie is gelukt.

Gaat de **animatie te snel**, dan heeft het geen zin om speed aan te passen. Want als je de animatie in de room zet, heeft de animatie standaard de snelheid 30. Wat je wel kunt doen is bijvoorbeeld het ene **plaatje te kopiëren** en dan 10 keer te plakken en het andere plaatje ook.
Je hebt dan eerst 10 keer plaatje 1 en dan 10 keer plaatje 2. De animatie gaat dan 10 keer langzamer.

14. Als je tevreden bent over de animatie klik dan op het **groene vinkje** om het plaatje op te slaan en daarna op **OK** om de sprite op te slaan.

Een sprite maken die andere kant op kijkt

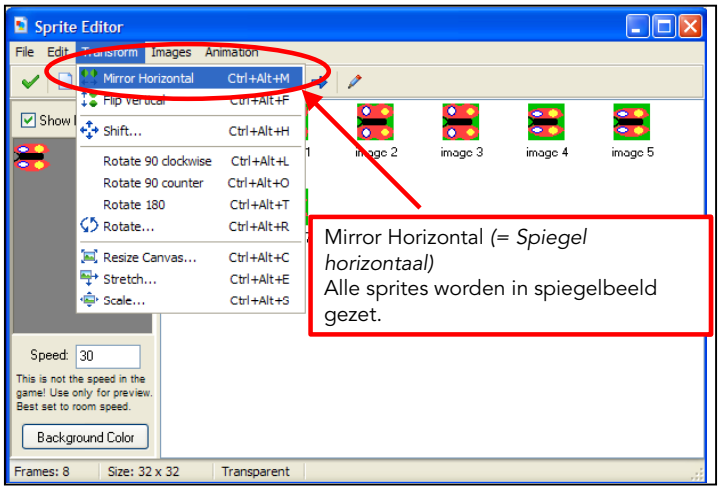
In het voorbeeld voorbeeld-animatie zag je ook een sprite spr_vlinderLinks. Deze sprite kijkt naar links.

Heb je al een sprite gemaakt die naar rechts kijkt en heb je ook een sprite nodig die naar links kijkt? Dat kan het snelst als volgt.

1. Klik met je **rechtermuisknop** op de **sprite**.
2. Klik op **Duplicate** (= verdubbel)

Je hebt nu een tweede sprite die er hetzelfde uitziet als de eerste.

3. Verander de **naam** van deze sprite (in het voorbeeld hierboven dus spr_vlinderLinks)
4. Klik op **Edit sprite**
5. Klik op **Transform** (= *transformeren ofwel omvormen*) en dan op **Mirror Horizontal** (= *spiegel horizontaal*). Het plaatje 'kijkt' nu de andere kant op.
6. Klik op het **groene vinkje** om het plaatje op te slaan vinkje en daarna op **OK** om de sprite op te slaan.



Sprite wisselen wanneer de ik-figuur van richting verandert

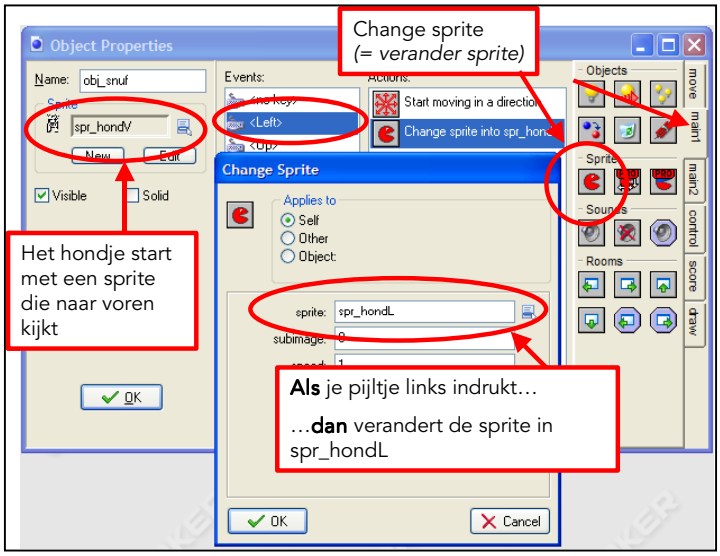
In veel spellen kijkt de ik-figuur naar links, als je naar links loopt, en naar rechts, als je naar rechts loopt. In deze paragraaf leer je hoe je dat programmeert.

Voorbeeld bekijken

1. **Start** Game maker of had je nog je spel geopend, klik dan op **Opslaan** ..
2. Open uit de map Voorbeeldgames **Voorbeeld-spriteWisselen1**.
3. Klik op **Run the game** . Bestuur de ik-figuur met de **pijltjes**

Je ziet dat als je pijltje links indrukt, de hond naar links kijkt. En als je pijltje rechts indrukt kijkt de hond naar rechts.

4. Klik op **<ESC>** om het spel af te sluiten en terug te keren naar **Game Maker**.
5. Dubbelklik op **obj_snuf** en klik op **<left>**.



.

Zelf programmeren

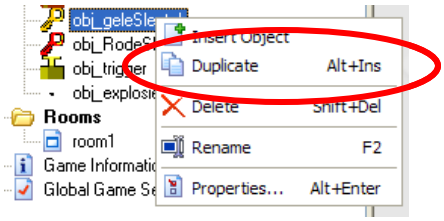
6. **Open** de **laatste versie** van je eigen spel. Sla je spel op met een naam waarbij je het versienummer 1 hoger maakt.
7. Dubbelklik op obj_IK.
8. Klik op event **'Keyboard event <left>**
9. Sleep de action 'Change the sprite' naar het vak Actions.
10. Kies een sprite die naar links kijkt.

11. Doe ditzelfde voor de andere Events.

Kopiëren van objecten

Voor een nieuw level kun je de objecten gebruiken die je al gemaakt hebt. Maar je kunt ook nieuwe objecten maken. Als je voor de nieuwe levels objecten wilt, die net een beetje anders werken, kun je objecten het best kopiëren.

