Het ontwerpen van spellen met Game Maker

Versie 6.1

Geschreven door Mark Overmars

Vertaald door:
Piet Geelen
&
Nederlandse Game Maker Community
www.game-maker.nl

Klik hier om te zien wie meegeholpen hebben.

Wat is er nieuw in deze versie?

In vergelijking tot vorige versies is versie 6.0 van *Game Maker* aanzienlijk gewijzigd. Hoewel de interface er nagenoeg hetzelfde uitziet, zijn er in de kern van het programma toch vele manieren verbeteringen aangebracht. Dat heeft tot gevolg dat versie 6.0 in vele opzichten niet compatibel is met voorgaande versies. Het blijft echter wel mogelijk oudere bestanden te laden (met de extensie `.gmd`) maar die zullen hoogstwaarschijnlijk zonder wijgingen aan te brengen niet werken. Hieronder worden de belangrijkste wijzigingen uit de doeken gedaan.

Het verwerken van grafische bestanden

De graphics engine voor de spellen is volledig op de schop gegaan. Vanaf nu is die gebaseerd op Direct3D. Dat betekent dat er een hoop mogelijkheden zijn bijgekomen, maar dat het ook leidt tot enkele incompatibiliteitsproblemen met vorige versies.

- De nieuwe engine maakt gebruik van Direct3D versie 8.0. Verder is het van belang dat je DirectX versie 8.0 of later op je PC moet hebben staan, anders kun je de spellen niet spelen. Daarnaast heb je een videokaart nodig met minimaal 8MB geheugen (maar liever nog 16 MB of meer). Als dat niet het geval is zal het spel een waarschuwing in beeld brengen en vervolgens stoppen.

- Nu is het mogelijk om sprites te vergroten of te verkleinen, te draaien, gedeeltelijk transparant te maken (alpha blending) en de kleur met een andere kleur te mengen. Om dit mogelijk te maken zijn nieuwe acties toegevoegd en de oude actie om een sprite in te stellen voor een instantie is verwijderd. Dit werkt overigens alleen in de geregistreerde versie.
- De variabele `image_single` is verwijderd. Daarvoor in de plaats is de variabele `image_index` gekomen om het juiste subplaatje te kiezen waarvoor je de `image_speed` op 0 dient te zetten.
- Er is maar één tekenkleur die gebruikt wordt voor polygonen, lijnen en tekst. Als je bijvoorbeeld een rechthoek met een zwarte rand wilt tekenen moet je nu eerst de rechthoek tekenen met de opvulkleur, de kleur veranderen in zwart en dan de omtrek tekenen. In de map tekenacties is een extra actie toegevoegd waardoor het mogelijk wordt om alleen maar een omtrek te tekenen. Er zijn ook een aantal acties waarmee je vormen met een kleurgradiënt kunt tekenen.
- Ook het gebruik van fonts om letters te tekenen is compleet veranderd. Zie hieronder.
- De meeste tekenfuncties zijn veranderd. Sommige zijn van naam veranderd, bij andere zijn de argumenten veranderd. Er is een groot aantal tekenfuncties bijgekomen, bijvoorbeeld die om samengestelde polygonen te tekenen. Daarnaast zijn er functies gekomen die met het tekenen van windows en views te maken hebben, etc. Je kunt het beste het juiste hoofdstuk kiezen uit het GML-deel om meer gedetailleerde informatie hierover te vinden.
- Ook de resource functies met betrekking tot sprites en achtergronden zijn veranderd. Veel functies om sprites en achtergronden te kunnen bewerken zijn verwijderd omdat deze door de aanwezigheid van de vele nieuwe tekenfuncties overbodig zijn geworden. Daarnaast zijn zijn ook de functies load-on-use, discarding sprites en keuze van geheugen verwijderd.
- In de voorkeursinstellingen van spellen is een aantal nieuwe opties toegevoegd die betrekking hebben op de grafische mogelijkheden. De exclusive mode is verwijderd.
- Er is ook een aantal functies toegevoegd waarmee het mogelijk wordt om driedimensionale objecten te tekenen. Ook al blijft *Game Maker* een 2D spelontwikkelomgeving, kun je deze functies gebruiken om sommige 3D spellen te ontwerpen.

Fonts

In versie 6 worden de fonts op een totaal verschillende manier behandeld. Om een speciaal font te gebruiken in je spel moet je een font resource aan het spel toevoegen. In de font resource geeft je aan welk fonttype je wilt gebruiken, de lettergrootte en of de letters normaal, vet of cursief moeten zijn. Dan moet je ook nog aangeven welke karakters (de range) je wilt gebruiken. Kijk in het hoofdstuk over fonts voor meer gedetailleerde informatie. Als je eenmaal een font resource aan het spel hebt toegevoegd moet je een actie gebruiken om het font daadwerkelijk in te stellen waarna hetgebruikt kan worden om tekst te tekenen. Hiervoor bestaat ook een nieuwe functie. de nieuwe

font resource heeft als voordeel dat dat font niet op de computer hoeft te staan waarop het spel wordt gespeeld. Het is ook mogelijk om geroteerde tekst en vergrote of verkleinde tekst te tekenen, de tekst gedeeltelijk transparant te maken en zelfs kleurgradiënten toe te passen in tekst.

Geluid en muziek

Ook de geluidsengine is compleet opnieuw ontworpen. Daardoor wordt het mogelijk om bijvoorbeeld geluidseffecten toe te passen en zelfs 3D geluid. Ook is het mogelijk meerdere midi bestanden tegelijkertijd af te spelen. Het kan wel mogelijk zijn dat geluiden enigszins anders klinken dan in vorige versies, vooral midi bestanden.

Data bestanden

de data bestand resource is verwijderd. Daarvoor in de plaats is een totaal ander mechanisme gekomen om aan te geven welke bestanden in een stand-alone spel moeten worden verpakt. Dat kun je vinden in de voorkeursinstellingen van het spel. Lees de beschikbare informatie daarover nauwkeurig door om te begrijpen wanneer bestanden worden verpakt en waar zijn worden geplaatst als ze worden uitgepakt zodat het spel ze kan vinden als het gespeeld wordt.

Tiles

Alle instellingen over tiles worden opgeslagen in de achtergrond resource. Dat is zo als je als achtergrond een set tiles wilt gebruiken. Je moet dat aangeven als je de achtergrond resource aanmaakt. Je kunt daar ook de grootte van de tiles instellen.

Spelinformatie

Je kunt nu een aantal instellingen wijzigen op welke manier je de spelinformatie wilt laten zien. In het bijzonder kun je de titelbalk van het spelvenster instellen, de positie en de grootte, of er wel of geen kaders zichtbaar moeten worden en of het spel al dan niet moet doorgaan.

Acties

De acties zijn hergegroepeerd in een kleiner aantal libraries. Oude acties die te maken hadden met compatibiliteit zijn er uitgehaald. Een aantal andere acties is ook verwijderd, met name de aanmaak van een achtergrond met een kleurgradiënt. Het blijft nog steeds mogelijk om libraries die gemaakt zijn door anderen toe te voegen, voorwaarde is wel dat zij gebruik maken van functies die niet zijn gewijzigd. De layout van de acties libraries zou nog kunnen worden aangepast.

Andere veranderingen

Er zijn nog vele wijzigingen en toevoegingen. Hieronder staan de belangrijkste.

- De mogelijkheid om de hoeken van de sprites automatisch af te ronden.
- De mogelijkheid om de grootte van het spelvenster te wijzigen.
- Geroteerde views.
- Transparent, colorized, or texture mapped primitives.
- Een verbeterd helpbestand.
- Vloeiendere (maar minder) overgangen.
- De mogelijkheid om de room niet automatisch te laten tekenen.
- Het sneller laden van spellen.
- Een betere synchronisatie met de verversingsfrequentie van het scherm om tearing te voorkomen.
- Een betere mogelijkheid om te wisselen van applicaties.
- De mogelijkheid om event te dupliceren en meerdere acties tegelijk te kunnen kopiëren.
- Afzonderlijke globale muis event voor de verschillende knoppen.
- ...

Het gebruik van Game Maker

Game Maker is een programma waarmee je eenvoudig je eigen computerspellen kunt maken. Dit

Het spelen van computerspellen is leuk. Maar wat eigenlijk veel leuker is, is het maken van je eigen computerspellen en anderen die laten spelen. Helaas is het maken van computerspellen niet eenvoudig. Commerciële computerspellen zoals je ze tegenwoordig kunt kopen hebben een ontwikkeltijd van ongeveer 3 jaar waarmee een ontwikkelteam van tussen de 10 en 50 personen betrokken is. De beschikbare budgetten bedragen al gauw in de miljoenen dollars. Bovendien zijn deze mensen ook nog eens bijzonder goed op hun vakgebied: programmeurs, ontwerpers, geluidstechnici, enz.

Betekent dit nu dat het onmogelijk is om je eigen computerspelen te maken? Gelukkig niet. Natuurlijk moet je niet verwachten dat je je eigen *Quake* of *Age of Empires* in een paar weken kunt maken. Maar dat is ook helemaal niet nodig. Een stuk eenvoudigere spellen dan *Tetris*, *pacman*,

Space Invaders, enz. zijn ook best aardig om te spelen en veel eenvoudiger om te maken. Helaas vereisen zij ook nog goede kennis van programmeren om met grafisch werk, geluid, interacties enz. te kunnen omgaan.

Maar nu bestaat het programma *Game Maker*. Dat is ontwikkeld om het voor jou een stuk eenvoudiger te maken om zulke spellen te maken. Er komt zo goed als geen programmeerwerk meer bij kijken. Een intuïtieve en eenvoudig te gebruiken interface waarin je alles met verplaatsen in een scherm kunt bereiken, maakt het mogelijk om snel je spellen te kunnen maken. Je kunt afbeeldingen importeren of zelf maken, evenals sprites (animated gifs), geluiden, ruimtes enz. en ze daarna meteen gebruiken. Je kunt eenvoudig je objecten definiëren die in een spel voorkomen en er gedrag aan toekennen. Bovendien is het mogelijk spellen te ontwikkelen waarin meerdere levels voorkomen met verschillende moeilijkheidsgraden. Indien je zelf alles in de hand wilt houden is dat ook mogelijk; het programma bevat namelijk een eigen programmeertaal, die jou de mogelijkheid biedt om je programma volledig naar jouw hand te zetten. Daarvoor moet je toch wel enige programmeerkennis hebben.

Game Maker is vooral bedoeld om tweedimensionale spellen mee te ontwikkelen. Dus geen driedimensionale spellen als *Quake*. Maar laat dat vooral geen teleurstelling zijn. Veel grote spellen zoals *Age of Empires*, de *Command and Conquer*-serie en *Diabolo* gebruiken tweedimensionale sprite-technieken, hoewel ze erg driedimensionaal lijken. Bovendien is het maken van tweedimensionale spellen eenvoudiger en sneller.

Van *Game Maker* bestaan twee versies: een gratis versie en een geregistreerde versie. De gratis versie is ook echt gratis. Je mag de spellen die je ermee maakt vrij verspreiden, je kunt er zelfs geld voor vragen. Bekijk hiervoor de licentievoorwaarden die te vinden zijn op de website en in de map programma's als je Game Maker hebt geïnstalleerd. Maar je wordt toch ten zeerste aangemoedigd om je kopie van *Game Maker* te laten registreren. Daardoor krijg je de beschikking over een aantal extra features in *Game Maker* en bovendien verdwijnt dan ook het logo als je spellen speelt. Bovendien draag je dan bij aan de verdere ontwikkeling van *Game Maker*.

In deze handleiding wordt alles uitgelegd wat je moet weten over *Game Maker* en hoe je ermee spellen kunt maken. Je moet je echter wel realiseren dat het maken van spellen niet zomaar iets is. Er zijn bij het maken van spellen echt een aantal dingen belangrijk: wat voor soort spel wordt het, hoe moet de omgeving eruit zien, welke figuren spelen een rol, welke acties komen er in voor, welke geluiden voeg je er aan toe, enz. Begin eerst met het ontwerpen van een zeer eenvoudig spelletje om inzicht te krijgen hoe het werkt. Je zult merken dat het veel plezier oplevert om spellen te maken.
Bekijk eens de volgende website:

http://www.gamemaker.nl/

daar zul je veel voorbeelden, ideeën en hulp kunnen vinden. Bovendien is er een forum waaraan je kunt deelnemen. Na flink wat oefenen wordt jij misschien zelf een super spelontwikkelaar.

Veel plezier met het maken van spellen in *Game Maker*.

Installatie

Waarschijnlijk heb je het programma al geïnstalleerd, maar als dat nog niet het geval is, kun je hier de installatieprocedure volgen. Klik na het downloaden op `gmaker.exe`. Volg de instructies die op het scherm verschijnen. Je kunt het programma overal op je PC installeren, maar toch kun je het beste de standaard installatiesuggesties volgen. Als de installatie eenmaal voltooid is, zul je in het startmenu een nieuwe programmagroep aantreffen waar je *Game Maker* mee kunt starten en de documentatie kunt opvragen.

Als je *Game Maker* voor het eerst opstart wordt je gevraagd of je het programma in de **Simple** or **Advanced** mode wilt draaien. Indien je nog nooit met een spelontwikkelprogramma hebt gewerkt en je bent geen ervaren programmeur, kun je het beste kiezen voor de simple mode (dus selecteer dan **No**). In de simple mode zijn er veel minder functies zichtbaar op het scherm. In een later stadium kun je eenvoudig overschakelen op de advanced mode door het betreffende item in het **File**-menu te kiezen.

Binnen de installatiemap (standaard `C:\Program Files\Game_Maker6\`) zijn een aantal andere mappen aangemaakt:

- **`examples`** bevat een aantal voorbeelden van spellen die je kunt gebruiken en veranderen, bedoeld als oefeningen.
- **`lib`** bevat een aantal libraries met actions. Als je later nog extra actionlibraries wilt toevoegen, dan moet je ze in deze map plaatsen.
- **`sprites`** in deze map staan sprites die je vrij kunt gebruiken. Standaard wordt er slechts een beperkt aantal sprites tijdens de installatie in de map geplaatst maar op de *Game Maker* website (http://www.gamemaker.nl/) kun je een aantal resource packs downloaden waarin extra sprites, geluiden, achtergronden, etc. te vinden zijn.
- **`backgrounds`**, **`sounds`** in deze mappen bevinden zich afbeeldingen van achtergronden en geluiden.

Requirements

Om *Game Maker* te gebruiken heb je een moderne Pentium PC nodig waarop als besturingssysteem Windows 98SE, 2000, Me, XP, of hoger geïnstalleerd is. Verder heb je een

videokaart met minimaal 16MB nodig voor de meeste spellen. Alleen voor de simpelste spellen kun je volstaan met een videokaart met 8MB. De monitor moet minimaal een resolutie van 800x600 ondersteunen met 65000 (16-bit) kleuren. Daarnaast is DirectX versie 8.0 of hoger vereist op je PC. De nieuwste versie ervan kun je downloaden van de Microsoft website: http://www.microsoft.com/windows/directx/.)
Bij het ontwerpen en testen van je spellen heb je behoorlijk wat RAM-geheugen nodig (minimaal 64 MB, maar hoe meer hoe beter). Tijdens het afspelen van de spellen kun je met minder geheugen toe, maar het is wel afhankelijk van het type spel dat je hebt gemaakt.

Registratie

Game Maker kan vrij gebruikt worden zonder te betalen. Maar de ongeregistreerde versie zal tijdens het spelen een klein logo in beeld hebben. Om de extra features te verkrijgen, het logo te laten verdwijnen en verdere ontwikkeling van de software mogelijk te maken, raden we je ten zeerste aan om je kopie van *Game Maker* te laten registreren. Na registratie heb je de beschikking over de volgende extra features:

- Geen *Game Maker* logo tijdens het spelen van de spellen.
- Rotated, color blended and translucent sprites.
- Extra actions voor bijvoorbeeld CD-muziek, geroteerde tekst, en colorized shapes.
- Speciale geluidseffecten en in de ruimte geplaatste geluiden.
- Een aantal extra tekenfuncties waaronder textured polygons.
- Een particle systeem waarmee je vuurwerk, vlammen, regen en andere effecten kunt maken.
- 3D-graphics functies.
- De mogelijkheid om multiplayer games te maken die over een netwerk kunnen worden gespeeld.
- Functies om resources te maken of te veranderen (sprites, backgrounds, etc.)tijdens het spel.
- Een verzameling functies om datastructuren te maken en te gebruiken.
- Functies om bewegingen te plannen.
- De mogelijkheid om *Game Maker* uit te breiden door het gebruik van DLLs.

De kosten van registratie voor *Game Maker* bedragen slechts 15 Euro of een gelijke hoeveelheid geld in andere valuta, op dit moment bijvoorbeeld US $ 18. Er zijn verschillende manieren waarop je je kopie kunt laten registreren. De eenvoudigste manier is de online-registratie door gebruik te maken van een veilige betaling middels en creditcard of door middel van een PayPal account. Je kunt ook geld overmaken op onze bankrekening of cash geld opsturen. Details hierover kun je vinden op de *Game Maker* registratie website:

http://www.gamemaker.nl/registration.html

Ga om je kopie van *Game Maker* te registreren naar bovenstaande website. Dat kan door het programma te starten en daarna te kiezen voor **Registration** uit het **Help**-menu. Klik vervolgens op de knop **Go to Registration Webpage** in de linkerkolom van het popup-venster dat dan in beeld verschijnt. Van daaruit wordt je rechtstreeks geleid naar de pagina op de *Game Maker*-website waar je de verschillende betaalopties in beeld krijgt, inclusief de online betaling.

Als het geld van de registratie is ontvangen, krijg je spoedig daarna een e-mail met de naam waarop de kopie is geregistreerd en de key en bovendien informatie hoe je die gegevens in het programma kunt verwerken. Om de key in het programma te verwerken moet je opnieuw kiezen voor **Registration** uit het **Help**-menu. Klik vervolgens op de knop **Enter a Registration Key** in de linkerkolom van het popup-venster dat dan in beeld verschijnt. Typ vervolgens de naam en de key in en klik daarna op **OK**. Als je daarbij geen fouten maakt, zal het programma zijn geregistreerd.

Indien je in het bezit bent van een registratiekey van versie 5 van *Game Maker* kun je deze key gebruiken om je versie 6 van *Game Maker* te registreren. Om dat voor elkaar te krijgen ga je naar **Registration** uit het **Help**-menu. Als het goed is staat er aan de linkerkant een knop **Convert a Version 5 Key** (zo niet, dan ben je niet in het bezit van een geldige registratie van versie 5 of je versie 6 is al geregistreerd.) Bovendien verscijnt er dan een tekst in beeld waarin beschreven staat hoe je keys kunt converteren. Lees de aanwijzingen allereerst rustig en zorgvuldig door alvorens je op de knop drukt.

Het globale idee

Alvorens te duiken in de mogelijkheden van *Game Maker* is het goed om eerst een idee te krijgen wat het globale idee achter dit programma is. Spellen die gemaakt zijn met *Game Maker* spelen zich af in één of meer rooms. (Rooms zijn vlak, niet 3D, maar zij kunnen er 3D-achtig uitzien). In deze rooms plaats je objecten, die je met het programma kunt definiëren. Typische objecten zijn: muren, bewegende ballen, monsters, een hoofdfiguur, enz. Sommige objecten zoals muren staan in de room en doen verder niets. Andere objecten zoals een hoofdfiguur kunnen bewegen door de room en reageren op input van de speler (toetsenbord, muis, joystick) en op andere objecten. Bijvoorbeeld, als de hoofdfiguur een monster ontmoet kan dat zijn dood betekenen. Objecten zijn de belangrijkste onderdelen van de spellen die met *Game Maker* worden gemaakt. Laten we daarom op objecten iets dieper ingaan.

Allereerst hebben objecten een uiterlijk (afbeelding) nodig om ze zichtbaar te maken op het beeldscherm. Zulke afbeeldingen noemen we *sprites*. Een sprite is soms een enkele afbeelding

maar het kan ook voorkomen dat hij bestaat uit meerdere afbeeldingen na elkaar (animated gif). Dan lijkt het bijvoorbeeld of een persoon loopt, een bal draait, een ruimtevaartuig explodeert, enz. Tijdens het spel kan de sprite van een object veranderen. (een persoon ziet er anders uit als hij naar links of naar rechts loopt) In *Game Maker* kun je je eigen sprites maken of je kunt ze in het spel halen door kant en klare files te openen (bijvoorbeeld animated GIF's).

Met objecten gebeuren bepaalde dingen. Deze gebeurtenissen noemen we *events*. Objecten kunnen bepaalde *actions* (handelingen) verrichten als een event plaatsvindt. Er kan een groot aantal events plaatsvinden en er is een groot aantal handelingen die verricht kunnen worden door de objecten. Bijvoorbeeld: er is een *creation event* als een object wordt gemaakt. (Om preciezer te zijn: als een instantie van een object wordt gemaakt; er kunnen meerdere instanties van hetzelfde object zijn). Als een bal wordt gemaakt kun je die bal een beweging meegeven zodat hij begint te bewegen als hij gemaakt wordt. Als twee objecten elkaar treffen treedt er een *collision event* (botsing) op. In zo'n geval kun je ervoor zorgen dat de bal stopt of een beweging krijgt in een andere richting, bijvoorbeeld de tegenovergestelde richting. Je kunt bij die botsing ervoor zorgen dat er een geluid wordt geproduceerd. In *Game Maker* bestaat de mogelijkheid om *sounds* (geluiden) toe te voegen aan events. Je kunt ook *sounds* in *Game Maker* definiëren. Als een speler van een spel een toets van het toetsenbord indrukt is er sprake van een *keyboard event*, waarna een object de juiste handeling kan gaan verrichten, bijvoorbeeld bewegen in een bepaalde richting. Ik hoop dat het idee een beetje duidelijk is. Voor elk object dat je ontwerpt kun je actions aangeven voor verschillende events. Dit is de manier om het gedrag van een object te definiëren.

Als je eenmaal je objecten hebt gedefinieerd, wordt het tijd om de *rooms* waarin die objecten zich gaan bevinden te ontwerpen. Die rooms kunnen gebruikt worden voor de verschillende levels van het spel of om verschillende plaatsen aan te geven. Er bestaan actions waardoor je van de ene room in de andere kunt komen. Rooms hebben op de eerste plaats een *background* (achtergrond). Dit kan eenvoudig een kleur zijn of een afbeelding. Zulke achtergrondafbeeldingen kun je met Game Maker ontwerpen of je kunt ze importeren in *Game Maker* als bestaand bestand. (De achtergrond kan een hoop dingen doen maar voorlopig beschouwen we het maar als iets dat de room er mooi laat uitzien.) Daarna kun je de objecten in de room plaatsen. Je kunt meerdere instanties van een zelfde object in de room plaatsen. Zo hoef je maar één muurobject te definiëren terwijl je het op vele plaatsen in de room kunt gebruiken. Bovendien kun je ook meerdere instanties van hetzelfde monsterobject in je spel hebben, zolang ze maar hetzelfde gedrag hebben.

Nu ben je zover om het spel uit te voeren. De eerste room verschijnt in beeld met de objecten daarin die acties uitvoeren die ze bij hun creation event hebben meegekregen. Ze beginnen op elkaar te reageren dankzij de actions in de collision events maar ze kunnen ook reageren op de events van de speler die worden ingegeven met de muis of het toetsenbord.

Samenvattend kunnen we zeggen dat de volgende zaken (vaak worden deze resources genoemd) een belangrijke rol spelen:

- *objecten*: dit zijn de echte entiteiten in een spel
- *rooms*: de plaatsen (levels) waarin de objecten leven
- *sprites*: animated) images die gebruikt worden om de objecten weer te geven
- *sounds*: deze kunnen in spellen zowel worden gebruikt als achtergrond als ook voor effecten
- *backgrounds*: de afbeeldingen die gebruikt worden als achtergrond in de rooms

Er bestaan echter nog een aantal andere bronnen: paden (*paths*), *scripts*, *fonts* en tijdlijnen (*time lines*). Dit zijn belangrijke onderdelen die vooral in wat meer gecompliceerde spellen een rol spelen. Je komt deze dingen alleen tegen als je *Game Maker* in de advanced mode gebruikt. Deze onderdelen worden later in aparte hoofdtukken voor gevorderden behandeld.

Een spelvoorbeeld nader bekeken

Het is goed om allereerst eens te kijken hoe een erg eenvoudig spel gemaakt kan worden. We nemen aan dat je *Game Maker* in de simple mode gebruikt. Allereerst moeten we omschrijven welk spel we gaan maken. (dit moet je altijd als eerste doen; dat scheelt je later een hoop tijd.) Het spel dat we gaan maken is erg eenvoudig: een bal botst tegen muren die in een rechthoek zijn geplaatst. Het is de bedoeling dat je de bewegende bal met de linker muiskop aanklikt. Elke keer als dat gelukt is wordt er een geluid geproduceerd, bijvoorbeeld van een geweerschot, en wordt de score opgehoogd met 1.

Zoals uit de beschrijving is af te leiden zijn er 2 objecten nodig: de bal en de muur. Bovendien hebben we ook twee verschillende sprites nodig: een voor het muurobject en een voor het computerspel. Tot slot willen we een geluid horen als we erin geslaagd zijn de bal met de muis aan te klikken. We zullen slechts 1 room gebruiken waarin het spel zich afspeelt. (Als je het spel niet zelf wilt maken, kun je het ook openen; het staat in de map `Examples` onder de naam `hit the ball.gm6`.)

Laten we allereerst de sprites maken:

1. Selecteer uit het menu **Add** de optie **Add Sprite**. (je kunt ook het geschikte icoon gebruiken op de werkbalk) Er verschijnt een venster. Vul in het veld **Name** "wall" in.
2. Kies vervolgens de **Load Sprite** knop en kies een geschikte afbeelding (die kun je onder andere vinden in de map "maze"). Dat is alles. Je kunt daarna het venster sluiten.
3. Op dezelfde manier als boven beschreven kun je een "ball"-sprite maken.

Vervolgens voegen we een geluid toe.

1. Selecteer uit het menu **Add** de optie **Add Sound**. Een ander venster verschijnt nu in beeld. Geef het geluid dat je kiest een naam en kies daarna **Load Sound**.
2. Kies een bijpassend geluid, om te controleren of het inderdaad geschikt is kun je op de "play"-knop drukken.
3. Als je het geschikte geluid gevonden hebt kun je het venster sluiten.

Nu moeten we de 2 objecten gaan maken.

1. Allereerst het object "wall". We kiezen weer van het menu **Add** de optie **Add Object**. Ook nu verschijnt er weer een venster maar dat zit ietwat ingewikkelder in elkaar. Aan de linkerkant kun je enige algemene informatie over het object vinden.
2. Geef het object een geschikte naam en kies uit het "drop down-menu" de juiste "wall"-sprite. Aangezien een muur onbeweeglijk is, moet je een vinkje zetten in de box **Solid**. Voorlopig laten we het hierbij.
3. Op een zelfde manier als onder 1 maken we een tweede object en noemen dat "ball", kies daarbij de juiste sprite en maak de "ball" niet solid. Vervolgens gaan we aan de bal enig gedrag toekennen. In het midden zie je een lege lijst met events. Onderaan daarvan zie je een knop met het label **Add Event**. Als je daarop drukt krijg je alle mogelijke events te zien. Kies daarvan de **creation event**. Die event wordt nu aan de lijst toegevoegd. Helemaal aan de rechterkant van het venster zie je alle mogelijke acties, in groepen ondergebracht. Van de **move**-groep kies je de actie met de 8 rode pijlen en sleept die naar het midden in de lijst. Hierdoor kan het object in een bepaalde richting gaan bewegen. De "speed" kun je op 8 laten staan. Meteen als je de actie in de lijst hebt gesleept verschijnt er een nieuw venster waarin je de richtingen van de beweging van het object kunt aangeven. Kies alle 8 de pijlen waarmee je wilt aangeven dat het object alle kanten op kan gaan (random). De *speed* kun je op 8 laten staan. Sluit daarna het venster. Nu kan de bal gaan bewegen in een onbepaalde richting zo gauw hij wordt gecreëerd.
4. Vervolgens moeten we gaan aangeven wat er gebeurt als een bal tegen de muur botst. Druk opnieuw op de knop **Add Event**. Zoek de "collision event"-knop (met de 2 kleine rode pijlen die elkaar raken, de vijfde van boven, links) en selecteer vervolgens in het drop-down-menu het "wall"-object. Voor deze actie maken we gebruik van de "bounce"-actie (om te controleren wat elke actie doet, kun je de muis erboven houden en dan verschijnt er een tekstveld met de actie erin)
5. Tenslotte moeten we nog definiëren wat er gebeurt als de gebruiker de linker muisknop indrukt op de bal. Zoek de juiste event en selecteer de linker muisknop van het pop-up menu. Voor deze event hebben we een paar acties nodig: één om het geluid te laten afspelen (kan gevonden worden in de groep van **main1**-actions. Kies het juiste geluid.

Daarna een om de score bij te houden (in de groep **score**). Vul de waarde 1 in en plaats tevens een vinkje in de box Relative (Dit betekent dat er 1 bij de huidige score wordt opgeteld.). Daarna nog twee om de bal op een nieuwe plaats te laten beginnen en in een nieuwe (random) richting te laten verplaatsen (op dezelfde manier als bij de creation event) Als je bij het invullen van de gegevens ergens een fout maakt kun je deze weer herstellen door op een actie te dubbelklikken en de waarden te wijzigen.

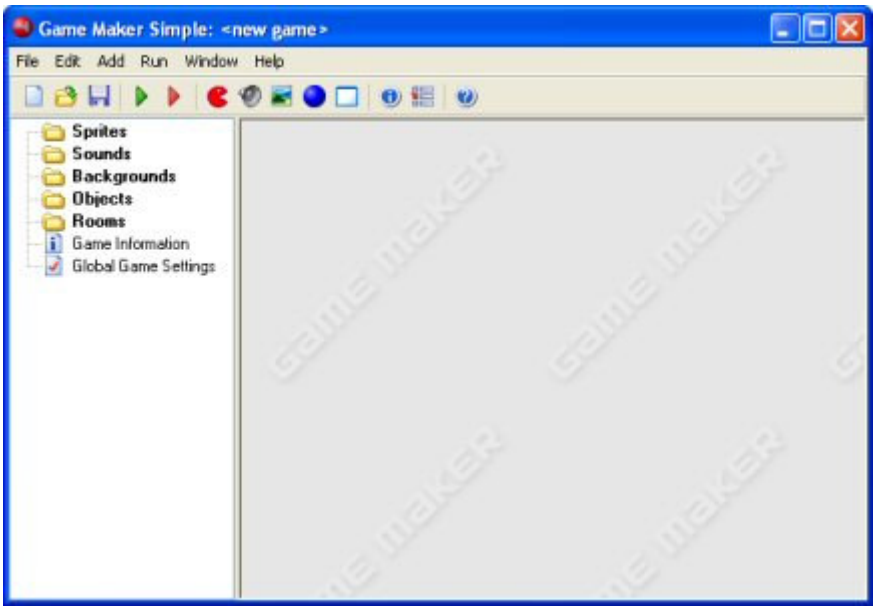
Onze objecten zijn nu klaar. Wat nog overblijft is het maken van een ruimte. Selecteer uit het **Add**-menu **Add a new room**. Aan de rechterkant zie je een lege ruimte. Aan de linkerkant zie je enige mogelijkheden waarmee je die room kunt wijzigen, zoals hoogte en breedte van de ruimte. Links onder staat een pop-up menu waaruit je objecten kunt kiezen die je aan de room kunt toevoegen op een door jou te kiezen plaats. Je kunt met een muisklik objecten overal plaatsen. Met de rechter muisknop kun je de objecten weer verwijderen. Plaats nu een muur in de room en vervolgens 1 of 2 ballen erin. Klaar is Kees.

Nu is het tijd om ons spel te testen. Druk op de **Run**-knop en wacht af wat er gebeurt. Als je geen fouten hebt gemaakt beginnen de ballen rond te bewegen. Probeer dan de ballen met de muis aan te klikken en kijk dan eens wat er gebeurt. Het spel is altijd te stoppen door op <Esc> te drukken. Als iets niet goed is uitgevoerd heb je de mogelijkheid veranderingen aan te brengen.

Gefeliciteerd!! Je hebt nu je eerste spelletje in elkaar gezet. Maar ik denk dat het nu tijd is om iets meer over **Game Maker** te weten te komen.

De algemene userinterface

Als je het programma *Game Maker* start verschijnt het volgende venster op het scherm:



In feite is dit het venster dat je te zien krijgt als je *Game Maker* in de simple mode opstart. In de advanced mode krijg je nog een aantal extra items te zien.) Aan de linkerzijde zie je verschillende bronnen: Sprites, Sounds, Backgrounds, Objects, Rooms en nog twee: Game Information en Global Game Settings. Bovenaan zijn de bekende menubalk en werkbalk te zien. In dit hoofdstuk zal ik kort de verschillende menu-items, knoppen enz. beschrijven. In de hoofdstukken verderop zal ik een aantal van die zaken gedetailleerd beschrijven. Merk op dat vele dingen op verschillende manieren kunnen worden bereikt: door een commando uit het menu, door op een knop te drukken of door met de rechtermuisknop op een resource te klikken.

File menu

In het file menu kun je enkele van de gebruikelijke commando's vinden zoals het laden en opslaan van files plus een paar speciale:

- **New**. Kies dit commando om een nieuw spel te gaan ontwerpen. Als het huidige spel is veranderd, krijg je de vraag of die veranderingen moeten worden opgeslagen. Er is ook een knop op de werkbalk voor de optie New.
- **Open**. opent een bestaand spel. *Game Maker*-files hebben de extensie .gm6. (Je kunt ook .gmd-files openen die met een oudere versie van *Game Maker* zijn gemaakt. Het kan echter zijn dat deze niet helemaal correct werken in versie 6 van *Game Maker*.) Op de werkbalk is ook een icoontje te vinden voor de optie open te vinden. Je kunt ook een spel openen door een file in het *Game Maker* venster te slepen.
- **Recent Files**. Gebruik dit submenu om spellen te openen die de onlangs hebt geopend.
- **Save**. slaat een spel op onder zijn huidige naam. Als er van te voren geen naam was toegekend aan het spel vraagt Game Maker jou om een naam te geven. Dit commando is alleen maar te gebruiken als er wijzigingen in het spel zijn aangebracht. Ook voor de optie save vind je een knop op de werkbalk.
- **Save As**. slaat het spel op onder een andere naam. Je wordt gevraagd een nieuwe naam aan het spel te geven.
- **Create Executable**. Gebruik dit commando om een stand-alone versie van je spel te maken dat je aan andere mensen kunt geven.
- **Advanced Mode**. Als je hierop klikt zal*Game Maker* switchen yussen de simple en advanced mode. In de advanced mode zijn extra commando's en resources beschikbaar.
- **Exit**. Dat spreekt voor zich. Klik hierop om *Game Maker* te beëindigen. Als je het openstaande spel hebt gewijzigd, zal *Game Maker* vragen of de wijzigingen moeten worden opgeslagen.

Edit menu

Het edit menu bevat een aantal commando's dat betrekking heeft op de op dat moment geselecteerde voorwerp (object, sprite, geluid, enz.) of groep voorwerpen. Afhankelijk van het type voorwerp kan het zijn dat een of meerdere commando's niet beschikbaar zijn.

- **Insert resource**. Voeg een nieuwe instantie toe van het huidig gekozen type voorwerp vóór het huidige exemplaar. Als je een groep voorwerpen hebt geselecteerd zal het voorwerp in de groep worden toegevoegd. Een venster zal op het beeldscherm verschijnen waarin je de eigenschappen van het voorwerp kunt wijzigen. In de volgende hoofdstukken zal hierover in detail worden ingegaan.
- **Duplicate**. Hiermee kun je een kopie maken van het huidige voorwerp en voegt het vervolgens toe. Er verschijnt een venster waarin je het voorwerp kunt veranderen.Makes a copy of the current resource and adds it.
- **Delete**. Verwijdert het op dat moment geselecteerde voorwerp (of groep voorwerpen). Wees daar voorzichtig mee. Er is geen optie om dit ongedaan te kunnen maken. Daarvoor word je echt gewaarschuwd.
- **Rename**. Geeft het voorwerp een andere naam. Dat kun je ook realiseren in het eigenschappenvenster van het voorwerp.
- **Properties**. Met dit commando kun je het eigenschappenvenster van een voorwerp oproepen om de eigenschappen te bewerken. Merk op dat alle eigenschappenvensters binnen het hoofdscherm verschijnen. Je kunt vele voorwerpen tegelijkertijd bewerken. Je kunt het eigenschappenvenster ook oproepen door op het voorwerp dubbel te klikken.

Merk op dat al deze commando's ook op een andere manier aangeroepen kunnen worden. Door met de rechtermuis op een voorwerp of voorwerpengroep te klikken verschijnt er een pop-up menu.

Add menu

In dit menu kun je nieuwe voorwerpen van elk type object toevoegen. Merk op dat er ook voor elk type een knop op de werkbalk voorkomt en dat er voor elk type een sneltoetscombinatie bestaat.

Run menu

Dit menu wordt gebruikt om het spel te runnen. Dat kan op twee manieren:

- **Run normally**. Het spel runt op de gebruikelijke manier. Dat wil zeggen op de meest efficiente wijze en alsof het draait als een executable spel.
- **Run in Debug mode**. Het spel runt in de debug mode. Op deze manier kun je bepaalde aspecten van het spel controleren, je kunt het spel stopzetten en rechtstreeks naar

bepaalde delen van het spel gaan. Dit is vooral handig als er bepaalde dingen niet goed gaan in het spel, maar maak hier alleen gebruik van als je al wat meer inzicht hebt in het werken men *Game Maker*.

Window menu

In dit menu vind je enkele gebruikelijke commando's om windowmanagement toe te passen in het hoofdvenster. Je kunt daar kiezen voor:

- **Cascade**. Zet alle vensters in een cascade onder elkaar, zodat ze slechts gedeeltelijk zichtbaar zijn.
- **Arrange Icons**. Groepeert alle eigenschapvensters met een icoon (met name handig als je het hoofdvenster van grootte gaat veranderen).
- **Close All**. Deze optie sluit alle eigenschappenvensters, maar eerst wordt er nog gevraagd om de wijzigingen op te slaan of niet.

Help menu

In dit menu vind je enige commando's terug die je kunnen helpen, zoals:

- **Contents**. Met dit commando roep je de online versie van deze handleiding op.
- **Registration**. Hoewel de standaardversie van *Game Maker* gratis is, willen we je toch aanmoedigen om je kopie te laten registreren. Hierdoor krijg je de beschikking over een aantal extra features en draag je bij tot de verdere ontwikkeling van het programma. Je vindt hier de informatie die je nodig hebt om je programma te registreren. Als je programma. na betaling, is geregistreerd, kun je dit commando gebruiken om de ontvangen registratiekey in te geven.
- **Web site**. Deze optie verbindt je met de *Game Maker* website, waar de meest actuele informatie over de meest recente versie van *Game Maker* kunt vinden samen met een verzameling spellen, documentatie en resources. Ik adviseer je om minstens eenmaal per maand op de site te kijken voor nieuwe informatie.
- **About *Game Maker***. Dit geeft enige korte informatie over de gebruikte versie van *Game Maker*.

The resource explorer

Aan de linkerkant van het hoofdvenster vind je de resource explorer. Je treft hier een in boomvorm een lijst met alle voorwerpen (objecten, sprites, achtergronden etc.) aan die in je spel voorkomen. Hij werkt hetzelfde als de Windows verkenner, waarvan ik aanneem dat je daar aardig mee uit de

voeten kunt. Als een bepaald item aan de linkerkant voorzien is van een +(plus-) teken, houdt dat in dat dit item meerdere items in zich heeft, net als een map, waarin nog meerdere mappen zitten. Als je erop klikt, komen de onderliggende voorwerpen tevoorschijn en verandert het plusteken in een minteken. Als je daarna op het minteken klikt, wordt de map weer gesloten en verandert het minteken weer in een plusteken. Je kunt de namen van voorwerpen veranderen, alleen de namen op het hoogste niveau zijn standaard en daardoor onveranderbaar. Je moet dan het voorwerp selecteren (met een enkele klik van de muis) en daarna op de naam klikken. Om de eigenschappen van een voorwerp aan te passen moet je op het voorwerp dubbelklikken. Met de rechtermuisknop kun je dezelfde commando's bereiken als met het Edit menu.

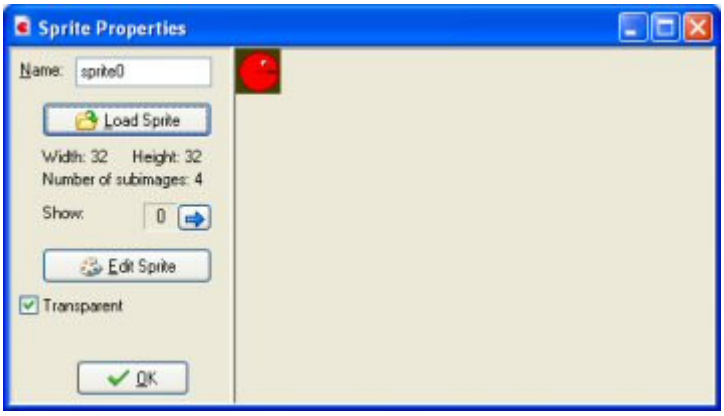
Je kunt in de resource explorer de volgorde van de voorwerpen wijzigen door met de rechtermuisknop een voorwerp te selecteren, de muisknop ingedrukt te houden en vervolgens naar de gewenste plaats te slepen en daarna de muisknop los te laten. Dit kan ook met hele groepen. Het is overigens wel zo dat je voorwerpen alleen maar in de juiste groep kunt verplaatsen. Het is dus niet mogelijk geluiden te verslepen naar de spritesgroep of omgekeerd.

Het definiëren van sprites

Sprites zijn de visuele representaties van alle objecten in het spel. Een sprite is of éé plaatje op zich, getekend met elk willekeurig programma, of een set van plaatjes, die werken als een animatie. Bijvoorbeeld, de volgende vier plaatjes vormen een sprite in Pacman waarbij het naar rechts beweegt.

Wanneer je begint met het ontwerp van een spel, ga je sprites verzamelen als objecten in jou spel. Een sprite collectie, zowel statisch als geanimeerd, wordt door *Game maker* bijgeleverd. Andere sprites kun je vinden op het web in de vorm van ”animated gifs”.

Om een sprite toe te voegen kies je **Add Sprite** in het **Add**-menu, of maak gebruik van de corresponderende knop in de werkbalk. Er verschijnt dan het volgende venster in beeld:



Bovenaan kun je de naam van de sprite aangeven. Alle sprites (en alle andere resources) hebben een naam. Je kunt het beste kiezen voor een duidelijk omschrijvende naam. Zorg ervoor dat alle bronnen een verschillende naam hebben. Gebruik als het even mogelijk is alleen letters en getallen en het underscore symbool (_) in de naam van een sprite (of een andere bron) en laat de naam beginnen met een letter. Gebruik absoluut geen spaties! Dit is vooral van belang als je later scriptcode gaat gebruiken.

Om een sprite te laden moet je klikken op de knop **Load Sprite**. Er verschijnt dan een bestandsvenster dat je de mogelijkheid biedt om uit de gewenste map je sprite te laden.In *Game Maker* is het mogelijk om grafische bestanden met verschillende extensies te gebruiken. Als je een animated gif laadt vormen de verschillende plaatjes van de animated gif ook de verschilllende sprites. Op het moment dat een sprite is geladen, verschijnt het plaatje aan de rechterkant in het venster. Bij een animated gif komen alle plaatjes tegelijk in het rechter venster te staan. Met de pijltjes-toetsen kun je dan elk plaatje selecteren.

Het keuzeveld met de naam **Transparent** geeft aan of de rechthoekige achtergrond van de sprite al dan niet transparant moet zijn. De meeste sprites zijn transparant. De achtergrondkleur wordt bepaald door de kleur van de pixel linksonder in de hoek van het plaatje. Zorg er in ieder geval voor dat geen enkele pixel van het plaatje deze kleur heeft. (Let op! GIF-files bepalen vaak hun eigen transparantiekleur. Deze kleur wordt niet door *Game Maker* gebruikt!)

Met de knop **Edit Sprite** kun je een sprite bewerken of zelfs compleet nieuwe sprites maken.

Geluiden en muziek

De meeste spellen hebben bepaalde geluidseffecten en bepaalde achtergrondmuziek. Sommige nuttige geluidseffecten worden geleverd op de *Game Maker*-website. Veel meer geluidseffecten zijn te vinden op andere plaatsen op het web.

Om bepaalde geluidsbronnen toe te voegen aan je spel, gebruik je het item **Add Sound** in het **Add**-menu of je kunt gebruik maken van de corresponderende button in de menubalk. Het volgende scherm verschijnt:

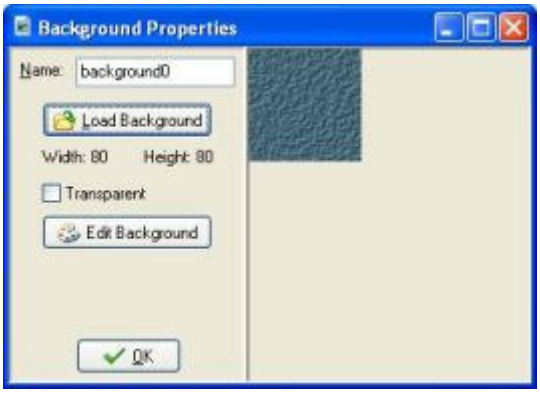
Om een geluid te laden, klik je op de knop met **Load Sound**. Een dialoogscherm verschijnt, waarin je het geluidsbestand kunt selecteren. Er zijn twee soorten geluidsbestanden, ”wave files” en ”midi files”. ”Wave files” worden gebruikt voor korte geluidseffecten. Deze files gebruiken veel geheugenruimte, maar worden onmiddellijk afgespeeld. Gebruik deze files voor al je geluidseffecten in je spel. ”Midi files” beschrijven muziek op een andere manier. Zij gebruiken veel minder geheugen, maar blijven beperkt tot instrumentale achtergrondmuziek. Eén ”midi file” kun je laten spelen en niet meerdere tegelijkertijd.

Wanneer je een muziek file laadt, worden zijn lengte en aard vertoond. Je kunt het beluisteren terwijl je gebruik maakt van de playbutton. Bovendien is er een knop **Save Sound** om het geluid op te slaan. Deze knop is niet echt vereist, maar het kan handig zijn als je het oorpronkelijke geluid kwijt bent.

Achtergronden

Het derde type basisbronnen aanwezig in *Game Maker* zijn de achtergronden. Het zijn meestal grote afbeeldingen die als achtergrond (of voorgrond) worden gebruikt voor de ruimtes waarin het spel zich afspeelt. Vaak worden achtergrondafbeeldingen zo gemaakt dat zij een gebied bedekken zonder visuele problemen (oneffenheden). Je kunt op deze manier de achtergrond vullen met een bepaald patroon. Een aantal van deze bedekkende achtergronden kun je vinden op de *Game Maker*-website. Veel meer kunnen er gevonden worden op andere plaatsen op het web.

Om een achtergrond bij te voegen, gebruik je het item **Add Background** in het **Add**-menu of maak je gebruik van de corresponderende knop in de taakbalk. Het volgende scherm verschijnt:



Klik op de knop **Load Background** om een achtergrondplaatje te laden. *Game Maker* ondersteunt vele typen afbeeldingbestanden. Achtergrond afbeeldingen kunnen geen animated GIF's zijn! het keuzeveld **Transparent** geeft aan of de achtergrond gedeeltelijk transparant is of niet. De meeste achtergronden zijn niet transparant waardoor de default waarde 'not' is. De transparantiekleur wordt bepaald door de kleur van de pixel in de linker benedenhoek van de afbeelding.

Je kunt de achtergrond veranderen of een nieuwe creëren door gebruik te maken van de knop **Edit Background**.

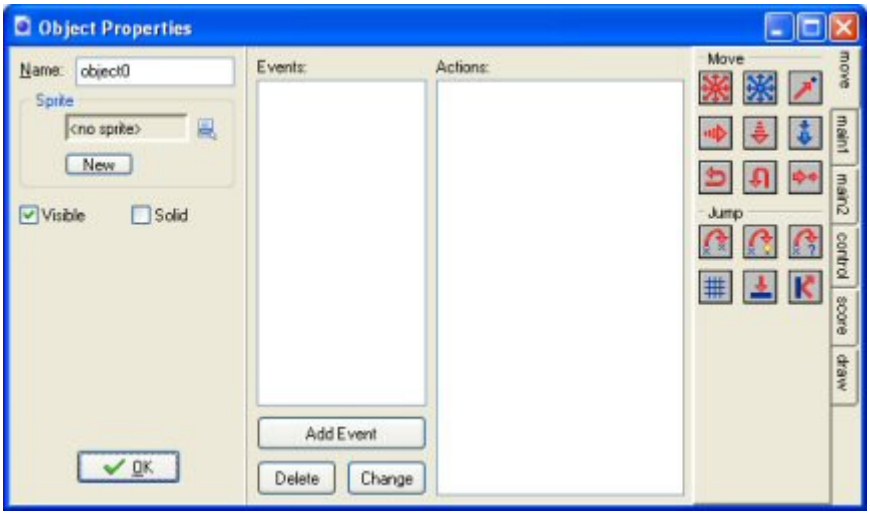
Het definiëren van objecten

We kunnen nu al leuke plaatjes en geluiden hebben toegevoegd aan ons spel, maar ze doen niets. We komen nu aan de meest belangrijke bron van *Game Maker*, de objecten. Objecten zijn entiteiten in het spel die iets doen. Meestal hebben zij een grafische representatie als sprite, zodat je ze kunt zien. Zij vertonen gedrag in de zin dat zij kunnen reageren op bepaalde gebeurtenissen. Alle dingen die je kunt zien in het spel (behalve de achtergrond) zijn objecten (of preciezer; zij zijn instanties van objecten). De figuren als monsters, ballen, muren enz. zijn allemaal objecten. Er kunnen ook objecten aanwezig zijn, die je niet ziet, maar die wel aspecten van het spel bepalen.

Het is belangrijk dat je het verschil tussen 'sprites' en 'objecten' realiseert. 'Sprites zijn alleen maar (geanimeerde) plaatjes die geen gedrag vertonen. Objecten hebben meestal een 'sprite' die hen representeert, maar het zijn de objecten die het gedrag vertonen. Zonder 'objecten' is er geen spel!

Het is ook belangrijk dat je het verschil tussen 'objecten' en 'instanties' begrijpt. Een object beschrijft een bepaalde entiteit; bijv. een monster. Er kunnen meerdere instanties van dit object in het spel bestaan. Wanneer we praten over een instantie bedoelen we één bepaalde instantie van het object. Wanneer we praten over het object, bedoelen we alle instanties van dit object.

Om een object toe te voegen aan het spel, kies je **Add Object** in het **Add**-menu. Het volgende scherm verschijnt:



Het is tamelijk ingewikkeld. In het linker deel van het scherm zie je algemene informatie over het object. In het midden deel zijn verschillende mogelijke events zichtbaar, die het object kunnen overkomen. In het volgende hoofdstuk wordt hier nader op ingegaan. Aan de rechterkant van het scherm zie je verschillende acties die het object kan ondernemen. Events en action worden beschreven in de volgende hoofdstukken.

Zoals altijd geef je het object een naam. Vervolgens wijs je een sprite toe aan het object. Hiertoe klik je met de linker muis knop op de sprite box of de menuknop ernaast. Een menu verschijnt met de mogelijke sprites. Selecteer degene die je wilt gebruiken voor het object. Als je nog geen sprite beschikbaar hebt, klik dan op de knop **New** om een sprite toe te voegen en eventueel aan te passen door middel van de knop **Edit**. Dit gaat een stuk sneller dan eerst een resource in de lijst opsporen en dan aangeven dat je hem wilt bewerken.

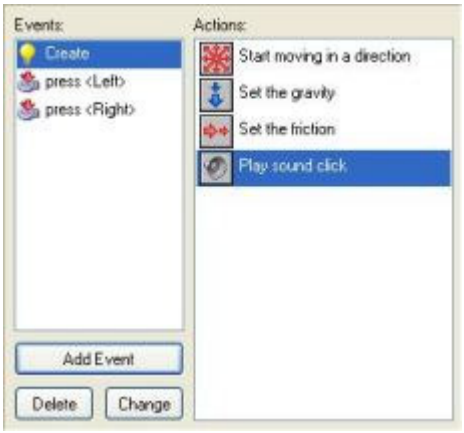
Onder de knop **New** vind je twee checkboxes. **Visible** geeft aan of instanties van objecten zichtbaar zijn. De meeste objecten zijn zichtbaar, maar soms is het handig om onzichtbare objecten te hebben. Bijvoorbeeld wanneer het richtingspunten zijn voor een bewegend monster. Onzichtbare objecten reageren op gebeurtenissen en andere instanties die met hen in botsing komen. De box die **Solid** heet, geeft aan dat het om ondoordringbare harde objecten gaat; bijv. een muur. Botsingen met solid objects worden anders afgehandeld dan botsingen met non-solid objects. Gebruik **Solid** alleen voor niet bewegende objecten.

Events (gebeurtenissen)

Game Maker gebruikt een gebeurtenis gedreven benadering (event driven approach). Dat betekent dat instanties van objecten gebeurtenissen ontvangen (een soort boodschappen dat er iets is

gebeurd). Ze kunnen hierop reageren door bepaalde acties uit te voeren. Voor ieder object moet je vastleggen op welke gebeurtenissen het moet reageren en welke acties het object dan moet ondernemen. Dit lijkt misschien gecompliceerd, maar is eigenlijk vrij gemakkelijk. Voor de meeste gebeurtenissen hoeft het object niets te doen. Voor de gebeurtenissen waar wel op gereageerd moet worden, kun je een 'drag and drop' benadering kiezen om de acties vorm te geven.

In het midden van het objecteigenschappen scherm zien je een lijst van events waarop het object moet reageren. In het begin is deze lijst leeg. Je kunt events toevoegen aan de lijst door op de knop **Add Event** te drukken. Dan verschijn er een venster met daarin de verschillende typen events Daarvan kun je er een selecteren die je wilt toevoegen. Soms verschijnt er een extra venster met daarin nog meer keuzes. Bijvoorbeeld, voor de keyboard event moet je key selecteren, dan verschijnt er onderin een lijst met allerlei mogelijkheden, met beschrijvingen, waaruit je een keuze kunt maken. Een event in de lijst wordt geselecteerd, die kan dan worden aangepast. Je kunt een event wijzigen door er op te gaan staan met de muis en er met de linkermuisknop op te klikken. Aan de rechterkant worden deze acties gerepresenteerd door kleine icoontjes. Zij worden gegroepeerd in een aantal door tabbladen aangegeven pagina's. In het volgende hoofdstuk wordt beschreven welke acties mogelijk zijn en wat zij inhouden. Tussen de gebeurtenissen en acties is een lijst. Deze lijst bevat de acties voor de geselecteerde gebeurtenis. Om acties toe te voegen aan de lijst, trek je hen met de muis van de rechterkant naar de lijst. Zij worden onder elkaar geplaatst, met een korte beschrijving erbij. Voor iedere actie word je gevraagd een aantal parameters mee te geven. Deze worden ook beschreven in het volgende hoofdstuk. Dus na een paar acties wordt de situatie bijv. als volgt:



Nu kun je beginnen om acties aan een ander event toe te voegen. Klik daarvoor op de juiste event met de linkermuisknop en sleep vervolgens de acties in de lijst.

Je kunt de volgorde waarin de acties in de lijst staan veranderen, door gebruik te maken van drag-and-drop. Wanneer je de <Alt> knop vasthoudt terwijl je sleept, maak je een kopie van de actie. Je kunt zelfs drag-and-drop gebruiken tussen actielijsten van de verschillende objecten. Wanneer

je klikt op een actie met de rechter muisknop, verschijnt een menu waarin je een actie kunt verwijderen (dit kan ook door de <Del> knop te gebruiken) maar ook kunt knippen en plakken. (je kunt meerdere acties voor knippen, kopiëren of verwijderen selecteren door de <Shift> toets of de <Ctrl> toets te gebruiken. Druk op <Ctrl><A> om alle acties te selecteren.) Als je je muis stil houdt boven een actie verschijnt er een langere beschrijving van die actie in beeld. Kijk in het volgende hoofdstuk om meer te weten te komen over acties. Er bestaat een grote collectie verschillende events. Zoals je kunt zien in het object properties scherm, hebben sommige menunamen een menu symbool naast zich. Dit betekent dat er een collectie van events is. Wanneer je klikt op het menu, of rechts klikt op de eventnaam, verschijnt een menu waarin je een event kunt kiezen die je wilt veranderen. Hier volgt een beschrijving van de verschillende events. (normaal gebruik je maar een paar van deze).

Om een geselecteerde event te verwijderen, samen met alle acties, moet je klikken op de knop met de naam **Delete**. (Events zonder enige actie erin worden automatisch gewist als je het venster sluit, dus die hoef je niet handmatig te wissen.) Als je acties aan een ander event wil toekennen (bijvoorbeeld omdat je besloten hebt een andere toets ervoor te kiezen), klik dan op de knop met de naam **Change** en kies voor de nieuwe event die je wilt gebruiken (Die event hoeft nog niet in de lijst voor te komen.) Het is ook mogelijk om een event, inclusief acties, te dupliceren door met je rechtermuisknop op de eventlijst te klikken en in het pop-up menu wat dan verschijnt te kiezen voor **Duplicate Event**

Zoals boven al is aangegeven kun je een event aan de lijst toevoegen door te klikken op de knop **Add Event**. Het volgende venster verschijnt dan in beeld:

Hier kies je een event die je wilt toevoegen aan de lijst. Soms verschijnt er een venster met extra keuzemogelijkheden. Hieronder volgt een beschrijving van de diverse events. (Houd er rekening mee dat je in de praktijk heel vaak slechts enkele ervan zult gebruiken.)

Create event

Deze event treedt op wanneer een instantie van een object wordt gecreëerd. Het wordt meestal gebruikt om een instantie in beweging te zetten en/of bepaalde variabelen in te stellen voor deze instantie.

Destroy event

Deze event treedt op wanneer de instantie wordt vernietigd. Eigenlijk treedt dit event op vlak voordat de instantie wordt vernietigd, zodat de instantie nog bestaat wanneer de event wordt uitgevoerd. Meestal wordt deze event niet gebruikt, behalve wanneer het de bedoeling is om de score te wijzigen, of om een ander object te creëren.

Alarm events

Iedere instantie heeft 8 alarm klokken. Je kunt deze alarmklokken inschakelen door bepaalde acties te gebruiken(zie het volgende hoofdstuk). De alarmklok tikt dan af tot 0 en op dat moment wordt de alarm event gegenereerd. Om de acties voor een bepaalde alarm klok aan te geven, moet het eerst geselecteerd worden in het menu. Alarm klokken zijn zeer nuttig. Je kunt hen gebruiken om bepaalde dingen te laten gebeuren in de tijd. Bijvoorbeeld om de bewegingsrichting van een monster na iedere 20 stappen te wijzigen. (In dit geval moet één actie in de event de klok opnieuw instellen).

Step events

De step event treedt op bij iedere stap in het spel. Hier kun je acties plaatsen die continu uitgevoerd moeten worden. Bijvoorbeeld; wanneer een object een ander object moet volgen, kun je hier de bewegingsrichting aanpassen zodat het tweede object gevolgd wordt. Met deze event moet je voorzichtig zijn. Geef niet veel van deze gecompliceerde acties in de step event van objecten mee, wanneer er veel instanties van dit object zijn. Dit vertraagt het spel. Eigenlijk zijn er drie verschillende step events. Meestal heb je alleen de default nodig. Wanneer je het menu gebruikt, kun je de begin en eind step event laten plaatsvinden. De normale step event wordt uitgevoerd vlak voordat de instanties in hun nieuwe posities worden geplaatst. De eind step event wordt uitgevoerd aan het einde van de step, vlak voor het tekenen. Dit wordt gebuikt om de sprite te laten veranderen afhankelijk van de richting.

Collision events

Wanneer twee instanties botsen (sprites overlappen) treedt een collision event op. Eigenlijk treden twee collision events op, voor iedere instantie één. De instantie kan op deze event reageren. Selecteer in het menu het object waarmee je een collision event wilt definiëren. Daarna worden in het collision event de acties geplaatst.

Er bestaat een verschil in wat er gebeurt, wanneer een instantie botst met een solide object of dat een instantie botst met een niet-solide object. Wanneer er geen acties in de collision event zijn opgenomen, gebeurt er niets. De huidige instantie blijft gewoon bewegen; zelfs wanneer het andere object een solide object is. Wanneer de collision event acties bevat, gebeurt het volgende:

Botsing met een solide object: de instantie wordt teruggeplaatst op zijn voorafgaande plaats (dus vóór de botsing plaatsvindt). De event wordt uitgevoerd. Uiteindelijk wordt de instantie naar zijn

nieuwe positie verplaatst. Dus wanneer de event de bewegingsrichting omkeert, botst de instantie op de muur zonder te stoppen. Als er nog steeds een botsing plaatsvindt, wordt de instantie op zijn vorige plaats gehouden. Op die manier houdt hij echt op te bewegen.

Wanneer het andere object een niet-solide object is, wordt de instantie niet teruggezet. De event wordt gewoon uitgevoerd, met de instantie op zijn huidige positie. Er wordt geen tweede check gedaan op botsingen. Als je erover nadenkt is dit logisch, wat er gebeurt. Omdat het object niet solide is, kunnen we er gewoon over heen bewegen. De event geeft aan dat dit gebeurt.

Er zijn veel mogelijkheden voor de collision event. Instanties kunnen het gebruiken om tegen muren te botsen. Je kunt het gebruiken om objecten te vernietigen wanneer zij geraakt worden door bijv. een kogel.

 **Keyboard events**

Wanneer de speler een toets indrukt, vindt een keyboard event plaats voor alle instanties van alle objecten. Er is een verschillende event voor iedere toets. In het menu kun je de toets kiezen, waarvoor je de keyboard event wilt definiëren en daar acties naar toe brengen. Slechts een paar objecten hebben events nodig voor slechts een paar toetsen. Er zijn twee verschillende manieren waarin je keyboard events kunt ontvangen. Zij zijn continue, d.w.z. zolang als de toets wordt ingedrukt vindt een keyboard event plaats in iedere stap. Wanneer zij niet continue zijn vindt er een slechts één keyboard event plaats, wanneer de toets wordt ingedrukt en opnieuw wanneer de toets weer wordt ingedrukt. Er zijn twee speciale keyboard events. Ten eerste: <No key>. Deze event treedt op in iedere stap wanneer geen toets wordt ingedrukt. De tweede heet <Any key> en treedt op wanneer elke willekeurig toets wordt ingedrukt. Wanneer de speler meerdere toetsen indrukt, treden voor alle ingedrukte toetsen events op. Let op: wanneer toetsen van het numerieke toetsenbord worden ingedrukt vinden alleen de corresponderende events plaats, wanneer ook <NumLock> is ingedrukt.

 **Mouse events**

Een muis event vindt plaats, wanneer de muiscursor ligt binnen het sprite gebied die de instantie representeert. Afhankelijk van welke muisknop wordt ingedrukt, krijg je de linkermuisknopevent, de rechtermuisknopevent, de geenmuisknopevent of de middelstemuisknopevent. Deze button-events worden in iedere step opnieuw uitgevoerd zolang de muisknop ingedrukt is. De press-events worden slechts eenmaal uitgevoerd als de muisknop wordt ingedrukt. De release-events worden uitgevoerd als een muisknop wordt losgelaten. Deze events vinden alleen plaats, zolang de muis boven de instantie is. Indien je op een willekeurige plaats in de room wilt reageren op de muis press-events of muis release-events moet je gebruikmaken van de global mouse events die je in het submenu kunt vinden. Er zijn twee speciale mouse events. de mouse enter event treedt op wanneer de muis een instantie binnengaat. De mouse leave event treedt op als de muis eeen

instantie uitgaat. Deze events kun je vooral toepassen om bijvoorbeeld van sprite te veranderen of een geluid af te spelen. Daarnaast zijn er ook nog een aantal events die betrekking hebben op de joystick. Je kunt acties aangeven voor de vier verschillende hoofdrichtingen van de joystick. (in diagonale richting treden beide events op.) Op deze manier kun je acties definiëren voor maximaal 8 joystick-knoppen. Dit kun je voor zowel de primaire als de secundaire joystick doen.

Other events

Er zijn een aantal andere events die bruikbaar kunnen zijn in bepaalde spellen. Zij staan hieronder genoemd:

- **Outside**: Deze event vindt plaats wanneer een instantie totaal buiten de kamer ligt. Dit is een goed moment om het te vernietigen.
- **Boundary**: Deze event vindt plaats wanneer de instantie de grens van de kamer snijdt.
- **Game start**: Deze gebeurtenis treedt op bij alle instanties in de eerste kamer, wanneer het spel start. Het treedt op voor de room start-event (zie hieronder) en vóór de creation events van de instanties in de kamer. Deze event wordt gedefinieerd in slechts één "controller"-object en wordt gebruikt om achtergrondmuziek te starten of om sommige variabelen te initialiseren of data te laden.
- **Game end**: Deze event vindt plaats voor alle instanties wanneer het spel eindigt. Opnieuw wordt dit event gedefinieerd door slechts één object. Het kan bijv. worden gebruikt om data in een bestand op te slaan.
- **Room start**: Deze event treedt op bij alle instanties die in de room aanwezig zijn, op het moment dat de room start. Het treedt op vóór de creatie events.
- **Room end**: Deze event treedt op bij alle bestaande instanties op het moment dat de room wordt beëindigd.
- **No more lives**: *Game Maker* heeft een built-in lives systeem. Er bestaat geen actie om het aantal levenden in te stellen en te veranderen. Wanneer het aantal levenden kleiner of gelijk aan 0 wordt, treedt deze event op. Het wordt gebruikt om het spel te beëindigen of opnieuw op te starten.
- **No more health**: *Game Maker* heeft een built-in health systeem. Er bestaat een action om de health in te stellen en te veranderen. Indien de waarde van de health kleiner of gelijk wordt dan 0, treedt deze event op. Het is typisch een event om het aantal levens te evrminderen of het spel opnieuw op te starten.
- **End of animation**: Zoals boven al aangegeven, bestaat een animatie uit een aantal plaatjes die achtereenvolgend zichtbaar worden. Nadat het laatste plaatje zichtbaar is geweest, begint de eerste weer. De event treedt precies op dat moment op. Dit kan worden gebruikt om een animatie te veranderen of om een instantie te vernietigen.
- **End of path**: Deze event treedt op als een instantie aan het eind van het ingestelde pad is gekomen.

- **User defined**: Er bestaan 8 van zulke events. Normaliter komen die nooit voor tezij je ze met behulp van een stuk GML-code oproept.

 **Drawing event**

Zichtbare instanties tekenen hun sprites in iedere stap van het scherm. Wanneer je acties in de drawing event specificeert, wordt de sprite niet getekend, maar worden de gespecificeerde acties uitgevoerd. Dit kan worden gebruikt om iets anders dan de sprite te tekenen of veranderingen aan te brengen in de sprite parameters. Er zijn een aantal drawing actions die speciaal bedoeld zijn voor de drawing event. De drawing event wordt niet alleen uitgevoerd wanneer het object zichtbaar is. Onafhankelijk van wat je hier tekent, zijn collision events gebaseerd op de sprite die is geassocieerd met de instantie.

 **Key press events**

Deze event lijkt op de keyboard-event maar hij treedt alleen op als de toets indrukt, maar dan wel maar 1 keer. Dit is erg handig als een actie slechts eenmaal moet worden uitgevoerd, bijvoorbeeld bij het afvuren van kogels uit een kanon.

 **Key release events**

Deze event lijkt op de keyboard-event maar hij treedt alleen op als de toets loslaat, maar dan wel maar 1 keer.

In bepaalde gevallen is het belangrijk dat je weet hoe *Game Maker* omgaat met de events in volgorde van uitvoering. Dat gebeurt als volgt:

- Begin step events
- Alarm events
- Keyboard, Key press, en Key release events
- Mouse events
- Normal step events
- (nu worden alle instanties op hun nieuwe posities geplaatst)
- Collision events
- End step events
- Drawing events

De creation, destroy en andere events worden uitgevoerd als de overeenkomstige zaken gebeuren.

Actions (acties)

In *Game Maker* vinden acties plaats. Acties worden geplaatst in events van objecten. Wanneer een event optreedt, worden de acties zichtbaar als gedrag van de objecten. Er is een groot aantal

verschillende acties beschikbaar en het is belangrijk dat je begrijpt wat zij doen. In dit hoofdstuk worden de acties beschreven die voorkomen in de simple mode. Houd er rekening mee dat sommige van deze acties alleen maar in de geregistreerde versie van *Game Maker* aanwezig zijn. Dat wordt bij de betreffende acties aangegeven.

Alle acties kun je vinden in de tabbladen aan de rechterzijde van het Object property venster. Er zijn zes sets met acties. Door op de correcte tab te klikken, vind je de set. Wanneer je de muis boven één van de acties houdt, zie je een korte omschrijving van de functie.

Een stukje herhaling:
Om een actie in een event te plaatsen moet je het van de tabbladen naar de actielijst slepen. Je kunt de volgorde in de lijst veranderen op dezelfde manier. Wanneer de <Alt> toets wordt vastgehouden tijdens het slepen, wordt een kopie gemaakt van de actie. Je kunt slepen en kopiëren tussen de lijsten in verschillende object property vensters. Gebruik de rechter muisknop om acties te verwijderen (of gebruik de <Del> toets) en voor kopieer en plak acties.

Wanneer je een actie instelt, verschijnt er meestal een pop-up scherm, waarin je bepaalde parameters voor die actie kunt invullen. Twee typen parameters verschijnen in veel acties. Bovenaan kun je aangeven naar welke instantie de actie verwijst. De default waarde is `self`; de instantie waarin de actie als eerste zichtbaar is gemaakt. Meestal is dit wat je wilt. Wanneer sprake is van een collision event, kun je ook specificeren naar welke andere instantie betrokken bij de botsing, de actie moet verwijzen. Op die manier kun je bijvoorbeeld andere instanties vernietigen. Je kunt ook kiezen om de actie te laten verwijzen naar alle instanties van een object. Op die manier kun je bijvoorbeeld alle rode ballen in blauwe ballen veranderen. De tweede wijze waarop parameters nodig zijn is de box met de naam **Relative**. In deze box geef je waarden mee die relatief zijn aan de huidige waarden. Op die manier kun je iets toevoegen aan de huidige score zonder de huidige score te moeten veranderen in een nieuwe score. De andere parameters komen hieronder aan bod. Je kunt altijd parameterwaarden veranderen door te dubbelklikken op een actie.

Move actions (bewegingsacties)

De eerste set van acties bestaat uit acties die met de beweging van objecten te maken hebben. De volgende acties bestaan:

 **Start moving in a direction**

Gebruik deze actie om de instantie in een bepaalde richting te laten bewegen. Je kunt de richting aangeven door gebruik te maken van de pijltjestoetsen. Gebruik de middelste knop om een beweging te stoppen. Je moet ook de snelheid van beweging specificeren. Deze snelheid is

gegeven in pixels per step. De default waarde is 8. Het verdient de voorkeur om geen negatieve waarden te gebruiken. Je kunt meerdere richtingen specificeren. In dit geval wordt random een keuze gemaakt. Op die manier kun je een monster bijv. zowel links als rechts laten bewegen.

 **Set direction and speed of motion**

Een tweede manier om een beweging te specificeren met de blauwe pijlen. Hier kun je een precieze beweging specificeren. Het betreft een hoek tussen 0 en 360 graden. 0 betekent naar rechts. De richting is tegen de wijzers van de klok in. Bijvoorbeeld; 90 betekent een opwaartse beweging. Wanneer je een willekeurige richting wilt, type je `random(360)`. Zoals je verderop zult zien, betekent `random` een random getal dat kleiner is dan de geïndiceerde waarde. Zoals je misschien hebt gezien bestaat er een checkbox met het label **Relative** draagt. Wanneer je deze aanvinkt, wordt de nieuwe beweging toegevoegd aan de voorafgaande. Bijv. wanneer een instantie naar boven beweegt en je voegt een beetje beweging naar links toe, dan wordt de nieuwe beweging naar linksboven.

 **Move towards a point**

Met deze actie kun je bewegingen op een andere manier specificeren. Je stelt een positie en een snelheid vast en de instantie begint in de richting van die positie en met die snelheid te bewegen. (Het stopt niet bij die positie!) Bijvoorbeeld: wanneer je een kogel in de richting van de positie van een ruimteschip wilt laten vliegen, kun je een positie `spaceship.x, spaceship.y` gebruiken. Wanneer je de **Relative** box aanvinkt, kun je de positie specificeren die relatief is aan de huidige positie van de instantie. (De snelheid wordt niet relatief berekend!)

 **Set the horizontal speed**

De snelheid van een instantie bestaat uit een horizontale en een verticale component. Met deze actie kun je de horizontale snelheid aanpassen. Als de horizontale snelheid positief is, betekent dat een beweging naar rechts, een negatieve horizontale snelheid betekent een snelheid naar links. Als deze snelheid wordt gewijzigd blijft de verticale snelheid gelijk. Maak gebruik van de optie relatief om de snelheid te verhogen of te verlagen.

 **Set the vertical speed**

Op een gelijke manier kun je met deze manier de verticale snelheid van een instantie wijzigen.

 **Set the gravity**

Met deze actie kun je graviditeit voor een bepaald object creëren Je specificeert een richting (hoek tussen 0 en 360 graden) en een snelheid en bij iedere stap wordt de grootte van de snelheid in een bepaalde richting, toegevoegd aan de huidige beweging van de objectinstantie. Meestal heb je een hele kleine snelheidsverhoging nodig (ca. 0.01) Je zult bij deze actie meestal een neerwaartse beweging willen hebben (270 graden). Wanneer je de **Relative** box aanvinkt, kun je de

neerwaartse beweging (graviditeit) en richting vergroten. In tegenstelling tot de echte wereld, kan graviditeit in alle richtingen optreden.

 **Reverse horizontal direction**

Met deze actie wordt de horizontale beweging van de instantie omgedraaid. Dit gebruik je bijv. wanneer een object tegen een verticale muur botst.

 **Reverse vertical direction**

Met deze actie wordt de verticale beweging van de instantie omgedraaid. Dit kan bijv. worden gebruikt wanneer het object tegen een horizontale muur botst.

 **Set the friction**

Frictions vertragen instanties wanneer zij bewegen. Je specificeert de grootte van de friction. Bij iedere stap wordt dit getal van de snelheid afgetrokken tot de snelheid uiteindelijk 0 wordt. Meestal gebruik je een heel klein getal (bijv. 0.01).

 **Jump to a given position**

Bij gebruik van deze actie kun je de instantie in een bepaalde positie brengen. Daartoe moet je de x- en y-coördinaten specificeren en vervolgens wordt de instantie neergezet op deze positie, vanuit zijn referentiepunt. Wanneer je **Relative** aanklikt, wordt de nieuwe positie relatief berekend t.o.v. de huidige positie van de instantie. Deze actie wordt vaak gebruikt om instanties continu te laten bewegen. In elke step verhogen we de positie een beetje

 **Jump to the start position**

Deze actie zet de instantie terug op de plaats waar hij is ontstaan.

 **Jump to a random position**

Deze actie beweegt de instantie naar een random positie in de room. Er worden alleen posities gekozen, waarbij de instantie nergens een solide instantie doorsnijdt. Je kunt deze specificeren. Wanneer je positieve waarden kiest, worden coördinaten gekozen die veelvouden van integers zijn van de geïndiceerde waarden. Dit kan gebruikt worden om de instantie uit te lijnen binnen de grenzen van je spel (als die er zijn). Je kunt horizontale en verticale waarden afzonderlijk invoeren.

 **Snap to grid**

Met deze actie kun je de actie van een instantie binnen een rooster houden. Je kunt horizontale en verticale waarden meegeven (grootte van de cellen op het rooster). Op deze manier zorgen je ervoor dat instanties binnen het rooster blijven.

 **Move to contact position**

Met deze actie kun je een instantie in een bepaalde richting laten bewegen totdat een positie is bereikt dat deze instantie een object heeft bereikt. Indien er een botsing optreedt op die positie blijft de instantie staan. Aan de andere kant blijft de instantie precies staan op de plek vlak voordat een botsing zou optreden. Je kunt met deze actie de precieze richting maar ook de maximale afstand van de beweging aangeven. Bijvoorbeeld, als een instantie naar beneden valt, kun je een maximale afstand instellen waarbinnen de instantie een object tegenkomt en kan botsen. Daarbij is het ook mogelijk om aan te geven of het alleen maar gaat om solid objects of alle typen objecten. Deze actie gebruik je dus als je er zeker van wil zijn dat de instantie stopt met bewegen als het in contact komt met een ander object dat bij de botsing betrokken is.

 **Bounce against objects**

Wanneer deze actie wordt geplaatst in de collision event van een bepaald object, zal het object op een natuurlijke manier terugbotsen vanaf een muur. Wanneer je de parameter 'precise' op false zet, worden horizontale en verticale muren correct behandeld. Wanneer je deze parameter op true zet, worden ook scheve en gebogen muren correct behandeld. Echter deze gaan langzamer. Je kunt ook aangeven of je wilt laten stuiten van solide objecten of van alle objecten. Het stuiten kan nooit volledig correct zijn, omdat dit van vele eigenschappen afhankelijk is. In de meeste situaties is het effect echter goed genoeg.

Main actions, set 1

De volgende groep acties gaat over het creëren, veranderen en vernietigen van instanties van de objecten, over geluiden en rooms.

 **Create an instance of an object**

Met deze actie kun je een instantie van een object creëren Je specificeert het object dat je wilt creëren en de positie van de nieuwe instantie. Wanneer je de **Relative** box aanvinkt, wordt de positie berekend relatief aan de positie van de huidige instantie. Het creëren van instanties tijdens het spel is zeer bruikbaar. Een ruimteschip kan kogels creëren, een bom kan een explosie creëren In veel spelletjes is een controller element aanwezig die van tijd tot tijd monsters of andere objecten creëren. Voor het creëren van nieuwe instanties wordt de creation event uitgevoerd.

 **Create an instance of an object with a speed and direction**

Deze actie werkt eigenlijk op dezelfde manier als de actie hierboven, maar heeft twee extra velden. Je kunt meteen ook bij het creëren van een nieuwe instantie de snelheid en de richring waarin de instantie moet gaan bewegen aangeven. Houd er wel rekening mee dat als je de box **Relative** aanvinkt, alleen maar de positie van de instantie relatief is en niet de snelheid en de richting van de beweging. Bijvoorbeeld, als je een kogel op een persoon wilt afschieten moet je gebruik maken

van een truc. Als positie kies je (0,0) en je vinkt de **Relative** box aan. Als richting kies je de huidige richting van de instantie. Dit kun je bewerkstelligen door het woord `direction` in te typen. (In feite is dit een variabele die de huidige bewegingsrichting aangeeft als een instantie beweegt.)

 **Change the instance**

Met deze actie kun je de huidige instantie in een ander object laten veranderen. Bijvoorbeeld: je kunt een bom in een explosie laten veranderen. Alle 'settings' zoals beweging of waarden van variabelen zullen hetzelfde blijven. Je kunt aangeven of de destroy event voor het huidige object en de creation event voor het nieuwe object moet gelden.

 **Destroy the instance**

Met deze actie wordt de huidige instantie vernietigd. De destroy event wordt voor deze instantie uitgevoerd.

 **Destroy instances at a position**

Met deze actie vernietig je alle instanties die in hun box begrenzing een bepaalde positie bevatten. Dit kan belangrijk zijn wanneer je een bom wilt laten exploderen. Wanneer je de **Relative** box aanvinkt, wordt de positie relatief berekend t.o.v. de positie van de huidige instantie.

 **Change the sprite**

Gebruik deze actie om de sprite voor de instantie te veranderen. Je geeft aan welke nieuwe sprite moet worden getoond. Je kunt ook subimages weergeven. Normaal gesproken gebruik je 0 om de eerste subimage te laten zien tenzij je een speciaal plaatje wilt laten zien. Bij het gebruik van -1 wil je dat er geen ander subimage wordt getoond. Tenslotte verander je de snelheid waarme de subimages in de animatie moeten worden vertoond. Als je slechts een subimage wilt zien, dus geen animatie, dan zet je de snelheid op 0. Als de snelheid >1 zullen de subimages worden overgeslagen, indien de snelheid <1 dan zullen de subimages meerdere malen worden getoond. Gebruik geen negatieve snelheid. Het veranderen van sprites is een belangrijk kenmerk. Bijv. wanneer je een sprite van figuur wilt laten veranderen, afhankelijk van de richting in welke het loopt. Dit kan worden bereikt wanneer je verschillende sprites maakt voor iedere vier richtingen. Binnen de keyboard events voor de pijltjestoetsen wordt de bewegingsrichting van de sprite gespecificeerd.

 **Transform the sprite**

Gebruik deze actie om de grootte en de orientatie van de sprite voor een instantie te wijzigen. Gebruik de scale factors om het groter of kleiner te maken. Het instellen van de hoek van de sprite gaat tegen de wijzers van de klok in, dus de sprite draait naar links. Bijvoorbeeld, om de sprite in de bewegingsrichting af te beelden moet je de waarde `direction` ingeven. Dat is onder andere

belangrijk voor een auto. Je kunt er ook voor kiezen of een sprite horizontaal gespeigeld moet worden afgebeeld verticaal onderste boven. ***Deze actie is alleen maar beschikbaar in de geregistreerde versie.***

 **Set sprite blending**

Normaliter wordt een sprite zo weergegeven als hij is ontworpen. Door het toepassen van deze actie kun je de kleur van de sprite veranderen. De kleur wordt gemengd met de kleuren van de sprite. Als je wilt dat de sprite meerdere kleuren kan aannemen, kun je de sprite het beste in zwart-wit ontwerpen en vervolgens de mengkleur gebruiken om de werkelijke spritekleur weer te geven. Bovendien is het ook mogelijk om een alfa tranparantie aan te geven. Met een waarde 1 is de sprite ondoorschijnend (opaque) en als de waarde 0 is, dan is de sprite volledig doorschijnend. Met een waarde tussen 0 en 1 zal je meer of minder de achtergrondkleur door de sprite heen kunnen zien. Vooral voor het maken van explosies is dit een geweldige actie. ***Deze actie is alleen maar beschikbaar in de geregistreerde versie.***

 **Play a sound**

Met deze actie speel je een geluidsbron af die je aan het spel hebt toegevoegd. Je kunt aangeven welke muziek je wilt laten spelen en of het eenmaal speelt (default) of dat het steeds weer opnieuw begint. Meerdere wave geluiden kunnen tegelijkertijd worden afgespeeld, maar er kan slechts één midi-file tegelijkertijd worden afgespeeld. Wanneer je een midi bestand start wordt het vorige midibestand gestopt.

 **Stop a sound**

Deze actie stopt het geïndiceerde geluid. Wanneer meerdere instanties van dit lied aan het spelen zijn, worden deze allemaal gestopt.

 **If a sound is playing**

Deze actie test of het aangegeven geluid wordt afgespeeld. als dat het geval is wordt de volgende actie uitgevoerd. Zo niet, wordt de actie overgeslagen. Je kunt **Not** kiezen om ervoor te zorgen dat de volgende actie toch wordt uitgevoerd, ook als het aangegeven geluid niet wordt afgespeeld. Bijvoorbeeld, je kunt controleren of een bepaalde achtergrondmuziek wordt afgespeeld en indien dat niet het geval is kun je ervoor kiezen om een bepaalde achtergrondmuziek te starten.

 **Go to previous room**

Beweeg naar de voorgaande kamer. Je kunt het soort overgangseffect tussen de kamers aangeven. Experimenteer om te zien, wat het beste werkt. Wanneer je in de eerste kamer bent, krijg je een foutmelding.

 **Go to next room**

Beweeg naar de volgende kamer. Je kunt de overgang aangeven.

 **Restart the current room**

De huidige kamer wordt opnieuw opgestart. Je kunt de overgang aangeven.

 **Go to a different room**

Met deze actie kun je naar een bepaalde kamer gaan. Je geeft aan naar welke kamer en welk overgangseffect.

 **If previous room exists**

Deze actie controleert of een vorige room bestaat. Als dat het geval is wordt de volgende actie uitgevoerd. Het is gebruikelijk om deze test te gebruiken voordat je naar de vorige room gaat.

 **If next room exists**

Deze actie controleert of een volgende room bestaat. Als dat het geval is wordt de volgende actie uitgevoerd. Het is gebruikelijk om deze test te gebruiken voordat je naar de volgende room gaat.

Main actions, set 2

Hieronder staan enkele acties die te maken hebben met timing, het geven van berichten aan de gebruiker, en het spel als geheel.

 **Set an alarm clock**

Met deze actie kun je één van de acht alarmklokken voor een instantie instellen. Je kunt het aantal stappen en de alarmklok instellen. Na het ingestelde aantal stappen, ontvangt de instantie de corresponderende alarm event. Je kunt de waarde verhogen of verlagen door de **Relative** box aan te vinken. Wanneer je de alarmklok instelt op een waarde minder dan of gelijk aan 0, wordt de event niet gegenereerd.

 **Sleep for a while**

Met deze actie kun je de scène een tijdje bevriezen. Dit wordt vaak gebruikt aan het begin of eind van een level, wanneer je de speler een boodschap wilt meegeven. Je specificeert het aantal milliseconden dat het spel wordt stilgelegd. Je moet ook aangeven of het scherm opnieuw getekend moet worden om de meest recente situatie weer te geven.

 **Display a message**

Met deze actie kun je een boodschap meegeven in een dialog box. Simpelweg wordt de boodschap ingetypt. Wanneer je een #-symbool in de tekst gebruikt, wordt dit geïnterpreteerd als het begin

van een nieuwe regel. Wanneer de boodschap tekst start met een quote of dubbele quote, wordt het geïnterpreteerd als een expressie. Verderop volgt meer informatie over expressies.

 **Show the game information**

Met deze actie toon je een venster met de informatie over het spel.

 **Restart the game**

Met deze actie start je het spel weer op vanaf het begin.

 **End the game**

Met deze actie wordt het spel beëindigd.

 **Save the game**

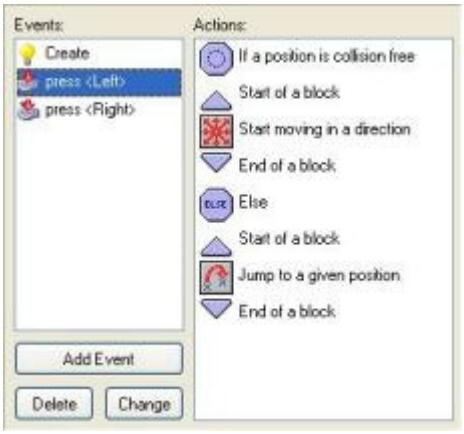
Met deze actie kun je de huidige spel status opslaan. Je specificeert de bestandsnaam (het bestand wordt gecreëerd in de directory van het spel). Het spel kan later weer opgestart worden met de volgende actie. (Houd er echter wel rekening mee dat allen de basale elementen van het spel worden opgeslagen. Dingen die niet worden opgeslagen zijn: geluiden die worden afgespeeld, en bijzondere zaken zoals inhouden van datastructuren, particles, enz.)

 **Load the game**

Laad het spel in vanuit het bestand. Specificeer de bestandsnaam. Zorg ervoor dat het opgeslagen bestand over hetzelfde spel gaat en gemaakt is in dezelfde versie van *Game Maker*, zo niet dan volgt een foutmelding. (Om precies te zijn: het spel wordt geladen aan het einde van de huidige stap. Sommige acties na deze actie vinden niet in het geladen spel plaats maar worden nog uitgevoerd in het huidige spel.)

Control actions

Een zijn aantal acties waarmee je kunt controleren of andere acties worden uitgevoerd. De meeste van deze acties stellen een vraag, bijvoorbeeld of een positie leeg is. Wanneer het antwoord ja (true) is, wordt de volgende actie uitgevoerd; zo niet dan wordt deze overgeslagen. Wanneer je meerdere acties uitgevoerd wilt zien, dan wel overgeslagen afhankelijk van de uitkomst, dan kun je deze acties bij elkaar stoppen in een block door start block en end block acties rondom hen te zetten. Er kan ook een ander deel code zijn dat wordt uitgevoerd als een antwoord nee (false) is. Dus een vraag kan er als volgt uitzien:



In dit voorbeeld wordt de vraag gesteld of een positie voor de huidige instantie botsvrij is. Zo ja, dan start de instantie in een bepaalde richting. Zo niet, dan springt de instantie in een gegeven positie.

Voor alle vragen is er een veld, gelabeld **NOT**. Wanneer je dit veld aanvinkt, wordt het resultaat van de vraag omgedraaid. Dus wanneer het resultaat true was, wordt dit false en omgekeerd. Dit zorgt ervoor bepaalde acties te laten plaatsvinden, wanneer een vraag niet juist is.

Voor vele vragen kun je instellen dat zij worden toegepast op alle instanties van een bepaald object. In dit geval is het resultaat alleen maar waar als het geldt voor alle instanties van het object. Bijvoorbeeld, je kunt controleren of voor alle ballen de positie iets meer naar rechts collision vrij is.

De volgende questions en gerelateerde acties zijn beschikbaar: (Let erop dat zij allemaal een verschillend gevormd icoon hebben en een verschillende achtergrondkleur, waardoor zij gemakkelijker van andere acties kunnen worden onderscheiden.)

If a position is collision free

Deze vraag wordt met true beantwoord, wanneer de huidige instantie, geplaatst op de geïndiceerde positie geen collision genereert met een object. Je kunt de positie zowel absoluut als relatief genereren. Je kunt ook aangeven of dit alles alleen geldt voor solide objecten of voor alle objecten. Deze actie wordt veelvuldig gebruikt om te controleren of een instantie naar een bepaalde positie kan verplaatsen.

If there is a collision at a position

Dit is het omgekeerde van de voorgaande actie. Het geeft de waarde true terug, wanneer er een botsing ontstaat als het huidige object naar een bepaalde gegeven positie wordt verplaatst (opnieuw of voor solide objecten of voor alle objecten)

 **If there is an object at a position**

Deze vraag wordt met true beantwoord, wanneer een instantie die geplaatst is op de aangewezen positie een instantie van het aangewezen object ontmoet.