1. Klik met je rechtermuisknop op een object en klik dan op duplicate.
   Na het kopiëren geef je het object een naam en past events of acties aan



Sprite of object wisselen wanneer ander object van richting verandert

Een ik-figuur die naar links kijkt als je naar links loopt en naar rechts kijkt als je naar rechts loopt is niet heel moeilijk. De actie sprite-wisselen reageert op event pijltjestoets indrukken. Bij een object dat tussen twee muurtjes heen en weer loopt is dat wat lastiger. Want je hebt alleen maar één event: botsing met muurtje. Je weet dan niet of je de sprite moet veranderen in één die naar links kijkt of één die naar rechts kijkt. Dat hangt namelijk af van welke richting het object gaat ná de botsing, naar links of naar rechts.

Voorbeeld bekijken

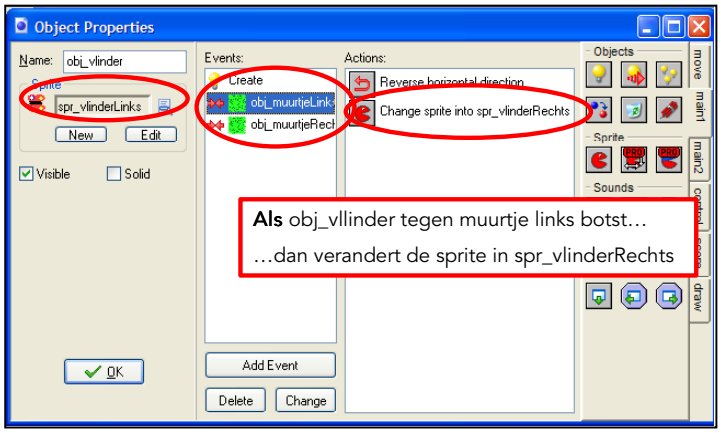
Een truc is dat je twee verschillende muurtjes maakt, obj_muurtjeLinks en obj_muurtjeRechts. Als het object tegen obj_muurtjeLinks botst verandert de sprite in een sprite die naar rechts kijkt. En als het object tegen obj_muurtjerechts botst verandert de sprite in een sprite die naar links kijkt.

1. **Start** Game maker of had je nog je spel geopend, klik dan op **Opslaan** ..
2. Open uit de map Voorbeeldgames **Voorbeeld-spriteWisselen2**.
3. Klik op **Run the game** .

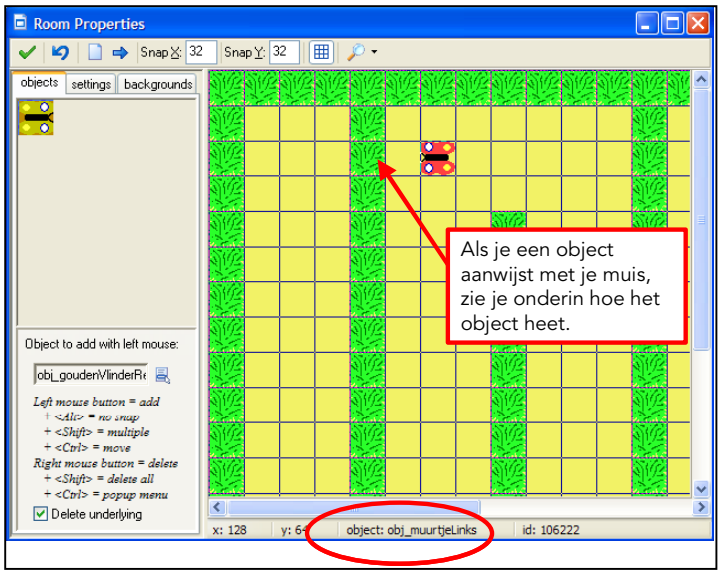
Je ziet dat beide vlinders tussen de muurtjes heen en weer gaan. Als ze naar links fladderen kijken ze naar links, en als ze naar rechts fladderen kijken ze naar rechts.

4. Klik op **<ESC>** om het spel af te sluiten en terug te keren naar **Game Maker**.
5. Dubbelklik op **obj_vlinder**.

   Je ziet nu twee events botsing, namelijk botsing met obj_muurtjeLinks en botsing met obj_muurtjeRechts.
6. Klik op **botsing met obj_muurtjeLinks**. Je ziet dat er een actie is toegevoegd, namelijk de actie **Change sprite into spr_vlinderRechts**. (*=verander sprite in spr_vlinderRechts*)



7. Dubbelklik op **room1**.



8. Wijs met je **muis** het **muurtje links** van de rode vlinder aan.

Onder in de grijze balk zie je dat dit muurtje obj_muurtjeLinks is.

Let wel op dat je dan in het speelveld alle muurtjes waar dat nodig is vervangt door obj_muurtjeLinks of obj_muurtjeRechts.

Zelf programmeren

Heb jij objecten die op en neer gaan in plaats van heen en weer? Dan heb je sprites nodig die naar boven en een die naar beneden kijkt. En muurtjes obj_muurBoven en obj_muurOnder. Verder gaat het hetzelfde als hieronder.

9. **Open** de **laatste versie** van je eigen spel. Sla je spel op met een naam waarbij je het versienummer 1 hoger maakt.
10. Zorg ervoor dat je een **sprite** hebt die naar links **kijkt** en een **sprite** die naar **rechts** kijkt. Hoe je dat handig doet? **Zie hoofdstuk 'Een sprite maken die de andere kant opkijkt'**.
11. **Maak** 2 nieuwe objecten, **obj_muurLinks** en **obj_muurRechts**. Dit kun je het makkelijkst doen door obj_muur te kopiëren. **Zie hoofdstuk: 'Kopiëren van objecten'** .