 **If the number of instances is a value**

Je specificeert een object en een aantal. Wanneer het huidige aantal instanties van een object gelijk is aan dit aantal, levert de vraag de waarde true op. In het andere geval 'false'. Je kunt ook laten checken of het aantal instanties kleiner is dan een gegeven waarde of hoger. Dit wordt bijv. gebruikt om te laten checken of alle instanties van een bepaald type zijn verdwenen of niet. Meestal gebeurt dit op het moment dat een level van een spel eindigt.

 **With a change perform next action**

You specify the number of sides of a dice which is then thrown. Then if the dice lands on one, the result is true and the next action is performed. This can be used to put an element of randomness in your game. For example, in each step you can generate with a particular chance a bomb or a change of direction. The larger the number of sides of the dice, the smaller the chance. You can actually use real numbers. For example if you set the number of sides to 1.5 the next action is performed two out of three times. Using a number smaller than 1 makes no sense.

 **If the user answers yes to a question**

e specificeert een vraag. Een dialog wordt getoond aan de speler met een jaknop en een neeknop. Het resultaat is 'true' wanneer de speler ja antwoordt.

 **If an expression is true**

Dit is de meest algemene vraag actie. Je kunt een willekeurige expressie meegeven. Wanneer de expressie evalueert naar true (een getal groter of gelijk aan 0.5) wordt de volgende actie uitgevoerd. Hieronder vind je meer informatie hierover.

 **If a mouse button is pressed**

Geeft de waarde 'true' terug, wanneer de aangegeven muisknop wordt ingedrukt. Een standaard gebruik is de step event. Je kunt checken of een muisknop is ingedrukt en wanneer dat het geval is, bijv. naar die positie gaan (gebruik de jump to point actie met waarden `mouse_x` en `mouse_y`)

 **If instance is aligned with grid**

Geeft de waarde 'true' terug wanneer de positie van de instantie op het rooster ligt. Je specificeert de horizontale en verticale ligging op het rooster. Dit is goed te gebruiken wanneer bepaalde acties, bijv. een draai maken, alleen zijn toegestaan, wanneer de instantie op het rooster ligt.

 **Start of block**

Geeft de start van een blok van acties aan.

 **End of block**

Geeft het einde van een blok van acties aan.

 **Else**

Achter het actiedeel volgt het else gedeelte, dat wordt uitgevoerd wanneer het resultaat van de vraag false is.

 **Repeat next action**

Deze actie wordt gebruikt om de volgende actie te een aantal malen te herhalen (of een blok van acties) . Je kunt simpelweg het aantal keren meegeven.

 **Exit the current event**

Bij deze actie wordt geen enkele verdere actie in deze event meer uitgevoerd. Dit wordt vaak gebruikt na een vraag. Bijvoorbeeld, wanneer een positie vrij is hoeft er niets gedaan te worden en beëindigen we de event. In dit voorbeeld worden de volgende acties alleen uitgevoerd, wanneer er een botsing optreedt.

Indien je meer controle wilt uitoefenen wat er allemaal in je spel gebeurt, kun je gebruik maken van de ingebouwde programmeertaal GML die beschreven wordt in deel 4 van deze documentetaie. Het geeft je meer flexibiliteit dan het gebruik van acties. Het gebruik van eigen variabelen kan ook van groot belang zijn bij het maken van een spel. De volgende acties hebben daar betrekking op.

 **Execute a piece of code**

Wanneer deze actie optreedt, wordt een formulier zichtbaar, waarin code kan worden getypt. Dit kunnen simpele functies zijn of meer complexe code. Gebruik deze code actie alleen voor kleine stukjes code. Voor grotere stukken code kun je beter gebruik maken van scripts waarover in deel 2 van de documentatie veel meer staat.

 **Comment**

Gebruik deze actie om een commentaarregel toe te voegen aan de actielijst. De zin wordt getoond in italic font. Het doet niets wanneer de event wordt uitgevoerd. Commentaar toevoegen helpt je te herinneren wat je events werkelijk doen. Realiseer je dat het gaat om een actie. Dus als het geplaatst is in een blok code warvoor geldt dat de uitkomst true is, dan zal deze actie worden uitgevoerd, ook al zie je er niets van en doet het niets.

 **Set the value of a variable**

Er zijn veel ingebouwde variabelen in het spel. Met deze actie kun je deze ingebouwde variabelen veranderen. Je kunt ook eigen variabelen creëren en hen waarden toekennen. Je specificeert de naam van de variabele en de nieuwe waarde ervan. Wanneer je de **Relative** checkbox aanvinkt, wordt de waarde toegekend aan de huidige waarde van de variabele. Denk eraan dat dit alleen mogelijk is wanneer de variabele al een toegekende waarde heeft. Hieronder volgt meer informatie.

 **If a variable has a value**

Met deze actie kun je controleren wat de waarde van een bepaalde variabele is. Als de waarde gelijk is aan de opgegeven waarde zal de vraag true als antwoord geven, in in het andere geval false. Je kunt ook controleren of een waarde groter of kleiner is dan een opgegeven waarde. Hieronder volgt nog meer informatie over variabelen. Je kunt overigens deze actie ook gebruiken om 2 expressie met elkaar te vergelijken.

 **Draw the value of a variable**

Met deze actie kun je de waarde van een variabele op een door jou gewenste positie op het scherm weergeven. Houd er wel rekening mee dat deze actie alleen maar gebruikt kan worden in de draw event van een object.

Score actions

In de meeste spellen wordt gewerkt met scores voor de speler. Bovendien wordt er meestal ook een aantal levens aan de speler toebedeeld. En tenslotte heeft de speler een bepaalde mate van levenskracht (health). De volgende acties maken het mogelijk om te gaan met de score, de levens en de levenskracht van de speler.

 **Set the score**

Game Maker heeft een ingebouwd mechanisme om te score te registreren en weer te geven. De score staat meestal in een apart deel van het speelvenster afgebeeld. Deze actie kun je gebruiken om de score te wijzigen. Je geeft dan gewoon een nieuwe waarde voor de score op. Meestal moet de score worden opgehoogd. Zet in dat geval een vinkje in de box **Relative**.

 **If score has a value**

Met deze vraagactie kun je controleren of de score reeds een bepaalde waarde heeft bereikt. Je kunt die waarde aangeven en tevens bepalen of de waarde groter of kleiner dan of gelijk aan die waarde moet zijn om die actie uit te voeren.

 **Draw the value of score**

Met deze actie kun je de waarde van de score op een bepaalde plaats op het scherm tekenen. Je bepaalt de precieze positie en de titel die boven de score moet komen te staan. De score wordt in een standaard font getekend. Deze actie kan alleen maar uitgevoerd worden in de drawing event van het object.

 **Display the highscore table**

Voor elk spel wordt een toptien van highscores bijgehouden. Deze actie toont de lijst met highscores. Als de score die door een speler is behaald tussen de tien beste waarden inzit, wordt die score op de juiste plaats in die lijst weergegeven en kan de speler zijn naam aan die score toevoegen. Je kunt al dan niet een achtergrondafbeelding toevoegen, wel of geen kader gebruiken, welke kleur de nieuwe en bestaande scores hebben en welk font je daarvoor wilt gebruiken.

 **Clear the highscore table**

Deze actie verwijdert alle scores uit de highscoretabel

 **Set the number of lives**

Game Maker bezit ook een ingebouwd systeem voor het instellen en bijhouden van het aantal levens tijdens een spel. Met deze actie kun je het aantal levens wijzingen (meestal is dat met 1 verlagen). Normaliter start je met 3 levens aan het begin van een spel en verlaag of verhoog je dat aantal afhankelijk van wat er tijdens het spel gebeurt. Vergeet niet een vinkje te plaatsen in de box **Relative** als je het aantal levens wilt verhogen of verlagen. Op het moment dat het aantal levens 0 wordt (of kleiner dan 0) wordt de "no more lives" event gestart.

 **If lives is a value**

Met deze vraagactie kun je controleren of het aantal levens een bepaalde waarde heeft bereikt. Je kunt zelf aangeven welk aantal dat moet zijn en of de waarde groter of kleiner dan of gelijk aan die opgegeven waarde dient te zijn.

 **Draw the number of lives**

Met deze actie kun je de waarde van het aantal levens op een bepaalde plaats op het scherm tekenen. Je bepaalt de precieze positie en de titel die boven het aantal levens moet komen te staan. De aantal levens wordt in een standaard font getekend. Deze actie kan alleen maar uitgevoerd worden in de drawing event van het object.

 **Draw the lives as image**

In plaats van het weergeven van het aantal levens als een cijfer is het vaak aardiger om daarvoor een aantal kleine plaatjes te gebruiken. Deze actie is daar precies voor bedoeld. Je bepaalt de

precieze positie en het plaatje waardoor het aantal levens precies op die plaats als afbeeldingen wordt getekend. Deze actie kan alleen maar uitgevoerd worden in de drawing event van het object.

 **Set the health**

Game Maker bezit een ingebouwd systeem voor het bijhouden van de levenskracht van een speler. Je kunt deze actie gebruiken om de waarde ervan te veranderen. Als beginwaarde wordt meestal de waarde 100 gebruikt, hetgeen staat voor maximale levenskracht. Als de waarde 0 bereikt is, dan is alle levenskracht verdwenen. Je kunt hier eenvoudig een nieuwe waarde voor de levenskracht opgeven. Vaak wil je ervoor zorgen dat de waarde van de levenskracht afneemt, soms ook toeneemt. Zorg er dan voor dat je een vinkje plaatst in de box **Relative**. Als de waarde van de levenskracht kleiner of gelijk aan 0 is geworden, wordt een out of health event gestart.

 **If health is a value**

Met deze vraagactie kun je nagaan of de levenskracht een bepaalde waarde heeft bereikt. Je kunt zelf de waarde aangeven en of de levenskracht groter of kleiner dan of gelijk aan die opgegeven waarde dient te zijn.

 **Draw the health bar**

Met deze actie kun je de levenskracht in de vorm van een balk tekenen. Bij de waarde 100 is de balk volledig ingekleurd, als de waarde 0 is, is de balk leeg. Je kunt de positie en grootte van die balk op het scherm bepalen en de kleur en achtergrond van de balk.

 **Set the window caption information**

Normaliter worden in de titelbalk de naam van de room en de score weergegeven. Met deze actie kun je dat veranderen. Je kunt hier aangeven of je de score, het aantal levens en de levenskracht wel of niet wilt laten zien en wat de titel voor elk van deze onderdelen moet zijn.

Drawing actions (tekenacties)

Normaal wordt in elke step van het spel voor elke instantie de sprite opnieuw getekend in de room. Je kunt dit wijzigen door door acties te laten uitvoeren in de drawing event. (Let er wel op dat deze acties alleen worden uitgevoerd als een instantie zichtbaar (visible) is!) De volgende acties zijn beschikbaar. Deze acties hebben alleen zin in de drawing event. Op andere plaatsen worden ze genegeerd.

 **Draw a sprite image**

Je specificeert een sprite door de positie (absoluut of relatief in verhouding tot de huidige instantie) en het plaatje (subplaatje) van de sprite (subplaatjes worden genummerd vanaf 0 en hoger). Wanneer je de volgende subplaatje wilt tekenen gebruik je nummer -1.

 **Draw a background image**

Je geeft het achtergrond figuur mee door de positie te bepalen (absoluut of relatief) en of het figuur verspreid moet worden over de hele kamer of slechts gedeeltelijk.

 **Draw a text**

Je specificeert de tekst en de positie. Het symbool # in de tekst wordt geïnterpreteerd dat je een nieuwe regel begint (gebruik \# om het symbool # zelf te krijgen.) Op die manier kun je meerregelige teksten maken. Wanneer de tekst start met een punt of dubbele punt, wordt het geïnterpreteerd als expressie. Bijvoorbeeld, je gebruikt

```
'X: ' + string(x)
```

om de waarde van de x-coördinaat van een instantie weer te geven. (De variabele x bevat de huidige x-coördinaat. De functie `string()` geeft de waarde weer als een string. De operator + combineert de twee strings.)

 **Draw a text transformed**

Deze actie lijkt veel op de vorige actie maar nu kun je ook nog de tekstgrootte variëren door horizontale en verticale schaalwaardes op te geven en bovendien kun je de text roteren door een hoekgrootte op te geven. ***Deze actie is alleen maar beschikbaar in de geregistreerde versie.***

 **Draw a rectangle**

Je specificeert de coördinaten van de twee tegenovergestelde hoeken van de rechthoek; absoluut of relatief t.o.v. de huidige instantie positie.

 **Draw a horizontal gradient**

Deze actie tekent ook een rechthoek maar nu kun je ook een kleurgradiënt van links naar rechts instellen. Je bepaalt de grootte van de rechthoek en tevens de 2 te gebruiken kleuren. ***Deze actie is alleen maar beschikbaar in de geregistreerde versie.***

 **Draw a vertical gradient**

Deze actie tekent ook een rechthoek maar nu kun je ook een kleurgradiënt van boven naar beneden instellen. Je bepaalt de grootte van de rechthoek en tevens de 2 te gebruiken kleuren. ***Deze actie is alleen maar beschikbaar in de geregistreerde versie.***

 **Draw an ellipse**

Je specificeert de coördinaten van de twee tegenovergestelde hoeken van de omliggende rechthoek; absoluut of relatief t.o.v. de huidige instantie positie.

 **Draw a gradient ellipse**

Ook hier wordt een ellips getekend, maar nu kun je een kleurgradiënt instellen van binnenuit naar buiten. ***Deze actie is alleen maar beschikbaar in de geregistreerde versie.***

 **Draw a line**

Je specificeert de coördinaten van de twee eindpunten van de lijn; absoluut of relatief t.o.v. de huidige instantie positie.

 **Draw an arrow**

Deze actie tekent een pijl. Jij geeft de coördinaten aan van de twee eindpunten van de lijn en de grootte van de pijlpunt.

 **Set the colors**

Deze actie stelt je in staat om de kleuren van te tekenen vormen, lijnen en tekst in te stellen. (dit heeft geen invloed op de mnanier waarop sprites en achtergronden worden getekend.)

 **Change fullscreen mode**

Met deze actie kun je kiezen om het spel te spelen op een volledig scherm of in een window met beperkte afmetingen.

 **Take a snapshot image of the game**

Met deze actie kun je een snapshot maken van je spel en opslaan als een .bmp-bestand. Je moet dan alleen nog een naam geven aan dat bestand. ***Deze actie is alleen maar beschikbaar in de geregistreerde versie.***

Het gebruik van expressies en variabelen

In veel acties is het nodig om waarden aan parameters mee te geven. In plaats van alleen een getal mee te geven, kun je ook een formule intypen, bijv. 32*12. Je kunt ook veel gecompliceerdere expressies meegeven. Bijv. wanneer je de horizontale snelheid wilt verdubbelen, kun je dit aangeven door `2*hspeed`. De variabele `hspeed` geeft de huidige horizontale snelheid weer van de instantie. Er zijn een groot aantal variabelen die je kunt gebruiken. Een aantal van de meest belangrijke zijn:

x de x-coördinaat van de instantie

y de y-coördinaat van de instantie

hspeed de horizontale snelheid (pixels per step)

vspeed de verticale snelheid (pixels per step)

direction de huidige bewegingsrichting in graden (0-360)

speed de huidige snelheid in deze richting

visible of een object zichtbaar(1) is of niet(0)
image_index Deze variabele geeft aan welk subplaatje van de huidige sprite op dit moment wordt getoond. Als je het verandert en de snelheid op 0 zet (zie onder) kun je een vast subplaatje in beeld brengen.
image_speed Deze variabele geeft de snelheid aan waarmee de subplaatjes worden getoond. De standaard waarde is 1. Als je deze waarde groter dan 1 maakt zullen een aantal plaatjes niet getoond worden waardoor de animatie sneller verloopt. Als je de waarde kleiner maakt dan 1 zal de animatie langzamer verlopen doordat subplaatjes meerdere malen worden getoond.
score de huidige waarde van de score
lives het huidige aantal levens
health de huidige levenskracht (0-100)
mouse_x x-positie van de muis
mouse_y y-positie van de muis

Je kunt de meeste van deze variabelen wijzigen door de set variabele actie. Je kunt ook je eigen variabelen definiëren door hen een waarde toe te kennen, gebruik echter geen relatieve omdat ze nog niet bestaan. Je kunt deze variabelen vervolgens gebruiken in expressies. De variabelen die je op deze manier creëert zijn lokale variabelen voor de huidige instantie. Ieder object heeft een eigen kopie ervan. Om een globale variabele te creëren moet je het woord global en een punt ervoor zetten.

Je kunt refereren aan de waarden van de variabelen voor andere objecten door de objectnaam en een punt ervoor te zetten. Bijv. wanneer je een bal wilt laten bewegen naar de plaats waar de munt is, kun je de positie als volgt aangeven (`coin.x` , `coin.y`). In het geval van een collision event, kun je refereren aan de x-coördinaat van de andere objecten door `other.x` te schrijven. In conditionele expressie kun je gebruik maken van vergelijkingen als < kleiner dan), >, enz.

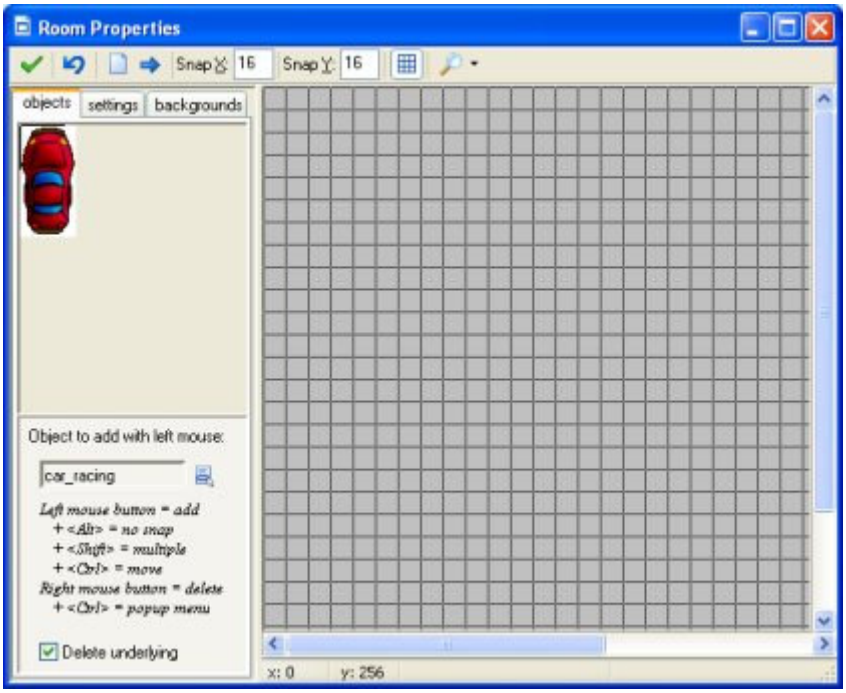
In je expressies kun je ook gebruik maken van functies. Bijv. de functie `random(10)` geeft een random reëel getal onder de 10 aan. Je kunt zo bijvoorbeeld de snelheid van de bewegingsrichting instellen op een random waarde. Veel meer functies bestaan. Die worden beschreven in deel 4 van de documentatie.

Het maken van rooms

Nu je voorwerpen met hun gedrag in de vorm van gebeurtenissen en acties hebt bepaald, is het tijd om de rooms of de levels te creëren waarin het spel plaatsvindt. Elk spel heeft minstens één room. In deze rooms plaatsen wij instanties van de objecten. Zodra het spel begint wordt de eerste room getoond en de instanties daarin komen tot leven door de acties in hun creation event.

Er is vele mogelijkheden om rooms te maken. Naast het instellen van een aantal eigenschappen en het toevoegen van instanties van objecten, kun je ook achtergronden toevoegen, views definiëren, en tiles toevoegen. Het merendeel van deze opties worden later besproken. In dit hoofdstuk zullen we alleen aandacht besteden aan een aantal standaard instellingen, het toevoegen van instanties van objecten en het toevoegen van achtergrondafbeeldingen.

Om een room te creëren, kies **Add Room** van het **Add**-menu. Het volgende venster zal in beeld verschijnen:



Bovenaan het venster bevindt zich een werkbalk. Hier kun je de grootte van cellen van de grid instellen hetgeen noodzakelijk is omdat die moet overeenstemmen met de grootte van de objecten. Je kunt ook aangeven of je de gridlijnen, de achtergronden, etc. wel of niet wilt zien. Soms is het nuttig om tijdelijk een aantal dingen van de room te verbergen. Houd er wel rekening mee dat als je instanties van objecten aan de room toevoegt deze altijd zichtbaar zullen zijn, ongeacht de instellingen van de view. Er zijn ook knoppen om in een keer alle instanties uit de room te verwijderen of om alle instanties een paar pixels te verplaatsen. Als je negatieve getallen gebruikt zullen de instanties naar links verplaatst worden. Dit kan handig zijn als je bijvoorbeeld van plan bent de room te vergroten. Je kunt deze instelling ook gebruiken om instanties buiten de room te plaatsen, hetgeen soms erg handig kan zijn. Tenslotte is er ook nog een **Undo** knop waarmee je de laatste bewerking ongedaan mee kunt maken en een **OK** knop om de veranderingen op te slaan. (Als je rechtsboven op het kruisje klikt zal het venster gesloten worden zonder dat de wijzigingen worden opgeslagen.)

Aan de linkerkant tref je 3 tabbladen aan (in de advanced mode zijn het er 5). In het tabblad **objects** kun je instanties van objecten aan de room toevoegen. In het tabblad **settings** kun je een aantal instellingen van de room bepalen. In het tabblad **backgrounds** kun je afbeeldingen aan de achtergrond toevoegen.

Het toevoegen van instanties

Aan de rechterkant van het room ontwerpvenster zie je de room. In het begin is die leeg met een grijze achtergrond

Om instanties aan de room toe te voegen moet je eerst op het tabblad **objects** klikken, als dat tabblad al niet zichtbaar is. Selecteer daarna het object dat je wilt toevoegen door te klikken op de knop met het menu-icoon (of door te klikken in het gebied met de afbeeldingen aan de linkerkant). aan de linkerkant verschijnt nu de afbeelding van het object. (houd er rekening mee dat als je de oorspronkelijke sprite hebt gewijzigd er een kruis op de afbeelding staat. Dit geeft aan of de instanties wel of niet goed passen binnen de grid.) Klik nu met je linkermuisknop rechts in de room. Je merkt dat er een instantie van het object op die plek in de room, passsend in de grid, verschijnt. Indien de de <Alt>-toets ingedrukt houdt terwijl je instantie plaatst in de room, zul je merken dat diee niet mooi uitgelijnd in de grid geplaatst zal worden. Als je echter de muisknop ingedrukt houdt terwijl je de instantie in de ruimte plaatst, zal deze wel correct in de grid geplaatst worden. Als je de <Shift>-toets ingedrukt houdt en met ingedrukte muisknop beweegt in de room, zul je merken dat meerdere instanties correct in de grid worden geplaatst. Op die manier kun je de room vullen met instanties volgens bepaalde patronen.

Je zult merken dat bij het over elkaar heen plaatsen van instanties de oorspronkelijke instantie zal verdwijnen. Vaak zal dit ook de bedoeling zijn, maar het kan zijn dat dat niet het geval is. Dit kan vermeden worden door het vinkje in de box **Delete underlying** aan de linkerkant te verwijderen.

Als je de positie van een instantie in de room wilt wijzigen moet je de <Ctrl>-toets ingedrukt houden en met de linkermuisknop klikken op de instantie, de muisknop ingedrukt houden en vervolgens de instyantie naar de gewenste plaats in de room verslepen. (Gebruik daarbij ook de <Alt>-toets voor precieze positionering.)

Als je de <Ctrl> toets ingedrukt houdt terwijl de rechtermuisknop indrukt op een instantie, verschijnt er een menu. Hier kun je het object verwijderen, een exacte positie intypen voor de instantie of de onderste instantie naar boven plaatsen of omgekeerd.

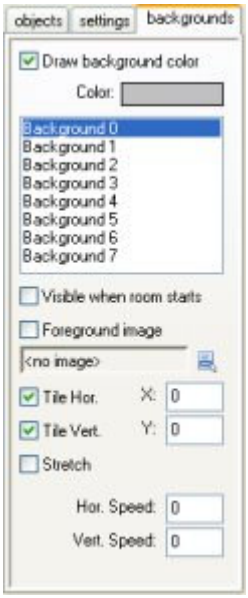
Room setting

Each room has a number of settings that you can change by clicking on the **settings** tab.

Elke room heeft een naam. Je kunt hem het beste een zinvolle naam geven. Er is ook een titel. Deze titel wordt getoond in de venstertitelbalk wanneer het spel gespeeld wordt. Je kunt de breedte en de hoogte van de room (in pixels) bepalen. Maak deze niet te groot. Het zal het spel vertragen en zal veel resources vergen. Ook kan je de snelheid van het spel bepalen. Dit is het aantal stappen per seconde. Hoe hoger de snelheid, hoe vloeiender de beweging. Maar je zult een snelle computer nodig hebben om het spel te spelen.

Setting the background

With the tab **backgrounds** you can set the background image for the room. Actually, you can specify multiple backgrounds. The tab page looks as follows:



At the top you will see the background color. You can click on it to change it. The background color is only useful if you don't use a background image that covers the whole room. Otherwise, best uncheck the box labeled **Draw background color** because this will be a waste of time.

Aan de linkerzijde zie je een lijst van 8 achtergronden. Je kan ze allemaal definiëren maar je zal er meestal maar één of twee nodig hebben. Om een achtergrond te definiëren, selecteer je hem eerst in de lijst. Vervolgens klik je **Visible when room starts** aan, anders kun je het niet zien. De naam van de achtergrond zal vet worden wanneer hij wordt gedefinieerd. Kies nu op een achtergrondplaatje in het menu. Er zijn een aantal instellingen die je kunt aanpassen. Ten eerste kan je aangeven of het achtergrondplaatje de room horizontaal en/of verticaal moet opvullen, als tapijttegels in een kamer. Je kunt ook de positie van de achtergrond in de room bepalen (dit zal ook het opvullen van de room beïnvloeden). Een andere optie is het oprekken van de achtergrond. De achtergrond wordt dan zodanig opgeschaald dat het de hele room zal opvullen. De verhouding tussen breedte en hoogte wordt dan niet gehandhaafd. Tenslotte kan je de achtergrond laten scrollen door hem een horizontale of verticale snelheid te geven. Het resultaat hiervan zal een beetje schokkerig zijn.

Er is nog één checkbox: **Foreground image**. Wanneer je deze checkbox aanvinkt, wordt de achtergrond eigenlijk een voorgrond, die voor al het andere wordt getekend. Zo'n plaatje zal natuurlijk gedeeltelijk doorzichtig moeten zijn om van enig nut te zijn.

Het verspreiden van spellen

Met behulp van de informatie uit de vorige hoofdstukken ben je in staat om spellen te maken. Je wilt natuurlijk dat mensen je spellen spelen af die af zijn. Je kunt ze natuurlijk je .gm6-bestanden geven zodat ze de spellen in *Game Maker* kunnen spelen, maar dat is waarschijnlijk niet wat je wilt. In de eerste plaats wil je waarschijnlijk dat mensen je gemaakte spellen niet kunnen wijzigen en op de tweede plaats wil je natuurlijk dat ook mensen de spellen kunnen pelen als ze niet in bezit zijn van *Game Maker*. Met andere woorden: je wilt zelfstartende stand-alone spellen maken.

Hetis erg gemakkelijk om in *Game Maker* zelfstartende stand-alone spellen te maken. In het **File** menu selecteer je het item **Create Executable**. Je krijgt dan de vraag om het spel een naam te geven. Geef het spel een naam en druk op **OK** en je hebt een stand-alone spel dat je aan iedereen kunt geven. Je kunt ook het icoon voor het stand-alone spel aanpassen in door te klikken op de optie Change Global Game Settings in het **Add** menu.

Als je eenmaal een zelfstartend stad-alone spel hebt gemaakt zoals hierboven beschreven, kun je dus de file weggeven aan andere mensen of op je website plaatsen waar het eventueel gedownload kan worden. Je bent volledig vrij in het verspreiden van je spellen, gemaakt met *Game Maker* op wat voor manier dan ook. Zelfs het verkopen van de spellen is toegestaan. Dit betekent natuurlijk ook dat de sprites, de afbeeldingen en geluiden mee verspreid worden of verkocht. Let er wel op dat alle gebruikte zaken vrij van rechten zijn. Kijk in de bijgesloten licentievoorwaarden voor meer informatie hierover.

Onder normale omstandigheden is het nuttig om je spellen te zippen, samen met enige documentatie, zoals een readme-bestand. In windows XP kan dat rechtstreeks gebeuren, daarvoor moet je kijken in het rechtermuisknop menu kijken, maar bovendien zijn er meerdere gratis zipprogramma's op het internet te vinden. Een alternatief is om een installer voor je spel te maken. Ook hiervoor bestaan een aantal freeware programma's op het internet.

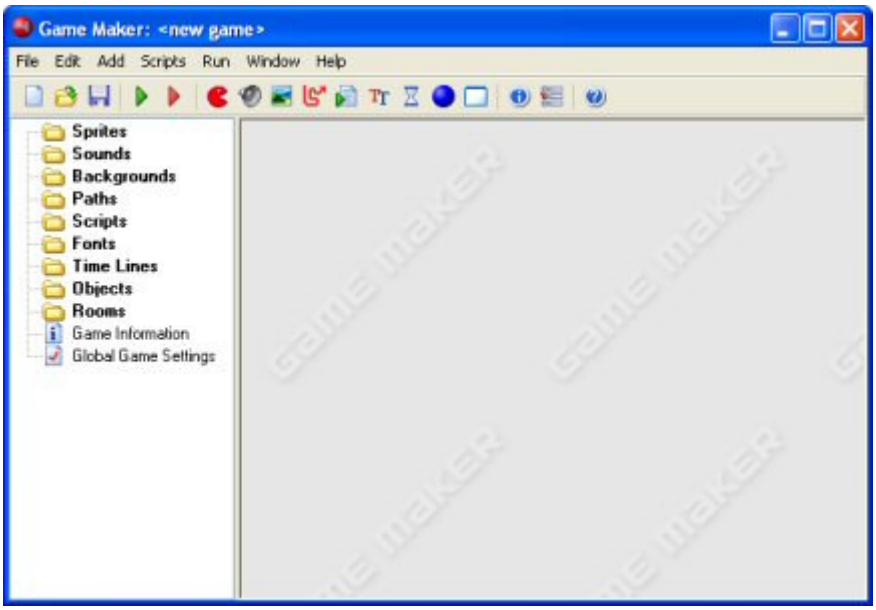
Gevorderd gebruik

Dit gedeelte geeft informatie over het meer gevorderd gebruik van *Game Maker*.

De advanced mode

Tot nu toe hebben we ons bezig gehouden met de eenvoudige features van *Game Maker*. Maar er is veel meer mogelijk. Maar om dit mogelijk te maken moet je *Game Maker* draaien in de advanced mode. Dit kun je eenvoudig voor elkaar krijgen. Simpelweg in het **File** menu op het item **Advanced mode** klikken. Om echt alles in volle omvang te kunnen waarnemen moet je *Game Maker* op nieuw opstarten of op zijn minst je spel opslaan en opnieuw laden.

Als je *Game Maker* in de advanced mode start, krijg je het volgende scherm in beeld:



Het bevat alle opties die ook in de simple mode te zien zijn maar daarnaast nog een aantal resources, knoppen en menu-items. Bovendien, hebben de resources, zoals we in de volgende hoofdtukken zullen zien, extra opties. Hier bespreken we de extra menu-items.

File menu

In het file menu vind je de volgende extra commando's:

- **Merge Game.** Met deze optie kun je alle resources (sprites, geluiden, objecten, rooms, etc.) van een spel overhevelen naar het huidige spel. Dit is erg handig als je gedeeltes gaat maken die je opnieuw wilt gebruiken (bijvoorbeeld, menu systemen). (Houd er wel rekening mee dat alle resources en instanties en tiles een nieuw id krijgen, hetgeen tot problemen kan leiden als je ze gebruikt in scripts.) Het is jouw verantwoordelijkheid dat de resources in beide files verschillende namen hebben, als dat niet het geval is, ontstaan geheid problemen.
- **Preferences.** Hier kun je een aantal voorkeursinstellingen van *Game Maker* instellen. Deze zullen worden herinnerd door *Game Maker*, ook als je het spel afsluit en daarna weer opent. Kijk hieronder in de lijst van alle mogelijkheden.

Voorkeursinstellingen

Om de voorkeursinstellingen te maken kies je het item **Preferences** uit het menu **File**. Deze worden bewaard, ook als je *Game Maker* afsluit en weer opent om verder te gaan. Je kunt de volgende instellingen maken:

- **Show recently edited games in the file menu.** Als deze optie is aangevinkt, zullen de laatste 8 bewerkte spellen (of minder, als je nog geen 8 spellen hebt bewerkt) in het **file** menu zichtbaar worden door op de optie **Recent Files** te gaan staan.
- **Load last opened file on startup.** Als deze optie is aangevinkt, wordt in *Game Maker* automatisch het laatst bewerkte spel geopend.
- **Keep backup copies of files.** Als deze optie is aangevinkt, zal het programma automatisch een backup maken van het spel waarmee je bezig bent. Deze backup krijgt de extensie gb0-gb9. Deze spellen zijn gewoon in *Game Maker* te openen. Er wordt aangeraden om minsten één backup van je werk op te slaan!
- **Maximal number of backups.** Hier kun je aangeven hoeveel (1-9) verschillende kopiën er door het programma moeten worden onthouden.
- **Show progress while loading and saving files.** Als deze optie is aangevinkt, zal tijdens het laden of opslaan een voortgangsindicator in beeld verschijnen.
- **At startup check for, and remove old temporary files.** *Game Maker* en spellen met *Game Maker* gemaakt, zorgen ervoor dar er tijdelijke bestanden worden geproduceerd. Normaliter worden deze files automatisch verwijderd, maar soms, bijvoorbeeld als een spel crasht, worden ze bewaard. Indien deze optie is aangevinkt zal *Game Maker* bij de start nagaan of zo'n file bestaat en die daarna meteen verwijderen.
- **Run games in secure mode.** Als deze optie is aangevinkt, zal geen enkel spel dat jij met *Game Maker* hebt gemaakt en op jouw computer draait externe programma's kunnen opstarten, files kunnen wijzigen of verwijderen die niet tot het spel behoren. (Dit is een veiligheidsmaatregel tegen Trojaanse paarden, maar het is geenszins zeker of deze veiligheid in alle gevallen zal werken. Je kunt geen schadeclaims indienen als dat wel het geval mocht blijken.). Als deze optie is aangevinkt betekent dat dus dat spellen die gebruikmaken van externe files niet goed werken. Deze optie werkt alleen tijdens het spelen van het spel in *Game Maker*. Maak je van het spel een stand-alone executable, zal het NIET in de veilige modus werken.
- **Show the origin and bounding box in the sprite image.** Als deze optie is aangevinkt in het eigenschappenvenster van de sprite, worden in de spriteafbeelding de oorsprong en bounding box aangegeven.
- **In object properties, show hints for actions.** Als deze optie is aangevinkt in het eigenschappenvenster van een object, zal er een beschrijving getoond worden als je met de muis over een actie gaat.
- **When closing, remove instances outside the room.** Als deze optie is aangevinkt, waarschuwt het programma als zich instanties, of tiles buiten de room bevinden en geeft je de mogelijkheid om ze te verwijderen.
- **Remember room settings when closing the form.** Als deze optie is aangevinkt, onthoudt het programma een aantal roominstellingen, zoals het wel of niet laten zien van

de grid, wel of niet verwijderen van onderliggende objecten, enz. als je later dezelfde room weer wilt bewerken.

- **Scripts and code and colors.** Kijk hiervoor in het hoofdstuk over scripts voor meer informatie over de voorkeursinstellingen.
- **Image editor.** Standaard gebruikt *Game Maker* een ingebouwd beeldbewerkingsprogramma. Als je echter een ander beeldbewerkingsprogramma wilt gebruiken, kun je dat hier instellen.
- **External sound editors.** Je kunt hier aangeven welke externe geluidsbewerkingsprogramma's je wilt gebruiken voor de verschillende soorten geluidsfiles. Merk op dat *Game Maker* geen ingebouwd geluidsbewerkingsprogramma bevat hetgeen betekent dat als je hier niets invult, je geluiden niet kunt bewerken.

Edit menu

In het edit menu vind je de volgende extra commando's:

- **Add group.** Resources kunnen worden gegroepeerd. Dit is erg handig als je van plaan beent erg grote spellen te gaan maken. Bijvoorbeeld, je kunt alle geluiden die aan een object gekoppeld zijn in een groep stoppen, of je kunt alle objecten die samen in één level voorkomen in een groep stoppen. Dit commando voegt een groep toe in het huidige resourcetype. Je moet die groep wel een naam geven. Binnen een groep kun je ook subgroepen aanmaken, enz. Zoals onder wordt aangegeven, is het mogelijk om resources in groepen te slepen.
- **Find Resource.** Met dit commando kun je een naam intypen van een resource en meteen het passende eigenschappenvenster openen.
- **Show Object Information.** Met behulp van dit commando kun je in één oogopslag alle objecten in het spel zien.

Add menu

In dit menu kun je extra resources toevoegen. Merk op dat er voor al die resources ook op de toolbar een knop te vinden is en een toetscombinatie op het toetsenbord.

Scripts menu

In het scripts menu vind je de volgende extra commando's:

- **Import Scripts.** Kan gebruikt worden om nuttige scriptfiles te importeren.

- **Export Scripts.** Kan gebruikt worden om jouw script op te slaan in een bestand waardoor het door anderen gebruikt kan worden. Als je een script resource selecteert, wordt alleen dat script opgeslagen. Als je een groep scripts selecteert, worden alle scripts uit die groep opgeslagen. Als je de root resource selecteert worden alle scripts die er zijn opgeslagen. Je kunt deze optie ook gebruiken door met je muis op een script te gaan staan of op een groep en vervolgens op de rechtermuisknop te klikken waardoor de optie in beeld verschijnt (in een menu).
- **Show Built-in Variables.** Geeft een gesorteerd overzicht van alle ingebouwde variabelen, zowel de locale als de globale variabelen.
- **Show Built-in Functions.** Geeft een gesorteerd overzicht van alle ingebouwde functies.
- **Show Constants.** Geeft een gesorteerd overzicht van alle ingebouwde constanten and constanten, gedefinieerd in de spelopties.
- **Show Resource Names.** Geeft een gesorteerd overzicht van alle resource namen. Je kunt op de resourcesnaam klikken om die resource te kunnen bewerken.
- **Search in Scripts.** Je kunt hiermee in een script naar een bepaalde string zoeken. Je kunt dan op een van de gevonden plaatsen klikken, waarna de cursor automatisch in het script naar die plaats gaat om zao dat script bijvoorbeeld te kunnen editen.
- **Check Resource Names.** Hiermee kun je alle namen van de resources checken. Namen die onjuist zijn worden aangegeven, datzelfde geldt ook voor dubbele resource namen, of als een resourcenaam de naam is van een variabele, een functie of een constante. Je kunt op die namen klikken waarna de resource wordt geopend en je aanpassingen kunt verrichten.
- **Check All Scripts.** Met deze optie kun je alle scripts op fouten controleren. Je kunt op de aangegeven plaatsen klikken, waarna de cursor op die plaats terechtkomt om zo je script te kunnen bewerken.

Meer over sprites

Er betstaan meerdere gevorderde gebruiksmogelijkheden om je eigen sprites te maken.

Het bewerken van sprites

Tot nu toe hebben we de sprites steeds uit bestanden geladen. Het is echter ook mogelijk om ze te maken en te wijzigen binnen *Game Maker*. Dit doe je door te dubbelklikken op een sprite (of door een nieuwe te maken); het sprite eigenschappenvenster zal verschijnen. Klik nu op de knop **Edit Sprite**. Een nieuw formulier zal verschijnen met daarin alle subplaatjes waaruit sprite is opgebouwd.

Het sprite edit-venster ziet er als volgt uit:

Rechts zie je de verschillende plaatjes van de sprite. Merk op dat in **Game Maker** alle subplaatjes van een sprite dezelfde grootte moeten hebben. Aan de linkerzijde zie je hoe de geanimeerde sprite eruit ziet. (Als je de animatie niet ziet, zet dan een vinkje in de box **Show Preview**.) Onder de preview kan je de snelheid van de animatie en de achtergrondkleur instellen. Op deze wijze krijg je een idee van de animatie in de spelsituatie. (Let op: deze snelheid is alleen bedoeld voor de preview. De snelheid van de animatie tijdens het spel hangt af van de snelheid van de room.)

De sprite editor bevat vele commando's om een sprite te maken en te veranderen. Dit gebeurt allemaal via menu's. (Voor sommige zijn er knoppen op de toolbar.) Sommige commando's bewerken op individuele plaatjes. Hiervoor moet je eerst een subplaatje met de muis selecteren.

File menu

Het file menu bevat een aantal commando's die betrekking hebben op het laden en opslaan van sprites.

- **New.** Creëert een nieuwe, lege sprite. Je moet de grootte van de sprite instellen. (Denk eraan dat alle plaatjes van een sprite dezelfde grootte moeten hebben.)
- **Create from file.** Maak een sprite uit een bestand. Vele bestandtypes kunnen worden gebruikt. Op deze manier maak je een sprite die uit één enkel plaatje bestaat, behalve als je geanimeerde GIF-bestanden gebruikt die in subplaatjes verdeeld zijn. Merk op dat de transparantiekleur de kleur is van de pixel die zich helemaal linksonder in het plaatje bevindt, niet de transparantiekleur van het GIF-bestand. Je kunt er ook voor kiezen voor meerdere afbeeldingen (multiple images), die dan worden geladen. Ze moeten echter wel allemaal even groot zijn.
- **Add from file.** Voeg een plaatje (of plaatjes) uit een bestand toe aan de huidige sprite. Als de plaatjes niet dezelfde grootte hebben kunt je kiezen waar ze neer wil zetten of dat je ze uit wil rekken. Je kunt er ook voor kiezen voor meerdere afbeeldingen (multiple images), die dan worden geladen. Ze moeten echter wel allemaal even groot zijn.

- **Save as GIF.** Slaat de sprite op als een animated gif-bestand.
- **Save as strip.** Slaat de sprite op als een bitmap, met alle afbeeldingen naast elkaar.
- **Create from strip.** Maakt het mogelijk dat je een sprite uit een strip kunt maken. Hieronder volgt er meer informatie over.
- **Add from strip.** Gebruik dit om afbeeldingen van een strip toe te voegen. Hieronder volgt er meer informatie over.
- **Close saving changes.** Sluit het formulier, veranderingen worden opgeslagen. Als je de veranderingen niet wilt bewaren, klik op de afsluitknop van het venster.

Edit menu

Het edit menu bevat een aantal commando's die betrekking hebben op de huidig geselecteerde sprite. Je kunt hem knippen naar het klembord, een plaatje van het klembord plakken, de huidige sprite leegmaken of verwijderen, en sprites naar links en rechts in de opeenvolging verschuiven. Tenslotte is er een commando om een individueel plaatje te bewerken in het ingebouwde tekenprogramma. (Kijk onder voor meer informatie.)

Transform menu

In het transform menu kan je een aantal wijzigingen in de plaatjes uitvoeren.

- **Mirror horizontal.** Horizontaal spiegelen van de plaatjes.
- **Flip vertical.** Verticaal spiegelen van de plaatjes.
- **Shift.** Hier kan je de plaatjes horizontaal en verticaal verplaatsen over een bepaalde waarde.
- **Rotate.** Je kunt de plaatjes 90 graden, 180 graden, of een willekeurig aantal graden roteren. In het laatst geval kan je ook de kwaliteit bepalen. Experimenteer voor de beste resultaten.
- **Resize Canvas.** Hier kan je de grootte van het canvas veranderen. Je kunt aangeven waar de oude plaatjes op het nieuwe canvas worden geplaatst.
- **Stretch.** Hier kan je de plaatjes tot een nieuwe grootte uitrekken. Je kunt de schaal en de kwaliteit aangeven.
- **Scale.** Dit commando schaalt de plaatjes (maar niet de beeldgrootte!). Je kunt de schaal, de kwaliteit, en de positie van de huidige plaatjes aangeven.

Images menu

In het images menu kan je een aantal operaties op de plaatjes uitvoeren.