12. **Vervang** in je **levels** bij het linker muurtje waar het object tegen gaat botsen in **obj_muurLinks** en het rechter muurtje in **obj_muurRechts**.
13. **Vervang** bij het object de event botsing met obj_muur in event **botsing met obj_muurLinks**. (Dit kan het makkelijkst door met je rechtermuisknop op de event te klikken en dan te kiezen voor Change event (=*verander event*))
14. **Voeg** bij event botsing met obj_muurLinks de **actie Change sprite toe**. Kies als sprite de **sprite die naar rechts kijkt**.
15. **Kopieer (Duplicate)** event botsing met **obj_muurLinks** en kies botsing met **obj_muurRechts**.
16. Dubbelklik op actie **Change sprite** en verander de sprite in een **sprite die naar links kijkt**.

Een andere manier om dit op te lossen is, door niet twee verschillende obj_muur te maken, maar twee verschillende objecten, die heen en weer gaan, te maken. Dit is bij de gouden vlinder gedaan.

Als het ene object tegen een muur botst verandert deze in de ander met actie **Change an instance** .
Zorg ervoor dat **perform events** op **yes** staat.
En zorg dan er voor dat bij **event Create** bij actie **Move in a direction** er bij Directions één pijl is aangeklikt en niet twee.

Kijk verder bij **obj_goudenVlinderLinks** hoe een en ander is geprogrammeerd.

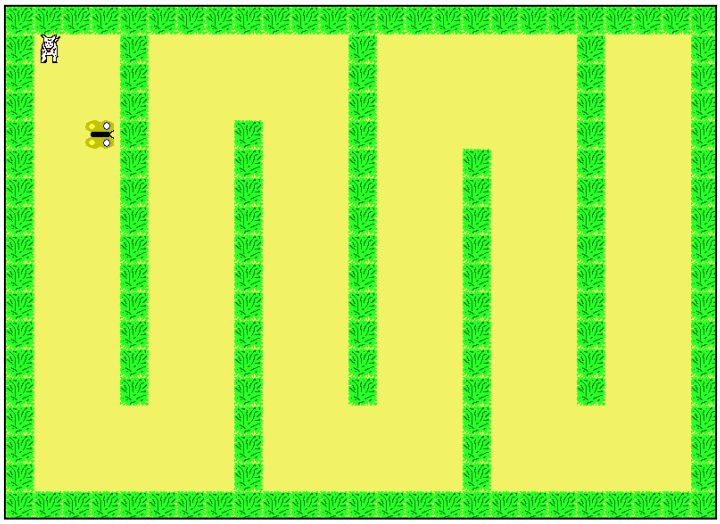
Soms wel, soms niet: gebruik van dobbelsteen

Als je wat verrassing in je game wil inbouwen, dat soms iets wel gebeurt en soms niet, dan kun je gebruik maken van de actie Test change, dobbelsteen.

Voorbeeld bekijken

1. **Start Game maker of** had je nog je spel geopend, klik dan op **Opslaan** ..
2. Open uit de map Voorbeeldgames **Voorbeeld-dobbelsteen**.
3. Klik op **Run the game** .

Het spel heeft nu maar één level, waarin 1 gouden vlinder. Als je de vlinder pakt springt de vlinder naar een willekeurige andere plek in het doolhof. Zo hou je altijd één vlinder. Maar deze keer is er iets bijzonders.



4. Pak de **gouden vlinder**.
5. Wat gebeurt er met een gouden vlinder als je die hebt gepakt?

   .....
6. Pak nog een paar keer de vlinder.
   a. Schrijf telkens op wat er gebeurt met de kleur
      i. je pakt een .....vlinder, dat wordt een .....
      ii. je pakt een .....vlinder, dat wordt een .....
      iii. je pakt een .....vlinder, dat wordt een .....
      iv. je pakt een .....vlinder, dat wordt een .....
      v. je pakt een .....vlinder, dat wordt een .....
      vi. je pakt een .....vlinder, dat wordt een .....

vii. je pakt een .....vlinder, dat wordt een .....
viii. je pakt een .....vlinder, dat wordt een .....
ix. je pakt een .....vlinder, dat wordt een .....
x. je pakt een .....vlinder, dat wordt een .....

b. Wat gebeurt er als je een gouden vlinder pakt?.....
c. Is dat altijd zo?.....
d. Wat gebeurt er als je een rode vlinder pakt?.....
e. Is dat altijd zo?.....

7. Klik op **<ESC>** om het spel af te sluiten en **terug** te keren naar **Game Maker**
8. Dubbelklik op **obj_goudenVlinder**. Welke acties zijn geprogrammeerd bij botsing met obj_IK? Vul in:

   Als obj_goudenSleutel en obj_IK botsten,

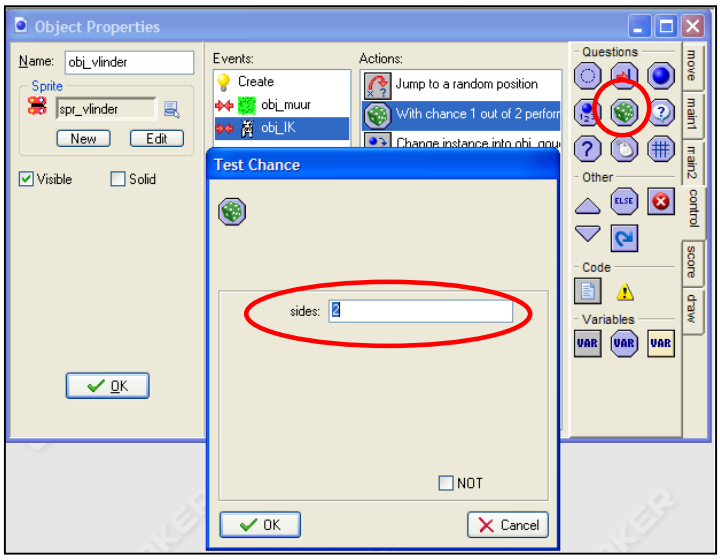
   dan .....,

   en dan .....
9. Dubbelklik op **obj_vlinder** en klik op de event **Botsing** met **obj_IK**.