- **Cycle left.** Verschuif alle plaatjes één plaats naar de linkerzijde. Hierdoor begint de animatie op een ander punt.
- **Cycle right.** Verschuif alle plaatjes één plaats naar rechts.
- **Black and white.** Maakt een sprite zwart wit (beïnvloedt de kleur van de transparantie niet!).
- **Colorize.** Hier kan je de kleur (tinten) van de plaatjes veranderen. Gebruik de schuifbalk om de verschillende kleuren te kiezen.
- **Colorize Partial.** Hier kun je de kleur (hue) van gedeeltes van afbeeldingen wijzigen. Je kunt een bestaande kleur kiezen met een aantal kleuren er omheen om vervolgens aan te geven die rage kleuren te vervangen door een nieuwe. Dit is handig als je bijvoorbeeld de kleur van een shirt van een speler wilt vervangen.
- **Shift Hue.** Dit is een andere manier om de kleur van afbeeldingen te veranderen. Maar bij deze handeling worden de kleuren boven dan de aangegeven waarde aan de plaatjes toegevoegd, heeftgeen tamelijk interessante effecten kan geven.
- **Intensity.** Hier kan je de intensiteit veranderen door waarden voor de kleurenverzadiging en de lichtheid van de plaatjes te verstrekken.
- **Fade.** Hier bepaal je een kleur en een waarde. De kleuren in de plaatjes veranderen nu langzaam naar deze kleur.
- **Transparency.** Hier kan je het niveau van gaasdeur (screen-door) transparantie aangeven. Dit wordt bereikt door een aantal pixel doorzichtig te maken.
- **Blur.** Om de plaatjes te vertroebelen worden de kleuren een beetje gemengd, zodat het vager wordt. Hoe hoger de waarde, hoe vager het wordt.
- **Crop.** Dit maakt de plaatjes zo klein mogelijk. Dit is zeer nuttig omdat hoe groter de beelden, hoe meer videogeheugen het spel zal gebruiken. Je kunt een kader aangeven rond het plaatje om een transparantie probleem te vermijden.

Je zult met deze commando's moeten experimenteren om sprites te krijgen zoals jij ze wilt.

Animation menu

In het animation menu kan je nieuwe animaties maken uit de huidige animaties. Er zijn vele opties en je zou er een beetje mee moeten experimenteren om de effecten te maken die je wilt. Vergeet ook niet dat je een animatie altijd kunt opslaan en later aan de huidige animatie kunt toevoegen. Ook kan je altijd een aantal lege plaatjes toevoegen en ongewenste schrappen. Ik zal kort de verschillende mogelijkheden aangeven.

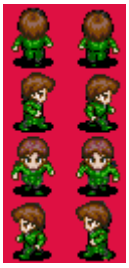
- **Set Length.** Hier kan je de lengte van je animatie veranderen. De animatie wordt vaak genoeg herhaald om het aantal frames te creëren je aangeeft. (Normaal wilt je dat dit een veelvoud van het huidige aantal frames zal zijn.)

- **Stretch.** Dit bevel verandert ook de lengte van de animatie. Maar deze keer, worden frames verdubbeld of verwijderd om het juiste aantal te krijgen. Als je het aantal frames verhoogt gaat de animatie langzamer en als je het aantal vermindert het sneller gaat.
- **Reverse.** Zoals te verwachten keert dit commando de animatie om. Hij wordt dus achterstevoren afgespeeld.
- **Add Reverse.** Dit keer wordt de omgekeerde animatie toegevoegd, hierdoor verdubbeld het aantal kaders. Dit is zeer nuttig om een voorwerp naar links en rechts te laten gaan, de kleur te veranderen en weer terug te gaan, enz.. Misschien wil je wel het dubbele eerste en middelste frame die zo ontstaan verwijderen.
- **Translation sequence.** Je kunt een animatie maken waarin het plaatje een klein beetje verplaatst bij elke stap. Je moet het aantal frames aangeven en de totale waarde van de horizontale en verticale beweging.
- **Rotation sequence.** Creëert een animatie waarin het plaatje roteert. Je kunt of omwenteling met de wijzers van de klok mee kiezen of tegen de klok in. Bepaal het aantal frames en de totale hoek in graden (360 is een volledige draai). Je moet misschien de grootte van het canvas aanpassen om ervoor te zorgen het totale plaatje tijdens de omwenteling zichtbaar blijft.
- **Colorize.** Creëert een animatie die het plaatje in een bepaalde kleur verandert.
- **Fade to color.** Creëert een animatie die het plaatje langzaam in een bepaalde kleur verandert.
- **Disappear.** Laat het plaatje verdwijnen met behulp van gaasdeur (screen-door) transparantie.
- **Shrink.** Krimpt het plaatje tot er niets overblijft. Je kunt de richting aangeven.
- **Grow.** Laat het plaatje groeien vanuit niets.
- **Flatten.** Vlakt het plaatje uit tot niets in een bepaalde richting.
- **Raise.** Laat het plaatje groeien vanuit een bepaalde richting.
- **Overlay.** Bedekt de animatie met een andere animatie of plaatje uit een bestand.
- **Morph.** 'Morft' (Verandert) de animatie naar een animatie of plaatje uit een bestand. Merk op dat dit 'morfen' het best werkt als de twee animaties hetzelfde gebied van het plaatje gebruiken. Anders verdwijnen halverwege de animatie bepaalde pixels en andere verschijnen plotseling.

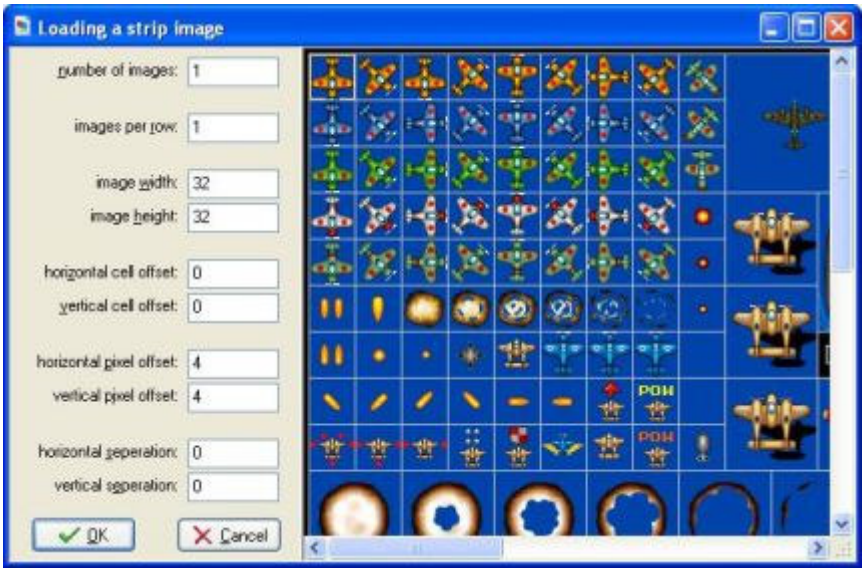
In het bijzonder zijn de laatste twee commando's zeer krachtig. Bijvoorbeeld, om een voorwerp op te blazen, voeg je een aantal exemplaren toe en een aantal lege kaders. Gebruik dan het commando overlay met een animatie van een explosie. (Zorg ervoor dat het aantal plaatjes klopt.) Een andere manier om dit te doen is je animatie 'morfen'of omvormen naar een animatie van een explosie. Met wat oefening kunt je geweldige sprites maken.

Strips

Zoals hierboven al vermeld, worden sprites normaal opgeslagen als animated gif-files of als strips. Een strip is één grote bitmap die alle plaatjes naast elkaar opslaat. Het enige probleem is dat de grootte van individuele subplaatjes niet in het plaatje wordt opgeslagen. Daarbij komt dat vele stripbestanden die te vinden zijn op het internet meerdere sprites opslaan in één bestand. Bijvoorbeeld, het volgende stuk van een stripbestand bevat vier verschillende animaties.



Om individuele sprites uit dergelijke bestanden te selecteren, kun je kiezen voor **Create from Strip** of **Add from Strip** van het **File** menu. Na aangeven van het juiste stripbestand zal het volgende venster in beeld verschijnen:



Rechts zie je (een deel van) het stripplaatje dat je selecteerde. Aan de linkerzijde kan je een aantal parameters opgeven die aangeven in welke subplaatjes je geïnteresseerd bent. Merk op dat één of meerdere rechthoeken in het plaatje aangeven welke plaatjes zijn geselecteerd. De volgende parameters kunnen worden opgegeven:

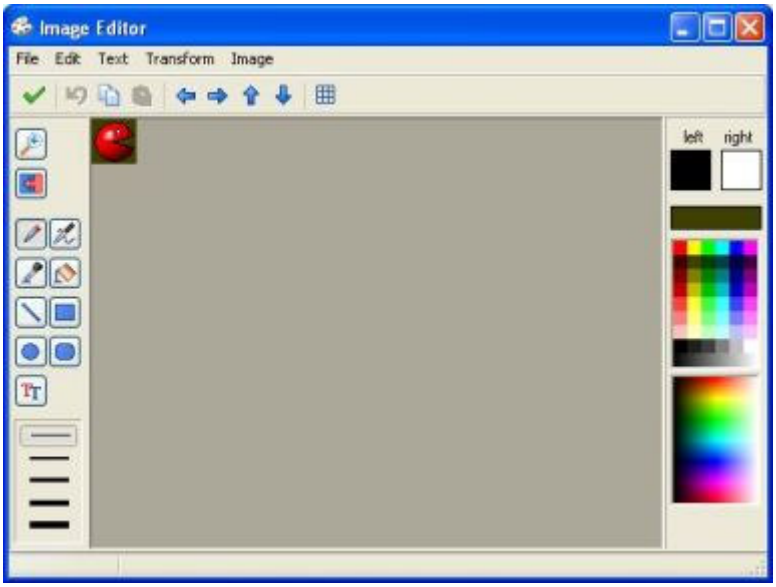
- **Number of images.** Dit is het aantal plaatjes dat je uit de strook wilt nemen.
- **Images per row.** Hoeveel van de plaatjes die jij wilt staan er per rij. Bijvoorbeeld, door dit op 1 te zetten zult je een verticale opeenvolging van plaatjes selecteren.
- **Image width.** Breedte van de individuele beelden.
- **Image height.** Hoogte individuele beelden.

- **Horizontal cell offset.** Als je niet de plaatjes linksboven wilt selecteren, kan je hier aangeven hoeveel plaatjes horizontaal zouden moeten worden overgeslagen.
- **Vertical cell offset.** Hier geef je aan hoeveel plaatjes verticaal over te slaan.
- **Horizontal pixel offset.** Soms is er wat extra ruimte bij de linkerbovenkant. Hier geef je dit aantal aan (in pixels). Here you indicate this amount (in pixels).
- **Vertical pixel offset.** Verticale hoeveelheid extra ruimte.
- **Horizontal separation.** In sommige strips zijn er lijnen of lege ruimtes tussen de plaatjes. Hier kan je de horizontale waarde aangeven om tussen de plaatjes (in pixel) over te slaan.
- **Vertical separation.** Verticale waarde om tussen de plaatjes over te slaan.

Zodra je de correcte reeks plaatjes geselecteerd hebt, klik je op **OK** om je sprite te creëren. Denk er alsjeblieft aan dat je plaatjes die door anderen zijn gemaakt slechts mag gebruiken wanneer je hun toestemming hebt of wanneer ze freeware zijn.

Het bewerken van individuele subplaatjes

Je kunt de individuele subplaatjes ook bewerken. Daartoe selecteer je een subplaatje en kies je **Edit Image** van het **Image** menu. Dit zal een ingebouwd beeldbewerkingprogrammaatje openen. Realiseer je wel dat dit een beperkt programma is dat hoofdzakelijk gemaakt is om kleine veranderingen in bestaande plaatjes aan te brengen en niet om nieuwe plaatjes te ontwerpen. Daarvoor kun je beter een volledig tekenprogramma gebruiken en gebruik bestanden (of kopiëren en plakken) om het plaatje in *Game Maker* te krijgen. Je kunt bovendien een extern beeldbewerkingsprogramma in de voorkeuren in *Game Maker* zetten.



Het formulier hierboven toont het plaatje in het midden en een aantal basistekenknoppen aan de linkerzijde. Hier kan je in en uitzoomen, pixel, lijnen en rechthoeken tekenen, tekst plaatsen, enz..

Merk op dat de kleur afhangt van het feit of je de linker of rechter muisknop gebruikt. Voor sommige tekeninstrumenten kunnen je eigenschappen instellen (zoals lijnbreedte of zichtbaarheid van de omtrek). Er is een speciale knop die alle pixel van een bepaalde kleur verandert in een andere kleur. Dit is in het bijzonder nuttig om de achtergrondkleur te veranderen die voor transparantie wordt gebruikt. Op de toolbar staan een aantal speciale knoppen om alle pixels in het plaatje in een bepaalde richting te bewegen. Ook kan je aangeven om een rooster te tonen wanneer het plaatje wordt gezoomd (dit werkt alleen bij een zoomfactor van minstens 4).

Rechts in het venster kun je de te gebruiken kleuren selecteren (één voor de linkermuisknop en één voor de rechter). Er zijn vier manieren om de kleur te veranderen. Ten eerste kun je met de muisknop (links of rechts) in één van de 16 basiskleuren klikken. Merk op dat er een speciaal kleurvakje is die de kleur van de pixel linksonder in het plaatje aangeeft, deze kleur wordt gebruikt als transparantiekleur als de sprite doorzichtig is. Je kunt deze kleur gebruiken om een deel van je plaatje doorzichtig te maken. De tweede manier is in het plaatje met de veranderende kleur te klikken. Hier kun veel meer kleurenkiezen. Je kunt de muisknop constant ingedrukt houden om de kleur te zien die je selecteert. Ten derde, kun je met de linkermuis in de box klikken die de linker en rechter kleur aangeven. Een kleurendialoogvenster komt in beeld waarin je de kleur kunt selecteren. Tenslotte kun je aan de linkerzijde het kleurendruppelbuisje selecteren en op een positie in het plaatje klikken om die kleur te kopiëren.

In de menu's zal je dezelfde transformatie en edit commando's aantreffen die ook beschikbaar zijn in de sprite editor. Dit keer zijn ze echter slechts op het huidige plaatje van toepassing. (Wanneer een sprite meerdere plaatjes bevat, zijn commando's die verandering van de grootte teweeg brengen, zoals stretch, niet beschikbaar. Je kunt het plaatje ook als bitmap bewaren. Er zijn twee extra commando's in het **Image** menu:

- **Clear.** Maak het plaatje leeg met de linkerkleur (die dan automatisch de kleur van de transparantie wordt).
- **Gradient fill.** Met dit commando kan je het plaatje vullen met een geleidelijk aan veranderende kleur (niet erg nuttig om sprites te maken, maar het ziet er aardig uit, en kan voor achtergronden worden gebruikt, die hetzelfde programma gebruiken).

Merk op dat er geen mechanisme is om delen van het plaatje te selecteren. Ook missen we een aantal prettige tekenroutines. Hiervoor zou je meer geavanceerd tekenprogramma moeten gebruiken (of eenvoudig het programma Paint dat in Windows zit). De gemakkelijkste manier om dit te doen is de kopiëerknop te gebruiken om zo het plaatje op het klembord te zetten. In je tekenprogramma plak je het plaatje om het te krijgen. Dan kun je het bewerken en daarna weer naar het klembord kopiëren. Daarna kan je in *Game Maker* het gewijzigde plaatje weer plakken.

Geavanceerde sprite instellingen

In de Avanced mode komen in het sprite eigenschappenformulier een aantal geavanceerde opties die wij hier zullen behandelen.

Ten eerste zijn er opties met betrekking tot het controleren van botsingen. Telkens wanneer twee instanties samenkomen wordt een collision event gegenereerd. De botsingen worden gecontroleerd op de volgende manier. Elke sprite heeft een bounding box. Deze box is dusdanig dat het ondoorzichtige deel van alle subplaatjes bevat. Wanneer de bounding boxen overlappen, wordt er gecontroleerd of twee pixel in de huidige subplaatjes van twee sprites overlappen. Deze tweede verrichting is moeizaam en vereist extra geheugen en voorbewerken. Dus als je niet geïnteresseerd bent in het nauwkeurige controleren van botsingen tussen sprites, kan je beter het vakje **Precise collision checking** niet aanvinken. In dat geval wordt slechts gecontroleerd op het overlappen van bounding boxen. Je kunt de bounding box ook veranderen. Dit is haast nooit nodig, maar soms wil je de bounding box kleiner maken, zodat dat de botsingen met sommige uitstekende delen van de sprite niet in acht worden genomen.

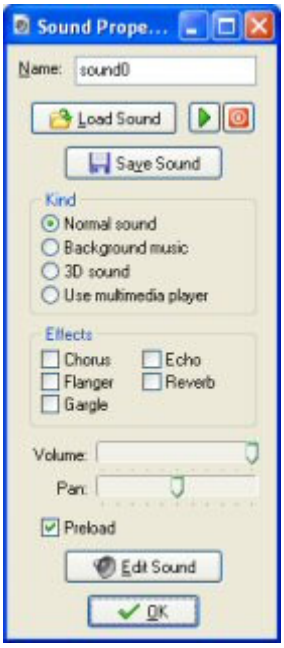
De hoeken van sprites kunnen een beetje geblokt uitzien. Om dit te vermijden kun je het beste een vinkje plaatsen in de box **Smooth edges**. In dat geval worden de pixels in de hoeken van de sprite gedeeltelijk transparant getekend. Daardoor zien de sprites er een stuk fraaier uit. (Gebruik deze optie niet als de sprites groter gemaakt moeten worden, want in dat geval zal er tussen de delen een deels transparante lijn te zien zijn.) Het effect is alleen tijdens het spelen zichtbaar, niet in de editor.

Gedurende het spelen van een spel worden sprites omgezet in textures. Deze moeten in het videogeheugen worden geplaatst (op de grafische kaart) voordat ze kunnen worden gebruikt. Als er een vinkje staat in de box **Preload texture** gebeurt dat laden meteen waardoor er geen vetraging optreedt tijdens het spel. Als je echter een grote hoeveelheid grote sprites hebt die niet meteen in het begin van het spel gebruikt worden is het raadzaam deze optie uit te zetten. *Game Maker* zal dan de benodigde textures uit het geheugen halen en ze in het videogeheugen plaatsen en ze eruit verwijderen als ze niet meer nodig zijn. (swapfunctie)

Tenslotte kun je de oorsprong van de sprite aangeven. Dit is het punt in de sprite dat overeenkomt met zijn positie in de room. Als je een instantie op een bepaalde plaats neerzet, wordt daar de oorsprong geplaatst. Standaard is het de linker bovenkoek van de sprite, maar soms is het handiger om het centrum van de sprite als oorsprong te gebruiken of enig ander belangrijk punt. Je kunt er zelfs voor kiezen om de oorsprong buiten de sprite te plaatsen. Het is ook mogelijk om de oorsprong van een sprite in te stellen door op de afbeelding te klikken (als de oorsprong in de afbeelding wordt getoond).

Meer over geluiden en muziek

In advanced mode you have a lot more control over the sounds and pieces of music you add to your game. When you add a sound resource the following form will show:



Naast de knoppen voor het laden, het opslaan en het afspelen van geluiden, gaan we nu een aantal andere instellingen die te maken hebben met geluid hier bespreken.

Allereerst kun je het type geluid aangeven . 4 Soorten zijn mogelijk. Normale geluiden worden gebruikt om geluidseffecten te krijgen en daarvoor gebruik je meestal .wav files. (Maar je kunt daar ook wel .mid files voor gebruiken.) Je kunt meerdere normale geluiden tegelijk afspelen. Het is zelfs mogelijk om meerdere kopieën van hetzelfde geluid tegelijk af te spelen. Wat achtergrondmuziek betreft, die lijken op normale geluiden, maar daarvan kun je er slechts één op enig moment afspelen. Dat betekent dat als je op een bepaald moment een bepaald achtergrondmuziekje wilt spelen, de huidige achtergrondmuziek, als die er al is, meteen wordt gestopt. Voor achtergrondmuziek worden standaard midi files gebruikt. 3D muziek kun je ook gebruiken, daarvoor kun je met behulp van speciale functies instellingen maken. Die worden alleen maar gebruikt voor geavanceerde geluidseffecten.

Geluidsbestanden worden normaal gesproken afgespeeld door gebruik te maken van DirectX. Daardoor heb je vele mogelijkheden tot je beschikking, maar je bent wel beperkt tot het gebruik van wave en midi files. Als je andere geluiden wilt afspelen, zoals mp3 bestanden, moet je kiezen voor het gebruik van de media player. Deze heeft echter wel maar beperkte instelmogelijkheden. Je kunt dan geen volumeveranderingen of geluidseffecten gebruiken en bovendien kun je dan maar één geluid tegelijkertijd afspelen. Merk op dat midi files anders kunnen klinken als ze met de media

player worden afgespeeld, bij gebruik als achtergrondmuziek of normale geluiden. Dat komt omdat de media player gebruik maakt van de hardware synthesizer (die op elke machine anders kan zijn) terwijl anders een softwareversie wordt gebruikt (die op alle machines hetzelfde klinken). Gebruik bij voorkeur géén mp3 files in je spellen. Die files moeten gedecomprimeerd worden hetgeen processortijd kost waardoor je spel trager kan worden. Het feit dat de file-grootte kleiner is betekent dus niet dat zij minder geheugen in in gebruik nemen. Bovendien worden mp3 files ook niet dooe alle machines ondersteund. Dat betekent dan dat jouw spellen niet op alle machines zijn te spelen.

Op de tweede plaats kun je een aantal geluidseffecten aan het spel toevoegen zoals chorus of echo (alleen beschikbaar in de geregistreerde versie van *Game Maker*!) Je kunt elke combinatie selecteren. Het resultaat is meteen te beluisteren. (Als je GML gebruikt kun je zelfs de parameters van deze effecten wijzigen.)

Bovendien is het mogelijk om een standaard geluidsvolume in te stellen en aan tegeven uit welke speaker het geluid moet komen.

Van alle geluiden kun je aangeven of ze tijdens het laden van het spel moeten worden geladen in het geheugen. Als een geluid wordt afgespeeld, moet het in het audio geheugen worden geladen. Als je een geluid van te voren laadt, dat gebeurt dus bij het laden van het spel, kan het meteen gebruikt worden als het nodig is in het spel. Als dat niet het geval is zal het geluidpas geladen worden als het voor het eerst gebruikt wordt in het spel. Dit spaart wel geheugen maar kan wel een vertraging opleveren als het geluid voor het eerst gebruikt wordt.

Game Maker heeft geen ingebouwde geluidseditor. Maar in de Preferences kun je externe geluidseditors opgeven om je geluiden te kunnen bewerken. Als je dat gedaan hebt, kun je met de knop **Edit Sound** deze geluidsbewerkingseditors selecteren en je geluid bewerken. (Tijdens het bewerken van geluiden zal het *Game Maker* venster verborgen zijn. Op het moment dat je het bewerken afsluit, kom je weer terug in het *Game Maker* venster.)

Meer over achtergronden

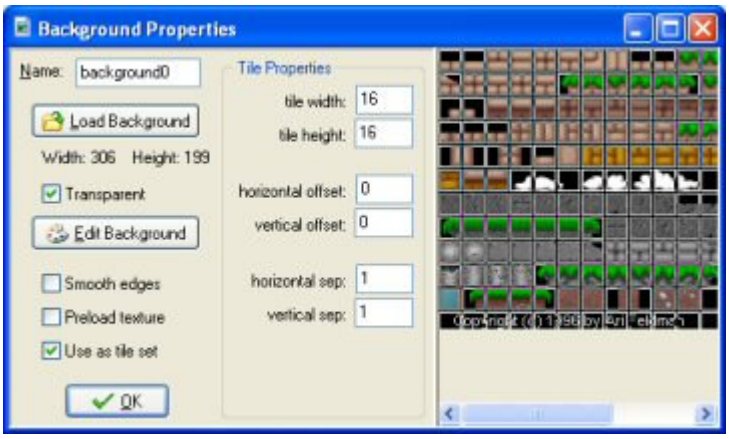
Naast het laden van uit een bestand, kun je achtergronden ook zelf maken. Daartoe klik je op **Edit Background**. Een klein ingebouwd tekenprogramma wordt geopend waarin je je achtergrond kunt maken of veranderen. Houd er rekening mee dat dit geen volwaardig programma is. Voor meer geavanceerde mogelijkheden moet je een ander tekenprogramma gebruiken. Er is één optie die bijzonder nuttig is. In het **Image menu** vind je een commando **Gradient Fill**. Dit kan worden gebruikt om een leuke achtergronden met kleurverloop te creëren.

In de advanced mode vind je in het achtergrond eigenschappenvensterthe een aantal geavanceerde opties:

De hoeken van sprites kunnen een beetje geblokt uitzien. Om dit te vermijden kun je het beste een vinkje plaatsen in de box **Smooth edges**. In dat geval worden de pixels in de hoeken van de sprite gedeeltelijk transparant getekend. Daardoor zien de sprites er een stuk fraaier uit. (Gebruik deze optie niet als de sprites groter gemaakt moeten worden, want in dat geval zal er tussen de delen een deels transparante lijn te zien zijn.) Het effect is alleen tijdens het spelen zichtbaar, niet in de editor!

Gedurende het spelen van een spel worden sprites omgezet in textures. Deze moeten in het videogeheugen worden geplaatst (op de grafische kaart) voordat ze kunnen worden gebruikt. Als er een vinkje staat in de box **Preload texture** gebeurt dat laden meteen waardoor er geen vetraging optreedt tijdens het spel. Als je echter een grote hoeveelheid grote sprites hebt die niet meteen in het begin van het spel gebruikt worden is het raadzaam deze optie uit te zetten. *Game Maker* zal dan de benodigde textures uit het geheugen halen en ze in het videogeheugen plaatsen en ze eruit verwijderen als ze niet meer nodig zijn. (swapfunctie)

Soms vind je het nuttig om een achtergrond te gebruiken die is opgebouwd uit tiles (tegels), een set kleine plaatjes die samen een grote achtergrond vormen. Bij het maken van de rooms kun je de subimages op verschillende plaatsen neerzetten. Dat is nogal belangrijk om de levels er fraai uit te laten zien. Als je de achtergrond als een tile set wilt gebruiken moet je in de box **Use as tile set** een vinkje plaatsen. Het formulier kan er dan bijvoorbeeld als volgt gaan uitzien:



Je kunt een aantal instellingen voor de tile set maken. Vooral het instellen van de hoogte en de breedte van elke tile. (Je kunt slechts 1 grootte opgeven. Je moet er wel voor zorgen dat als je meerdere tiles gebruikt ze van dezelfde grootte moeten zijn. Als je tiles hebt van verschillende groottes, kun je beter meerdere sets van tiles maken.) Je kunt ook de precieze plaats bepalen waar de tile linksboven begint. Tenslotte kan een scheiding tussen de tiles worden aangegeven

(normaal 0 of 1). Voor meer informatie over het gebruik van tiles kun je het beste kijken in het hoofdstuk over het ontwerpen van rooms.

Een waarschuwing is hier wel op zijn plaats. Als je kiest voor gescheiden borders tussen de sprites en interpolatie gebruikt tussen pixels (zie global game setttings) kan dat resulteren in scheuren tussen de tiles. Daarom is het belangrijk dat de pixels rondom de tiles precies overeenkomen met de pixels binnen de tiles om dit effect te voorkomen.

Meer over objecten

Wanneer je een object creëert in de advanced mode, kan je wat meer geavanceerde settings veranderen.

Depth

Ten eerste, kunt je de **Depth** (diepte) van de instanties van het object instellen. Wanneer de instanties op het scherm worden getekend worden zij getekend in volgorde van diepte. De instanties met de grootste diepte worden eerst getekend. De instanties met de kleinste diepte worden het laatst getekend. Wanneer de instanties dezelfde diepte hebben, worden zij getekend in de volgorde waarin zij werden gecreëerd. Als je wilt waarborgen dat een object bovenop ligt, moet je de andere een negatieve diepte geven. Als je ervoor wilt zorgen het onder andere instanties ligt, geef het dan een grote positieve diepte. Je kunt de diepte van een instantie tijdens het spel ook veranderen door gebruik te maken van de variable depth.

Persistent objects

Ten tweede, kun je er voor zorgen dat een object permanent aanwezig blijft. Een blijvend object zal blijven bestaan wanneer het zich van de ene room naar de volgende beweegt. Het verdwijnt slechts wanneer je het uitdrukkelijk vernietigt. Zo moet je het object in de eerste room zetten en dan zal het beschikbaar zijn in alle rooms. Dit is prettig wanneer je bijvoorbeeld een hoofdfiguur hebt die zich van room naar room beweegt. Het gebruiken van blijvende objecten is een krachtig mechanisme maar ook een die gemakkelijk tot fouten leidt.

Parents

Elk object kan een ouderobject hebben. Wanneer een object een ouder heeft, erft het het gedrag van de ouder. Anders gezegd, het object is een soort speciaal geval van het ouderobject. Bijvoorbeeld, als je 4 verschillende ballen hebt, genoemd ball1, ball2, ball3 en ball4, die zich alle hetzelfde gedragen maar verschillende sprites hebben, kun je ball1 tot de ouder van andere drie

maken. Nu moet je slechts gebeurtenissen voor ball1 bepalen. Andere zullen de gebeurtenissen erven en zullen zich op precies dezelfde manier gedragen. Bovendien als je voor instanties van ball1 (de parent) acties opgeeft, worden die automatisch doorgegeven aan de andere (children). bijvoorbeeld, als je alle instanties van ball1 vernietigt zullen ook alle instanties van de andere ballen worden vernietigd. Dit bespaart heel wat werk.

Meestal zullen de objecten zich bijna hetzelfde moeten gedragen, maar er kunnen ook kleine verschillen optreden. Bijvoorbeeld, één monster kan zich op en neer kunnen bewegen en een ander van links naar rechts. Voor de rest hebben zij precies hetzelfde gedrag. In dit geval bijna zouden alle events dezelfde acties moeten hebben op één of twee na. Opnieuw kunnen wij één object tot de ouder van de andere maken. Maar in dit geval bepalen wij ook bepaalde events voor het kindobject. Deze events overrulen de ouderevents. Wanneer een event voor het kindobject acties bevat, worden deze uitgevoerd in plaats van de acties van de ouder. Als je ook de ouderevent wilt uitvoeren kan je de zogenaamde "geërfde" event oproepen door de de juiste actie te gebruiken.

In feite is het een goede gewoonte om in zo'n gevallen één basisobject te maken. Dit basisobject dient alle standaard gedrag te bezitten maar wordt echter nooit in het spel gebruikt. Alle in feite gebruikte objecten hebben dit basisobject als ouder. Ouderobjecten kunnen op hun beurt ook weer ouderobjecten bezitten, etc. (Maar je mag echter geen cyclische overerving gebruiken!) Op deze manier kun je een stamboom van overerving van objecten maken. Dit is erg nuttig om het spel gestructureerd te houden en je wordt dan ook aangespoord om dit mechanisme van overerving te leren gebruiken.

Er is ook een tweede gebruik van het ouderobject. Het erft ook het botsingsgedrag voor andere objecten. Laat me dit met een voorbeeld duidelijk maken. Veronderstel dat je vier verschillende vloerobjecten hebt. Wanneer een bal de vloer raakt moet het van richting veranderen. Dit moet in de botsingsgebeurtenis van de bal met de vloer worden bepaald. Omdat er vier verschillende vloeren zijn moet de code voor elk van de vier verschillende botsingsevents van de bal ingevuld worden. Maar wanneer je één vloer de ouder van andere drie maakt, hoef je slechts die ene botsingsevent met die ene te vloer bepalen. Door overerving zullen de andere botsingen dezelfde event uitvoeren. Dit scheelt een hoop kopieerwerk.

Zoals al eerder aangegeven zul je er rekening mee moeten houden dat als je een ouderobject gebruikt dat ook consequenties heeft voor de kindobjecten. Dat gebeurt bijvoorbeeld als je een actie wilt laten uitvoeren door instanties van een bepaald object. Maar het treedt ook op als je gebruik maakt van het `with()` statement in scriptcode (Zie verderop). Bovendien treedt het op als je gebruik maakt van functies zoals `instance_position`, `instance_number`, etc. Tenslotte werkt het ook als je verwijst naar variabelen in andere objecten. In bovengenoemd voorbeeld betekent dat als je invult `ball1`.speed = 10, dat ook geldt voor ball2, ball3 en ball4.

Masks

Wanneer twee instanties in botsing komen, treedt er een botsingsevent op. Om te bepalen of twee instanties elkaar raken worden sprites gebruikt. Dit is in de meeste gevallen prettig, maar soms wil je botsingen met instanties van een andere vorm. Bijvoorbeeld, als je een isometrisch spel maakt, hebben de objecten typisch een hoogte (om hen een 3D uiterlijk te geven). Maar voor botsingen wil je slechts de grondvorm van de sprite gebruiken. Dit kan worden bereikt door een afzonderlijke sprite te creëren die als botsingsmasker voor het object wordt gebruikt.

Informatie

De knop **Show Information** stelt je in staat om alle beschikbare informatie over een object weer te geven en te printen. Dit is vooral van belang om overzicht te houden over alle acties en events. Zeker bij complexe spellen kun je al snel dat overzicht verliezen.

Meer over acties

In de advanced mode komen een aantal extra acties voor die we hier zullen beschrijven.

Meer move actions

Enkele extra move acties zijn beschikbaar in de advanced mode. Het gaat om de volgende acties:

 **Set a path for the instance**

Met deze actie kun je aangeven dat een instantie zich volgens een bepaald pad moet bewegen. Je bepaalt het pad dat de instantie moet volgen en geeft daarbij ook de snelheid in pixels per step aan. Als de snelheid een positieve waarde heeft begint de instantie aan het begin van het pad, bij een negatieve snelheid begint hij aan het eind van het pad. Vervolgens moet je aangeven wat het eindgedrag van de instantie moet zijn, dus als hij het hele pad heeft afgelegd. Je kunt ervoor kiezen om de beweging te stoppen, om opnieuw vanaf het huidige punt te gaan bewegen (dat betekent het pad opnieuw afleggen als het pad tenminste een gesloten circuit is) of de beweging omdraaien. Tenslotte heb je de mogelijkheid om aan te geven dat het pad als iets absoluuts gezien moet worden, dat wil zegggen dat de postie van de instantie exact die is, die in het pad staat aangegeven (dit is nuttig als je het pad op een speciale plaats in de room hebt ontworpen) of als iets relatiefs hetgeen inhoudt dat het startpunt van het pad de huidige plaats van de instantie in de room is (eindpunt als de speed negatieve waarde heeft). Kijk in het hoofdstuk over paden om hierover meer te weten te komen.

 **End the path for the instance**

Gebruik deze actie om het pad voor de instantie te stoppen.

 **Set the position on the path**

Met deze actie kun je de huidige positie van de instantie op het pad wijzigen. De waarde dient tussen de 0 en de 1 te liggen (0 = beginpunt, 1 = eindpunt).

 **Set the speed for the path**

Met deze actie kun je de snelheid van de instantie op het pad wijzigen. Een negatieve snelheid zorgt ervoor dat de instantie terug gaat volgens het pad. Je kunt de snelheid op 0 zetten om de beweging tijdelijk stop te zetten.

 **Perform a step towards a point**

Deze actie moet in een step event geplaatst worden om de instantie naar een bepaalde plaats op het pad te verplaatsen. Als de instantie die plaats reeds bereikt heeft zal hij niet verder bewegen. Je kunt aangeven waarheen de instantie verplaatst moet worden (positie), met welke snelheid (de grootte van de step) en of de beweging gestopt moet worden als de instantie botst tegen een solid object of botst tegen welke instantie dan ook.

 **Step towards a point avoiding objects**

Dit is een erg krachtige bewegingsactie. Ook deze dient in een stepevent geplaatst te worden. Ook hier wordt de instantie verplaatst naar een bepaalde positie. Maar in dit geval probeert de instantie obstakels te omzeilen. Als de instantie op een solid object afgaat (of welke instantie dan ook) dan zal het zijn bewegingsrichting veranderen om zo om het object heen te bewegen. Deze benadering is niet 100% gegarandeerd, maar in de meeste gevalllen zal het werken en zal de instantie ongeschonden zijn doel bereiken. Voor nog moeilijkere gevallen bestaan er bewegingsplanningfuncties. Jij geeft aan waarheen een instantie moet bewegen en met welke snelheid (stepgrootte) en of solid objects (of welke instanties dan ook) vermeden dienen te worden.

Meer main actions

Enkele extra main acties zijn beschikbaar in de advanced mode. Het gaat om de volgende acties:

 **Set a time line**

(Alleen beschikbaar in de advanced mode.) Met deze actie bepaal je de karakteristieke tijdlijn voor een instantie van een object. Jij bepaalt de tijdlijn, de startpositie (0 is het begin). Je kunt deze actie ook gebruiken om een tijdlijn te beëindigen door te kiezen voor **No Time Line** als waarde.

 **Set the time line position**

(Alleen beschikbaar in de advanced mode.) Met deze actie kun je de positie in de huidige tijdlijn veranderen (zowel absoluut als relatief). Dit kun je gebruiken om bepaalde delen van de tijdlijn over te slaan of juist bepaalde delen te herhalen. Bijvoorbeeld, als je een tijdlijn met een looping wilt maken, op het laatste moment, dien je deze actie te gebruiken om de positie terug op 0 te zetten. Je kunt het ook gebruiken om te wachten totdat er iets speciaals gebeurt in het spel. Je hoeft alleen maar de test actie toe te voegen, indien de returnwaarde niet true is, zet dan de tijdlijnpositie relatief op -1.

 **Show a video**

Met deze actie kun je tijdens het spel een video/filmbestand afspelen. Je geeft precies aan om welke file het gaat en of die full screen getoond moet worden of in een beperkt venster. Zorg er wel voor dat de file bestaat. Je kunt de file samen met het spel verspreiden of het in een datafile zetten en het exporteren.

 **Replace a sprite from a file**

Deze actie kun je gebruiken om een bestaande sprite te vervangen door een sprite uit een bestand. Je geeft aan welke sprite je wilt vervangen, de bestandsnaam (.bmp, .jpg, or .gif) en het aantal subplaatjes in de sprite als je hem als bmp of jpg bestand laadt. Voor een gif bestand wordt het aantal subplaatjes automatisch bepaald aan de hand van het aantal subplaatjes van het te laden gif-bestand. Andere instellingen zoals wel of niet transparant, worden niet gewijzigd. Deze actie kun je gebruiken om te voorkomen dat je alle sprites moet opslaan in het programma zelf. Bijvoorbeeld, aan het begin van een level kun je de door jou gebruikte sprites vervangen door de eigenlijke sprites die je wilt gebruiken. Zorg ervoor dat je géén sprites vervangt van instanties die op dit moment ergens in de room actief aanwezig zijn. Dit kan ongewenste resultaten opleveren bij botsingen. ***Deze actie is alleen maar beschikbaar in de geregistreerde versie.***

 **Replace a sound from a file**

Met deze actie kun je een geluid vervangen door een geluidsbestand (.wav, .mid, or .mp3). Je geeft aan welk geluid vervangen dient te worden en geeft bovendien aan welk geluidsbestand geladen dient te worden. Daarmee voorkom je dat in het spel alle geluiden dienen te worden opgeslagen. Bijvoorbeeld, je kunt er op deze manier voor zorgen dat verschillende achtergrondmuziekjes kunnen worden gedraaid, je hoeft alleen maar aan te geven welk muziekje je wilt horen. Verander géén geluiden als ze worden afgespeeld. ***Deze actie is alleen maar beschikbaar in de geregistreerde versie.***

 **Replace a background from a file**

Met deze actie kun je een achtergrond vervangen door een achtergrond uit een bestand (.bmp, or .jpg). Geef de te veranderen achtergrond aan en kies ook het bestand dat deze achtergrond zal

vervangen. Daarmee voorkom je dat alle achtergronden in het programma moeten worden opgeslagen. Verander echter niet de achtergrond die op dit moment zichtbaar is. ***Deze actie is alleen maar beschikbaar in de geregistreerde versie.***

Meer control actions

Enkele extra control acties zijn beschikbaar in de advanced mode. Het gaat om de volgende acties:

 **Execute a script**

Met deze actie kun je een stuk script, dat je aan het spel hebt toegevoegd, uitvoeren. Je specificeert het script en de maximaal 5 argumenten voor het script.

Call the inherited event

Deze actie is alleen zinvol als het object een ouderobject bezit. Het verwijst naar een corresponderende event in het ouderobject.

Meer draw actions

de volgende extra draw actie is beschikbaar in de advanced mode:

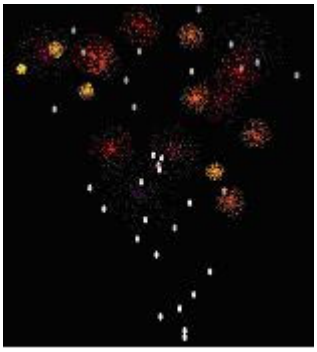
 **Set a font for drawing text**

Hiermee kun je de font instellen, welke vanaf dat moment gebruikt wordt om tekst te tekenen. Je kunt alleen maar fonts kiezen die door jou van te voren als font resources zijn gedefinieerd. Indien je kiest voor **No Font** zal standaard Arial 12pnt gebruikt worden.

Particle actions

Een set acties die te maken hebben met particles (partikels) is beschikbaar onder de **Extra** tab. ***Deze acties zijn alleen maar beschikbaar in de geregistreerde versie van Game Maker***.

Partikelsystemen zijn bedoeld om speciale effecten te onwerpen. Partikels zijn kleine elementen (voorgesteld door een pixel of een of andere kleine vorm). Zulke partikels bewegen in de room rond volgens vooropgestelde patronen en kunnen terwwijl ze bewegen van kleur veranderen. Veel van die partikels tesamen kunnnen explosies, vuurwerk, vlammen, regen, sneeuw, vallende sterren, vallend puin etc. vormen.



Game Maker bevat een uitgebreid partikelsysteem hetgeen middels functies kan worden benaderd. Een beperkter systeem kan door de volgende acties toegankelijk worden gemaakt.

Een partikelsysteem kan overweg met meerdere typen partikels. Nadat je een partikelsysteem hebt aangemaakt moet je als eerste aangeven welke typen partikels gebruikt gaan worden. Met onderstaande acties kun je 16 typen partikels specificeren. Elke type heeft een vorm, een grootte, een beginkleur en een eindkleur. De kleur verandert langzaam van de beginkleur in de eindkleur. Partikels hebben een beperkte levensduur. Partikels hebben ook een snelheid en een bewegingsrichting. Tenslotte hebben zwaartekracht en wrijving invloed op de partikels.

Nadat je de partikeltypen hebt gespecificeerd, moet je ze op bepaalde plaatsen in de room laten ontstaan. Je kunt ofwel een groot aantal partikels van een bepaald type op een plaats in de room laten ontstaan ofwel je kunt met regelmatige snelheid een aantal partikels in de room laten verschijnen. Partikels komen als het ware uit speciale kokers (emitters). Het partikelsysteem kan maximaal 8 van deze kokers tegelijk aan het werk hebben. Dus na het maken van de partikeltypen moet je ook nog de emitters ontwerpen en ze aangeven of ze de partikels ineens naar buiten laten komen of geleidelijk in een stroom.

Hieronder vind je de complete set acties. Je zult ermee moeten experimenteren om het gewenste effect te bereiken.

 **Create the particle system**

Deze actie creëert het partikelsysteem. Het moet worden aangeroepen voordat enige andere actie kan worden gebruikt. Je hoeft het slechts eenmaal aan te roepen. Je kunt opgeven op welke diepte de partikels worden getekend. Als je gebruik maakt van een zeer grote positieve diepte, zullen de partikels achter de instanties verschijnen. Bij een grote negatieve diepte zullen ze juist vóór de instanties verschijnen.

 **Destroy the particle system**

Deze actie vernietigt het partikelsysteem, waardoor al het geheugen vrijkomt dat het in beslag

neemt. Vergeet deze actie niet te gebruiken als je bijvoorbeeld naar een andere room gaat omdat partikelsystemen behoorlijk veel geheugenruimte in beslag nemen.

 **Clear all particles in the system**

Deze actie zorgt ervoor dat alle zichtbare partikels worden verwijderd uit de room. Deze actie heeft geen effect op het stoppen van de emitters als je gebruik maakt van emitters die de partikels geleidelijk gedurende langere tijd in de room verspreiden, hetgeen betekent dat na het verwijderen van de huidige partikels er meteen weer nieuwe in de room verschijnen (zie verderop).

 **Create a type of particle**

Met deze actie ontwerp je een partikeltype. Je hebt de keuze uit 16 verschillende typen. Je hebt verder de mogelijkheid om zijn vorm te kiezen, de minimale en maximale grootte (als de partikels in de room verschijnen hebben ze een random grootte tussen minimale en maximale grootte), de beginkleur en de kleur waarin hij gedurende zijn bestaan zal veranderen. Houd er rekening mee dat met deze actie alleen het type wordt gemaakt en niet de eigenlijke partikels. Daarvoor heb je de emitters nodig (zie verderop).

 **Set the life time for a particle type**

De levensduur van een partikel bestaat uit een beperkt aantal steps. Daarna verdwijnt hij uit beeld. Met deze actie kun je de levensduur van een partikeltype instellen. Je geeft twee waarden op. De eigenlijke levensduur van een partikel wordt random gekozen tussen die twee waarden in.

 **Set the motion for a particle type**

Met deze actie kun je de snelheid en de bewegingsrichting van een partikeltype bepalen. Ook nu weer geef je een minimum en maximumwaarde op en de eigenlijke waarde zal random gekozen worden tussen die twee waarden in. Bijvoorbeeld, om een partikel in een random richting te laten bewegen geef je als minimum- en maximumwaarde de getalle 0 en 360 op. Je kunt ook wrijving specificeren. Deze waarde wordt iedere keer van de huidige waarde afgetrokken waardoor de snelheid steeds lager wordt totdat hij uiteindelijk 0 wordt. (Je kunt natuurlijk een deeltje ook steeds sneller laten bewegen door een negatieve waarde bij wrijving in te vullen.)