Object obj_vlinder lijkt wel op object obj_goudenVlinder, alleen zij verandert niet altijd maar soms. Dat kun je programmeren met een dobbelsteen.

Je ziet dat bij botsing met obj_IK, obj_vlinder naar een andere plek springt. Dan zie je de actie **Test chance**, een dobbelsteen.

10. **Dubbelklik** op de actie Test chance.



Bij sides (=*zijdes*) staat 2. Dat wil zeggen dat je gooit met een dobbelsteen met 2 kanten. Dat kan in het echt natuurlijk niet, maar op de computer kan alles. Je kunt het nu vergelijken met het gooien van een munt, die heeft wel twee kanten. Bij kop zal de volgende actie worden uitgevoerd, namelijk veranderen in een gouden vlinder. Bij munt blijft de rode vlinder een rode vlinder. Het is dus elke keer een verassing of er een rode of een gouden vlinder komt.

Zelf programmeren

11. **Open** de **laatste versie** van je eigen spel. Sla je spel op met een naam waarbij je het versienummer 1 hoger maakt.
12. Denk goed na waar je toeval in wil bouwen. Sleep bij die event de actie **Test change** naar het vak actions. **Test** of het goed werkt.

Let op: als er na toeval meerdere acties moeten worden gedaan, bijvoorbeeld veranderen in iets anders en een geluidje, zet de actie dan in een blok. (**Start block** en **End block** staan op tabblad Controls)

Objecten willekeurig rond laten lopen

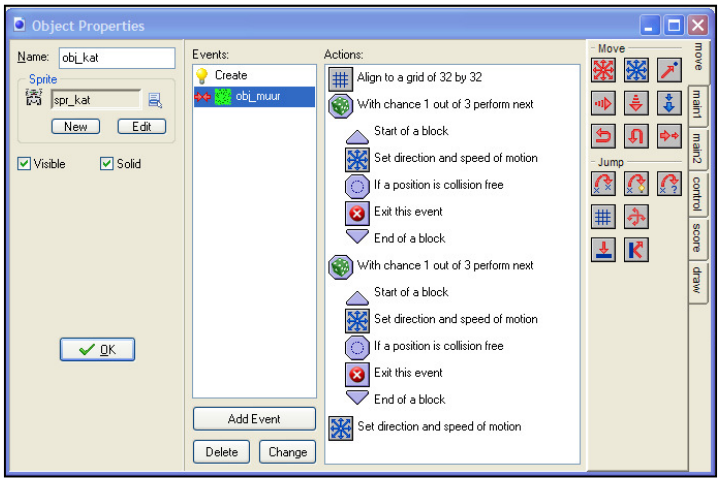
Je kunt in een doolhof monsters heen en weer laten lopen, maar het is ook leuk wanneer ze kris-kras door het doolhof lopen.

Voorbeeld bekijken

1. **Start Game maker of** had je nog je spel geopend, klik dan op **Opslaan** ..
2. Open uit de map Voorbeeldgames **Voorbeeld-willekeurigLopen**.
3. Klik op **Run the game** .

Je ziet in dit voorbeeld dat katten willekeurig rondlopen in het doolhof.

4. Klik op **<ESC>** om het spel af te sluiten en **terug** te keren naar **Game Maker.**
5. Dubbelklik op **obj_kat**. Klik dan op **Collision with obj_muur** (=*botsing met obj_muur*). Je ziet daar heel veel acties.



Het principe is als volgt.
Als obj_kat tegen obj_muur botst, dan springt de kat eerst terug naar een hokje.
Dan draait obj_kat telkens een kwartslag.
Dan kijkt hij of hij kan lopen.
Kan hij niet lopen, omdat er iets staat, dan draait hij nog een kwartslag.
Net zo lang tot hij wel kan lopen.
Zodra de kat kan lopen, wordt de event gestopt en loopt de kat in die richting verder.
Het gedrag van de kat zou zo wel voorspelbaar zijn, terwijl de bewegingen van de kat in level 2 dat toch niet zijn. Dat komt omdat je een blok acties niet altijd laat uitvoeren, maar af en toe. Met de actie Test chance .

Zelf programmeren

6. **Open** de laatste versie van je eigen spel. Sla je spel op met een naam waarbij je het **versienummer** 1 hoger maakt.
7. Omdat het programmeren best lastig is zie je op de **volgende pagina** van alle acties **schermafdrukken**.

Vul de velden precies zo als in de schermafdrukken. Maak geen typfouten.

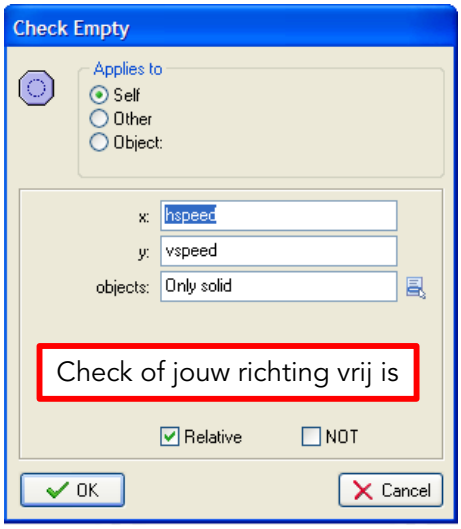
1.

2.

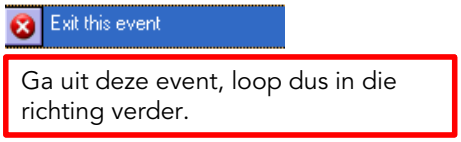
3.

4.