 **Set the gravity of a particle type**

Met deze actie stel je de hoeveelheid en de richting van de zwaartekracht in voor een bepaald type partikel. 270 betekent een zwaartekracht omlaag.

 **Create secondary particles**

Deze actie is een beetje gecompkiceerd. Partikels zijn in staat om nieuwe partikels tijdens hun bestaan of aan het eind van hun bestaan te maken. Met deze actie kun je dat instellen. Je kunt aangen welk type partikel en hoeveel ervan gemaakt moet worden bij elke stap gedurende zijn

bestaan en als hij aan zijn eind komt. Wees hier erg voorzichtig mee. Je kunt op deze manier een enorme hoeveelheid partikels laten ontstaan waardoor het systeem aanzienlijk kan worden vertraagd. Voor het aantal kun je ook negatieve waarden invullen. Een negatieve waarde x betekent dat bij elke step de kans dat een nieuw partikel ontstaat -1/x is. Dus als je na iedere 4 steps een tweede partikel wilt laten onstaan, moet je als waarde -4 invullen. Het onstaan van secundaire partikels is vooral prima om effecten als rijen partikels of exploderende partikels te verkrijgen.

 **Create a particle emitter**

Deze actie zorgt ervoor dat een partikel-emitter kan ontstaan. Partikels worden namelijk door emitters gemaakt. Je kunt 8 emitters tegelijk in werking hebben. Kies de emitter, bepaal zijn vorm, grootte en positie (in de vorm van een bounding box).

 **Destroy an emitter**

Deze actie vernietigt de aangegeven emitter. Houd er wel rekening me dat de partikels die door deze emitter zijn uitgestoten niet worden verwijderd door deze actie.

 **Burst a number of particles from an emitter**

Zelfs al heb je een partikeltype en een emitter bepaald, dan nog zijn er geen partikels. Je moet de emitter de opdracht geven om partikels te maken. Met deze actie geef je de emitter de opdracht om een bepaald aantal partikels van een bepaald type te produceren. Al de partikels worden tegelijkertijd gevormd. Je kunt ook een negatief getal als waarde opgeven. Een negatieve waarde x betekent dat er een partikel wordt gemaakt met een kans van -1/x. Als je dus bijvoorbeeld een partikel wilt laten ontstaan met een kans van 25 procent, kies dan de waarde -4.

 **Stream particles from an emitter**

Met deze actie geef je de emitter de opdracht om een bepaald aantal partikels geleidelijk in de ruimte te brengen. Er komt daardoor een stroom partikels op gang. Deze stroom gaat zolang door totdat je de opdracht geeft de stroom partikels te stoppen of de emitter te vernietigen. Als aantal kun je ook een negatief getal opgeven. een negatieve waarde betekent dat er in iedere step de kans dat een partikel wordt gemaakt -1/x is. Dus als je bijvoorbeeld om de 4 steps een partikel wilt laten ontstaan, moet je de waarde -4 gebruiken.

Extra actions

In de **Extra** tab komen ook een aantal acties voor die te maken hebben met het afspelen van CD's. ***Deze acties zijn alleen maar beschikbaar in de geregistreerde versie van Game Maker***.

 **Play a CD**

Met deze actie ben je in staat om enkele tracks van een CD in de standaard CDROM-drive af te spelen. Je geeft daarbij de begin- en de eindtrack aan.

 **Stop the CD**

Deze actie stopt het spelen van de CD die nu wordt afgespeeld.

 **Pause the CD**

Met deze actie wordt het afspelen van het huidige nummer van de CD onderbroken.

 **Resume the CD**

Met deze actie gaat het afspelen van de CD verder na een pauze.

 **If a CD exists in the drive**

Als zich een CD in de CDROM-drive bevindt, wordt de volgende actie uitgevoerd.

 **If the CD is playing**

Als er op dit moment een CD in de standaard CDROM-drive wordt afgespeeld, wordt de volgende actie uitgevoerd.

Tenslotte zijn er twee extra acties die in bepaalde spellen van pas kunnen komen.

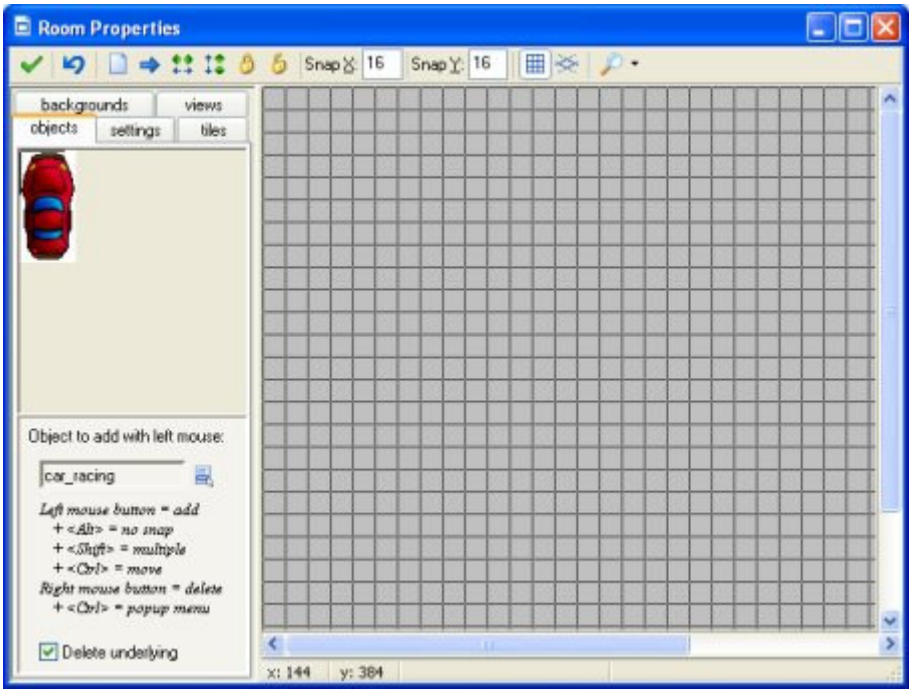
 **Set the mouse cursor**

Deze actie is te gebruiken om de windows muiscursor te vervangen door een sprite. Je bepaalt de sprite en geeft verder aan of de windows muiscursor nog getoond moet worden. De sprite mag een animated gif zijn. Houd er wel rekening mee dat de sprite alleen maar te zien is in de room, dus niet er buiten!

 **Open a webpage in a browser** Met deze actie kun je een webadres aangeven. De webpagina wordt dan in de standaard browser op de computer geopend. (Deze actie kun je overigens ook gebruiken om andere documenten te openen.) Deze actie werkt niet in de secure mode.

Meer over rooms

Rooms hebben in *Game Maker* vele opties. Eerder hebben we alleen de aller belangsrijkste behandeld. In dit hoofdstuk bespreken we de andere opties. Als je het room venster in de advanced mode opent, ziet het er als volgt uit:



Zoals je kunt zien zijn er een aantal nieuwe knoppen op de werkbalk te zien. Er zijn knoppen om de instantie zowel horizontaal als vertikaal te sorteren. Dit kan nuttig zijn als instanties elkaar gedeeltelijk overlappen. (Bij het werken met tiles komen deze knoppen goed van pas.) Er zijn ook knoppen waarmee je alle instanties kunt vergrendelen of ontgrendelen. Vergrendelde instanties kunen niet verschoven of verwijderd worden. Dit voorkomt dat er per ongeluk instanties verdwijnen. Door gebruik te maken van het menu dat verschijnt door met de rechtermuisknop te klikken (houd de <Ctrl> toets ingedrukt en klik met de rechtermuisknop op een instantie) kun je ook individuele instantie vergrendelen en ontgrendelen.

Tenslotte kun je aangeven dat je gebruik wilt maken van een isometrische grid. Dit is vooral erg nuttig als je isometrische spellen wilt maken. Allereerst lopen de gridlijnen diagonaal in de room. Daarnaast is ook de snapping van de instanties anders. (Het werkt het beste als de oorsprong van de instantie zich in de linker bovenhoek bevindt, hetgeen standaard het geval is.) Finally, you can indicate that you want to use an isometric grid. This is very useful when creating isometric games. First of all, the grid lines now run diagonally. Also the snapping of instances is different. (It works best when the origin of the instance is at the top left corner as is default.)

Er zijn ook twee nieuwe tabbladen die we hieronder zullen bespreken.

Advanced settings

Er zijn nog 2 aspecten in het tabblad **settings** die we nog niet hebben besproken. Op de eerste plaats is er een keuzeveld **Persistent**. Normaal gesproken is het zo dat als je een room verlaat en er in een later stadium weer terug komt, de oorspronkelijke instellingen van die room zijn te zien.

Dit is aardig als je meerdere levels hebt, maar niet echt handig als je een RPG hebt gemaakt. In dat soort spellen is het te doen gebruikelijk dat je de room weer aantreft zoals je hem een tijd geleden hebt verlaten. Als je nu een vinkje plaatst in de box **Persistent** gebeurt dat ook. De roomstatus wordt in het geheugen opgeslagen waardoor de room er exact hetzelfde uitziet als je er weer terugkomt. Alleen bij het herstarten van het spel worden de oorspronkelijke instellingen geladen. Er bestaat echter een uitzondering hierop. Indien je sommige objecten persistent hebt gemaakt, zullen de instanties ervan niet in de room achterblijven maar met de hoofdpersoon meegaan naar een volgende room.

Vervolgens is er ook nog een knop **Creation code**. Die geeft je de mogelijkheid om in een veld een stuk scriptcode in GML (zie deel 4) in te typen dat bij het onstaan van de room wordt uitgevoerd. Dit kan nuttig zijn om enkele roomvariabelen in te vullen, bepaalde instanties te laten ontstaan, etc. Het is belangrijk dat je begrijpt wat er precies gebeurt als je in een bepaalde ruimte van het spel komt.

- Allereerst krijgen de instanties in de huidige room (als die bestaat) een zogenaamde room-end event. Vervolgens worden de niet-persistente instanties verwijderd (er wordt geen destroy event gegenereerd!).
- Daarna worden de persistente instyanties uit de vorige room aan de nieuwe room toegevoegd.
- Alle nieuwe instanties worden gecreëerd en hun creation events worden uitgevoerd (als de room niet persitent is of nog niet van te voren is bezocht).
- Als het gaat om de eerste room wordt voor alle instanties de game-start event gegenereerd.
- Daarna wordt de room creation code uitgevoerd.
- Tenslotte krijgen alle instanties een room-start event.

Op die manier, bijvoorbeeld, kan door de room-start events variabelen gebruikt worden die in de room creation code zijn geplaatst en je kunt in de creation code ook verwijzen naar instanties (zowel nieuwe als persistente) in de room.

Verder is er nog een optie. In het pop-up menu, als je met je rechtermuisknop klikt op een instantie terwijl je de <Ctrl> toets hebt ingedrukt, kun je een creation code opgeven voor een bepaalde instantie. Deze code wordt uitgevoerd als de room start, net vóórdat de creation event van die instantie wordt uitgevoerd. Dit is erg handig omdat je op die manier speciale parameters aan een bepaalde instantie kunt meegeven, terwijl andere die niet hebben.

Het toevoegen van tiles

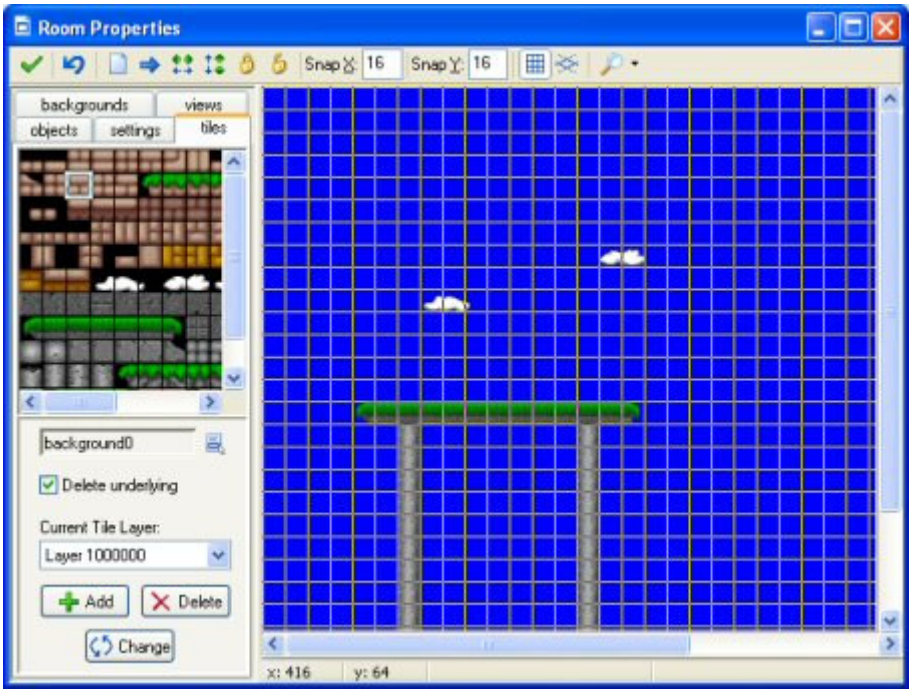
Je kunt ook een zogenaamde "betegelde" achtergrond maken (een achtergrond met tiles). De reden daarvoor is als volgt: in de meeste spellen wil je achtergronden die er fraai uitzien. In een doolhofspel bijvoorbeeld wil je dat de muren er ook als muren uitzien en in platformspellen wil je prachtig getekende achtergronden met bomen etc. In *Game Maker* is dat allemaal mogelijk door verschillende objecten te maken en je room uit deze objecten op te bouwen. Maar het probleem is dat dat allemaal erg veel werk is, een grote hoeveelheid resources in beslag neemt, hetgeen betekent dat het spel al heel gauw erg traag wordt door het gebruik van zoveel verschillende objecten. Om bijvoorbeeld een aardig uitziende muur voor een doolhofspel te maken heb je als je start al ongeveer 15 verschillende gevormde muurobjecten nodig.

De standaard manier om dit te bereiken, zoals dat in veel spellen gebruikelijk is, is dat de muren en andere statische objecten in feite op de achtergrond worden getekend. Maar, zul je je afvragen, hoe weet het spel nu dat een object tegen een muur botst als die muur alleen maar getekend is op de achtergrond? Daar is een truc op bedacht. Je gebruikt slechts één muurbject in je spel. Als je de room gaat maken plaats je die objecten daar waar op de achtergrond de muren zijn getekend en vervolgens maken we die objecten onzichtbaar. Als je nu het spel gaat spelen, zie je alleen maar de getekende achtergrond, de muurobjecten zijn niet te zien, maar tijdens het spel zal er wel op die onzichtbare objecten gereageerd worden, bijvoorbeeld dat als je tegen de muur oploopt je niet verder kunt en om de muur heen moet.

Deze methode kun je overal voor alle objecten toepassen die niet van vorm of positie veranderen. (Je kunt ze niet gebruiken voor animated gif's.) Voor plaformspellen heb je waarschijnlijk één vloerobject en één muurobject nodig terwijl het eruitziet alsof je op gras, boomtakken etc. loopt.

Om tiles aan je achtergrond toe te voegen moet je eerst een achtergrond resource aan je spel toevoegen die de tiles bevatten. Als je wilt dat je tiles gedeeltelijk tranparant zijn moet je er wel voor zorgen dat achtergrondafbeelding tranparant is. Als je vervolgens de achtergrond resource aan je spel toevoegt moet je aangeven dat het gebruikt wordt als een tile set. Vervolgens geef je de grootte op voor elke tile en of er wel of geen ruimte tussen de tiles gelaten moet worden zoals dat besproken is in het hoofdstuk over de achtergrond resources.

Klik nu om je room te maken op de tab **tiles**. Het volgende venster wordt weergegeven (in feite hebben we al enige tiles in de room geplaatst).



Links boven is de set met tiles afgebeeld. Om de set te selecteren moet je op de menuknop beneden klikken om de juiste achtergrondafbeelding te krijgen.

Nu kun je tiles toevoegen aan de achtergrond door de tile te selecteren aan de linker bovenkant en deze op de juiste plaats in de room neer te zetten door te klikken met je linkermuisknop. Het werkt dus op precies dezelfde wijze als je instanties aan de room toevoegt. Standaard worden onderliggende tiles verwijderd tenzij je het (standaard) vinkje hebt verwijderd in de box **Delete underlying**. Met de rechtermuisknop kun je tiles uit de room verwijderen. Als je de <Shift> toets ingedrukt houdt kun je meerdere tiles toevoegen. Als je de <Ctrl> toets ingedrukt houdt, kun je tiles naar een nieuwe plaats verslepen. Het gebruik van de <Alt> toets voorkomt dat de tiles niet goed in de grid geplaatst worden. Ook is er een pop-up menu als je de <Ctrl> toets ingedrukt houdt en op een tile met de rechtermuisknop klikt. Met de knoppen in de werkbalk kun je alle tiles wissen, verplaatsen, sorteren vergrendelen of ontgrendelen. (Dat functioneert alleen maar in de huidige laag; zie onder.)

Soms wil je een gedeelte van de achtergrond in de room hebben dat niet precies de grootte heeft van een tile. Dat kun je als volgt voor elkaar krijgen. Klik met de linkermuisknop linksboven in de afbeelding terwijl je de <Alt> toest ingedrukt houdt. Nu is het mogelijk om een gebied van variabele grootte te selecteren en dat op de zelfde manier als een tile in de room plaatsen.

Tiles kunnen in lagen op verschillende diepten geplaatst worden. Op de bodem zie je de huidige diepte. Standaard staat die op 1000000 ingesteld hetgeen betekent dat normaal gesproken dat achter alle instanties is. Dat betekent dat de instanties vóór de tiles bewegen. Je kunt gebruik meken van de **Add** knop om nieuwe tile lagen aan te maken, elk met een verschillende diepte.

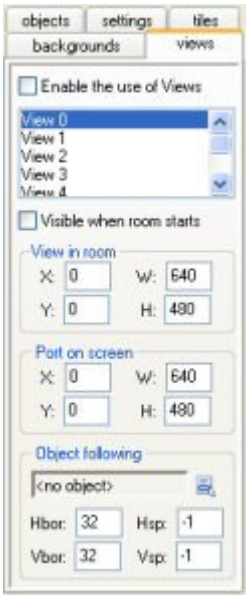
Negatieve dieptewaarden kun je gebruiken om tiles vóór de instanties te plaatsen. Als je bovendien ook nog de instanties verschillende diepten meegeeft, kun je ze tussen verschillende tile lagen door laten gaan. Als je op de **Delete** toets klikt kun je een laag verwijderen met al de aanwezige tiles erin. (Er moet altijd minimaal 1 laag aanwezig zijn.) Als je op **Change** klikt, kun je de diepte van een tile laag veranderen. Als je de diepte hetzelfde maakt als een andere laag zullen de twee lagen worden gemengd.

Het gebruik van tiles is een krachtig instrument dat zoveel mogelijk moet worden toegepast. Het is vele malen sneller dan het gebruik van objecten en de tile afbeeldingen worden slechts éénmaal opgeslagen. Op die manier kun je zeer grote rooms met tiles gebruiken met een minimum aan geheugenverbruik.

Views

Tenslotte is er ook nog een tab **views**. Dat geeft je de mogelijkheid om verschillende delen van je room op verschillende plaatsen op je scherm af te beelden. Er bestaan vele manieren om views te gebruiken. Allereerst kan het voorkomen dat je in een aantal spellen alleen maar een klein gedeelte van de room op elk moment wilt laten zien. Bijvoorbeeld bij de meeste platformspellen volgt de view de hoofdpersoon in het spel. Bij een spel dat je met zijn tweeën speelt wil je vaak het scherm opsplitsen waarbij je in het ene deel de ene speler bezig ziet en in het andere deel de andere speler. een derde manier is dat een deel van de room moet scrollen (bijvoorbeeld dat deel met de hoofdpersoon) terwijl een ander deel statisch is (bijvoorbeeld een status panel) Dan kan op eenvoudige wijze in *Game Maker* gerealiseerd worden.

Als je op de tab **views** klikt, verschijnt het volgende informatievenster:



Bovenaan bevindt zich de box **Enable the use of Views**. Als het vinkje is geplaatst wordt het gebruik van views ingeschakeld. Daaronder zie je een lijst van minstens 8 views die je kunt definiëren. Onder de lijst kun je informatie over de views opgeven. Allereerst moet je aangeven of de view te zien moet zijn als de room wordt gestart. Zorg ervoor dat er minstens één view zichtbaar is. De views die zichtbaar zijn, zijn vetgedrukt.

Een view wordt als een rechthoek in de room gedefinieerd. Dat is het gebied dat in de view moet worden getoond. De geeft de plaats aan van de linker bovenhoek en de hoogte en de breedte van het gebied. Vervolgens moet je aangeven waar dit gebied op het scherm moet worden getoond. Dit wordt de (view)port genoemd. Opnieuw geef je de plaats en de grootte op. Als je gebruik maakt van een single view is de plaats (0,0). Merk op dat de grootte van de port anders kan zijn dan de grootte van de view. In dat geval zal de grootte van de view aangepast worden aan de grootte van de port. (met behulp van scriptcode is het ook mogelijk de view te roteren.) De ports kunnen elkaar overlappen. In dat geval worden ze weergegeven in de volgorde die je hebt ingesteld allemaal boven elkaar.

Zoals hierboven is aangegeven wil je vaker een view gebruiken om een bepaald object te volgen. Dat object kun je op de bodem aangeven. Als er meerdere instanties van dit object aanwezig zijn wordt alleen de eerste door een view gevolgd. (in scriptcode is het ook mogelijk om een bepaalde instantie van een object te volgen in een view.) Normaliter moet het mogelijk zijn dat een karakter een beetje ronddwaalt in de room zonder dat de view verandert. Alleen als het karakter vlak in de buurt van de grenzen van de view komt, moet de view veranderd worden. Je kunt aangeven hoeveel ruimte rondom het object zichtbaar moet blijven. Tenslotte kun je ook de snelheid waarmee de views veranderen aangeven. Dat kan betekenen dat het lijkt alsof het karakter uit beeld gaat lopen, maar het geeft een veel natuurlijkere manier van verandering in het spel. Als je de view in een keer wilt wijzigen moet je de waarde -1 gebruiken.

Fonts

Als je tekst wilt tekenen in je spel gebeurt dat standaard met het font Arial 12 pnt. Maar ik kan me voorstellen dat je leukere fonts wilt gebruiken voor je teksten. Om verschillende fonts te kunnen gebruiken moet je eerst font resources aanmaken. In elke font resource geef je aan welk type font daarvoor gebruikt moet worden. Door middel van acties om fonts te gebruiken kun je in je spel verschillende fonts toepassen. action to set a font.

Om een font resource aan je spel toe te voegen moet je gebruik maken van de optie **Add Font** in het **Add** menu of van de juiste knop op de werkbalk. Het volgende venster verschijnt in beeld:

Zoals altijd moet je ook aan de font resource een naam geven. Vervolgens kun je het juiste font uit de lijst halen. Verder kun je de grootte aanpassen en aangeven of de letters vetgedrukt en/of cursief moeten zijn. Houd er rekening mee dat grote letters veel geheugenruimte in beslag nemen. Daarom is het verstandig om geen letters te gebruiken met een fontgrootte graoter dan pakweg 32. (Het is mogelijk om de grootte te veranderen tijdens het spelen.) Een voorbeeld van het aangegeven font is aan de onderkant van het venster te zien.

Een font bestaat uit 256 karakters, genummerd van 0 t/m 255. Maar in het algemeen gebruik je maar een klein gedeelte van deze karakterset. Vandaar dat standaard aleen maar de karakters met de nummers 32 t/m 127 opgeslagen zijn in de font. Hoe meer karakters je opslaat in de font, hoe meer geheugenruimte er in beslag wordt genomen. Je kunt de range aan karakters wijzigen. Om te zien welke karakters er in een set voorkomen en wat de index ervan is, kun je gebruik maken van de Character Map die je kunt vinden in het Windows Start menu onder Accessoires/System Tools. Sommige standaard ranges kunnen worden aangegeven door gebruik te maken van de knoppen: **Normal** (range 32 t/m 127), **All** (range 0 t/m 255), **Digits** (alleen de 10 cijfers) en **Letters** (alleen de hoofd- en kleine letters). Er kunnen ook andere ranges worden aangegeven door de eerste en de laatste index aan te geven. Als het karakter niet in de range ligt, zal het worden weergegeven door een spatie.

Normaal gesproken heb je een groot aantal fonts op je computer geïnstalleerd. Er bestaan websites waar je nog eens honderden fonts kunt downloaden. De kans bestaat echter dat als jij voor je spel één of meerdere bijzondere fonts hebt gebruikt die fonts bij iemand anders niet op de computer zijn geïnstalleerd. Om dit te vermijden zorgt *Game Maker* ervoor dat alle gebruikte fonts die je in het spel gebruikt in de game-file worden opgeslagen. Deze optie is overigens alleen beschikbaar als je een stand-alone executable file maakt. Dus iemand die jouw stand-alone files krijgt hoeft dus die fonts niet op zijn computer te hebben staan. Maar als je iemand een editable file geeft met allerlei

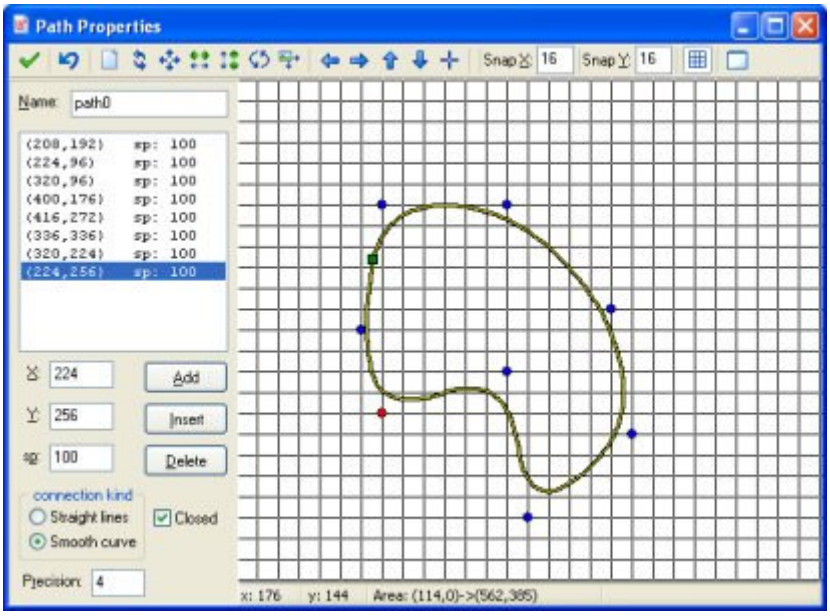
aardige fonts erin verwerkt, kun je beter die persoon ook voorzien van die fonts, om er zo mee te kunnen werken.

Paden (paths)

In geavanceerdere spellen wil je dat instanties bepaalde routes afleggen in de rooms. Hoewel je dit kunt bewerkstelligen door gebruik te maken van bijvoorbeeld timer events of scriptcode, blijft het toch een tamelijk ingewikkelde aangelegenheid. Het gebruik van path resources maakt het echter een stuk eenvoudiger. Het idee is tamelijk simpel. Je definieert een path door het gewoon te tekenen. Vervolgens plaats je een actie in de creation event van het object die het object vertelt die bepaalde route te volgen. In dit hoofdstuk wordt dat in detail uitgelegd.

Het definiëren van paden

Om een pad aan je spel toe te voegen moet je kiezen voor de optie **Add Path** uit het **Add** menu. Het volgende venster verschijnt dan op je scherm (in dit geval hebben we al een klein pad getekend).



Links boven in het scherm kun je, zoals gebruikelijk, de padnaam opgeven. Daaronder staan de punten waarmee je het pad kunt definiëren. Elk punt heeft zowel een positie als een snelheid (aangegeven met sp). Afhankelijk van hoe je het pad in het spel gebruikt kan de positie absoluut zijn, dat betekent dat de instantie waarvoor dit pad bedoeld is later exact op die plaats in de room zijn weg vervolgt, of relatief, hetgeen betekent dat een instantie altijd aan het begin van het pad begint en daarna zijn weg vervolgt. De snelheid moet als volgt worden opgevat. Als een instantie het pad volgt is de standaard snelheid 100. Een lagere waarde betekent een vermindering van zijn

snelheid terwijl een hogere waarde leidt tot een toename van de standaard snelheid. (Dus het geeft een procentuele waarde aan van de feitelijke snelheid.) Snelheid zal worden geïnterpoleerd tussen de verschillende punten waardoor de snelheid gelijkmatig verandert.

Om een punt aan het pad toe te voegen moet je klikken op de knop **Add**. Er wordt dan een kopie gemaakt van het geselecteerde punt. Nu kun je de feitelijke positie en snelheid wijzigen door de waarden in de daarvvor bestemde invoervelden te veranderen. Elke keer als je een punt in de lijst selecteert kun je de waarden ervan wijzigen. Klik op **Insert** om een nieuw punt vóór het huidige punt in te voegen. en klik op **Delete** om het huidige punt te verwijderen.

Aan de rechterkant van het venster zie je het feitelijke pad. De rode stip geeft het geselecteerde controlepunt aan. De blauwe stippen geeven de andere controlepunten aan. Het groene vlak geeft het startpunt van het pad aan. Je kunt het pad ook wijzigen door gebruik te maken van de muis. Klik waar dan ook met de linkermuisknop op de afbeelding om een punt toe te voegen. Klik op een bestaand punt, houd de linkermuisknop ingedrukt en sleep het punt waarheen je wil. Daardoor zal het pad verander worden. Als je de <Shift> toets ingedrukt houdt terwijl je op een punt klikt, voeg je een extra punt toe. Tenslotte kun je met de rechtermuisknop punten uit het pad verwijderen door erop te klikken. (houd er wel rekening mee dat je op deze manier niet de snelheid kunt wijzigen.) Normaliter worden de punten op de hoekpunten van de grid weergegeven. Je kunt de gridinstellingen veranderen met behulp van de werkbalk aan de bovenkant. Daar kun je ook aangeven of de grid wel of niet zichtbaar moet zijn. Als je een punt precies wilt positioneren moet je de <Alt> toets ingedrukt houden als je een punt toevoegt of van plaats verandert.

Je kunt de vorm van het pad op twee manieren beïnvloeden. Allereerst kun je het type verbinding gebruiken. Daarbij heb je de keuze uit rechtlijnige verbindingen of vloeiende lijnen. Daarnaast kun je aangeven of het pad gesloten moet zijn of niet.

Op de werkbalk vind je een aantal belangrijke controls. Van links naar rechts hebben ze de volgende betekenis. De eerste knop geeft aan dat je klaar bent en het venster wilt sluiten waarbij de wijzigingen worden opgeslagen. (Als je de wijzigingen niet wilt opslaan moet je op het kruisje rechtsboven in het venster klikken, waardoor het venster sluit zonder opslag van de wijzigingen.) Daarnaast bevindt zich de knop waarmee je de laatste wijziging ongedaan kunt maken.

Het volgende stel knoppen stelt je in staat om het pad te wissen, om de volgorde waarin het pad wordt afgelopen om te draaien, het pad te verplaatsen, het pad horizontaal te spiegelen, het pad verticaal om te draaien, het pad te draaien onder een bepaalde hoek en om het te vergroten of te verkleinen. Daarnaast zijn er knoppen om de view te verplaatsen (niet het pad zelf; het feiteljke viewgebied wordt aangegeven beneden in de statusbalk) en om de view te centreren.

Zoals al eerder boven is aangegeven kun je de waarden aangeven waar de punten moeten komen en of de grid zichtbaar moet zijn. Tenslotte is er een knop die je in staat stelt om een speciale room als achtergrond te laten zien voor het pad. Door hiervan gebruik te maken kun je eenvoudig het pad op een bepaalde plaats in de room plaatsen, bijvoorbeeld op een racecircuit, waardoor later de instanties de juiste route kunnen volgen. (dit heeft alleen maar zin als je gebruik maakt van absolute paden; zie onder.)

Het toekennen van paden aan objecten

Om een pad toe te kennen aan een instantie van een object, kun je de path actie in enig event plaatsen, bijvoorbeeld in de creation event. In deze actie moet je het pad specificeren uit het drop down menu. Daar kun je ook nog enkele andere waarden aangeven.

Je moet het pad aangeven dat gevolgd dient te worden en de snelheid in pixels per step. Als de snelheid een positieve waarde heeft, start de instantie aan het begin van het pad. Als de waarde negatief is, start de instantie aan het eind. Wees je er wel van bewust dat toen je het pad definieerde je de feitelijke snelheid relatief specificeerde aan deze opgegeven snelheid. Er bestaat ook een action om de snelheid waarmee het pad wordt afgelopen kan worden gewijzigd. Je kunt dit bijvoorbeeld gebruiken om een instantie sneller of langzamer te laten lopen op het pad. Merk daarbij op dat dan de normale snelheid van de instantie wordt ontkent (in feite zet je die op 0) als het pad wordt uitgevoerd. Zaken als wrijving en zwaartekracht hebben ook geen invloed op de beweging van een instantie als die zich beweegt op zijn pad.

Vervolgens moet je het gedrag van de instantie aan het eind van het pad aangeven, met andere woorden, wat moet er precies gebeuren als het eind van het pad bereikt is? Je kunt ervoor kiezen om de beweging op het eind van het pad te stoppen. Je kunt er ook voor kiezen om het pad opnieuw te laten aflopen, hetgeen betekent dat de instantie terugspringt naar de beginpositie van het pad en opnieuw zijn weg vervolgt. Een derde optie is het pad vanaf deze positie opnieuw te doorlopen, waarbij het eindpunt van de voorige doorloop het beginpunt is van de nieuwe doorloop van het pad (dat is hetzelfde als het pad gesloten is). Tenslotte kun je ervoor kiezen om de beweging om te draaien en het pad in omgekeerde volgorde te laten aflopen. Houd er rekening mee dat aan het eind van een pad een event optreedt; zie onder.

Tenslotte kun je aangeven of het pad absoluut of relatief moet zijn. een absoluut pad wordt uitgevoerd op de plek waar het is gedefinieerd. De instantie wordt op het beginpunt geplaatst en vervolgt zijn weg naar het eindpunt (bij een positieve snelheid). Dit is bijvoorbeeld nuttig als je een bepaald racecircuit hebt waarop je een pad hebt gemaakt. Wanneer je kiest voor en relatief pad zal een instantie vanaf zijn huidige positie het pad volgen. Dit is nuttig wanneer een instantie een beweging ter plekke moet maken. Bijvoorbeeld, ruimteschepen in het space invader spel kunnen

een bepaalde draaibeweging maken op hun huidige positie om bijvoorbeeld vijanden te kunnen ontwijken.

Als je een instantie op een andere plek op het pad wilt neerzetten kun je gebruik maken van actie om de path position in te stellen. Een path position ligt altijd tussen 0 en 1, waarbij 0 betekent op de startpositie en de 1 de eindpositie van het pad. Merk op dat in elke stap de richtingsvariabele automatisch wordt aangepast aan de positie waarop de instantie zich bevindt op het pad. Je kunt deze variabele gebruiken om de correcte oriëntatie van de sprite te kiezen.

Als je scripts of kleine stukjes code gebruikt heb je meer controle over de manier waarop het pad wordt uitgevoerd. Er bestaat een functie om het pad van een instantie te starten. De variabele `path_position` geeft de huidige positie aan op het pad (tussen 0 en 1, zoals boven is vermeld). De variabele `path_speed` geeft de snelheid langs het pad weer. Een variabele `path_scale` kun je gebruiken om de grootte van het pad te bepalen. De waarde 1 betreft de oorspronkelijke grootte. Een grotere waarde betekent dat het pad is vergroot; een kleinere waarde maakt het kleiner. De variabele `path_orientation` geeft de richting aan waarin het pad dient te worden uitgevoerd (in graden tegen de wijzers van de klok in). Dit stelt je in staat om het pad in verschillende richtingen uit te voeren (bijvoorbeeld van onder naar boven of van links naar rechts). Er is ook een variabele die het eingedrag controleert. Tenslotte zijn er een groot aantal functies die vragen naar eigenschappen van het pad (bijvoorbeeld de x- en y-coördinaten van bepaalde posities) en er bestaan functies om paden te maken. Er bestaan zelfs functies om botsingsvrije paden te maken voor een instantie om een bepaald doel te bereiken. Voor details moet je kijken in deel 4 waarin de taal GML uitvoerig aan bod komt.

Je zult je wel afvragen wat er gebeurt als een instantie botst tegen een andere instantie terwijl hij zijn pad bewandelt. In feite gebeurt hetzelfde als had die instantie een normale snelheid. Als het om een solid object gaat zal dit worden teruggeplaatst. Als beide instanties niet solid zijn zullen ze op hun nieuwe posities worden geplaatst. Vervolgens worden de collision event(s) uitgevoerd en wordt er nagegaan of de botsing is opgelost. Zo niet en de andere instantie is solid, dan zal de instantie stoppen, zoals het hoort (aannemende dat er een collision event gedefinieerd is). Bovendien wordt de `path_position` variabele niet opgehoogd. Als de instantie die de weg had geblokkerd verdwene is, zal de instantie zijn weg op het pad weer vervolgen. Om zelf collisions te kunnen afhandelen kan de variabele `path_positionprevious` handig zijn. Het onthoudt de vorige positie in het pad en je kunt de padpositie instellen met behulp van deze variabele om te voorkomen dat je zomaar verder gaat op het pad.

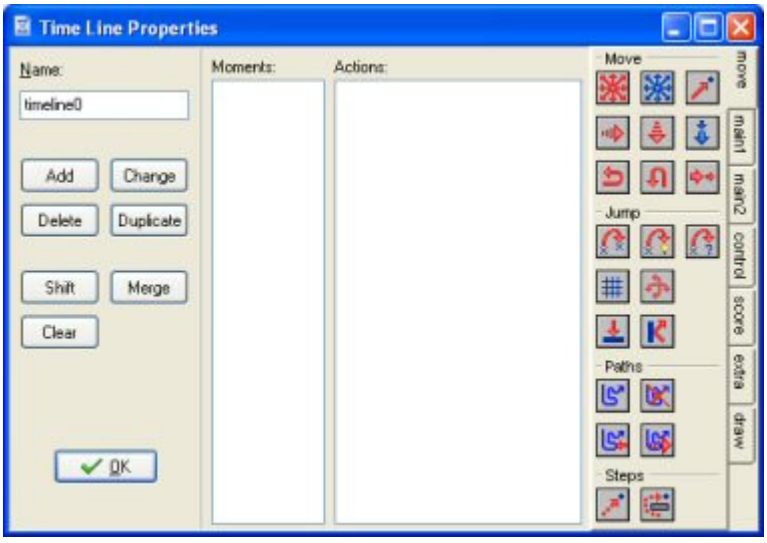
The path event

Zoals al boven is beschreven kun je aangeven wat er moet gebeuren als een instantie aan het eind van het pad is gekomen. Op dat moment treedt er een **End of Path** event op. Die kun je vinden in de tab **Other** events. Hier kun je acties in plaatsen. Je wilt bijvoorbeeld aan het eind de instantie laten verdwijnen of juist die instantie aan een nieuw (ander) pad laten beginnen.

Tijdlijnen (Time Lines)

In veel spellen moeten bepaalde dingen gebeuren op bepaalde tijden. Je kunt dit doen door gebruik te maken van alarm events maar als het geheel te gecompliceerd wordt, werkt dat ook niet meer. Hiervoor is nu juist de time line resource bedoeld. In een tijdlijn geef je precies aan wanneer wat moet gebeuren. Je kunt daarvoor ook alle acties gebruiken die je in verschillende events kunt toepassen. Als je eenmaal een tijdlijn hebt gemaakt kun je die koppelen aan en instantie van een object. Deze instantie zal dan precies op tijd die acties uitvoeren zoals dat in de tijdlijn is vastgelegd. Laten we eens een voorbeeld bekijken. Stel je wilt een bewaker maken. Deze bewaker moet eerst 20 steps naar links uitvoeren, daarna 10 naar boven, 20 naar rechts, 10 naar beneden en daarna stoppen. Om dat te bereiken, maak je een tijdlijn, waarin je aangeeft dat de bewegingsrichting in het begin naar links is. Op moment 20 stel je de bewegingsrichting in naar boven, op moment 30 naar rechts, op moment 50 naar beneden en op moment 60 stop je de beweging. Vervolgens koppel je deze tijdlijn aan de bewaker en de bewaker zal precies uitvoeren wat jij hem hebt opgedragen. Je kunt een tijdlijn ook gebruiken om je spel globaler te controleren. Creëeer een onzichtbare controller object, maak een tijdlijn die op bepaalde momenten vijanden in het spel laat ontstaan en koppel die aan het controller object. Als je ermee wilt gaan werken zul je merken dat dit een erg krachtig concept is.

Om een tijdlijn te maken moet je de optie **Add Time Line** uit het **Add** menu kiezen. Dan komt het volgende venster in beeld:



Het lijkt een beetje op het eigenschappenvenster van objecten. Aan de linker kant kun je de naam opgeven en er zijn knoppen om momenten toe te voegen aan de tijdlijn of om ze te veranderen. Daarnaast een lijst momenten. In deze lijst staan de momenten in time steps vermeld, waarop de opegegeven acties zullen plaatsvinden. Daarnaast is er de gewone lijst acties voor het gekozen moment en tenslotten een set van alle mogelijke beschikbare acties.

Om een moment aan de tijdlijn toe te voegen moet je op de knop **Add** klikken. Geef het moment aan (uitgedrukt in het aantal steps sinds de tijdlijn is opgestart). nu kun je acties in de lijst slepen net als bij events voor objecten. Er zijn ook knoppen om een geselecteerd moment te wissen, de tijd te veranderen en de tijdlijn te wissen.

Tenslotte zijn er twee speciale knoppen. Met de **Merge** knop kun je alle momenten in een tijdsinterval samenvoegen tot één. Met de **Shift** knop kun je alle momenten een bepaalde tijd voorwaarts of achterwaarts plaatsen in de tijdlijn. Let er wel op dat je op die manier geen negatieve time momenten krijgt. Die zullen nooit worden uitgevoerd.

Er zijn twee acties die gerelateerd zijn aan tijdlijnen.

 **Set a time line**

Met deze actie stel je de bijzondere tijdlijn voor een instantie van een object in. Je geeft de tijdlijn aan, de startpositie in tijd (0 is het begin). Je kunt deze actie ook gebruiken om een tijdlijn te beëindigen door te kiezen voor **No Time Line** als waarde.

 **Set the time line position**

Met deze actie kun je de positie in de huidige tijdlijn veranderen (absoluut of relatief). Je kunt dat gebruiken om bepaalde stukken van de tijdlijn over te slaan of juist te herhalen. Bijvoorbeeld, als je een tijdlijn wilt maken die iedere keer herhaald wordt, dan moet je op het laatste moment deze actie toevoegen om de positie terug op 0 te zetten. Je kunt deze actie ook gebruiken om te wachten totdat iets speciaals gebeurt. Dan moet je de test actie toevoegen, en als de returnwaarde niet true is, moet je de tijdlijn positie relatief op -1.

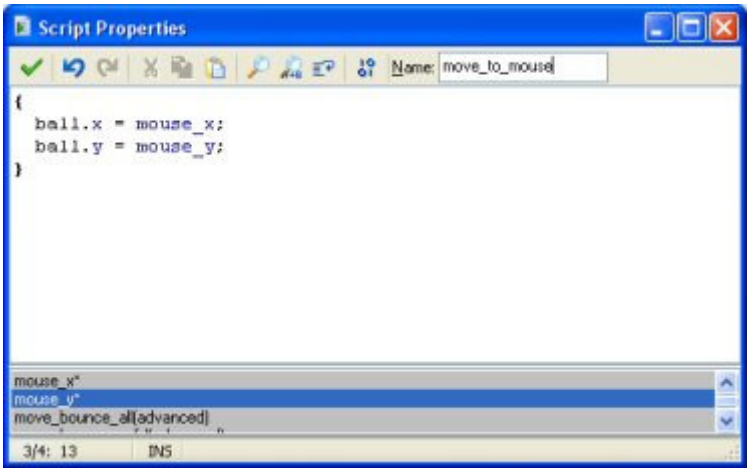
Scripts

Game Maker heeft een ingebouwde programmeertaal. Zodra je wat meer weet van *Game Maker* en het optimaal wilt gebruiken, is het raadzaam om te beginnen met het leren van deze taal. Er zijn twee manieren om de taal te gebruiken. Ten eerste kunnen je scripts maken. Dit zijn stukken code die je een naam geeft. Zij worden getoond in de resource tree en kunnen behalve opgeslagen worden in een bestand ook geladen worden uit een bestand. Zij kunnen worden gebruikt om een bibliotheek te vormen die de mogelijkheden van *Game Maker* uitbreidt. Daarnaast kun je een code

actie toevoegen aan één of andere gebeurtenis en een stuk code daar intypen. Het toevoegen van code acties werkt op de precies zelfde manier als het toevoegen van scripts op twee verschillen na. De code acties hebben geen naam en kunnen geen argumenten gebruiken. Voor de rest voer je code op precies dezelfde manier in als bij scripts. We concentreren ons verder in dit hoofdstuk op scripts.

Zoals al eerder aangegeven, wordt een script in GML-code (de ingebouwde programmeertaal)geschreven en is bedoeld om een bepaalde taak uit te voeren. Een script kan gebruik maken van input-variabelen die we argumenten noemen (soms ook parameters). Om een script van één of andere event uit te voeren, kan je de script actie gebruiken of code. In deze script actie geef je aan welk script je wilt uitvoeren, samen met de maximaal 5 argumenten. Als je een script gebruikt met een stuk code erin, werkt het op dezelfde manier alsof je een GM-functie gebruikt. In dat geval kun je maximaal 16 argumenten gebruiken. Scripts kunnen één waarde teruggeven. Dit wordt vaak gebruikt om rekenmodellen of wiskundige modellen te maken. Daarvoor wordt het **return** sleutelwoord gebruikt. Geen enkele code na het return sleutelwoord wordt uitgevoerd! Als een script een returnwaarde aangeeft kun je dit ook gebruiken als een functie als je returnwaardes gebruikt in andere acties.

Om een script aan je spel toe te voegen moet je de optie **Add Script** uit het **Add** menu kiezen. Er verschijnt dan het volgende venster in beeld (in dit voorbeeld is al een klein script toegevoegd dat het product van 2 argumenten uitrekent).



(In feite is dit een ingebouwde script editor. In de instellingen kun je ook aangeven dat je gebruik wilt maken van een externe editor.) Bovenaan rechts kun je de naam van het script aangeven. Er is een klein veld waarin je de code kunt intypen. Merk op dat onderaan het venster een lijst beschikbaar is met alle functies, ingebouwde variabelen en constantes. Hierin vind je alles wat je nodig hebt. Door erop te dubbelklikken kun je het item aan de code toevoegen (of gebruik <Ctrl>P). Je kunt er ook voor kiezen om deze lijst niet te laten zien. Je kunt dat in de voorkeursinstellingen aangeven. De editor heeft een aantal nuttige eigenschappen, waarvan een

aantal door middel van knoppen zijn weergegeven (klik op de rechtermuisknop om nog een aantal extra commando's te laten zien):

- Meervoudig ongedaan maken of herhalen ofwel door middel van het indrukken van een toets of in groepen (kan ingesteld worden in de voorkeursinstellingen)
- Intelligent van zelf inspringen waardoor uitlijning met de vorige regel optreedt (kan ingesteld worden in de voorkeursinstellingen)
- Intelligente tabinstelling waardoor een tabsprong optreedt tot het eerste karakter uit de vorige lijn (kan ingesteld worden in de voorkeursinstellingen)
- Gebruik <Ctrl>I om geselecteerde regels te laten inspringen en <Shift> <Ctrl>I om geselecteerde regels te laten terugspringen
- Kopiëren en Plakken
- Zoeken en Vervangen
- Gebruik <Ctrl> + up, down, page-up, of page-down om te scrollen zonder dat de cursorpositie verandert
- Gebruik F4 om het script of de resource op wiens naam de cursor staat te openen (werkt niet in de code actie; alleen maar in scripts)
- Het opslaan en laden van een script als tekstfile

Er is ook een knop aanwezig waarmee je kunt testen of een script correct is of niet. Niet alle aspecten van het script kunnen in dit stadium worden getest, maar het is wel mogelijk om de syntax en het gebruik van de aanwezige functies te testen.

Zoals je opgemerkt zult hebben, zijn delen van het script gekleurd. De editor is van bestaande objecten, ingebouwde variabelen en functies, enz. op de hoogte. Kleurcodering is een hulp bij het vermijden van fouten. In het bijzonder zie je onmiddellijk of je schrijffouten hebt gemaakt of een sleutelwoord als variabele hebt gebruikt. Het gebruik van kleurcodering maakt het echter wel langzaam. In de voorkeursinstellingen in het **File** menu kun je kleurcodering aan- en uitschakelen. Hier kan je de kleur voor de verschillende componenten van de programma's ook veranderen. (Als er iets mis gaat met de kleurcodering kun je tweemaal op F12 klikken, om het uit en daarna weer aan te schakelen.) Ook kan je de font veranderen die in scripts en code wordt gebruikt.

De scripts zijn uiterst nuttig om de mogelijkheden van *Game Maker* uit te breiden. Dit vereist echter dat je je scripts zorgvuldig moet ontwerpen. De scripts kunnen in libraries worden opgeslagen die aan je spel kunnen worden toegevoegd. Om een library te importeren, gebruik je **Import scripts** uit het **File** menu. Om je scripts in de vorm van een library op te slaan gebruik je **Export scripts**. De script libraries zijn eenvoudige tekstbestanden (hoewel zij de extensie .gml hebben). Bij voorkeur niet direct editen omdat zij een speciale structuur hebben. Sommige libraries met nuttige scripts zijn in het programma opgeslagen. (Om onnodig werk te vermijden bij het

laden van het spel kun je het beste, na het importeren van een library, scripts schrappen die je niet gebruikt.)

Als je scripts schrijft, kun je nogal snel fouten maken. In Game Maker bestaat de mogelijkheid om je scripts te testen. Daarvoor is een knop aanwezig. Als er een fout in het script zit zal deze, nadat het script is uitgevoerd, worden weergegeven. Daarbij wordt het type fout aangegeven en ook de plaats waar zich de fout in het script bevindt. Het zal zelden voorkomen dat je een pop-up venster tegenkomt waarop staat: "Unexpected error occurred during the game". Deze foutmelding houdt in dat er een of ander probleem met Windows of de hardware is opgetreden. Vaak ligt hier een eindeloze herhaling (recursie) aan ten grondslag, of een gebrek aan geheugen of dat het systeem niet voldoet aan de minimale hardware eisen, onjuiste drivers of firmware. In het algemeen kun je aannemen dat deze problemen buiten *Game Maker* liggen. Als je bepaalde dingen nauwkeuriger wilt testen, bestaat de mogelijkheid het spel in de debug-mode te draaien. Dan verschijnt er een venster waarin je veel aspecten van je spel kunt bekijken en testen.

In het **Run** menu heb je de mogelijkheid om het spel te laten stoppen op een punt (pauze), het stap voor stap af te spelen en het eventueel te herstarten. In het **Watch** menu kun je de waarden bekijken van bepaalde expresies. Gebruik **Add** om een expressie in te typen waarvan de waarde in elke stap van het spel wordt weergegeven. Op deze manier kun je bekijken of de dingen in het spel verlopen zoals ze zouden moeten verlopen. Je kunt een groot aantal expressies op die manier bekijken. Je kunt ze opslaan voor later gebruik (bijvoorbeeld nadat je een correctie hebt aangebracht in het spel). In het **Tools** menu vind je items waarmee je zelfs nog meer informatie kunt zien. Je kunt een complete lijst met instanties van het spel zien, alle globale variabelen (nou, in ieder geval de belangrijkste) en de locale variabelen van een instantie (gebruik of de objectnaam of het id van de instantie). Je kunt ook berichten bekijken die door je code zijn verzonden door gebruik te maken van de functie `show_debug_message(str)`. Tenslotte kun je het spel

opdrachten geven en de snelheid van het spel aanpassen. Als je erg ingewikkelde spellen gaat maken is het zeker de moeite waard om te leren omgaan met de debug opties.

Het afronden van je spel

Als je van plan bent om je spel te verspreiden moet je ervoor zorgen dat alle ingrediënten aanwezig zijn in je spel die een spel tot een prima speelbaar spel maken. Dat betekent dat je naast het spel zelf ook spelinformatie moet geven in de vorm van een gebruikershandleiding (in het spel dan wel apart in een file), dat je ervoor moet zorgen dat alle globale instellingen correct zijn, waardoor het spel meteen speelbaar is, en ervoor zorgen dat de snelheid waarmee het spel gespeeld kan worden OK is. Dit gedeelte geeft je de nodige informatie waarmee je dat kunt bereiken.

Spelinformatie

Een goed spel voorziet de speler van wat informatie over hoe het spel te spelen. Deze informatie wordt getoond wanneer de speler de <F1> toets indrukt tijdens het spel. Om de spelinformatie te creëren, dubbelklik op **Game Information** in de resource tree aan de linkerkant van het scherm. Een kleine teksteditor wordt geopend waar je de spelinformatie kunt intypen. Je kunt verschillende fonts, kleuren en stijlen gebruiken. Je kunt ook de achtegrondkleur instellen.

In het **File** menu kun je ook een aantal **Options** instellen. Hier kun je o.a. de titel van de spelinformatie tijdens het spelen van het spel opgeven. Bovendien kun je de grootte en de positie van het informatievenster (de waarde -1 zorgt ervoor dat het venster wordt gecentreerd) aangeven, of het een rand moet hebben en of je de speler het venster moet kunnen vergroten of verkleinen. Je kunt ervoor zorgen dat het informatievenster bovenop verschijnt en bovendien of het spel doorloopt als de informatie wordt geraadpleegt of dat het spel en pauze inlast.

een interessante optie is om het te doen alsof de informatie in het speelveld wordt geprojecteerd. Als je deze optie gebruikt (aanvinken) wordt het informatievenster op exact dezelfde plaas en met dezelfde grootte afgebeeld bovenop het speelveld. Als je de juiste achtergrondkleur kiest krijg je een aardig visueel effect. (Je zou ervoor kunnen kiezen om onderaan het informatievenster de mededeling te doen dat de speler op Escape moet drukken om verder te gaan met het spel.)

Een goed advies is: zorg ervoor dat je informatie kort maar correct is! Natuurlijk kun je ook je naam erbij vermelden; je hebt tenslotte het spel gemaakt. Alle voorbeeldspellen hebben een informatiebestand over het spel en hoe het werd gemaakt. als voorbeeld zou je daar eens kunnen kijken.

Als je een beetje fraaiere informatiefile wilt maken kun je bijvoorbeeld Word gebruiken. En selecteer dan het deel wat je wilt gebruiken, kopieer het en plak het in de *Game Maker* informatie editor. Voor de meer geavanceerde spellen zal je dit waarschijnlijk helemaal niet gebruiken, maar probeer het eens in bepaalde rooms om informatie te tonen over je spel.

Globale instellingen van je spel

In je spel kun je een groot aantal instellingen veranderen. Die hebben gevolgen voor de vorm van het speelveld, grafische eigenschappen, interacties, het laden van afbeeldingen, constanten en informatie over de maker van het spel. Je kunt bovendien aangeven welke files moeten worden gebruikt om een stand-alone versie van je spel te maken en hoe de foutenafhandeling moet gebeuren.

De instellingen kunnen worden aangepast door te dubbelklikken op **Global Game Settings** in de resource tree aan de linkerkant van het scherm. Daar zie je een ondercerdeling in diverse tabbladen. (Enige opties zijn alleen maar aanwezig in de advanced mode.)

Grafische opties

In dit tabblad kun je een aantal instellingen wijzigen die gerelateerd zijn aan het uiterlijk van je spel. Het is normaal gesproken nuttig om na te gaan wat de effecten zijn van wijzigingen in deze instellingen omdat die wel eens belangrijke verschillen kunnen opleveren qua uiterlijk. Maar houd er wel rekening mee dat andere personen totaal verschillende computers kunnen hebben en je zou toch willen dat kouw spel op zoveel mogelijk computers gespeeld zou kunnen worden.

Start in fullscreen mode
Als deze optie is aangevinkt, runt het spel in de full screen mode (beeldvullend), zo niet, draait het in een window, waarvan jij de grootte hebt bepaald.

Scaling
Hier kun je aangeven wat er gebeurt als het window groter is dan de room os als het spel beeldvullend wordt gespeeld. Je hebt nu 3 keuzemogelijkheden. Je kunt gebruik maken van een vaste grootte van het venster. De room wordt op schaal (met de door kou opgegeven waarde) in het midden van het beeld getekend. De waarde 100 betekent betekent standaard grootte. Deze optie gebruik je als je rooms en sprites erg klein zijn. De tweede optie is dat je de room kunt opschalen totdat hij window- of beeldvullend is, maar met behoud van de verhoudingen tussen breedte en hoogte. De derde optie is die dat het window of beeld totaal gevuld worden. Dat kan leiden tot vervorming van het beeld (vooral als je de gebruiker de mogelijkheid biedt de windowgrootte eigenhandig te laten aanpassen).

Interpolate colors between pixels

Als deze optie is aangevinkt zullen de kleuren van de pixels in sprites, achtergronden en tiles die niet juist zijn uitgelijnd met pixels op het scherm, worden geïnterpoleerd. Dit is vooral het geval als ze van grootte zijn veranderd of geroteerd of op posities staan waarvan de coördinaten geen geheel getal zijn. Interpolatie zorgt ervoor dat de beweging wat vloeiender verloopt maar het kan ook een effect geven waarbij de afbeeldingen vage randen hebben. (Bij tiles kan het leiden tot scheurtjes tussen de afbeeldingen als ze niet nauwgezet zijn ontworpen.)

Color outside the room region

Als de room niet even groot is als het window of het scherm is er een ruimte rondom die niet gebruikt wordt. Hier kun je de kleur van die ruimte opgeven.

Allow the player to resize the game window

Als de optie window mode is aangevinkt kan de speler het window groter of kleiner maken door met de muis aan de hoeken te slepen.

Let the game window always stay on top

Als de optie window mode is aangevinkt zorgt deze optie ervoor dat het spelvenster altijd boven andere vensters zichtbaar is.

Don't draw a border in windowed mode

Als de optie window mode is aangevinkt zorgt deze optie ervoor dat het spelvenster geen kader en titelbalk heeft.

Don't show the buttons in the window caption

Als de optie window mode is aangevinkt zorgt deze optie ervoor dat in het titelvenster de knoppen om het venster te sluiten of te minimaliseren of maximaliseren ontbreken.

Display the cursor

Deze optie geeft aan of je de muiscursor wel of niet wilt laten zien. Deze optie uitschakelen geeft normaal gesproken een fraaier beeld en maakt het spel in ieder geval sneller. (In *Game Maker* kun je eenvoudig je eigen cursorobject maken.)

Freeze the game when the form looses focus

Als deze optie is aangevinkt zal het spel stoppen met behoud van alle kenmerken van dat moment indien de speler een ander venster of een andere applicatie opent. Als daarna weer op het spelvenster wordt geklikt, gaat het spel verder waar het gebleven was.

Resolutie

In dit tabblad kun je de schermresolutie instellen waarin je spel gespeeld moet worden. Standaard wordt de resolutie niet gewijzigd. Soms wil je echter dat het spel gespeeld wordt in een lagere resolutie of je wilt de verversingsfrequentie van de monitor aanpassen waardoor je er zeker van bent dat de timing in het spel correct werkt. Als je de resolutie wilt aanpassen dien je een vinkje te plaatsen in de box **Set the resolution of the screen**.

Er zijn 3 instellingen die je kunt aanpassen. Allereerst is er de kleurdiepte. Dit geeft aan hoeveel bits er gebruikt worden om de kleur van een pixel te bepalen. Op dit moment laten de meeste computers alleen maar 16 bit (High Color) of 32 bit (Full Color) toe maar oudere computers laten ook 8 bit en soms 24 bit kleuren toe. Als je wilt dat je spellen voornamelijk op wat oudere computers gespeeld kunnen worden, kun je de kleurdiepte het beste op 16 bit zetten. In andere gevallen zet je de kleurdiepte op 32 bit of je laat hem onveranderd.

Daarnaast heb je ook nog de schermresolutie. Dat is de maat voor het aantal pixels (horizontaal en vertikaal) op het beeldscherm. Het wijzigen van de resolutie kan handig zijn als de rooms bijvoorbeeld erg klein zijn. In dit geval zou het handig zijn de resolutie te verlagen. Houd er wel rekening mee dat dit ook effect heeft op het draaien van andere applicaties. Dat kan vooral bij lage resoluties problemen opleveren. In het algemeen moet je dit alleen maar doen als het spel in de full screen mode gespeeld wordt. *Game Maker* zal dan automatisch na het beëindigen van het spel de beeldschermresolutie zijn zijn ooespronkelijke stand terugzetten.

Tenslotte kun je ook nog de verversingsfrequentie aanpassen. Dit getal geeft aan hoe vaak per seconde het beeld op het scherm wordt ververst. Als de snelheid van de room groter is dan de verversingsfrequentie zullen niet alle steps zichtbaar zijn. (Als je echter een verversingsfrequentie opgeeft die voor je monitor te hoog is of niet beschikbaar, zal de frequentie niet worden gewijzigd.)

Er bestaat ook nog een speciale instelling: **Use synchronization to avoid tearing**. Deze functie behoeft enige uitleg. Op het scherm wordt een aantal maal per seconde een beeld getekend, hetgeen afhankelijk is van de verversingsfrequentie. Als een room voor de helft opnieuw is getekend (van onder naar boven) dan zal in de bovenste helft het oude beeld nog zichtbaar zijn. Dit verschijnsel heet tearing. Om dit verschijnsel te vermijden moet je deze optie aanvinken. In dat geval wordt het nieuwe beeld alleen maar op het scherm gebracht als de verversing volledig is afgerond. Dat betekent dat bij iedere verversing een nieuw compleet beeld op het scherm getekend wordt. Het tekenen van de nieuwe afbeelding op het scherm is op die manier gesynchroniseerd met het verversen van het scherm. Het nadeel is dat we moeten wachten totdat de verversing is afgelopen. Daarnaast kan er ook een conflict optreden tussen de interne timing van het spel en de synchronisatie. Als je deze optie wilt gaan gebruiken kun je het beste de frequentie op een vaste waarde instellen (bijvoorbeeld 60) en de snelheid van de room op 9999.

Op die manier zal het spel met precies 60 frames per seconde runnen zonder taering. (Op LCD schermen houdt je altijd enige tearing, dat is onvermijdelijk.)

Verschillende andere opties

In de **Global Game Settings** is ook het tabblad **Other** aanwezig. Hier kun je een aantal extra opties instellen. Allereerst kun je een aantal standaard toetsten instellen:

Let <Esc> end the game
Als deze optie is aangevinkt, kun je met de <Esc> toets het spel beëindigen. Bij wat geavanceerdere spellen wil je dit helemaal niet laten gebeuren omdat je bijvoorbeeld, voordat het spel beëindigd wordt, je eerst nog gegevns wilt opslaan. In dat geval moet je het vinkje weghalen en je eigen acties voor de <Esc> toets ingeven. (Op het kruisje in de rechterbovenhoek van een venster zal ook een <Esc> toets event tot gevolg hebben.)

Let <F1> show the game information
Als deze optie is aangevinkt, kun je met de F1 toets de spelinformatie laten zien.

Let <F4> switch between screen modes
Als deze optie is aangevinkt, kun je met de F4 toets wisselen tussen de full screen en de window mode.

Let <F5> and <F6> load and save the game
Als deze optie is aangevinkt, kun je met de F5 toets de huidige spelsituatie opslaan en met F6 het laatst opgeslagen spel weer openen.

Daarnaast kun je ook nog prioriteiten aangeven in het spelproces. Deze prioriteit geeft aan hoeveel processortijd wordt toegekend aan het spel. In de normal mode probeert het besturingssysteem de processortijd op een redelijke manier te verdelen over de verschillende processen die plaatsvinden. Hoe hoger je de prioriteit instelt, des te meer processortijd er voor het spel wordt gereserveerd waardoor het spel vloeiender en sneller loopt. Maar dat betekent wel dat andere processen minder tijd krijgen (waaronder Windows processen, het kan betekenen dat zelfs de muis het op gegeven moment niet meer doet). Gebruik de optie dus zorgvuldig.

Opties voor het laden van het spel

Hier kun je bepalen wat er allemaal moet gebeuren als een spel wordt geladen. Allereerst kun je je eigen afbeelding laten laden. Vervolgens is het mogelijk een voortgangsbalk onderaan de afbeelging in beeld te brengen. Daarvoor heb je drie opties: ofwel géén voortgangsbalk, ofwel de standaardbalk wordt geladen, ofwel je kunt twee afbeeldingen specificeren: de achtergrond van de

voortgangsbalk en de voorgrond. Je kunt daarbij aangeven of de balk must be scaled (default) or clipped while it becomes longer. Zorg er in het tweede geval wel voor dat de afbeelding groot genoeg is om de balk te vullen. (Houd er rekening mee dat in dit geval twee afbeeldingen moeten worden gespecificeerd, dus niet slechts één.)

Het is mogelijk om aan te geven dat de afbeelding die geladen wordt transparant moet zijn. In dat geval wordt de pixel in de linker benedenhoek van de achtergrondafbeelding als transparante kleur gebruikt. Bovendien kan de alpha doorschijnendheid worden aangegeven. Een waarde 0 betekent volledig doorschijnend. De waarde 255 betekent volledig ondoorschijnend. (Beide instellingen werken alleen onder Windows 2000, XP of later.)

Vervolgens kun je aangeven welk icoon je wilt gebruiken voor stand-alone spellen. Daarvoor mag je alleen maar icoontjes gebruiken met een afmeting van 32x32 pixels. als je een ander type wilt gebruiken krijg je een waarschuwing.

Tenslotte kun je het unieke spel id wijzigen. dit id wordt gebruikt om de highscore-lijst op te slaan maar ook spelbestanden. Als je een nieuwere versie van je spel uitbrengt, maar je wilt niet de oude highscore-lijst gebruiken, dan moet je dit nummer veranderen.

Constanten

In dit tabblad kun je globale constanten definiëren die je vervolgens in alle scripts en stukken code kunt gebruiken, of als waarden voor acties. Elke constante heeft een naam en een waarde. Wat de naamgeving betreft moet je dezelfde regels in acht nemen als bij variabelen, dat wil zeggen dat ze moeten beginnen met een letter of het (_) underscore-teken en verder bestaat uit letters, cijfers of underscore-tekens. Het is wel aan te raden om de namen van de constanten zo te kiezen dat ze eenvoudig van elkaar zijn te onderscheiden. Een gebruikelijke conventie is om alleen maar hoofdletters en underscores te gebruiken.

Een waarde van een constante moet een constante expressie zijn. Dat wil zeggen dat het ofwel een constant getal is of een string (een tekenreeks voorzien van aanhalingstekens voor en achter) ofwel een expressie. De expressie wordt getest voordat er iets anders in het spel gebeurt. Op die manier kan de expressie niet verwijzen naar de huidige room met zijn instanties of scripts. Maar het kan wel de ingebouwde constanten en de namen van de resources bevatten.

Je kunt een constante toevoegen door op de knop **Add** te klikken en een constante verwijderen door op de knop **Delete** te klikken. Je kunt de naam of de waarde van een constante veranderen door er op te klikken. Er is ook een knop om alle constanten te wissen of ze te sorteren op naam.

Het insluiten van bestanden in stand-alone spellen

Zoals al eerder in deze handleiding is aangegeven heb je de mogelijkheid om stand-alone spellen te maken. Dan kan het voorkomen dat je extra bestanden voor die spellen nodig hebt. Bijvoorbeeld, je zou korte viedeo- of tekstbestanden willen gebruiken in je spel. In andere gevallen wil je een aantal DLL's of afbeeldingen of geluiden willen toevoegen die worden geladen tijdens het spelen van het spel. Je kunt die bestanden samen met het spel, bijvoorbeeld gezipt aanbieden, maar soms is het eenvoudiger om al die bestanden in het spelbestand te integreren. Op die manier hoef je maar één bestand te verspreiden.

Je kunt aangeven welke files allemaal dienen te worden opgenomen in het spelbestand. Bovenaan het venster is een lijst met bestanden die worden opgenomen in het spel. Gebruik **Add** om bestanden te selecteren die aan de lijst moeten worden toegevoegd (je kunt meerdere bestanden tegelijk selecteren). Gebruik **Delete** of **Clear** om één of meerdere of alle bestanden uit de lijst te verwijderen. (Houd er wel rekening mee dat de bestanden niet in het .gm6 bestand zijn opgenomen, maar alleen maar de namen van de bestanden. Dat betekent dat als je iemand je te bewerken .gm6 bestand stuurt, je ook de bestanden moet opsturen om er uiteindelijk een stand-alone van te kunnen maken.)

De bestanden in de lijst worden in de executable verpakt als deze wordt gemaakt. Als de game wordt gespeeld worden deze bestanden uitgepakt waardoor ze in het spel bereikbaar zijn. Het is belangrijk om te weten waar dit gebeurt. Je kunt daarvoor uit twee mogelijkheden kiezen. Standaard worden de files opgeslagen in de map waarin het te spelen spel staat. Dit is ook de werkmap voor het spel. Op die manier heeft het spel voldoende aan de bestandsnamen omdat hij dan automatisch in de werkmap naar die bestanden gaat zoeken. Je hoeft dus geen pad op te geven. Deze manier werkt prima als je het stand-alone spel hebt opgeslagen op een harde schijf maar zal absoluut niet werken als het is opgeslagen op een medium met alleen-lezen moegelijkheden, zoals een CDROM.

De tweede manier is om aan te geven dat de bestanden moeten worden uitgepakt in een tijdelijke map, die tijdens het spelen van het spel wordt aangemaakt. Als je deze optie kiest moet je wel een pad opgeven naar die tijdelijke map omdat anders het spel niet weet waar het de bestanden vandaan moet halen die nodig zijn om het spel te kunnen spelen. Het pad kun je instellen door gebruik te maken van de ingebouwde variabele `temp_directory`. Vergeet echter niet om in dit geval de backslash toe te voegen. Als je bijvoorbeeld een stukje video in je spel wilt laten zien kun je het volgende stukje code gebruiken:

```
{
  show_video(temp_directory+'\movie.avi',true,true);
}
```

Besef wel dat de tijdelijke map weer van de computer wordt verwijderd als het spel wordt beëindigd. Dat betekend dat je in dit geval geen spellen kunt opslaan of spelinformatie kunt bewaren. Gebruik deze optie alleen maar als je het spel vanaf een CDROM wilt draaien of als tijdens het spel geen informatie hoeft te worden weggeschreven.

Als een uitgepakt bestand reeds in de map bestaat dan wordt die standaard niet overschreven. Je kunt dat wijzigen door een vinkje te pplaatsen in de box **Overwrite existing files**. daarnaast worden normaliter de uitgepakte files niet verwijderd als het spel wordt beëindigd (tenzij die bestanden zich in de tijdelijke map bevinden, die compleet van de computer wordt verwijderd). Je kunt dit veranderen door een vinkje te plaatsen in de box **Remove at game end**.

Een waarschuwing is hier wel op zijn plaats. Als je je spel bent aan het testen, is de werkmap de map waarin het .gm6 bestand staat. Als je bestanden die moeten ingepakt worden bij het spel hier ook staan en je maakt de keuze om na het beëindigen van het spel alle files te verwijderen, dan ben je uiteraard ook die files kwijt die bij het spel moeten worden ingepakt. Dat zal wel niet de bedoeling zijn. Daarom kun je die bestanden beter in een andere map plaatsen en dus niet in de map waarin het .gm6 bestand staat!

Error opties

Hier kun je een aantal instellen die te maken hebben met de manier hoe fouten worden weergegeven.

Display error messages
Als deze optie is aangevinkt worden foutmeldingen aan de speler getoond. In de uiteindelijke versie kun je deze optie beter uitschakelen.

Write error messages to file `game_errors.log`
Als deze optie is aangevinkt worden foutmeldingen in een logfile weggeschreven met de naam `game_errors.log` in de map waarin het spel staat.

Abort on all error messages
Onder normale omstandigheden zijn sommige fouten fataal voor het spel terwijl andere best genegeerd kunnen worden. Als je deze optie aanvinkt worden alle fouten als fataal beschouwd hetgeen leidt tot het beëindigen van het spel. In de uiteindelijke versie kun je deze optie beter uitschakelen.

Treat uninitialized variables as 0
Een veel voorkomende fout is dat je gebruik maakt van een variabele voordat je er een waarde aan

hebt toegekend. Soms is dat moeilijk te vermijden. Als je deze optie aanvinkt zullen zulke nog niet geïnitialiseerde variabelen niet langer een foutmelding opleveren maar behandeld worden alsof ze de waarde 0 hebben. Maar wees toch voorzichtig daarmee. Het kan er toe leiden dat je niet meer alert bent op het maken van fouten.

Informatie over het spel

Hier kun je aangeven wie de maker van het spel is, de versie van het spel en enige informatie over het spel zelf. Daarnaast wordt ook bijgehouden wanneer de laatste veranderingen aan het spel zijn doorgevoerd. dat is vooral nuttig als je met meerdere mensen tegelijk aan een spel werkt of een nieuwe release uitbrengt. Deze informatie is niet toegankelijk tijdens het spelen van het spel.

Beschouwingen over snelheid

Als je ingewikkelde spellen maakt, wil je vast en zeker dat ze zo snel mogelijk werken. Hoewel *Game Maker* er alles aan doet om de spellen zo snel mogelijk te maken, hangt er veel af van de manier van ontwerpen. Het is helemaal niet moeilijk om een spel te maken dat enorme hoeveelheden geheugen gebruikt. In dit hoofdstuk zal ik wat hints geven voor het maken van snellere en kleinere spellen.

Wees ten eerste voorzichtig met het gebruik van sprites en achtergronden. Bewegende sprites gebruiken veel geheugen en het tekenen van veel sprites kost veel tijd. Dus maak je sprites zo klein mogelijk. Verwijder alle onzichtbare ruimte rond de sprite (het commando **crop** in de sprite editor doet dat automatisch). Hetzelfde geldt ook voor achtergrondafbeeldingen. Zorg ervoor dat je het gebruik van een achtergrondkleur uitzet wanneer je een achtergrondafbeelding hebt die de hele room bedekt.

Wanneer je gebruik maakt van fullscreen mode zorg er dan voor dat de grootte van je room (of window) nooit groter is dan het scherm. De meeste videokaarten kunnen plaatjes makkelijker opschalen naar een groter formaat dan omgekeerd. Indien mogelijk kun je het beste de cursor uitschakelen. Daardoor wordt het tonen van afbeeldingen erg vertraagd.

Wees ook voorzichtig met het gebruik van meerdere views. Voor elke view moet de room opnieuw getekend worden.

Naast de afbeeldingen zijn er nog andere aspecten die de snelheid beïnvloeden. Wees er zeker van dat je zo weinig mogelijk instanties hebt. In het algemeen moet je instanties die niet meer nodig zijn verwijderen (bijvoorbeeld als ze de room verlaten). Vermijd veel werk in de step event of de teken event van instanties. Vaak hoeven dingen niet in elke stap gecontroleerd te worden. Het

interpreteren van code gaat best snel, maar het moet wel geïnterpreteerd worden. Ook kosten sommige functies en acties veel tijd, met name die alle instanties moeten controleren (zoals bijvoorbeeld de bounce actie).

Denk erover na waar je de collision events wilt toepassen. Normaal gesproken heb je twee opties. Objecten die helemaal geen collision event hebben worden veel sneller behandeld, maar pas ze bij voorkeur toe in die objecten die slechts een paar instanties hebben.

Wees voorzichtig bij gebruik van grote geluidsbestanden. Deze nemen veel geheugenruimte in beslag en slecht te comprimeren. Misschien kun je eens bekijken of je je geluiden door middel van sampling kleiner kunt maken.

Tenslotte, wanneer je een spel wilt maken dat door veel mensen gespeeld kan worden, test je spel dan ook op oudere computers.

De Game Maker Language (GML)

Game Maker bevat een ingebouwde programmeertaal. Deze programmeertaal maakt een grotere flexibiliteit mogelijk en een betere controle dan bij de standaard acties. Deze taal noemen we GML (de *Game Maker* Language). In dit gedeelte beschrijven we hoe de taal GML werkt en geven tevens een overzicht van de ongeveer 1000 functies en variabelen die het mogelijk maken om alle aspecten van je spel te controleren.

Informatie over GML kan gevonden worden op de volgende pagina's:

GML Taal overzicht

Game Maker bevat een ingebouwde programmeertaal. Deze programmeertaal geeft je meer flexibiliteit en controle dan de standaard acties. Naar deze taal zullen we verwijzen als GML (de *Game Maker* Language). Er zijn een paar verschillende plekken waar je programma's kan typen in deze taal. Ten eerste als je scripts maakt. Een script is een programma in GML. Ten tweede, als je code actie toevoegt aan een event. In een code actie moet je ook een programma geven in GML. Ten derde in de room creatie code. En ten slotte, overal waar je een waarde moet opgeven in een actie, kan je ook een expressie geven in GML. Een expressie, zoals we zullen zien hier beneden is geen compleet programma, maar een stukje code wat resulteert in een waarde.

In dit hoofdstuk zullen we de structuur van programma's in GML uitleggen. Als je gebruik wilt maken van programma's in GML, zijn er een paar dingen waar je rekening mee moet houden. Ten eerste, voor al je resources (sprites, objects, sounds, enz.) moet je namen gebruiken die starten met een letter en alleen bestaan uit letters, cijfers en de underscore '_'. Anders kan je ze niet aanroepen vanuit een programma. Controleer of alle resources verschillende namen hebben. Houd ook rekening om resources niet self, other global, of all te noemen want deze hebben een speciale betekenis in de taal. Ook moet je geen namen gebruiken die hier beneden staan aangegeven.

Een programma

Een programma bestaat uit een aantal instructies, genaamd verklaringen. Een programma moet beginnen met het symbool '{' en eindigen met het symbool '}'. Tussen die symbolen staan de verklaringen. Verklaringen moeten gescheiden worden met een ';' symbool. Dus de structuur van elk programma is:

```
{
  <statement>;
  <statement>;
  ...
}
```

Er zijn een paar verschillende type verklaringen, die hier beneden worden uitgelegd.

Variabelen

Net als elke programmeertaal bevat GML variabelen. Variabelen zijn geheugen locaties die informatie opslaan. Ze hebben een naam zodat hen daarmee naam kan verwijzen. Een variabele in GML kan een getal of een tekstreeks opslaan. Variabelen hoeven niet worden gedeclareerd te worden, zoals in veel andere talen. Er zijn een grote hoeveelheid van ingebouwde variabelen. Sommige zijn algemeen, zoals `mouse_x` en `mouse_y` die aangeven wat de huidige muispositie is, terwijl andere lokaal zijn voor de instantie wat het programma uitvoert, zoals x en y die de huidige positie aangeven van de instantie. Een variabele heeft een naam die moet beginnen met een letter en kan alleen letters, nummers en de underscore symbool '_' bevatten. (De maximale lengte is 64 symbolen.) Als je een nieuwe variabele gebruikt dan is het lokaal voor de huidige instantie en is niet bekend voor programma's voor andere instanties (zelfs voor hetzelfde object). Je kan toch verwijzen naar variabelen in andere instanties; zie beneden.

Definities

Een definitie slaat een waarde op in een variabele. Een definitie heeft de vorm:

```
<variable> = <expression>;
```

Een expressie kan een simpele waarde zijn, maar kan ook veel ingewikkelder is. Liever dan definiëren van een waarde aan een variabele, kan je ook de waarde optellen bij de gebruikte variabele. Net zoals je het kan aftrekken met -=, vermenigvuldigen met *=, delen met /=, of m.b.v. bits operators |=, &\ of ^=.

Expressies

Expressies kunnen getallen (bijv. 3.4), tekstreeksen tussen enkele of dubbele aanhalingstekens (bijv. 'hallo' or “hallo”) of meer ingewikkelde expressies zijn. Voor expressies, zijn er de volgende binaire operators (in volgorde van belangrijkheid):

- && || ^^: combineert Boolean waarden (&& = and, || = or, ^^ = xor)
- < <= == != > >=: vergelijkingen, resulteert in true (1), waar, of false (0), onwaar
- | & ^: bitwise operators (| = bitwise or, & = bitwise and, ^ = bitwise xor)
- << >>: bitwise operators (<< = schuif links, > > = schuif rechts)
- + -: optelling, aftrekking
- * / div mod: vermenigvuldiging, delen, integer delen, and modulo

Daarnaast bestaan ook de volgende operators:

- !: not, veranderd true in false en false in true
- -: maakt de waarde negatief

- ~: maakt de waarde m.b.v. bits negatief

Als waarden kan je ook getallen, variabelen, of functies die waarden teruggeven gebruiken. Subexpressies kunnen tussen haakjes worden geplaatst. Alle operators werken voor getallen. Vergelijkingen werken ook voor tekstreeksen en + voegt tekstreeksen samen. (Merk op dat, in tegenstelling tot bepaalde talen, beide argumenten voor een Boolean operatie worden altijd uitgevoerd, zelfs wanneer het eerste argument al bepaald wat eruit komt.)

Voorbeeld

Hier is een voorbeeld met een paar definities:

```
{
  x = 23;
  str = 'hallo wereld';
  y += 5;
  x *= y;
  x = y << 2;
  x = 23*((2+4) / sin(y));
  str = 'hallo' + " wereld";
  b = (x < 5) && !(x==2 || x==4);
}
```

Extra variabelen

Je maakt nieuwe variabelen door ze een waarde naar te definiëren (het is niet nodig om ze eerst te declareren). Als je gewoon een variabele naam gebruikt, zal de variabele worden opgeslagen met alleen de huidige instantie. Verwacht dus niet om het te vinden als je werkt met een ander object (of een andere instantie van hetzelfde object). Je kan ook variabelen instellen en lezen in andere objecten door het plaatsen van de object naam met een punt voor de variabele naam.

Om globale variabelen te maken, die zichtbaar zijn voor alle object instanties, laat ze dan voorgaan met het woord `global` en een punt. Dus bijvoorbeeld zou je kunnen schrijven:

```
{
  if (global.doit)
  {
```

```
    // doe iets
    global.doit = false;
  }
}
```

Soms wil je variabelen alleen in een stukje code of in een script. Op deze manier vermijd je verspilling van geheugen en ben je zeker dat er geen naamconflict is. Het is ook sneller dan het gebruik van globale variabele. Om dit te bereiken moet je de variabelen declareren aan het begin van een stukje code met gebruik van het sleutelwoord var. De declaratie ziet er zo uit:

```
var <varname1>,<varname2>,<varname3>, ...
```

Bijvoorbeeld, zou je kunnen schrijven:

```
{
  var xx,yy;
  xx = x+10;
  yy = y+10;
  instance_create(xx,yy,ball);
}
```

Verwijzen naar variabelen in een andere instantie

Zoals eerder uitgelegd, kun je variabelen instellen in de huidige instantie door gebruik van een statement zoals:

```
x = 3;
```

Maar in een aantal gevallen wil je verwijzen naar variabelen in een andere instantie. Misschien wil je bijvoorbeeld de beweging van alle ballen stoppen, of misschien de hoofdpersoon verplaatsen naar een specifieke positie, of, in het geval van een botsing, zou je misschien de sprite van het andere object in de botsing willen instellen. Dit kan worden bereikt door voor de variabele naam de naam van het object en een punt te plaatsen. Dus bijvoorbeeld, zou je kunnen schrijven:

```
ball.speed = 0;
```

Dit zal de snelheid van alle instanties van het object ball veranderen. Er zijn een paar speciale "objecten".

- `self`: De huidige instantie die de actie uitvoert.
- `other`: De andere instantie die is betrokken bij een boting in een collision event.
- `all`: Alle instanties.
- `noone`: Totaal geen instantie (klinkt waarschijnlijk vreemd maar dit wordt handig zoals we later zullen zien).
- `global`: Geen instantie, maar een container wat globale variabelen opslaat.

Dus, bijvoorbeeld, kan je de volgende soort statements gebruiken:

```
other.sprite_index = sprite5;
all.speed = 0;
global.message = 'Een goed resultaat';
global.x = ball.x;
```

Nu zul je je wel afvragen wat de laatste definitie doet als er meer ballen zijn. Nou, de eerste is genomen en zijn x waarde is ingesteld in de globale waarde.

Maar wat nou als je de snelheid wilt instellen van één specifieke bal, i.p.v. alle ballen? Dit is een beetje ingewikkelder. Elke instantie heeft een unieke id. Als je instantie plaatst in een room in de room-editor, is de id van die instantie weergegeven als je de muis stil houd boven de instantie. Deze nummers zijn groter of gelijk aan 10000. Zo'n nummer kan ook worden gebruikt aan de linkse kant van de punt. Maar wees voorzichtig. De punt zal worden geïnterpreteerd als de decimale punt in een nummer. Om dit te vermijden, plaats haakjes er omheen. Dus bijvoorbeeld, als we er vanuit gaan dat de id van de bal 100032 is, zou je kunnen schrijven:

```
(100032).speed = 0;
```

Als je een instantie maakt in een programma, de oproep geeft de id terug. Dus een goed stukje programma is

```
{
  nnn = instance_create(100,100,ball);
  nnn.speed = 8;
}
```

Dit maakt de bal en stelt zijn snelheid in. Merk op dat we de instantie id hebben gedefinieerd in een variabele en hebben deze variabele gebruikt om het object aan te geven voor de punt. Dit is helemaal in orde. Laat ons proberen om dit duidelijker te maken. Een punt is eigenlijk een operator. Het neemt een waarde als linkse operator en een variabele (adres) als rechtse, en geeft het adres van de specifieke variabele in het aangegeven object of instantie. Alle objectnamen, en de speciale objecten zoals hierboven aangegeven variable vertegenwoordigen waarden en deze kunnen worden gebruikt worden zoals elk gewone waarde. Bijvoorbeeld, het volgende programma is in orde:

```
{
  obj[0] = ball;
  obj[1] = flag;
  obj[0].alarm[4] = 12;
  obj[1].id.x = 12;
}
```

Het laatste statement zou gelezen moeten worden zoals volgt. We nemen de id van de eerste vlag. Voor de instantie met die id stellen we de x coördinaat naar 12.

Object namen, de speciale objecten, en de instantie id's kunnen ook worden gebruikt in een aantal functies. Ze worden eigenlijk behandeld als constants in het programma.

Arrays

Je kan gebruik maken van 1- en 2-dimensionale series in GML. Gebruik gewoon de index tussen '[' en ']' for een 1-dimensionale serie, en de twee gescheiden met een komma tussen hun voor 2-dimensionale series. Op dit moment gebruik je een index van de serie die is gemaakt. Elke serie start bij index 0. Dus wees voorzichtig met het gebruik van grote indexen, omdat geheugen voor een grote serie is gereserveerd. Gebruik nooit negatieve indexen. Het systeem plaatst een limiet op 32000 voor iedere index en 1000000 op de totale grootte. Dus bijvoorbeeld zou je kunnen schrijven het volgende:

```
{
  a[0] = 1;
  i = 1;
  while (i < 10) { a[i] = 2*a[i-1]; i += 1;}
  b[4,6] = 32;
```

```
}
```

If statement

Een if statement heeft de vorm

```
if (<expression>) <statement>
```

of

```
if (<expression>) <statement> else <statement>
```

Het statement kan ook een blok zijn. De expressie zal worden geavaleerd. Als de afgeronde waarde is <=0 (**false**) dan wordt het statement na else uitgevoerd, anders (**true**) wordt het andere statement uitgevoerd. Het is een goede gewoonte om altijd gekrulde haakjes te plaatsen om statements. Dus gebruik als beste

```
if (<expression>)
{
  <statement>
}
else
{
  <statement>
}
```

Voorbeeld

Het volgende programma beweegt het object naar het midden van het scherm.

```
{
  if (x<200) {x += 4} else {x -= 4};
}
```

Repeat statement

Een repeat statement heeft de vorm

```
repeat (<expression>) <statement>
```

Het statement wordt het aantal keer dat is aangegeven als afgeronde waarde van de expressie herhaald.

Voorbeeld

Het volgende programma maakt vijf ballen op willekeurige posities.

```
{
  repeat (5) instance_create(random(400),random(400),ball);
}
```

While statement

Het while statement heeft de vorm

```
while (<expression>) <statement>
```

Zolang de expressie waar (true) is, wordt het statement (dat ook een blok kan zijn) uitgevoerd. Wees voorzichtig met je while lus. Je kan ze makkelijk voor eeuwig door laten draaien, waardoor het spel zal blijven hangen en niet zal reageren op input van de gebruiker.

Voorbeeld

Het volgende programma probeert het huidige object te plaatsen op een vrije positie (dit is ongeveer hetzelfde als de actie om objecten te verplaatsen naar een willekeurige positie).

```
{
  while (!place_free(x,y))
  {
    x = random(room_width);
    y = random(room_height);
  }
}
```

Do statement

Een do statement heeft de vorm

```
do <statement> until(<expression>)
```

Het statement (dat ook een blok kan zijn) wordt uitgevoerd totdat de expressie waar (true) is. Het statement wordt in ieder geval 1 keer uitgevoerd. Wees voorzichtig met je lus. Je kan het makkelijk voor eeuwig door laten draaien, waardoor je spel zal blijven hangen en niet meer reageert op input van de gebruiker.

Voorbeeld

Het volgende programma probeert het huidige object te plaatsen naar een vrije positie (dit is ongeveer hetzelfde als de actie om objecten te verplaatsen naar een willekeurige positie).

```
{
  do
  {
    x = random(room_width);
    y = random(room_height);
  }
  until (place_free(x,y))
}
```

For statement

Een for statement heeft de vorm

```
for (<statement1> ; <expression> ; <statement2>) <statement3>
```

Dit werkt als volgende. Eerst wordt statement 1 uitgevoerd. Dan wordt de expressie geëvalueerd. Als het waar (true) is, dan wordt statement 3 uitgevoerd; daarna statement 2 en dan wordt de expressie weer opnieuw geëvalueerd. Dit gaat door totdat de expressie onwaar (false) is.

Dit klinkt misschien ingewikkeld. Je zou het als volgende moeten interpreteren. Het eerste statement initialiseert de for-lus. De expressie test of de lus moet eindigen. Statement2 is het step statement die door gaat naar de volgende evaluatie lus.

Het meest gebruikte gebruik is een lus tussen een bepaalde range.