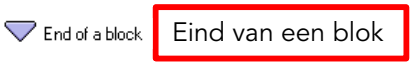
5.



6.



7.



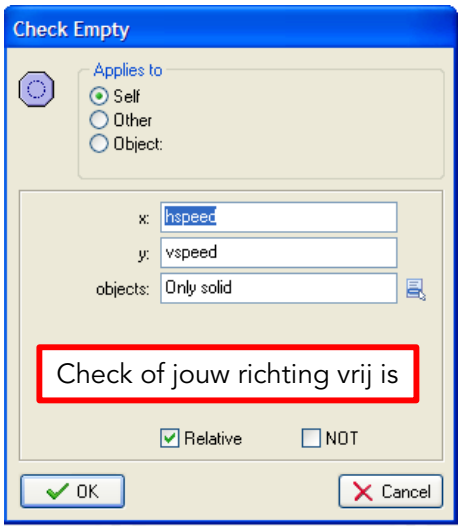
8.

9.

10.

Draai een halve slag en ga verder met snelheid 5

11.

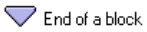


Check of jouw richting vrij is

12.

Ga uit deze event, loop dus in die richting verder.

13.



Eind van een blok

14.

Draai een driekwart om en ga verder met snelheid 5

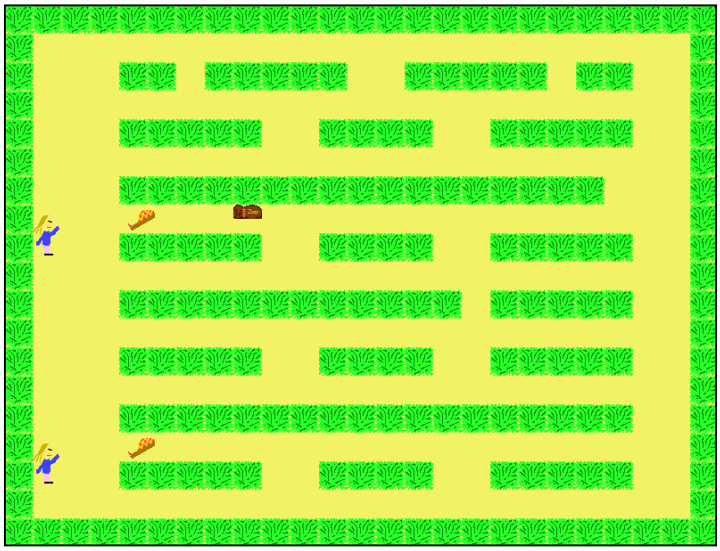
Heb je het niet echt begrepen? Niet erg, als je het maar heel precies overneemt, in de goeie volgorde.

De ik-figuur laten gooien of schieten

In een doolhof kun je de ik-figuur iets laten gooien of schieten. Maar hoe kun je dat programmeren?
In onderstaand voorbeeld gaan we er van uit dat gooien gebeurt door de spatiebalk in te drukken.

Voorbeeld bekijken

1. **Start Game maker of** had je nog je spel geopend, klik dan op **Opslaan** ..
2. Open uit de map Voorbeeldgames **Voorbeeld-gooien1**.
3. Klik op **Run the game** .
4. Druk de **spatiebalk** in.



Je ziet dat de ik-figuur een pantoffel gooit naar de slapende hond. Als de hond wordt geraakt wordt-ie wakker. Bij de onderste ik-figuur is er geen doel om te raken, maar verdwijnt de pantoffel als deze in de struikjes terecht komt.
Wat moet er allemaal gebeuren om de ik-figuur te laten gooien?

- Obj_IK moet gooien wanneer er op de spatiebalk wordt gedrukt.
- Wanneer het doel wordt geraakt, moet het doel veranderen in een geraakt doel

- Wanneer het doel wordt geraakt, blijft het gegooide niet doorvliegen, maar verdwijnt.
- Wanneer een doel niet wordt geraakt, moet het gegooide op een gegeven moment wel verdwijnen.

5. Klik op **<ESC>** om het spel af te sluiten en **terug** te keren naar **Game Maker.**

   Bij het gooien zijn 4 objecten betrokken:

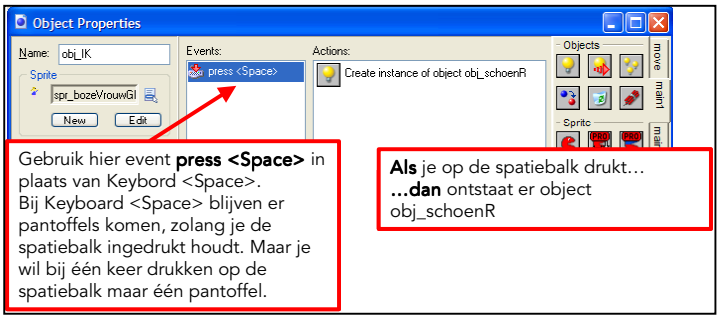
   - de ik-figuur → obj_IK
   - het gegooide → obj_schoen
   - het doel → obj_buldogSlaapt
   - het geraakte doel → obj_buldog

6. Dubbelklik op **obj_IK**.

   Er is hier één event geprogrammeerd:

   a. Event **press<Space>**.

   Als je op de **spatiebalk** drukt dan **ontstaat object obj_schoenR**.



7. Dubbelklik op **obj_schoenR**.

   Er zijn 3 events geprogrammeerd:

   a. event **Create:**

   als obj_schoenR ontstaat **dan** begint obj_schoenR te bewegen naar rechts.

   b. event **Collision with obj_muur**:

   als obj_schoenR botst met obj_muur **dan** verdwijnt obj_schoenR

   c. event **Collision with obj_buldogSlaapt**:

   als obj_schoenR botst met obj_buldogSlaapt,

dan verander obj_buldogSlaapt in obj_buldog

en dan verdwijnt obj_schoenR.

We hadden bij botsing van obj_schoenR met obj_buldogSlaapt ook alleen de sprite kunnen veranderen, maar dan houdt het object dezelfde eigenschappen. In dit geval is het beter om het object te veranderen in een ander object omdat we het geraakte doel obj_buldog misschien andere eigenschappen wilt geven dan het doel. Bijvoorbeeld dat obj_buldog heen en weer gaat rennen.

Zelf programmeren

8. **Open** de laatste versie van je eigen spel. Sla je spel op met een naam waarbij je het **versienummer** 1 hoger maakt.
9. Bedenk welke objecten bij jouw spel betrokken zijn bij het gooien:
    a. obj_IK is in jouw spel obj_IK
    b. obj_Doel is in jouw spel: .....
    c. obj_Gegooide is in jouw spel: .....
    d. obj_Raak is in jouw spel: .....
10. Het besturingsobject obj_IK moet gooien wanneer er op de spatiebalk wordt gedrukt.
    a. Voeg bij **obj_IK** de event **Key press** <**Space**> (=spatiebalk) toe.
    b. Sleep van het tabblad main1 de actie **Create Instance**
    c. Kies bij object **obj_Gegooide**
    d. Vink **relative** aan.
    e. Klik op **OK**
11. Wanneer het doel obj_Doel wordt geraakt, moet het veranderen in een geraakt doel: obj_Raak.
    a. Voeg bij het object **obj_Doel Collision** event met **obj_Gegooide** toe.
    b. Sleep vanaf het tabblad main1 de actie **Change instance** naar het vak Actions.
    c. Vul bij object **obj_Raak** in.
    d. Kies bij **perform events** (=voer de events uit) **yes**.

e. Klik op **OK**

Wanneer obj_Doel wordt geraakt, blijft het gegooide niet doorvliegen, maar verdwijnt.

f. Sleep vanaf tabblad main1 de actie **Destroy the instance** naar het vak Actions.

g. Klik op **OK**

12. Gegooide dingen, die geen doel raken moeten uiteindelijk verdwijnen.

a. Voeg bij **obj_Gegooide** event **Collision** with **obj_muur** toe

b. Sleep vanaf tabblad main1 de actie **Destroy instance** naar het vak Actions.

c. Klik op **OK.**

Schieten in plaats van gooien

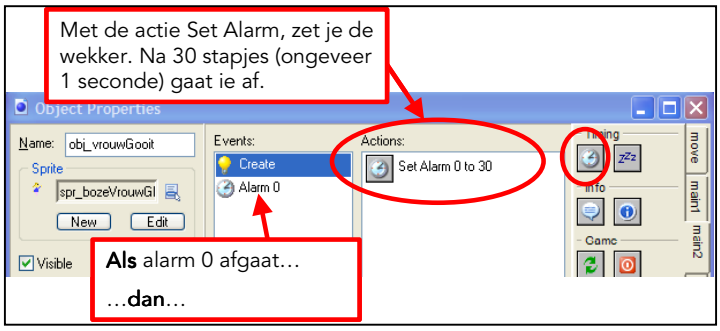
Als je de ik-figuur niet wilt laten gooien maar schieten, dan is het principe hetzelfde.

- Je hebt een sprite van de **ik-figuur** met een pistool voor de ik-figuur nodig.
- Het gegooide is een **kogel**. Je kunt een sprite als volgt laden.(Load sprite) Ga naar de map GamemakerBestanden, dan naar de map Sprites en dan naar de map Various. Kies de sprite **'dot'** en klik op openen.
- Je hebt een **doel** nodig, dat kan ieder object zijn.
- En je hebt een **geraakt doel** nodig. Daarvoor kun je als sprite de sprite **Explode**, de rookwolk, uit de map various kiezen.

Een wekker gebruiken

Soms wil je niet dat een actie meteen wordt uitgevoerd, maar dat er even tijd tussen zit. Daarvoor moet je 2 zaken programmeren.

- Door een bepaalde event wordt een wekker, of beter een kookwekker, gezet. Met de **actie Set alarm** (=*Zet een wekker*) (tabblad main2) kun je de **wekker zetten**. Als je bij **number of steps** (=*aantal stappen*) **30** invult, duurt het ongeveer **1 seconde** voordat de wekker afgaat.
  Bij elk object kun je 11 wekkers zetten: alarm0, alarm1, alarm2 enzovoorts.
- Als event **alarm** - dus **als** de wekker afgaat - **dan** moet er iets gebeuren.



In de volgende paragrafen vind je een aantal voorbeelden waarbij een wekker is gebruikt.

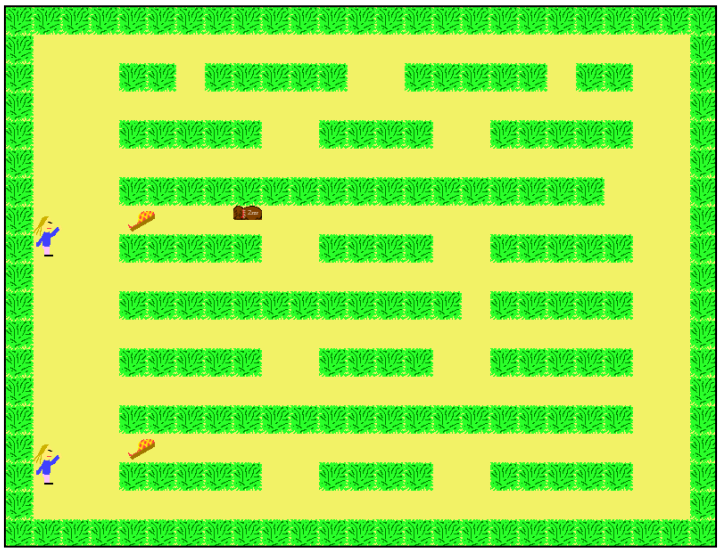
Een ander object laten gooien of schieten

In een doolhof kun je de ik-figuur iets laten gooien of schieten. Maar je kunt ook een ander object automatisch laten gooien of schieten.