Voorbeeld

Het volgende programma initialiseert een serie met een lengte van 19 en met de waarden 1 tot 10.

```
{
  for (i=0; i<=9; i+=1) list[i] = i+1;
}
```

Switch statement

In een aantal gevallen wil je je actie bepalen door een bepaalde waarde. Je kan dit doen door gebruik te maken van een aantal if statements maar het is makkelijker om gebruik te maken van het switch statement. Een switch statement heeft de volgende vorm:

```
switch (<expression>)
{
  case <expression1>: <statement1>; ... ; break;
  case <expression2>: <statement2>; ... ; break;
  ...
  default: <statement>; ...
}
```

Dit werkt als volgende. Ten eerste wordt de expressie uitgevoerd. Daarna wordt het vergeleken met de resultaten van de verschillede expressies na de case statements. De uitvoering gaat door na het eerste case statement met de juiste waarde, totdat een break statement is tegengekomen. Als geen case statement de juiste waarde heeft, dan gaat de uitvoering door na de default statement. (Het is niet verplicht om een default statement te hebben.) Merk op dat verschillende case statements geplaatst kunnen worden voor hetzelfde statement. De break is niet vereist. Als er geen break statement is gaat de uitvoering gewoon door met de code van het volgende statement.

Voorbeeld

Het volgende programma kiest een actie gebaseerd op de knop die is ingedrukt.

```
switch (keyboard_key)
{
  case vk_left:
  case vk_numpad4:
    x -= 4; break;
  case vk_right:
  case vk_numpad6:
```

```
        x += 4; break;
    }
```

Break statement

Het break statement heeft de vorm

```
    break
```

Indien het gebruikt wordt in een for-lus, een while-lus, een repeat-lus, een switch statement of een with statement, dan eindigt de lus of statement hier. Indien het gebruikt wordt buiten zo'n statement dan wordt het programma beëindigd (niet het spel).

Continue statement

Het continue statement heeft de vorm

```
    continue
```

Als het gebruikt wordt in een for-lus, een while-lus, een repeat-lus of een with statement, gaat het door met de volgende waarde in de lus of het statement.

Exit statement

Het exit statement heeft de vorm

```
    exit
```

Het eindigt de uitvoering van het huidige script of stukje code. (Het stopt niet de uitvoering van het spel! Hiervoor moet je de functie game_end(); gebruiken. Zie beneden.)

Functies

Een functie heeft de vorm van een functie naam, gevolgd door een of meer argumenten tussen haakjes, gescheiden door komma's.

```
    <function>(<arg1>,<arg2>,...)
```

Er zijn 2 typen functies. Ten eerste, is er een reusachtige collectie van ingebouwde functies, om alle aspecten van je spel te besturen. Ten tweede, elke script die je definieert in je spel kan gebruikt worden als functie.

Merk op dat voor een functie zonder argumenten nog steeds haakjes moeten worden gebruikt. Sommige functies geven waarden terug en kunnen gebruikt worden als expressies. Andere voeren gewoon commando's uit.

Merk op dat het onmogelijk is om een functie te gebruiken als de links kant van een definitie. Bijvoorbeeld, je kan niet schrijven `instance_nearest(x,y,obj).speed = 0`. Daarvoor in de plaats moet je dit schrijven `(instance_nearest(x,y,obj)).speed = 0`.

Scripts

Wanneer je een script maakt, wil je argumenten gebruiken die zijn meegegeven (indien je gebruikt maakt van een script in een actie, maar ook wanneer je een script oproept als een functie (of van een ander, of zelfs van hetzelfde script)). Deze argumenten zijn opgeslagen in de variabelen `argument0`, `argument1`, ..., `argument15`. Dus kunnen er maximaal 16 argumenten zijn. (Merk op dat als je een script oproept vanuit een actie, je alleen de eerste 5 argumenten opgegeven kunnen worden.) Je kunt ook `argument[0]` enz. gebruiken.

Scripts kunnen ook een waarde teruggeven, zodat ze gebruikt kunnen worden in expressies. Voor dit uiteinde gebruik je het return statement:

```
return <expression>
```

De uitvoering van het script eindigt bij het return statement!

Voorbeeld

Hier is de definitie voor een kleine script dat het kwadraat berekent van het argument:

```
{
  return (argument0*argument0);
}
```

Om een script aan te roepen vanuit een stukje code, doe je gewoon hetzelfde als dat je doet wanneer je functies oproept. Dat is, schrijf het script naam met de argument waardes tussen haakjes.

With constructies

Zoals eerder aangegeven, is het mogelijk om waarden te lezen en te veranderen van variabelen in andere instanties. Maar in een aantal gevallen wil je veel meer met andere instanties. Bijvoorbeeld, denk je eens in dat je alle ballen 8 pixels naar beneden wilt verplaatsen. Je denkt misschien dat je dit behaalt met dit stukje code

```
bal.y = bal.y + 8;
```

Maar dit in niet correct. De rechter kant van de definitie krijgt de waarde van de y-coördinaat van de eerste bal en voegt 8 eraan toe. Daarna is deze nieuwe waarde ingesteld als y-coördinaat voor alle ballen. Dus het resultaat is dat ze hetzelfde y-coördinaat krijgen. Het statement

```
bal.y += 8;
```

heeft precies hetzelfde effect omdat het een simpele afkorting van het eerste statement is. Dus hoe bereik je dit? Voor dit doel is er het **with** statement. Zijn globale vorm is

```
with (<expression>) <statement>
```

<expression> geeft 1 of meer instanties aan. Hiervoor kan je het instantie id gebruiken, de naam van een object (om alle instanties van 1 object aan te geven) of 1 van de speciale objecten (all, self, other, noone). <statement> wordt nu uitgevoerd voor elk van deze aangegeven instanties, alsof het de huidige (self) instantie is. Dus, om alle ballen 8 pixels naar beneden te verplaatsen kun je typen.

```
with (bal) y += 8;
```

Indien je meerdere statements wilt uitvoeren, plaats dan '{' en '}' om de statements heen. Dus bijvoorbeeld, om alle ballen te verplaatsen naar een willekeurige plek, kun je gebruiken

```
with (bal)
{
  x = random(room_width);
  y = random(room_height);
}
```

Merk op dat, in de statement(s), de aangeven instantie de self instantie is geworden. En de originele self instantie wordt de other instantie. Dus bijvoorbeeld, om alle ballen te verplaatsen naar de positie van de huidige instantie, kun je typen

```
with (bal)
{
  x = other.x;
  y = other.y;
}
```

Het gebruik van het with statement is erg krachtig. Laat me je nog een paar voorbeelden geven. Om alle ballen te vernietigen kun je typen

```
with (bal) instance_destroy();
```

Als een bom ontploft en je wilt alle instanties dichtbij de bom vernietigen, dan kun je typen

```
with (all)
{
  if (distance_to_object(other) < 50) instance_destroy();
}
```

Commentaar

Je kunt commentaar toevoegen aan je programma's. Alles op een regel na // wordt niet gelezen. Je kunt ook gebruik maken van meerdere regels commentaar door /* en */ om het commentaar te plaatsen. (Kleurencodering zou misschien niet goed kunnen werken hier! Druk op F12 om opnieuw de tekst te kleuren indien je een foutmelding krijgt.)

Functies en variabelen in GML

GML bevat een groot aantal van ingebouwde functies en variabelen. Met deze kun je elk gedeelte in het spel besturen. Voor alle acties zijn er corresponderende functies zodat je eigenlijk acties niet meer hoeft te gebruiken indien je liever code gebruikt. Maar er zijn veel meer functies en variabelen die aspecten in het spel besturen die niet alleen in acties gedaan kunnen worden. Dus indien je meer ingewikkelde spellen wilt maken wordt je sterk geadviseerd om de volgende

hoofdstukken door te lezen om een overzicht te krijgen van alles wat mogelijk is. Merk op dat deze variabelen en functies ook gebruikt kunnen worden als waarden voor acties geeft. Dus zelfs als je niet van plan bent om code te gaan gebruiken of scripts te schrijven, dan heb je toch voordeel bij het lezen van deze informatie.

De volgende overeenkomst wordt gebruikt in de volgende hoofdstukken. Variabele namen gemarkeerd met een * zijn alleen-lezen, dat houdt in dat hun waarden niet veranderd kan worden. Variabele namen met [0..n] na hun zijn series. De waarden die er tussen moeten liggen worden gegeven.

Berekeningen

Game Maker bevat een groot aantal functies op bepaalde dingen te verwerken. Hier is een complete lijst.

Constanten

De volgende wiskundige constanten bestaan:

`true` Gelijk aan 1.

`false` Gelijk aan 0.

`pi` Gelijk aan 3.1415...

Getallen functies

De volgende functies bestaan die omgaan met getallen.

`random(x)` Geeft een willekeurig getal tussen de 0 en de x. Het getal is altijd kleiner dan x.

`abs(x)` Geeft de absolute waarde van x.

`sign(x)` Geeft de sign van x (-1, 0 of 1).

`round(x)` Geeft x afgerond naar de dichtstbijzijnde rond getal.

`floor(x)` Geeft floor van x, dat is, x naar beneden afgerond naar een rond getal.

`ceil(x)` Geeft de ceiling van x, dat is, x naar boven afgerond naar een rond getal.

`frac(x)` Geeft de fractionele deel van x, dat is, het decimale gedeelte na de komma.

`sqrt(x)` Geeft de wortel van x. x kan niet negatief zijn.

`sqr(x)` Geeft x*x.

`power(x,n)` Geeft x tot de macht n.

`exp(x)` Geeft e tot de macht x.

ln(x) Geeft het natuurlijke logaritme van x.

log2(x) Geeft de log basis 2 van x.

log10(x) Geeft de log basis 10 van x.

logn(n,x) Geeft de log basis n van x.

sin(x) Geeft de sinus van x (x in radians).

cos(x) Geeft de cosinus van x (x in radians).

tan(x) Geeft de tangens van x (x in radians).

arcsin(x) Geeft de omgedraaide sinus van x.

arccos(x) Geeft de omgedraaide cosinus van x.

arctan(x) Geeft de omgedraaide tangens van x.

arctan2(y,x) Berekent arctan(Y/X) en geeft de hoek in het juiste kwadrant.

degtorad(x) Zet graden om in radians.

radtodeg(x) Zet radians om in graden.

min(val1,val2,val3,...) Geeft het minimum van de waarden. De functie kan maximaal 16 argumenten hebben. Ze moeten allemaal getallen zijn of allemaal een tekstreeks.

max(val1,val2,val3,...) Geeft het maximum van de waarden. De functie kan maximaal 16 argumenten hebben. Ze moeten allemaal getallen zijn of allemaal een tekstreeks.

mean(val1,val2,val3,...) Geeft het gemiddelde van de waarden. De functie kan maximaal 16 argumenten hebben. Het moeten allemaal getallen zijn.

median(val1,val2,val3,...) Geeft de mediaan van de waarden, dat is, de middelste waarde. (Als het aantal argumenten even is, dan wordt de kleinste van de 2 middelste waarden gegeven.) Deze functie kan maximaal 16 argumenten hebben. Alle argumenten moeten getallen zijn.

point_distance(x1,y1,x2,y2) Geeft de afstand tussen punt (x1,y1) en punt (x2,y2).

point_direction(x1,y1,x2,y2) Geeft de richting van punt (x1,y1) naar (x2, y2) in graden.

lengthdir_x(len,dir) Geeft de horizontale x-component van de vector afhankelijk van de aangegeven lengte en richting.

lengthdir_y(len,dir) Geeft de verticale y-component van de vector afhankelijk van de aangegeven lengte en richting.

is_real(x) Geeft aan of x een getal is (als tegenstelling van een tekstreeks).

is_string(x) Geeft aan of x is een tekstreeks (als tegenstelling van een getal).

Tekstreeks behandelende functies

De volgende functies werken met karakters en tekstreeksen.

chr(val) Geeft een tekstreeks die een karakter bevat met asci code val.

ord(str) Geeft de asci code van het eerste karakter in str.

real(str) Verandert een tekstreeks in een getal. str kan een min teken, een decimale punt en zelf een exponentieel gedeelte bevatten.

string(val) Verandert een getal waarde in een tekstreeks door gebruik te maken van een standaard formaat (geen decimale plaatsen als het een rond getal is, en anders 2 decimalen).

string_format(val,tot,dec) Verandert val in een tekstreeks gebruik makend van je eigen formaat: tot geeft aan de totaal aantal plaatsen en dec het aantal decimale plaatsen.

string_length(str) Geeft het aantal karakters in een tekstreeks.

string_pos(substr,str) Geeft de positie van substr in str (0=komt niet voor).

string_copy(str,index,count) Geeft een sub tekstreeks van str, begint op de positie index, met de lengte count.

string_char_at(str,index) Geeft het karakter in str die op de positie index staat.

string_delete(str,index,count) Geeft een kopie van str met het deel verwijderd dat start op positie index en met de lengte count.

string_insert(substr,str,index) Geeft een kopie van str met de substr toegevoegd op de positie index.

string_replace(str,substr,newstr) Geeft een kopie van str met de eerste keer dat substr voorkomt vervangen door newstr.

string_replace_all(str,substr,newstr) Geeft een kopie van str met alle keren dat substr voor komt vervangen door newstr.

string_count(substr,str) Geeft het aantal keer dat substr voorkomt in str.

string_lower(str) Geeft een kopie van str in kleine letter.

string_upper(str) Geeft een kopie van str in hoofdletters.

string_repeat(str,count) Geeft een tekstreeks een count aan kopieën van str.

string_letters(str) Geeft een tekstreeks van str alleen letters bevat.

string_digits(str) Geeft een tekstreeks van str alleen getallen bevat.

string_lettersdigits(str) Geeft een tekstreeks van str alleen letters en getallen bevat.

De volgende functies werken met het klembord voor het opslaan van tekst.

clipboard_has_text() Geeft aan of er tekst op het klembord staat.

clipboard_get_text() Geeft de huidige tekst op het klembord.

clipboard_set_text(str) Plaats de tekstreeks str op het klembord.

Werken met data en tijd

In *Game Maker* zijn er een aantal functies om te werken met data en tijd. Een datum-tijd combinatie is opgeslagen in een getal. Het deel voor de komma is het aantal dagen na 30/12/1899. Het deel na de komma is een deel van een 24 uur dag wat is verstreken. De volgende functies zijn er:

`date_current_datetime()` Geeft de datum-tijd waarde dat correspondeert met het huidige moment.

`date_current_date()` Geeft de datum-tijd waarde dat correspondeert met alleen de huidige datum (tijd wordt genegeerd).

`date_current_time()` Geeft de datum-tijd waarde dat correspondeert met alleen de huidige tijd. (datum wordt genegeerd).

`date_create_datetime(year,month,day,hour,minute,second)` Maakt de datum-tijd waarde dat correspondeert met de aangegeven datum en tijd.

`date_create_date(year,month,day)` Maakt de datum-tijd waarde dat correspondeert met de aangegeven datum.

`date_create_time(hour,minute,second)` Maakt de datum-tijd waarde dat correspondeert met de aangegeven tijd.

`date_valid_datetime(year,month,day,hour,minute,second)` Geeft of de aangegeven datum en tijd geldig is.

`date_valid_date(year,month,day)` Geeft of de aangegeven datum geldig is.

`date_valid_time(hour,minute,second)` Geeft of de aangegeven tijd geldig is.

`date_inc_year(date,amount)` Geeft een nieuwe datum dat is het amount aantal jaar na de aangegeven datum. amount moet een rond getal zijn.

`date_inc_month(date,amount)` Geeft een nieuwe datum dat is het amount aantal maanden na de aangegeven datum. amount moet een rond getal zijn.

`date_inc_week(date,amount)` Geeft een nieuwe datum dat is het amount aantal weken na de aangegeven datum. amount moet een rond getal zijn.

`date_inc_day(date,amount)` Geeft een nieuwe datum dat is het amount aantal dagen na de aangegeven datum. amount moet een rond getal zijn.

`date_inc_hour(date,amount)` Geeft een nieuwe datum dat is het amount aantal uren na de aangegeven datum. amount moet een rond getal zijn.

`date_inc_minute(date,amount)` Geeft een nieuwe datum dat is het amount aantal minuten na de aangegeven datum. amount moet een rond getal zijn.

`date_inc_second(date,amount)` Geeft een nieuwe datum dat is het amount aantal seconden na de aangegeven datum. amount moet een rond getal zijn.

`date_get_year(date)` Geeft het jaar dat correspondeert met de datum.

date_get_month(date) Geeft de maand dat correspondeert met de datum.
date_get_week(date) Geeft de week van het jaar dat correspondeert met de datum.
date_get_day(date) Geeft de dag van de maand dat correspondeert met de datum.
date_get_hour(date) Geeft het uur dat correspondeert met de datum.
date_get_minute(date) Geeft de minuut dat correspondeert met de datum.
date_get_second(date) Geeft de seconden dat correspondeert met de datum.
date_get_weekday(date) Geeft de dag van de week dat correspondeert met de datum.
date_get_day_of_year(date) Geeft de dag van het jaar dat correspondeert met de datum.
date_get_hour_of_year(date) Geeft de uur van het jaar dat correspondeert met de datum.
date_get_minute_of_year(date) Geeft de minuut van het jaar dat correspondeert met de datum.
date_get_second_of_year(date) Geeft de seconden van het jaar dat correspondeert met de datum.
date_year_span(date1,date2) Geeft het aantal jaren tussen 2 data. Het geeft incomplete jaren als een breuk.
date_month_span(date1,date2) Geeft het aantal maanden tussen twee data. Het geeft de incomplete maanden als een breuk.
date_week_span(date1,date2) Geeft het aantal weken tussen 2 data. Het geeft incomplete weken als een break.
date_day_span(date1,date2) Geeft het aantal dagen tussen 2 data. Het geeft incomplete dagen als een breuk.
date_hour_span(date1,date2) Geeft het aantal uur tussen twee data. Het geeft incomplete uren als een breuk.
date_minute_span(date1,date2) Geeft het aantal minuten tussen 2 data. Het geeft incomplete minuten als een breuk.
date_second_span(date1,date2) Geeft het aantal seconden tussen 2 data. Het geeft incomplete seconden als een breuk.
date_compare_datetime(date1,date2) Vergelijkt de twee datum-tijd waarden. Geeft -1, 0, of 1 afhankelijk van de eerste kleiner, gelijk aan, of groter is dan de tweede waarde.
date_compare_date(date1,date2) Vergelijkt de twee datum-tijd waarden waarbij alleen het datumgedeelte wordt geteld. Geeft -1, 0, of 1 afhankelijk van de eerste kleiner, gelijk aan, of groter is dan de tweede waarde.

date_compare_time(date1,date2) Vergelijkt de twee datum-tijd waarden waarbij alleen het tijdgedeelte wordt geteld. Geeft -1, 0, of 1 afhankelijk van de eerste kleiner, gelijk aan, of groter is dan de tweede waarde.
date_date_of(date) Geeft het datum gedeelte van de aangegeven datum-tijd waarde, waarbij het tijd gedeelte op 0 gezet wordt.
date_time_of(date) Geeft het tijd gedeelte van de aangegeven datum-tijd waarde, waarbij het datum gedeelte op 0 gezet wordt.
date_datetime_string(date) Geeft een tekstreeks aangegeven de gegeven datum en tijd in het standaard formaat van het systeem.
date_date_string(date) Geeft een tekstreeks aangegeven de gegeven datum in het standaard formaat van het systeem.
date_time_string(date) Geeft een tekstreeks aangegeven de gegeven tijd in het standaard formaat van het systeem.
date_days_in_month(date) Geeft het aantal dagen in de maand aangegeven bij de datum-tijd waarde.
date_days_in_year(date) Geeft het aantal dagen in het jaar aangegeven bij de aangegeven datum-tijd waarde.
date_leap_year(date) Geeft of het aangegeven jaar aangeven bij de datum-tijd waarde een schrikkeljaar is.
date_is_today(date) Geeft of de aangegeven datum-tijd waarde vandaag is.

Spel besturing

Er zijn veel variabelen en functies die je kunt gebruiken om de game play te definiëren. Deze beïnvloeden in het bijzonder de beweging en creatie van instanties, de timing, de rooms en de uitvoering van gebeurtenissen.

Informatie over spel besturing kan in de volgende pagina's worden gevonden:

Verplaatsen

Goed beschouwd is de beweging van object instanties een belangrijk aspect van spellen. Elke instantie heeft twee ingebouwde variabelen x en y die de positie van de instantie aangeven. Om precies te zijn, zij geven de plaats aan waar de oorsprong van de sprite zich bevind. Positie (0,0) is the linker bovenhoek van de room. Je kunt de positie van instanties wijzigen door de x en y variabele te wijzigen. Als je wilt dat het object gecompliceerde bewegingen moet kunnen maken moet je het zo doen. Normaal gesproken plaats je deze code in het step-event van het object.

Als het object zich met een constante snelheid en richting voortbeweegt, is er een makkelijker manier om dit te doen. Elke object instantie heeft een horizontale snelheid (`hspeed`) en een verticale snelheid (`vspeed`). Beide geven pixels per stap weer. Een positieve horizontale snelheid geeft een beweging naar rechts, een negatieve horizontale snelheid een beweging naar links. Positieve verticale snelheid is naar beneden gericht en negatieve verticale snelheid naar boven. Dus je hoeft deze variabelen (x en y) slechts een keer (bijvoorbeeld in het creatie-event) in te stellen om de object instantie een constante beweging te geven.

Er is nog een andere manier om beweging te specificeren, gebruikt makend van richting (in graden 0-359), en snelheid (mag niet negatief zijn). Je kunt deze variabelen instellen en uitlezen om zelf een beweging te specificeren. (Het programma wijzigt deze waarden naar de juiste `hspeed` en `vspeed`.) Ook is er de mogelijkheid om wrijving en zwaartekracht in te stellen. Tenslotte is er de functie `motion_add(dir,speed)` om een beweging toe te voegen aan de huidige beweging.

Om compleet te zijn, elke instantie heeft de volgende variabelen en functies die een relatie hebben met positie en beweging:

`x` De x-positie.
`y` De y-positie.
`xprevious` De vorige x-positie.
`yprevious` De vorige y-positie.
`xstart` De x-positie waarmee de instantie start in de room.
`ystart` De y-positie waarmee de instantie start in de room.
`hspeed` Horizontale component van de snelheid.
`vspeed` Verticale component van de snelheid.
`direction` De huidige richting (0-360, tegen de klok in, 0 = naar rechts).
`speed` De huidige snelheid (pixels per step).

friction Huidige wrijving (pixels per stap).
gravity Huidige hoeveelheid zwaartekracht (pixels per step).
gravity_direction Richting van de zwaartekracht (270 is naar beneden).
motion_set(dir,speed) Stelt de beweging in met de gegeven richting en snelheid.
motion_add(dir,speed) Voegt de beweging toe aan de huidige beweging (als een vectortoevoeging).

Er is een groot aantal functies beschikbaar die je help in het definiëren van bewegingen:

place_free(x,y) Geeft terug of de instantie geplaatst op positie (x,y) botsingvrij zou zijn. Dit wordt bijvoorbeeld gebruikt als controle voordat er daadwerkelijk naar de nieuwe positie wordt bewogen.
place_empty(x,y) Geeft terug of de instantie geplaatst op positie (x,y) iemand ontmoet. Dus deze functie neemt ook niet-solid instanties mee in de berekening.
place_meeting(x,y,obj) Geeft terug of de instantie geplaatst op positie (x,y) obj ontmoet. obj kan een object zijn in welk geval de functie true teruggeeft als een instantie van de object wordt ontmoet. obj kan ook een instantie id zijn, het speciale woord all betekend of de instantie een object ontmoet, of het speciale woord other.
place_snapped(hsnap,vsnap) Geeft terug of de instantie is uitgelijnd met de uitlijnafstand.
move_random(hsnap,vsnap) Beweegt de instantie naar een vrije, willekeurige, uitgelijnde positie, als de bijbehorende actie.
move_snap(hsnap,vsnap) Lijnt de instantie uit, als de bijbehorende actie.
move_wrap(hor,vert,margin) Verplaatst de instantie als het buiten de room is naar de andere zijde. `hor` geeft aan of er horizontaal moet worden verplaatst en `vert` geeft aan of er verticaal moet worden verplaatst. `margin` geeft aan hoe ver de oorsprong van de instantie buiten de room moet liggen alvorens te verplaatsen. Dus het is een marge om de room. Normaal gesproken gebruik je deze functie in de Outside-event.
move_towards_point(x,y,sp) Verplaatst de instantie met snelheid sp naar punt (x,y).
move_bounce_solid(adv) Stuitert tegen solid instanties, als de bijbehorende actie. adv geeft aan of geavanceerde stuiter moet worden gebruikt, die met schuine muren rekening houdt.
move_bounce_all(adv) Stuitert tegen alle instanties, in plaats van alleen tegen solid instanties.
move_contact_solid(dir,maxdist) Verplaatst de instantie in de richting tot

een contact positie met een solid object is gereikt. Als er geen botsing is op de huidige positie, wordt de instantie daar geplaatst dat er nog net geen botsing is. Als er al een botsing is wordt de instantie niet verplaatst. Je kunt de maximale afstand of te verplaatsen aangeven (gebruik een negatief getal voor een berekende afstand).

move_contact_all(dir,maxdist) Hetzelfde als de vorige functie, maar deze keer stop je als er een contact is met welk object ook, solid of niet.

move_outside_solid(dir,maxdist) Verplaatst de instantie in de richting tot het niet meer in een solid object ligt. Als er op de huidige positie geen botsing is wordt de instantie niet verplaatst. Ook kun je de maximale afstand aangeven (gebruik een negatief getal als je geen maximale afstand wilt).

move_outside_all(dir,maxdist) Hetzelfde als de vorige functie, maar deze keer verplaats je tot buiten elk object, solid of niet.

distance_to_point(x,y) Geeft de afstand van de botsingsrechthoek van de huidige instantie naar (x,y) terug.

distance_to_object(obj) Geeft de afstand van de instantie naar de dichtstbijzijnde instantie van object obj terug.

position_empty(x,y) Geeft terug of de positie (x,y) leeg is.

position_meeting(x,y,obj) Geeft terug of er op positie (x,y) een instantie is van obj. obj kan een object zijn, een instantie id, of een van de sleutelwoorden `self`, `other`, of `all`.

Paden

In *Game Maker* kun je paden definiëren en instanties opdracht geven zo'n pad te volgen. Hoewel je hiervoor acties kunt gebruiken, zijn er functies en variabelen die je meer mogelijkheden geven:

path_start(path,speed,endaction,absolute) Start een pad voor de huidige instantie. De `path` is de naam van het pad die je wilt starten. De `speed` is de snelheid waarmee het pad moet worden gevolgd. Een negatieve snelheid betekend dat de instantie het pad in omgekeerde volgorde volgt. `endaction` geeft aan wat er moet gebeuren aan het eind van het pad. De volgende waardes kunnen worden gebruikt:

0: stop het pad
1: ga verder vanaf de start positie van het pad (als het pad niet is gesloten wordt er naar de start positie gesprongen)
2: ga verder vanaf de huidige positie
3: keer over het pad terug, dus vermenigvuldig de snelheid met -1

Het argument `absolute` moet 'true' of 'false' zijn. Met true worden de absolute coördinaten van het pad gebruikt. Met false is het pad relatief naar de huidige positie van de instantie. Om preciezer te zijn, als de snelheid positief is, wordt het startpunt van het pad geplaatst op de huidige positie van de instantie en wordt het pad vanaf hier gevolgd. Als de snelheid negatief is wordt het laatste punt van het pad over de huidige positie van de instantie gelegd en wordt het pad vanaf hier achteruit gevolgd.

path_end() Beëindigd het volgen van een pad voor de huidige instantie.

path_index* Index van het huidige pad dat de instantie volgt. Deze kun je niet rechtstreeks wijzigen maar moet met de bovenstaande functies.

path_position Positie in het huidige pad. 0 is het begin van het pad. 1 is het eind van het pad. De waarde moet tussen 0 en 1 liggen.

path_positionprevious Vorige positie in het huidige pad. Dit kan worden gebruikt in bijvoorbeeld botsing gebeurtenissen om de positie op het pad terug te zetten naar de vorige positie.

path_speed Snelheid (in pixels per step) waarmee het pad moet worden gevolgd. Gebruik een negatieve snelheid om achteruit te bewegen.

path_orientation Oriëntatie (tegen de klok in) waarmee het pad wordt uitgevoerd. 0 is de normale oriëntatie van het pad.

path_scale Schaal van het pad. Vergroot om het pad te vergroten. 1 is de normale waarde.

path_endaction De actie die moet worden uitgevoerd aan het eind van het pad. Je kunt de bovenstaande waardes gebruiken.

Beweging plannen

Beweging plannen helpt je om instanties van een gegeven locatie naar een andere locatie te verplaatsen, ondertussen botsingen vermijdend met andere instanties (bijv. muren). Beweging plannen is een moeilijk probleem. Het is onmogelijk om algemene functies te geven die goed werken in alle situaties. Ook is het uitrekenen van botsingvrije bewegingen een tijdvretende operatie. Dus je moet voorzichtig zijn hoe en wanneer je het toepast. Houd deze opmerkingen in je achterhoofd wanneer je een van de volgende functies gebruikt.

Game Maker maakt verschillende soorten van beweging plannen mogelijk. De eenvoudigste manier laat elke instantie een stap zetten naar een bepaalde doel- positie, proberend rechtdoor te gaan als het mogelijk is, maar een andere richting te nemen als dat nodig is. Deze functies moeten worden gebruikt in het step-event van een instantie. Zij komen overeen met de beweging plannende acties die ook beschikbaar zijn:

mp_linear_step(x,y,stepsize,checkall) Deze functie laat de instantie een stap nemen recht naar de opgegeven positie (x,y). De grootte van de stap wordt aangegeven door `stepsize`. Als `checkall` 'true' is zal de instantie stoppen als het een andere instantie raakt. Als dit 'false' is stopt het alleen als het een solid instantie raakt. Merk op dat deze functie niet probeert een obstakel te ontwijken of er omheen gaat. In dat geval faalt het. De functie geeft terug of de doelpositie is bereikt.

mp_linear_step_object(x,y,stepsize,obj) Hetzelfde als de functie hier boven maar deze keer worden alleen instanties van `obj` aangemerkt als obstakel. `obj` kan een object zijn of een instantie id.

mp_potential_step(x,y,stepsize,checkall) Als de vorige functie, laat deze functie de instantie een stap voorwaarts nemen naar een bepaalde positie. Maar in dit geval probeert het obstakels te vermijden. Als de instantie in een solid instantie zal bewegen (of elke instantie als `checkall` 'true' is) zal de richting worden gewijzigd om de instantie te ontwijken en er omheen te bewegen. De zoektocht werk niet gegarandeerd met in de meeste eenvoudige gevallen zal het effectief naar het doel bewegen. De functie geeft terug of het doel is bereikt.

mp_potential_step_object(x,y,stepsize,obj) Hetzelfde als de functie hierboven maar deze keer worden alleen instanties van `obj` aangemerkt als obstakels. `obj` kan een object zijn of een instantie id.

mp_potential_settings(maxrot,rotstep,ahead,onspot) De vorige functie werkt met een aantal parameters die met deze functie kunnen worden gewijzigd. Globaal werkt de methode als volgt. Eerst wordt er geprobeerd recht naar het doel te bewegen. Het kijkt een aantal stappen vooruit die kunnen worden ingesteld met `ahead` (standaard 3). Verlaging van deze waarde betekend dat de instantie later zal starten met het wijzigen van de richting. Verhoging betekend dat de richting eerder zal worden gewijzigd. Als deze controle leidt tot een botsing wordt er gestart met het kijken naar links en naar rechts voor de beste richting. Het doet dit in stappen met de grootte `rotstep` (standaard 10). Verlaging geeft meer mogelijkheden maar is langzamer. De parameter `maxrot` is iets moeilijker uit te leggen. De instantie heeft een huidige richting. `maxrot` (standaard 30) geeft aan hoe groot de wijziging van de huidige richting mag zijn. Dus ook al kan het rechtstreeks naar het doel maar met een grote richtingsverandering zal het dus niet `maxrot` overschrijden. Als je `maxrot` groot maakt kan de instantie makkelijker het doel bereiken. Als je de waarde kleiner maakt wordt het pad vloeiender maar zou het langer kunnen duren voor het doel is bereikt (of zelfs zou kunnen mislukken). Als er geen stap kan worden gemaakt zal het gedrag afhankelijk zijn van de waarde van parameter `onspot`. Als `onspot` is 'true' (de standaard waarde),

zal de instantie draaien naar zijn doel met de waarde aangegeven door `maxrot`. Als dit 'false' is zal het helemaal niet verplaatsen. Op 'false' zetten is bruikbaar voor bijv. auto's maar verminderd de mogelijkheid om een pad te vinden.

Merk op dat de mogelijke nadering slechts plaatselijke informatie gebruikt. Dus het zal alleen een pad vinden als deze plaatselijke informatie voldoende is om de goede richting te bepalen. Het zal bijvoorbeeld (over het algemeen) mislukken om een pad te vinden door een doolhof.

De tweede soort functies rekent een botsingvrij pad uit voor de instantie. Als dit pad is berekend kun je deze toewijzen aan de instantie als route om naar het doel te bewegen. De berekening van het pad zal wat tijd nemen maar na dit zal de uitvoering van het pad snel zijn. Natuurlijk is dit alleen geldig als de situatie ondertussen onveranderd blijft. Als bijvoorbeeld obstakels veranderen moet je misschien het pad opnieuw berekenen. Merk opnieuw op dat deze functies kunnen mislukken. ***Deze functies zijn alleen beschikbaar in de geregistreerde versie van Game Maker.***

De eerste twee functies gebruiken de lineaire beweging en mogelijke veld doelen wat ook wordt gebruikt in de step functies.

`mp_linear_path(path,xg,yg,stepsize,checkall)` Deze functie berekend een rechtlijnig pad voor de instantie vanaf zijn huidige positie naar de positie (xg,yg) gebruik makend van de opgegeven stap grootte. Het gebruikt stappen als in de functie `mp_linear_step()`. Het opgegeven pad moet al bestaan en zal worden overgeschreven door het nieuwe pad. (Zie een later hoofdstuk hoe paden te maken en te verwijderen.) De functie zal teruggeven of een pad is gevonden. De functie zal stoppen en mislukking rapporteren als er geen rechtstreeks pad bestaat tussen start en doel. Als het mislukt wordt er toch een pad gemaakt dat loopt tot de positie waar de instantie is geblokkeerd.

`mp_linear_path_object(path,xg,yg,stepsize,obj)` Zelfde als de functie hierboven maar deze keer worden alleen instanties van `obj` aangemerkt als obstakels. `obj` kan een object of een instantie id zijn.

`mp_potential_path(path,xg,yg,stepsize,factor,checkall)` Deze functie berekend een pad voor de instantie vanaf zijn huidige positie en richting naar de positie (xg,yg) gebruikt maken van de opgegeven stapgrootte proberend botsing met obstakels te vermijden. Het gebruikt mogelijke veldstappen, als in de functie `mp_potential_step()` en ook de parameters die kunnen worden ingesteld met `mp_potential_settings()`. Het opgegeven pad moet al bestaan en zal worden overgeschreven door het nieuwe pad. (Zie een later hoofdstuk hoe paden te maken en te verwijderen.) De functie zal teruggegeven of een pad is

gevonden. Om te voorkomen dat de functie continu door zal rekenen moet factor groter zijn dan 1. De functie zal stoppen en mislukking rapporteren als het niet een pad kan vinden dat korter is dan de afstand tussen start en doel vermenigvuldigd met factor. Een factor van 4 is normaal goed genoeg maar als je grote omwegen verwacht is het beter factor groter te maken. Als het mislukt wordt het pad toch gemaakt in de richting van het doel maar zal deze niet bereiken.

`mp_potential_path_object(path,xg,yg,stepsize,factor,obj)` Hetzelfde als de functie hierboven maar deze keer worden allen instanties van `obj` aangemerkt als obstakels. `obj` kan een object zijn of een instantie id.

De andere functies gebruiken een veel complexer mechanisme gebruik makend van een op een grid (tabel of raster) gebaseerde benadering (ook wel A* algoritme genoemd). Het zal succesvoller zijn in het vinden van paden (hoewel het ook kan mislukken) en zal kortere paden vinden maar het vergt meer werk van jouw kant. Het globale idee is als volgt. Eerst maken we een grid op (het relevante deel van) de room. Je kunt er voor kiezen om een fijn grid (wat langzamer is) of een grof grid te gebruiken. Daarna wordt voor alle relevante object besloten welke cellen zij raken (gebruik makend van de botsingrechthoek of precieze controle). Deze cellen worden dan gemerkt als verboden. Dus een cel zal in zijn geheel verboden zijn, ook al wordt het slechts voor een klein gedeelte geraakt door een obstakel. Ten slotte specificeren we een start en een doel positie (die in vrije cellen moeten liggen) en de functie berekend het kortste pad (of bijna het kortste) tussen deze punten. Het pad zal door het midden van de cellen lopen. Als de cellen zo groot zijn dat de instantie geplaatst in het midden van een cel niet erbuiten steekt zal het succesvol zijn. Dit pad kun je nu aan een instantie geven als pad om te volgen.

De op grid gebaseerde benadering is zeer krachtig (is wordt gebruikt in veel professionele spellen) maar het verlangt van jou dat je zorgvuldig nadenkt. Je moet beslissen welk gebied en welke grootte goed genoeg zijn om het spel op te lossen. Ook moet je beslissen welke objecten moeten worden ontweken en of precieze controle belangrijk is. Al deze parameters beïnvloeden de efficiëntie van de benadering enorm.

Vooral de grootte van de cellen is cruciaal. Onthoud dat de cellen zo groot moeten zijn dat het bewegend object (geplaatst met zijn oorsprong in het midden van een cel) er niet buiten steekt. (Wees voorzichtig met de positie van de oorsprong van het object. Realiseer je ook dat je het pad kunt schuiven als de oorsprong van het object niet in het midden ligt!) Van de andere kant, hoe kleiner de cellen hoe meer mogelijke paden er ontstaan. Als je Cellen te groot maakt, zullen eventuele openingen tussen obstakels worden gesloten omdat alle cellen een obstakel hebben.

De actuele functies voor de grid gebaseerde benadering zijn als volgt:

mp_grid_create(left,top,hcells,vcells,cellwidth,cellheight) Deze functie maakt een grid. Het geeft een index terug die moet worden gebruikt in alle andere oproepen. Je kunt meerdere grids maken en beheren op hetzelfde moment. left en top gegeven de positie van de linker bovenhoek van de grid aan. hcells en vcells geven het aantal horizontale en verticale cellen aan. cellwidth en cellheight tenslotte geven de grootte van de cellen aan.

mp_grid_destroy(id) Vernietigd de aangegeven grid en haalt het uit het geheugen. Vergeet deze niet uit te voeren als je de grid niet meer gebruikt.

mp_grid_clear_all(id) Markeert alle cellen in de grid als vrij.

mp_grid_clear_cell(id,h,v) Zuivert de aangegeven cell. Cell 0,0 is de cell links boven.

mp_grid_clear_rectangle(id,left,top,right,bottom) Zuivert alle cellen die de aangegeven rechthoek raken (in room coördinaten).

mp_grid_add_cell(id,h,v) Markeert de aangegeven cell als verboden. Cell 0,0 is de cell links boven.

mp_grid_add_rectangle(id,left,top,right,bottom) Markeert alle cellen die de aangegeven rechthoek raken als verboden.

mp_grid_add_instances(id,obj,prec) Markeert alle cellen die een instantie van het obj raken als verboden. Je kunt ook een enkele instantie gebruiken door de instantie id op te geven. Ook kun je het sleutelwoord **all** gebruiken om alle instanties van alle objecten aan te geven. prec betekend of precieze botsingcontrole moet worden gebruikt (werkt alleen als precieze controle aan is gezet voor de sprite van de instantie).

mp_grid_path(id,path,xstart,ystart,xgoal,ygoal,allowdiag) Berekend een pad door de grid. path moet een bestaand pad zijn dat zal worden vervangen door het berekend pad. xstart en ystart geven de start van het pad en xgoal en ygoal het doel aan. allowdiag geeft aan of diagonale bewegingen zijn toegestaan of alleen horizontale en verticale. De functie geeft terug of het succesvol een pad heeft gevonden. (Merk op dat het pad onafhankelijk is van de huidige instantie; Het is een pad door de grid, niet een pad voor een specifieke instantie.)

mp_grid_draw(id) Deze functie tekent de grid met groene cellen voor lege en rode cellen voor verboden cellen. Deze functie is langzaam en alleen geleverd als 'debug' hulpmiddel.

Botsing controle

Wanneer beweging wordt gepland of goede acties moeten worden gezocht, is het vaak belangrijk om te weten of er op bepaalde plaatsen botsingen met andere objecten zijn. De volgende routines

kunnen hier voor worden gebruikt. Ze hebben allemaal drie algemene argumenten: Het argument `obj` kan een object zijn, het sleutelwoord **`all`**, of the id van een instantie. Het argument `prec` geeft aan of de controle precies moet zijn of slechts gebaseerd op de botsingrechthoek van de instantie. Precieze controle wordt alleen gedaan als de sprite van de instantie precieze controle aan heeft staan. Het argument `notme` kan op 'true' worden gezet om aan te geven dat de opgeroepen instantie niet hoeft te worden gecheckt. Al deze functies geven of the id van een van de instanties die botsen, of een negatieve waarde als er geen botsing is terug.

`collision_point(x,y,obj,prec,notme)` Deze functie test of op punt (x,y) een botsing is met onderdelen van object obj.

`collision_rectangle(x1,y1,x2,y2,obj,prec,notme)` Deze functie test of er een botsing is tussen de (opgevulde) rechthoek met de aangegeven tegenoverstaande hoeken en onderdelen van object obj. Je kunt dit bijvoorbeeld gebruiken om te testen of een gebied vrij van obstakels is.

`collision_circle(xc,yc,radius,obj,prec,notme)` Deze functie test of er een botsing is tussen de (opgevulde) cirkel gecentreerd op positie (xc,yc) met de gegeven straal en onderdelen van object obj. Je kunt dit bijvoorbeeld gebruiken om te test of er een object dicht bij een bepaalde locatie is.

`collision_ellipse(x1,y1,x2,y2,obj,prec,notme)` Deze functie test of er een botsing is tussen de (opgevulde) ellips met de aangegeven tegenoverstaande hoeken en onderdelen van object obj.

`collision_line(x1,y1,x2,y2,obj,prec,notme)` Deze functie test of er een botsing is tussen het lijnstuk van (x1,y1) tot (x2,y2) en onderdelen van object obj. Dit is een krachtige functie. Je kunt deze bijv. gebruiken om te testen of een instantie een andere instantie kan zien of dat er een muur tussen zit.

Instanties

In het spel zijn de basis eenheden de instanties van de verschillende objecten. Tijdens het spelen kun je een aantal aspecten van deze instanties wijzigen. Ook kun je nieuwe instanties aanmaken en instanties verwijderen. Naast de aan beweging gerelateerde variabelen hierboven besproken en de aan tekenen gerelateerde variabelen hieronder besproken, heeft elke instantie de volgende variabelen:

`object_index*` Index van het object waar dit een instantie van is. Deze variabele kan niet worden gewijzigd.

`id*` De unieke herkenning voor de instantie (>= 100000). (Merk op dat als er rooms worden gemaakt het id van een instantie onder de muis altijd wordt weergegeven.)

`mask_index` Index van de sprite gebruikt als mask (scherm) voor botsingen. Geef

dit een waarde van -1 om het hetzelfde te maken als de sprite_index.
solid Of de instantie solid is. Dit kan worden gewijzigd tijdens het spel.
persistent Of de instantie blijvend is en opnieuw zal verschijnen als er naar een andere room wordt gegaan. Vaak zet je 'blijvend' op bepaalde momenten uit. (Bijvoorbeeld als je naar de eerste room gaat.)

Er is een probleem met het gebruik van instanties. Het is niet zo makkelijk een enkele instantie te identificeren. Zij hebben geen naam. Als er slechts een instantie van een bepaald object is kun je de object naam gebruiken maar anders moet je het id van de instantie gebruiken. Dit is een unieke aanduider voor de instantie. Je kunt dit gebruiken in **with** statements and als een object aanduider. Gelukkig zijn er een aantal variabelen en routines die je helpen instantie id's te vinden.

instance_count* Aantal instanties die momenteel in de room bestaan.
instance_id[0..n-1]* De instantie id van een bepaalde instantie. Hier is n het aantal instanties.