Voorbeeld bekijken

1. **Start Game maker of** had je nog je spel geopend, klik dan op **Opslaan** ..

2. Open uit de map Voorbeeldgames **Voorbeeld-gooien2**.
3. Klik op **Run the game** .



Je ziet dat de vrouw nu automatisch een pantoffel gooit naar de slapende hond, telkens weer opnieuw.

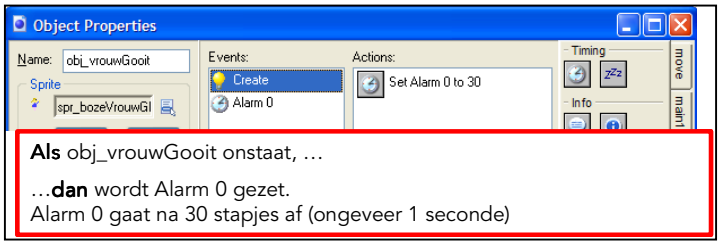
4. Klik op <ESC> om het spel af te sluiten en **terug** te keren naar **Game Maker.**

In de paragraaf 'De ik-figuur laten gooien of schieten' heb je geleerd wat je moet programmeren om te gooien of schieten. Maar nu is niet de event Press <space>, het indrukken van de spatiebalk, de event dat er moet worden gegooid, maar het feit dat de wekker afgaat.

.

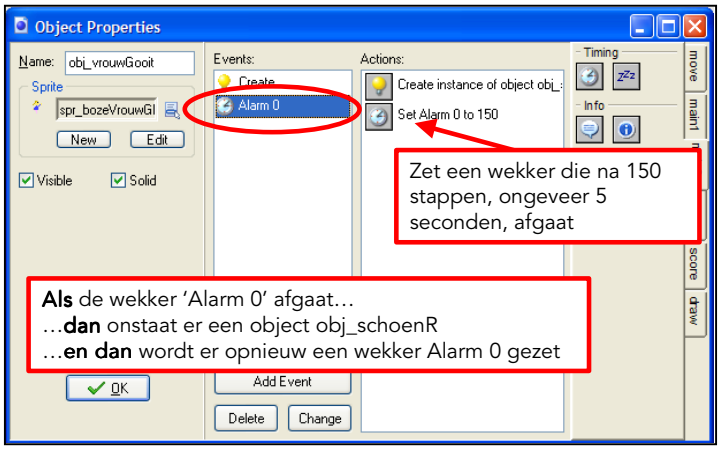
5. Dubbelklik op obj_vrouwGooit

De eerste wekker Alarm 0 wordt in dit voorbeeld gezet door de event Create



6. Klik op event Alarm0

Telkens **als** Alarm 0 afgaat, **dan** wordt een nieuw Alarm 0 gezet. Dus blijft obj_vrouw schoenen gooien.



Zelf programmeren

Alles is bijna hetzelfde als in het hoofdstuk 'De ik-figuur laten gooien of schieten'. Het is nu niet de ik-figuur die gooit maar een ander object. We noemen die voor het gemak obj_gooier

7. **Open** de laatste versie van je eigen spel. Sla je spel op met een naam waarbij je het **versienummer** 1 hoger maakt.
8. Bedenk welke objecten bij jouw spel betrokken zijn bij het gooien:
    a. obj_gooier is in jouw spel: .....
    b. obj_Doel is in jouw spel: .....
    c. obj_Gegooide is in jouw spel: .....
    d. obj_Raak is in jouw spel: .....
9. Het besturingsobject obj_gooier moet een alarm zetten wanneer het object ontstaat.
    a. Voeg bij **obj_gooier** de event **Create** toe.
    b. Sleep van het tabblad **main2** de actie **'Set Alarm'** naar het vak Actions
    c. Kies bij **number of steps** bijvoorbeeld **30**. (dus na 1 seconde)
    d. Kies bij **'in alarm no' Alarm 0.**
    e. Klik op **OK.**

    Het object obj_gooier moet gooien wanneer alarm 0 afgaat:

e. Voeg bij obj_gooier de event **Alarm 0** (=*wekker 0 gaat af*) toe.
f. Sleep van het tabblad **main1** de actie **Create Instance**
g. Kies bij object **obj_Gegooide**
h. Vink **relative** aan.
i. Klik op **OK.**

en dan moet er meteen ook een nieuwe wekker worden gezet:

j. Sleep van het tabblad **main2** de actie **Set Alarm** naar het vak Actions
k. Kies bij **number of steps** bijvoorbeeld **30**. (dus na 1 seconde)
l. Kies bij **'in alarm no' Alarm 0.**
m. Klik op **OK.**

13. Wanneer het doel obj_Doel wordt geraakt, moet het veranderen in een geraakt doel: obj_Raak.
    a. Voeg bij het object **obj_Doel Collision** event met **obj_Gegooide** toe.
    b. Sleep vanaf het tabblad main1 de actie **Change instance** naar het vak Actions.
    c. Vul bij object **obj_Raak** in.
    n. Kies bij **perform events** (=voer de events uit) **yes**.
    o. Klik op **OK.**

    Wanneer obj_Doel wordt geraakt, blijft het gegooide niet doorvliegen, maar verdwijnt.

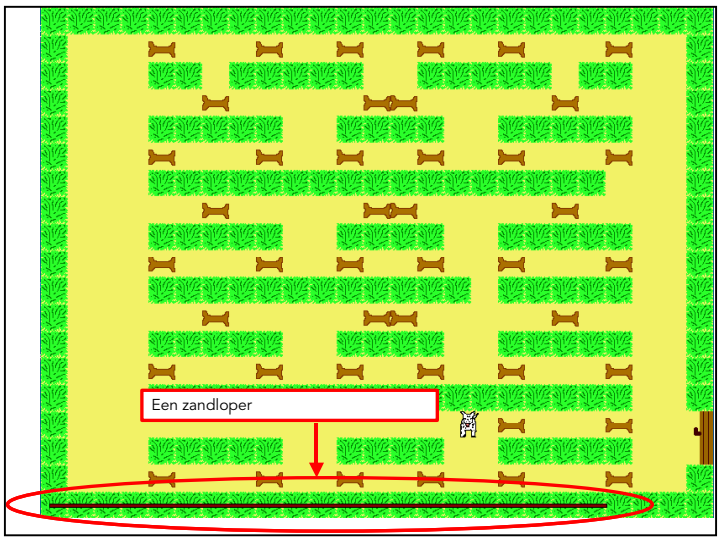
    p. Sleep vanaf tabblad main1 de actie **Destroy the instance** naar het vak Actions.
    q. Klik op **OK.**
14. Gegooide dingen, die geen doel raken moeten uiteindelijk verdwijnen.
    a. Voeg bij **obj_Gegooide** event **Collision** with **obj_muur** toe
    f. Sleep vanaf tabblad main1 de actie **Destroy instance** naar het vak Actions.
    g. Klik op **OK.**

De zandloper

Soms wil je dat een speler een level binnen een bepaalde tijd haalt. Daarvoor kun je een zandloper gebruiken.

Voorbeeld bekijken

1. **Start Game maker of** had je nog je spel geopend, klik dan op **Opslaan** ..
2. Open uit de map Voorbeeldgames **Voorbeeld-zandloper**.
3. Klik op **Run the game** .



Je ziet onder in het scherm een tijdslijn die steeds korter wordt. We noemen dit een zanloper.

4. Probeer binnen de tijd alle botten te verzamelen en ga dan naar het volgende level.
5. Wat gebeurt er als je het niet binnen de tijd haalt? .....
6. Klik op <ESC> om het spel af te sluiten en **terug** te keren naar **Game Maker.**

Voor een zandloper zijn 4 events bij **obj_controller** belangrijk.

- **Als** het level **onstaat**,

  dan begin je met **health** (=*gezondheid*) **100**.

  en dan wordt er een wekker **Alarm 0** gezet die na bijvoorbeeld **30 stappen**, ongeveer 1 seconde afgaat.

- **Als** de wekker **Alarm 0** afgaat,

  dan gaat er van **health 1 af**.

  en dan wordt opnieuw wekker **Alarm 0** gezet die na **30 stappen** af gaat.

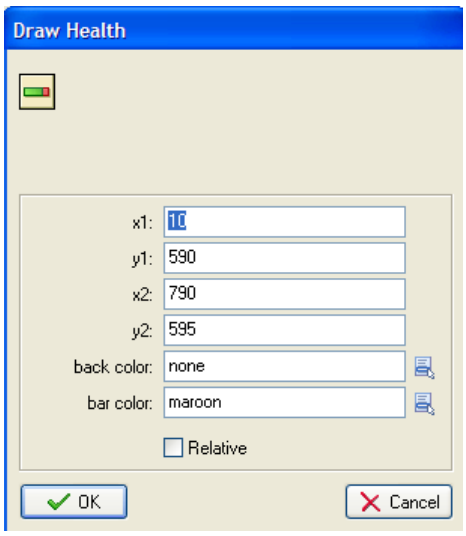
- **Als** je **geen levens** meer hebt,

  dan verschijnt er een **berichtje**

  en dan moet je **opnieuw beginnen**

- **Als** event **Draw** (=*het level wordt getekend*)

  dan moet de health balk worden getekend (**Draw the health bar**)



Zelf programmeren

7. **Open** de laatste versie van je eigen spel. Sla je spel op met een naam waarbij je het **versienummer** 1 hoger maakt.
8. Bekijk goed het object obj_controller in het voorbeeld.
9. Voeg de bovenstaande acties toe aan jouw obj_controller van het level waar je een zandloper wilt.

Snelheid zandloper aanpassen

Door in plaats van 15 in plaats van 30 te kiezen bij het aantal stappen, loopt de zandloper 2 keer zo snel. En door 60 in plaats van 30 te kiezen, loopt de zandloper 2 keer zo langzaam.

Plaats van de zandloper op het scherm aanpassen

Door bij de actie Draw health andere getallen in te vullen komt de tijdsbalk op een andere plaats op je scherm.

Hoofdstuk 11: Tenslotte

Dit was het einde van het tweede deel van de cursus. Er is nog een derde deel. Het derde deel is meer een naslagwerk. Hierin staan trucs om een platformspel te maken. Ook hier is het belangrijk om telkens eerst een ontwerp te maken, daarna dat wat je hebt ontworpen te bouwen en telkens te testen wat je hebt geprogrammeerd.

Succes verder met deel 3.
Je kunt Game Maker-deel 3 downloaden van de site van **www.gamemakerkids.nl**.

Overzicht van cursusboeken Game Maker

Game Maker-Introductie

Stap voor stap maak je een spel, waar al een begin mee is gemaakt, af.

Game Maker-Deel 1

Met dit cursusboek maak je helemaal zelf een compleet spel met veel mogelijkheden.

Game Maker-Deel 2

Je leert hoe je een spel ontwerpt. Waar moet je op letten. Wat maakt een spel tot een goed spel. Daarna ga je het zelf ontworpen spel maken.

Game Maker-Deel 3

In dit deel leer je nog meer trucs, zodat je ook andere spellen kunt maken, bijvoorbeeld een avonturenspel (platformgame).

Alle cursusboeken kunnen worden gedownload van **www.gamemakerkids.nl**.

Hoofdstuk 12: Bijlage - testschema voor de eindtest

Event	Actie	OK?
Als je geen toets indrukt	dan staat obj_IK stil	
Als je pijltje links indrukt	dan beweegt obj_IK naar links met snelheid...	

Event	Actie	OK?