Merk op dat de aanwijzing van de instanties van de instantie id's elke stap wijzigen, dus je kunt geen waarden van vorige stappen gebruiken. Laat me een voorbeeld geven. Stel dat elke unit in je spel een bepaalde kracht heeft en je wilt de sterkste hebben. Je kunt dan de volgende code gebruiken:

```
{
  maxid = -1;
  maxpower = 0;
  for (i=0; i<instance_count; i+=1)
  {
    iii = instance_id[i];
    if (iii.object_index == unit)
    {
      if (iii.power > maxpower)
        {maxid = iii; maxpower = iii.power;}
    }
  }
}
```

Na de lus zal maxid het id bevatten van de unit met de grootste kracht. (Verwijder geen instanties tijdens zo'n lus omdat ze automatisch zullen worden verwijderd uit de reeks en tot gevolg hebben dat je instanties overslaat.)

instance_find(obj,n) Geeft het id van de (n+1)'de instantie van type obj terug. obj kan een object zijn of het sleutelwoord all. Als deze niet bestaat wordt het speciale object noone teruggegeven. Merk op dat de aanduiding van de instantie tot de instantie id's elke stap wijzigen dus je kunt geen waarden van vorige stappen gebruiken.

instance_exists(obj) Geeft terug of een instantie van het type obj bestaat. obj kan een object zijn, een instantie id, of het sleutelwoord all.

instance_number(obj) Geeft het aantal instanties van type obj terug. obj kan een object of het sleutelwoord all zijn.

instance_position(x,y,obj) Geeft het id van de instantie van het type obj op positie (x,y) terug. Als er meerdere instanties op die positie zijn wordt de eerste teruggegeven. obj kan een object zijn of het sleutelwoord all. Als er geen instantie bestaat op (x,y) wordt het speciale object noone teruggegeven.

instance_nearest(x,y,obj) Geeft het id van de instantie van type obj het dichtst bij (x,y) terug. obj kan een object zijn of het sleutelwoord all.

instance_furthest(x,y,obj) Geeft het id van de instantie van type obj het verst van (x,y) terug. obj kan een object zijn of het sleutelwoord all.

instance_place(x,y,obj) Geeft het id van de instantie van type obj terug die de huidige instantie zal ontmoeten als deze op positie (x,y) wordt geplaatst. obj kan een object zijn of het sleutelwoord all. Als deze niet bestaat wordt het speciale object noone teruggegeven.

De volgende functies kunnen worden gebruikt om instanties te maken of te verwijderen.

instance_create(x,y,obj) Maakt een instantie van obj op positie (x,y). De functie geeft het id van de nieuwe instantie terug.

instance_copy(performevent) Maakt een kopie van de huidige instantie. Het argument geeft aan of het create-event moet worden uitgevoerd voor de kopie. De functie geeft het id van de kopie terug.

instance_destroy() Verwijderd de huidige instantie.

instance_change(obj,perf) Wijzigt de instantie in obj. perf geeft aan of de create- en destroy-events moeten worden uitgevoerd of niet.

position_destroy(x,y) Verwijderd alle instanties wiens sprite positie (x,y) bevatten.

position_change(x,y,obj,perf) Wijzigt alle instanties op (x,y) in obj. perf geeft aan of de create- en destroy-events moeten worden uitgevoerd.

Deactiveren van instanties

Als je grote rooms maakt, bijvoorbeeld in platform spellen, met een klein beeld, zullen veel instanties buiten het beeld liggen. Zulke instanties zijn toch actief en zullen hun gebeurtenissen (events) uitvoeren. Dit kan veel tijd kosten, wat vaak niet nodig is. (Het is bijvoorbeeld vaak niet nodig dat instanties buiten het beeld zich bewegen.) Om dit probleem op te lossen heeft *Game Maker* enkele functies om instanties te deactiveren en te activeren. Voordat je deze gebruikt moet je het je duidelijk zijn hoe ze werken.

Als je instanties deactiveert kun je je voorstellen dat ze zijn verwijderd uit het spel. Ze zijn niet meer zichtbaar en ze zullen geen gebeurtenissen meer uitvoeren. Dus voor alle acties en functies bestaan ze niet meer. Als je bijvoorbeeld alle instanties van een bepaald type verwijderd, zijn geactiveerde instanties niet verwijderd (omdat ze niet lijken te bestaan). Dus denk niet dat als een speler een sleutel oppakt hij hiermee een gedeactiveerde deur kan openen.

De meest cruciale misstap die je kunt maken is het deactiveren van de instantie die verantwoordelijk is voor het activeren. Om dit te vermijden geven sommige van de onderstaande routines de mogelijkheid om zichzelf niet te deactiveren.

Hier zijn de beschikbare routines:

`instance_deactivate_all(notme)` Deactiveert alle instanties in de room. Als notme 'true' is zal de instantie zelf niet worden gedeactiveerd (dit is gewoonlijk wat je wilt).
`instance_deactivate_object(obj)` Deactiveert alle instanties in de room van het gegeven object. Je kunt ook all gebruiken om aan te geven dat alle instanties moeten worden gedeactiveerd of het id van een instantie voor een enkele instantie.
`instance_deactivate_region(left,top,width,height,inside,notme)` Deactiveert alle instanties in de aangegeven rechthoek (dat is, die wiens botsing rechthoek (gedeeltelijk) in rechthoek ligt). Als inside false is worden de instanties die compleet buiten de rechthoek liggen gedeactiveerd. Als notme true is wordt de instantie zelf niet gedeactiveerd (wat je normaal is).
`instance_activate_all()` Activeert alle instanties in de room.
`instance_activate_object(obj)` Activeert alle instanties van het gegeven object in de room. Je kunt ook all gebruiken om aan te geven dat alle instanties moeten worden geactiveerd of het id van een instantie om een enkele instantie te activeren.
`instance_activate_region(left,top,width,height,inside)` Activeert alle instanties in de aangegeven rechthoek. Als inside false is worden de instanties die compleet buiten de rechthoek liggen geactiveerd.

Om bijvoorbeeld alle instanties buiten het beeld te deactiveren en die binnen het beeld te activeren, kun je de volgende code in de step-gebeurtenis van de bewegende figuur plaatsen:

```
{
  instance_activate_all();
  instance_deactivate_region(view_xview[0],view_yview[0],

view_wview[0],view_hview[0],false,true);
}
```

Vaak wordt er een iets grotere rechthoek gebruikt dan de grootte van het beeld.

Tijdmeting

Goede spellen hebben zorgvuldige tijdmeting nodig wanneer er dingen gebeuren. Gelukkig doet *Game Maker* de meeste tijdmeting voor je. Het zorgt er voor dat dingen in een constant tempo gebeuren. Dit tempo is gedefinieerd als de room wordt gemaakt. Maar je kunt het wijzigen met de algemene variabele room_speed. Je kunt bijvoorbeeld de snelheid van het spel langzaam ophogen, om het moeilijker te maken, door elke stap room_speed met een klein getal op te hogen (bijv. 0.001). Als je computer langzaam is kan deze het misschien niet presteren. Dit kan worden getest met de variabele fps die voortdurend het huidige aantal beelden per seconde bijhoudt. Tenslotte, voor wat geavanceerde tijdmeting kun je de variabele current_time gebruiken die het aantal milliseconden (1/1000 sec.) geeft vanaf dat de computer is gestart. Hier is de totale collectie variabelen beschikbaar (alleen de eerste kan worden gewijzigd):

room_speed Snelheid van het spel in de huidige room (in stappen per seconde).
fps* Aantal beelden per seconde die daadwerkelijk worden getekend.
current_time* Aantal milliseconden die zijn gepasseerd vanaf dat het systeem is gestart.
current_year* Het actuele jaar.
current_month* De actuele maand.
current_day* De actuele dag.
current_weekday* De actuele dag van de week (1=zondag, ..., 7=zaterdag).
current_hour* Het actuele uur.
current_minute* De actuele minuut.
current_second* De actuele seconde.

Soms wil je het spel voor een korte tijd stoppen. Hier kun je de sleep functie voor gebruiken.

sleep(numb) Slaap numb milliseconden.

Zoals je waarschijnlijk weet, heeft elke instantie 12 verschillende alarm klokken die je kunt instellen. Om waardes te wijzigen (of te krijgen) van de verschillende alarm klokken kun je volgende variabelen gebruiken:

alarm[0..11] Waarde van de aangegeven alarm klok. (Merk op dat alarm klokken alleen worden bijgehouden als de alarm-gebeurtenis van het object acties bevat!)

We hebben gezien dat je voor complexe tijdmeting vraagstukken tijdlijnen kunt gebruiken. Elke instantie kan een tijdlijn aan zich hebben gekoppeld. De volgende variabelen hebben een verband hier mee:

timeline_index Actuele index van de tijdlijn geassocieerd met de instantie. Je kunt deze instellen op een bepaalde tijdlijn om te gebruiken. Zet op -1 om het gebruiken van een tijdlijn voor een instantie te stoppen.
timeline_position Actuele positie in de tijdlijn. Je kunt dit wijzigen om bepaalde delen over te slaan of opnieuw uit te voeren.
timeline_speed Normaal wordt elke stap de positie van de tijdlijn vermeerderd met 1. Je kunt deze waarde wijzigen door deze variabele op een andere waarde te zetten. Je kunt ook niet-hele getallen als 0.5 gebruiken. Als de waarde groter is dan 1, kunnen er meerdere moments gebeuren in de zelfde stap. Zij zullen allemaal worden uitgevoerd in de goede volgorde, dus er zullen geen acties worden overgeslagen.

Rooms

Spellen werken in rooms. Elke room heeft een index die is gekoppeld aan de naam van de room. De actuele room is opgeslagen in de variabele room. Je kunt niet veronderstellen dat rooms zijn genummerd in de volgorde waarin ze worden uitgevoerd. Dus ga niet de variabele room vermeerderen of verminderen met een getal. In de plaats daarvan worden de onderstaande functies en variabelen gebruikt. Een stuk code zou er bijvoorbeeld zo uit kunnen zien:

```
{
  if (room != room_last)
  {
    room_goto_next();
  }
```

```
    else
    {
      game_end();
    }
}
```

De volgende variabelen en functies handelen over rooms.

room Index van de actuele room; kan worden gewijzigd om naar een andere room te gaan, maar beter is de onderstaande routines te gebruiken.
room_first* Index van de eerste room in het spel.
room_last* Index van de laatste room in het spel.
room_goto(numb) Ga naar de room met index numb.
room_goto_previous() Ga naar de vorige room.
room_goto_next() Ga naar de volgende room.
room_restart() Herstart de actuele room.
room_previous(numb) Geeft de index terug van de room voor numb (-1 = geen) maar ga er niet heen.
room_next(numb) Geeft de index terug van de room na numb (-1 = geen).
game_end() Beëindigd het spel.
game_restart() Herstart het spel.

Als een van bovenstaande functies worden opgeroepen om de room te wijzigen of het spel te herstarten of te beëindigen, realiseer je dat deze wijziging niet op datzelfde moment gebeurd. Het gebeurd nadat de actuele acties in zijn geheel is uitgevoerd. Dus de rest van het script wordt toch uitgevoerd, en het zelfde geldt voor eventuele opgeroepen scripts.

Rooms hebben a aantal eigenschappen meer:

room_width* Breedte van de room in pixels.
room_height* Hoogte van de room in pixels.
room_caption Titel regel voor de room die wordt getoond in de titel van het venster.
room_persistent Of de actuele room blijvend is.

Veel spellen bieden de speler de mogelijkheid het spel op te slaan en een opgeslagen spel te laden. In *Game Maker* gebeurd dit automatisch als de speler op <F5> drukt voor opslaan en <F6> voor

laden. Je kunt spellen opslaan en laden met een stuk code (merk op dat laden pas gebeurt aan het eind van de actuele stap).

game_save(string) Slaat het spel op in het bestand met de name tekenreeks.
game_load(string) Laadt het spel van het bestand met de naam tekenreeks.

Realiseer je dat alleen het basis spel wordt opgeslagen. Als je bijvoorbeeld een bepaald stuk muziek afspeelt, wordt de precieze positie van de muziek niet opgeslagen. Ook worden gewijzigde hulpbronnen niet opgeslagen. Andere dingen die niet worden opgeslagen zijn de inhoud van gegevensstructuren, particles, en multiplayer instellingen.

Score

Andere belangrijke aspecten van veel spellen zijn de score, de gezondheid en het aantal levens. *Game Maker* houd de score bij in de algemene variabele `score` en het aantal levens in de algemene variabele `lives`. Je kunt de score wijzigen door eenvoudig de waarde van deze variabele te wijzigen. Hetzelfde geldt voor `health` en `lives`. Als lives groter is dan 0 en kleiner of gelijk wordt aan 0 wordt de no-more-lives gebeurtenis uitgevoerd voor alle instances. Als je de score niet wilt laten zien in de titel, zet de variabele `show_score`, etc op false. Ook kun je de titel wijzigen. Voor meer ingewikkelde spellen kun je het beste de score zelf laten tekenen.

score De actuele score.
lives Het aantal levens.
health De actuele gezondheid (0-100).
show_score Of de score wordt worden vermeld in de venster titel.
show_lives Of het aantal levens moet worden vermeld in de venster titel.
show_health Of de gezondheid moet worden vermeld in de venster titel.
caption_score De titel gebruikt voor de score.
caption_lives De titel gebruikt voor het aantal levens.
caption_health De titel gebruikt voor de gezondheid.

Gebeurtenissen aanmaken

Zoals je weet, wordt *Game Maker* geheel door gebeurtenissen (events) aangestuurd. Alle acties gebeurden als resultaat van gebeurtenissen. Er zijn een aantal verschillende gebeurtenissen. Aanmaak en verwijder gebeurtenissen gebeuren als een instantie is gemaakt of verwijderd. In elke stap, behandeld het systeem eerst de alarm gebeurtenissen. Daarna de toetsenbord en muis gebeurtenissen en daarna de step-gebeurtenis. Nu worden de botsing-gebeurtenissen bekeken, waarna de instanties op hun nieuwe positie staan. Ten slotte wordt de draw-gebeurtenis gebruikt om de instanties te tekenen (merk op dat als er meerder views zijn de draw-gebeurtenis meerdere

keren per stap wordt uitgevoerd). Je kunt ook een gebeurtenis laten uitvoeren met behulp van een stuk code. De volgende functies bestaan:

event_perform(type,numb) Voert gebeurtenis numb van het aangegeven type van de actuele instantie uit. De volgende gebeurtenis soorten zijn aangemerkt:

ev_create
ev_destroy
ev_step
ev_alarm
ev_keyboard
ev_mouse
ev_collision
ev_other
ev_draw
ev_keypress
ev_keyrelease

Als er meerdere gebeurtenissen van het gegeven type zijn, kan numb worden gebruikt om de precieze gebeurtenis aan te geven. Voor de alarm gebeurtenissen is dit 0 tot 11. Voor de toetsenbord gebeurtenissen kun je de toetscode van de toets gebruiken. Voor muis (en joystick) gebeurtenissen kun je de volgende constanten gebruiken:

ev_left_button
ev_right_button
ev_middle_button
ev_no_button
ev_left_press
ev_right_press
ev_middle_press
ev_left_release
ev_right_release
ev_middle_release
ev_mouse_enter
ev_mouse_leave
ev_mouse_wheel_up
ev_mouse_wheel_down
ev_global_left_button
ev_global_right_button
ev_global_middle_button

ev_global_left_press

ev_global_right_press

ev_global_middle_press

ev_global_left_release

ev_global_right_release

ev_global_middle_release

ev_joystick1_left

ev_joystick1_right

ev_joystick1_up

ev_joystick1_down

ev_joystick1_button1

ev_joystick1_button2

ev_joystick1_button3

ev_joystick1_button4

ev_joystick1_button5

ev_joystick1_button6

ev_joystick1_button7

ev_joystick1_button8

ev_joystick2_left

ev_joystick2_right

ev_joystick2_up

ev_joystick2_down

ev_joystick2_button1

ev_joystick2_button2

ev_joystick2_button3

ev_joystick2_button4

ev_joystick2_button5

ev_joystick2_button6

ev_joystick2_button7

ev_joystick2_button8

Voor de botsing gebeurtenissen geef je de index van het andere object. Ten slotte, voor de andere gebeurtenis kunnen de volgende constanten worden gebruikt:

ev_outside

ev_boundary

ev_game_start

ev_game_end

ev_room_start

ev_room_end

ev_no_more_lives

ev_no_more_health

ev_animation_end

ev_end_of_path

ev_user0

ev_user1

ev_user2

ev_user3

ev_user4

ev_user5

ev_user6

ev_user7

ev_user8

ev_user9

ev_user10

ev_user11

ev_user12

ev_user13

ev_user14

ev_user15

Voor de step-gebeurtenis bestaat de index uit een van de volgende constanten:

ev_step_normal

ev_step_begin

ev_step_end

event_perform_object(obj,type,numb) Deze functie werkt het zelfde als de functie hierboven behalve dat je deze keer gebeurtenissen kunt aangeven in andere objecten. Merk op dat de acties in deze gebeurtenissen worden uitgevoerd voor de huidige instantie, niet voor de instanties van het opgegeven object.
event_user(numb) In de other-gebeurtenissen kun je ook 16 door de gebruiker gedefinieerde gebeurtenissen gebruiken. Deze worden alleen uitgevoerd als je ze oproept in met deze functie. numb moet tussen de 0 en 15 liggen.
event_inherited() Voert de geërfde gebeurtenis uit. Dit werkt alleen als de instantie een ouder (parent) object heeft.

Je kunt informatie krijgen over de actuele gebeurtenis die wordt uitgevoerd met behulp van de volgend alleen-lezen variabelen:

event_type* Type gebeurtenis die wordt uitgevoerd.

event_number* Nummer van de actuele gebeurtenis die wordt uitgevoerd.

event_object* De object index waarvan de actuele gebeurtenis wordt uitgevoerd.

event_action* De index van de actie die momenteel wordt uitgevoerd (0 is de eerste in de gebeurtenis, enz.).

Overige variabelen en functies

Hier zijn enkele variabelen en functies die verband hebben met fouten (errors).

error_occurred Geeft aan of er een fout is voorgekomen

error_last Regel die het laatste fouten bericht bevat

show_debug_message(str) Laat de tekenreeks in debug mode zien

De volgende functies bestaan die je toestaan om te controleren of bepaalde variabelen bestaan en waarmee je variabelen kunt instellen en hun waarde kunt krijgen. In al deze functies is de variabele name gebruikt als een tekenreeks!

variable_global_exists(name) Geeft terug of een algemene variabele met de gegeven naam (een tekenreeks) bestaat.

variable_local_exists(name) Geeft terug of een lokale variabele met de gegeven naam (een tekenreeks) bestaat voor de actuele instantie.

variable_global_get(name) Geeft de waarde van de algemene variabele met de gegeven naam (een tekenreeks) terug.

variable_global_array_get(name,ind) Geeft de waarde van index ind van de algemene reeksvariabele met de gegeven naam (een tekenreeks) terug.

variable_global_array2_get(name,ind1,ind2) Geeft de waarde van index ind1,ind2 van de algemene 2-dimensionale reeksvariabele terug met de gegeven naam (een tekenreeks).

variable_local_get(name) Geeft de waarde van de lokale variabele terug met de gegeven naam (een tekenreeks).

variable_local_array_get(name,ind) Geeft de waarde van index ind van de lokale reeksvariabele met de gegeven naam (een tekenreeks) terug.

variable_local_array2_get(name,ind1,ind2) Geeft de waarde van index ind1,ind2 van de lokale 2-dimensionale reeksvariabele terug met de gegeven naam (een tekenreeks).

variable_global_set(name,value) Stelt de algemene variabele met de gegeven naam (een tekenreeks) in met de gegeven waarde.

variable_global_array_set(name,ind,value) Stelt de index ind in de

algemene reeksvariabele met de gegeven naam (een tekenreeks) in met de gegeven waarde.

variable_global_array2_set(name,ind1,ind2,value) Stelt de index ind1,ind2 in van de algemene 2-dimensionale reeksvariabele in met de gegeven naam (een tekenreeks) met de gegeven waarde.

variable_local_set(name,value) Stelt de lokale variabele met de gegeven naam (een tekenreeks) in met de gegeven waarde.

variable_local_array_set(name,ind,value) Stelt de index ind in de lokale reeksvariabele met de gegeven naam (een tekenreeks) in met de gegeven waarde.

variable_local_array2_set(name,ind1,ind2,value) Stelt de index ind1,ind2 in de lokale 2-dimensionale reeksvariabele met de gegeven naam (een tekenreeks) in met de gegeven waarde.

Je kunt bijvoorbeeld schrijven:

```
{
  if variable_global_exists('ammunition')
    global.ammunition += 1
  else
    global.ammunition = 0
}
```

Je kunt deze functies ook gebruiken om variabelen in een script aan te geven in een soort van verwijzende manier, door hun namen aan te geven als een tekenreeks en de functies te gebruiken om ze te wijzigen.

Je kunt de programma belangrijkheid wijzingen met de volgende functie:

set_program_priority(priority) Stelt de voorrang id van het programma in. Je kunt een waarde tussen -3 en +3 aangeven. Een waarde van -3 betekend dat het programma alleen zal lopen als er geen andere processen tijd vragen, of anders gezegd, wanneer alle andere processen niets doen. Waarden van -2 en -1 zijn onder normaal, dus andere processen krijgen voorrang. 0 is de normale waarde. +1 en +2 geven een hogere voorrang id, wat kan resulteren in een hogere snelheid en een vloeiender spel. Maar andere processen zullen veel minder proces tijd krijgen. +3 geeft een real-time modus aan. In real-time modus wordt in principe alle tijd toegewezen aan het spel. Dit kan leiden tot ernstige problemen met andere toepassingen die lopen op de computer. Ook toetsenbord-gebeurtenissen en bijv.

het klikken op de afsluitknop zullen soms niet worden uitgevoerd door Windows. Dus gebruik dit alleen als je alle processortijd wilt hebben. Het is beter eerst voorzichtig te proberen voordat je het gebruikt en sla het spel op voordat je het speelt.

Gebruiker interactie

Er is geen enkel spel zonder interactie met de gebruiker. De normale manier om dit in *Game Maker* te doen, is door acties te plaatsen in muis- of toetsenbordevents. Maar soms heb je meer controle nodig. Vanuit een stuk code kun je controleren of bepaalde toetsen op het toetsenbord ingedrukt zijn en je kan de muispositie controleren en tevens kijken of knoppen zijn ingedrukt. Meestal controleer je deze zaken in het step-event van een controllerobject en onderneem je de gewenste acties.

Het toetsenbord

De volgende variabelen en functies bestaan voor keyboardinteractie:

`keyboard_lastkey` Code van de laatste ingedrukte toets. Kijk lager voor toetscodes. Het is mogelijk deze waarde te veranderen. Zet hem bijv. naar nul als je de waarde niet meer nodig hebt.
`keyboard_key` Code van de toets die momenteel is ingedrukt. Kijk lager voor toetscodes. 0 betekent dat er geen toets is ingedrukt.
`keyboard_lastchar` Laatste ingedrukte toets (als string).
`keyboard_string` Een string die maximaal de laatste 1024 karakters bevat. Deze string bevat alleen printbare karakters. Het reageert ook op de juiste manier wanneer je backspace gebruikt, dus door het laatste karakter te wissen.

Soms is het handig om toetsen te kopiëren naar andere toetsen. Het kan bijvoorbeeld zo zijn dat de speler de pijltjestoetsen en de numpadtoetsen mag gebruiken. In dat geval is het niet handig alle acties te kopiëren, maar is het mogelijk om de pijltjestoetsen als het ware te kopiëren naar de numpadtoetsen. Ook kan je een systeem inbouwen waarmee de speler zelf kan kiezen met welke toetsen hij speelt. De volgende bijbehorende functies zijn beschikbaar:

`keyboard_set_map(key1,key2)` Kopieert de eigenschappen van de toets met de toetscode key1 naar de toets met de toetscode key2.
`keyboard_get_map(key)` Geeft de toetscode van de toets waarvan de eigenschappen gekopieerd zijn.
`keyboard_unset_map()` Wist alle gekopieerde eigenschappen.

Om te controleren of een bepaalde toets of muistoets is ingedrukt kun je de volgende functies gebruiken. Dit is voornamelijk handig wanneer meerdere toetsen tegelijk worden ingedrukt.

keyboard_check(key) Geeft aan of een bepaalde toets is ingedrukt.
keyboard_check_pressed(key) Geeft aan of een toets sinds de vorige stap ingedrukt is.
keyboard_check_released(key) Geeft aan of een toets sinds de vorige stap is losgelaten.
keyboard_check_direct(key) Geeft aan of een bepaalde toets is ingedrukt door direct de hardware te controleren. Het resultaat houdt geen rekening met de applicatie die momenteel voorrang heeft. Het heeft toegang tot een aantal extra controles. Je kan bijvoorbeeld de keyboardcodes vk_lshift, vk_lcontrol, vk_lalt, vk_rshift, vk_rcontrol, en vk_ralt gebruiken om te controleren of de linker- of rechtershift, -control of -alt is ingedrukt.

De volgende functies kunnen worden gebruikt om de status van het toetsenbord te veranderen:

keyboard_get_numlock() Geeft aan of numlock aanstaat.
keyboard_set_numlock(on) Zet numlock aan (true) of uit (false).
keyboard_key_press(key) Doet net alsof de toets met toetscode key ingedrukt is.
keyboard_key_release(key) Doet net alsof de toets met toetscode key losgelaten wordt.

De volgende virtuele toetscodes bestaan:

vk_nokey toetscode die aangeeft dat er geen toets is ingedrukt
vk_anykey toetscode die aangeeft dat er minimaal één toets is ingedrukt
vk_left toetscode voor pijltjestoets 'links'
vk_right toetscode voor pijltjestoets 'rechts'
vk_up toetscode voor pijltjestoets 'boven'
vk_down toetscode voor pijltjestoets 'onder'
vk_enter entertoets
vk_escape escapetoets
vk_space spacetoets
vk_shift shifttoets
vk_control controltoets
vk_alt alttoets
vk_backspace backspacetoets
vk_tab tabtoets

vk_home hometoets

vk_end endtoets

vk_delete deletetoets

vk_insert inserttoets

vk_pageup pageuptoets

vk_pagedown pagedowntoets

vk_pause pausetoets/breaktoets

vk_printscreen printscreentoets/sysrqtoets

vk_f1 ... vk_f12toetscodes voor de toetsen F1 tot en met F12

vk_numpad0 ... vk_numpad9 toetscodes voor de numpadgetallen

vk_multiply vermenigvuldigtoets in het numerieke gedeelte van het toetsenbord

vk_divide deeltoets in het numerieke gedeelte van het toetsenbord

vk_add plustoets in het numerieke gedeelte van het toetsenbord

vk_subtract mintoets in het numerieke gedeelte van het toetsenbord

vk_decimal toets van het decimale teken in het numerieke gedeelte van het toetsenbord

Gebruik voor de lettertoetsen bijvoorbeeld dit `ord('A')`. (De hoofdletters!) Gebruik voor de cijfertoetsen bijvoorbeeld dit `ord('5')` om de <5> toets te krijgen. De volgende constanten kunnen alleen gebruikt worden in `keyboard_check_direct`:

vk_lshift linker shifttoets

vk_lcontrol linker controltoets

vk_lalt linker alttoets

vk_rshift rechter shifttoets

vk_rcontrol rechter controltoets

vk_ralt rechter alttoets

Je kan bijvoorbeeld de volgende code gebruiken in de step-event van het object, als je het object wil controleren met de pijltjestoetsen:

```
{
  if (keyboard_check(vk_left))  x -= 4;
  if (keyboard_check(vk_right)) x += 4;
  if (keyboard_check(vk_up))    y -= 4;
  if (keyboard_check(vk_down))  y += 4;
}
```

Natuurlijk is dit een stuk makkelijker met behulp van de toetsenbordevents.

Er zijn nog enkele functies met betrekking tot toetsenbord interactie.

keyboard_clear(key) Neutraliseert de status van de toets. Dit betekent dat het geen toetsenbord-events meer genereert, totdat de toets opnieuw wordt ingedrukt.
io_clear() Neutraliseert de status van alle toetsen.
io_handle() Update de status van alle toetsenbord- en muistoetsen.
keyboard_wait() Wacht totdat de gebruiker een toets indrukt.

De muis

De volgende variabelen en functies bestaan voor muisinteractie:

mouse_x* X-coordinaat van de muis in de room. Kan niet worden aangepast.
mouse_y* Y-coordinaat van de muis in de room. Kan niet worden aangepast.
mouse_button De muistoets die momenteel is ingedrukt. Gebruik als waarden: `mb_none`, `mb_any`, `mb_left`, `mb_middle`, of `mb_right`.
mouse_lastbutton Laatst ingedrukte muistoets.

Om na te gaan of een bepaalde toets is ingedrukt bestaan de volgende functies. Dit is met name handig als meerdere toetsen herhaaldelijk worden ingedrukt.

mouse_check_button(numb) Geeft terug of de aangegeven muistoets momenteel ingedrukt is (gebruik als waarden: `mb_none`, `mb_left`, `mb_middle`, of `mb_right`).
mouse_check_button_pressed(numb) Geeft terug of de aangegeven muistoets sinds de vorige stap is ingedrukt.
mouse_check_button_released(numb) Geeft terug of de aangegeven muistoets sinds de vorige stap is losgelaten.

Er zijn enkele extra functies met betrekking tot muis interactie.

mouse_clear(button) Neutraliseert de status van de aangegeven muistoets. Dit betekent dat het geen muisevents meer genereert, totdat de gebruiker het opnieuw indrukt.
io_clear() Neutraliseert de status van alle muistoetsen.
io_handle() Update de status van alle toetsenbord- en muistoetsen.
mouse_wait() Wacht totdat de gebruiker een toets indrukt.

De joystick

Er zijn sommige events geassocieerd met joysticks. Maar om gehele controle te krijgen over de joysticks, is er een hele set functies. *Game Maker* ondersteunt maximaal twee joysticks. Dus alle functies hebben een joystick-id nodig als argument.

`joystick_exists(id)` Geeft terug of joystick id (1 of 2) bestaat.

`joystick_name(id)` Geeft de naam terug van de joystick.

`joystick_axes(id)` Geeft de hoeveelheid assen van de joystick terug.

`joystick_buttons(id)` Geeft de hoeveelheid toetsen van de joystick terug.

`joystick_has_pov(id)` Geeft terug of de joystick point-of-view eigenschappen heeft.

`joystick_direction(id)` Geeft de keycode (vk_numpad1 to vk_numpad9) terug die overeenkomt met de richting van joystick id (1 of 2).

`joystick_check_button(id,numb)` Geeft terug of de aangegeven joysticktoets is ingedrukt (numb heeft bereik 1-32).

`joystick_xpos(id)` Geeft de positie van de x-as (-1 tot 1) terug van de joystick id.

`joystick_ypos(id)` Geeft de positie van de y-as (-1 tot 1) terug van de joystick id.

`joystick_zpos(id)` Geeft de positie van de z-as (-1 tot 1) terug van de joystick id, indien aanwezig.

`joystick_rpos(id)` Geeft de positie van de rudder-as (ofwel 4e as) terug.

`joystick_upos(id)` Geeft de positie van de u-as (ofwel 5e as) terug.

`joystick_vpos(id)` Geeft de positie van de v-as (ofwel 6e as) terug.

`joystick_pov(id)` Geeft de point-of-viewpositie van de joystick terug. Dit is een hoek tussen 0 en 360 graden. 0 is naar voren, 90 naar rechts, 180 naar achteren en 270 naar links. Als er geen point-of-viewrichting is aangegeven door de gebruiker wordt -1 geretourneerd.

Spel graphics

Een belangrijk deel van een spel zijn de spel graphics. Normaal zorgt *Game Maker* voor deze dingen en voor simpele spellen is er geen reden voor paniek. Maar soms wil je wat meer controle hebben. Voor sommige aspecten zijn er acties maar met scripts kan je nog meer controle hebben over aspecten. Dit hoofdstuk beschrijft alle variabelen en functies en geeft wat meer informatie over wat er gebeurd.

Sprites en plaatjes

Elk object heeft een eigen sprite. Dit is of een gewoon plaatje of het bevat meerdere plaatjes. Voor elke instantie tekent het programma het plaatje op het scherm, met zijn oorsprong(zoals gedefinieerd in de Sprite eigenschappen) op positie(x,y) van de instantie. Wanneer er meerdere plaatjes zijn, speelt hij alle plaatjes achter elkaar af met als gevolg een animatie. Er zijn een paar functies die invloed hebben op de manier hoe het plaatje is getekend. Deze kunnen gebruikt worden om het effect te veranderen. Iedere instantie heeft de volgende variabelen:

visible Wanneer het plaatje zichtbaar is(1) wordt het plaatje getekend, anders niet. Onzichtbare instanties zijn nog wel actief en kunnen colision gebeurtenissen creëren; Je kan ze alleen niet zien. Het op onzichtbaar zetten is handig om controller objecten te maken(maar ze niet solid om collision gebeurtenissen te voorkomen) of verborgen schakelaars.

sprite_index Dit is de index van de geselecteerde instantie. Je kan het veranderen om de instantie een andere sprite te geven. Je kan de namen van de verschillende sprite gebruiken als waarden. Veranderen van de sprite veranderd niks aan de geselecteerde sub-sprite.

sprite_width* Bepaald de breedte van een sprite. Deze waarde kan niet worden veranderd maar je zult het misschien nodig hebben.

sprite_height* Bepaald de hoogte van een sprite. Deze waarde kan niet worden veranderd maar je zult het misschien nodig hebben.

sprite_xoffset* Bepaald de horizontale compensatie van de sprite als gedefinieerd in de sprite eigenschappen. Deze waarde kan niet worden aangepast maar je zult het misschien nodig hebben.

sprite_yoffset* Bepaald de verticale compensatie van de sprite als gedefinieerd in de sprite eigenschappen. Deze waarde kan niet worden aangepast maar je zult het misschien nodig hebben.

image_number* De hoeveelheid van de subafbeeldingen voor de instantie(kan niet worden veranderd).

image_index wanneer de sprite meerdere subafbeeldingen heeft loopt het programma daardoorheen. Deze variabele verwijst naar de geselecteerde subafbeelding(ze zijn genummerd vanaf 0). Je kan het geselecteerde plaatje veranderen door de variabele aan te passen. Het programma zal gewoon doorgaan met afspelen, startend van zijn nieuwe index.(De waarde kan een gedeelte hebben. In dit geval wordt het altijd afgerond om de subafbeelding te verkrijgen dat wordt getekend.)

image_speed De snelheid waarmee wij door de subafbeeldingen lopen. Een waarde van 1 wijst erop dat we elke stap het volgende plaatje krijgen. De kleinere waarden zullen langzamere subafbeeldingen maken, door iedere subafbeelding vaker te tekenen. De grotere waarden zullen subafbeeldingen overslaan om de

beweging sneller te maken. Soms wil je misschien een subafbeelding laten zien en niet het programma erdoorheen te laten lopen. Daarvoor moet je de snelheid op 0 zetten en de goede subafbeelding selecteren. Bijvoorbeeld, veronderstel je hebt een voorwerp dat kan draaien en uw Sprite creëert die subafbeeldingen voor een aantal richtlijnen(linksdraaiend) heeft. Dan kan je in de Step gebeurtenis dit neerzetten:

```
{
  image_index = direction * image_number/360;
  image_speed = 0;
}
```

depth Normaal zijn de plaatjes getekend in de regel van de gecreëerde objecten. Je kunt dit veranderen door de beelddiepte te plaatsen. De standaardwaarde is 0, tenzij je het met een verschillende waarde in de object eigenschappen plaatst. Hoe hoger de waarde is hoe verder de instantie. (Je kunt negatieve waarden ook gebruiken). De instanties met hogere diepte zullen achter instanties met een lagere diepte liggen. Het plaatsen van de diepte zal garanderen dat de instanties worden getekend zoals je wilt (bijv. het vliegtuig voor de wolk). De achtergrond instanties zouden een hoge(positieve) diepte moeten hebben, en de voorgrondinstanties zouden een lage(negatieve) diepte moeten hebben.

image_xscale Een schaalfactor om grotere of kleinere beelden te maken. Een waarde van 1 wijst op de normale grootte. Je moet horizontale xscale en verticale yscale afzonderlijk plaatsen. Het veranderen van de schaal ruilt ook de waarden voor de beeldbreedte en hoogte en beïnvloedt botsingsgebeurtenissen aangezien je zou kunnen verwachten. Het veranderen van de schaal kan worden gebruikt om een 3-D effect te krijgen. Je kunt een waarde van -1 gebruiken om sprite horizontaal te weerspiegelen.

image_yscale Verticale yscale. 1 is geen schaal. Je kunt een waarde van -1 gebruiken om Sprite verticaal te weerspiegelen.

image_angle De hoek waarmee de Sprite wordt geroteerd. Je specificeert dit in graden, linksdraaiend. Een waarde van 0 wijst op geen omwenteling. ***Deze variabele kan alleen gebruikt worden in de geregistreerde versie!***

image_alpha Van de transparantie(de alpha-) waarde gebruikt voor het trekken van het beeld. Een waarde van 1 is het normale ondoorzichtige plaatsen; een waarde van 0 is volledig transparant.

image_blend gemengde kleuren gebruikt voor het tekenen van sprites. Een waarde van c_white is normaliter gebruikt. Wanneer je een verschillende waarde specificeert wordt de afbeelding gemengd met deze kleur. Dit kan worden gebruikt

om de Sprite te veranderen tijdens het spelen. ***Deze variabele kan alleen worden gebruikt in de geregistreerde versie!***

bbox_left* Linker kant van de bounding box die van de afbeelding van de instantie gebruikt wordt. (Neemt schaal mee).

bbox_right* Rechter kant van de bounding box van de Sprite van de instantie.

bbox_top* Bovenkant van de bounding box van de Sprite van de instantie.

bbox_bottom* Onderkant van de bounding box van de Sprite van de instantie.

Achtergronden

Elke room kan hoogstens 8 achtergronden hebben. Het heeft ook een achtergrond kleur. Alle aspecten van deze achtergronden kunnen veranderen met een stukje code met de volgende variabelen(merk op dat sommige series tussen 0 tot 7 zijn, bepalend van de achtergrond):

background_color Achtergrond kleur voor de room.

background_showcolor Bepaalt of de kleur te zien moet zijn of niet.

background_visible[0..7] Bepaalt of de achtergrond zichtbaar moet zijn.

background_foreground[0..7] Bepaalt of de achtergrond eigenlijk op de voorgrond moet.

background_index[0..7] Achtergrond afbeelding index.

background_x[0..7] X positie van de achtergrond.

background_y[0...7] Y positie van de achtergrond.

background_width[0...7]* Breedte van de achtergrond.

background_height[0...7]* Hoogte van de achtergrond.

background_htiled[0..7] Bepaalt of het horizontaal betegelt moet zijn.

background_vtiled[0..7] Bepaalt of het verticaal betegelt moet zijn.

background_xscale[0..7] Horizontale schaal factor voor de achtergrond. (Deze moet positief zijn; je kan niet een negatieve waarde gebruiken om horizontaal te spiegelen.)

background_yscale[0..7] Verticale schaal factor voor de achtergrond. (Deze moet positief zijn; je kan niet een negatieve waarde gebruiken om verticaal te spiegelen.)

background_hspeed[0..7] Horizontale scroll snelheid van de achtergrond (pixels per stap).

background_vspeed[0..7] Verticale scroll snelheid van de achtergrond (pixels per stap).

background_blend[0..7] Gemixte kleur wanneer de achtergrond getekend wordt. Een waarde van c_white is het normaliter. ***Alleen te gebruiken in de geregistreerde versie!***

background_alpha[0..7] Transparantie (alpha) waarde te gebruiken bij het tekenen the background. Een waarde van 1 is de normale waarde; een waarde van 0 is compleet doorzichtig.

Sprites en achtergronden tekenen

Objecten hebben normaal een sprite geassocieerd met ze om te tekenen. Maar je kan gebruikmaken van het draw event om andere dingen te tekenen. Deze sectie en de volgende paar geven je informatie over wat mogelijk is. Ten eerste, er zijn functies om sprites en achtergronden te tekenen op verschillende manieren. Deze geven je meer controle over de verschijning van de sprite. Je kan ook(delen van) achtergronden tekenen.

draw_sprite(sprite,subimg,x,y) Tekent subafbeelding subimg (-1 = normaal) van de sprite met index n met zijn oorsprong(x,y). (Zonder kleur menging en geen transparantie.)
draw_sprite_stretched(sprite,subimg,x,y,w,h) Tekent het plaatje gespreid zodat het wordt opgevuld tot in de linker- bovenhoek (x,y) een breedte w en hoogte h.
draw_sprite_tiled(sprite,subimg,x,y) Tekent de sprite betegeld zodat het de hele room opvult. (x,y) is de plaats waar een van de sprites is getekend.
draw_sprite_part(sprite,subimg,left,top,width,height,x,y) Tekent het geselecteerde deel van subafbeelding subimg (-1 = normaal) van de sprite met in de linker- bovenhoek het deel dat wordt getekend (x,y).

draw_background(back,x,y) Tekent de achtergrond op positie(x,y). (Zonder kleur menging en geen transparantie.)
draw_background_stretched(back,x,y,w,h) Tekent de achtergrond verspreid over de geselecteerde regio.
draw_background_tiled(back,x,y) Tekent de achtergrond betegeld zodat de achtergrond de gehele room bedekt.
draw_background_part(back,left,top,width,height,x,y) Tekent het geselecteerde deel van de achtergrond met de linker- bovenhoek op positie(x,y).

De volgende functies zijn uitgebreide versies van de voorgaande functies. ***Deze uitgebreide versies zijn alleen maar beschikbaar in de geregistreerde versie van game maker!***

draw_sprite_ext(sprite,subimg,x,y,xscale,yscale,rot,color,alpha) Tekent de sprite op schaal met factor xscale en yscale en draait linksom over graden. color is de gemengde kleur (gebruik c_white zodat hij niet mengt) en alpha

bepaald hoe transparant de factor met welk is samengevoegd met de achtergrond. Een waarde van 0 maakt de sprite onzichtbaar. Een waarde van 1 maakt het non-transparant. Deze functie kan mooie effecten maken (bijvoorbeeld deels transparante explosies).

draw_sprite_stretched_ext(sprite,subimg,x,y,w,h,color,alpha) Tekent de sprite gespreid zodat het de geselecteerde regio bedekt vanaf de linker-bovenhoek(x,y) en breedte w en hoogte h. Kleur bepaalt de gemengde kleur en alpha bepaald de transparantie.

draw_sprite_tiled_ext(sprite,subimg,x,y,xscale,yscale,color,alpha) Tekent de sprite betegeld zodat het de hele room bedekt maar nu met schaal factoren en een kleur en transparantie eigenschap.

draw_sprite_part_ext(sprite,subimg,left,top,width,height,x,y,xscale,yscale,color,alpha) Tekent het geselecteerde deel van subafbeelding subimg (-1 = normaal) van de sprite met de linker- bovenhoek op positie(x,y) maar nu met schaal factoren en een kleur en transparantie eigenschap.

draw_sprite_general(sprite,subimg,left,top,width,height,x,y,xscale,yscale,rot,c1,c2,c3,c4,alpha) De meest generale tekenfunctie. Het tekent het geselecteerde deel van subafbeelding subimg (-1 = normaal) van de sprite met de linker- bovenhoek op positie(x,y) maar nu met schaal factoren, een draaihoek, een kleur voor elk van de vier hoeken (linksboven, rechtsboven, rechtsonder, en linksonder), en een alpha transparantie waarde. Merk op dat draaiing plaats vindt om de linker- bovenhoek van het deel.

draw_background_ext(back,x,y,xscale,yscale,rot,color,alpha) Tekent de achtergrond op schaal en draait met de gemengde kleur(gebruik c_white voor geen menging) en transparantie alpha (0-1).

draw_background_stretched_ext(back,x,y,w,h,color,alpha) Tekent de achtergrond gespreid over de geselecteerde regio. color is de mengkleur en alpha bepaald de transparantie.

draw_background_tiled_ext(back,x,y,xscale,yscale,color,alpha) Tekent de achtergrond betegeld zodat het de gehele room bedekt maar nu met schaal factoren en een kleur en transparantie eigenschap.

draw_background_part_ext(back,left,top,width,height,x,y,xscale,yscale,color,alpha) Tekent het geselecteerde deel in de linker- bovenhoek op positie(x,y) maar nu met schaal factoren en een kleur en transparantie eigenschap.

draw_background_general(back,left,top,width,height,x,y,xscale,yscale,rot,c1,c2,c3,c4,alpha) De meest generale tekenfunctie. Het tekent het geselecteerde deel van de achtergrond met de linker- bovenhoek op

positie(x,y) maar nu met schaal factoren, een draaihoek, een kleur voor elk van de vier hoeken(linksboven, rechtsboven, rechtsonder, en linksonder), en een transparantie waarde. Merk op dat draaiing plaats vindt om de linker- bovenhoek van het deel.

Vormen tekenen

Er is een hele collectie functies beschikbaar om verschillende vormen te tekenen. Er zijn ook functies om tekst te tekenen (zie volgende sectie). Je kan ze alleen gebruiken in het Draw event van een object; deze functies maken normaliter niet uit ergens anders in code. Realiseer dat botsingen tussen instanties worden gevormd door hun sprites (of maskers) en niet door wat je eigenlijk tekent. De volgende teken functies bestaan die kunnen worden gebruikt om basis vormen te tekenen.

draw_clear(col) Leegt de gehele room in de kleur (geen alpha menging).
draw_clear_alpha(col,alpha) Leegt de gehele room in de kleur met alpha (handig voor oppervlakten).
draw_point(x,y) Tekent een put op positie (x,y) in de gewone kleur.
draw_line(x1,y1,x2,y2) Tekent een lijn vanaf (x1,y1) tot (x2,y2).
draw_rectangle(x1,y1,x2,y2,outline) Tekent een vierkant. `outline` bepaalt alleen of de buitenlijn moet worden getekend (true) of dat het binnenste moet worden opgevuld (false).
draw_roundrect(x1,y1,x2,y2,outline) Tekent een rond gemaakte rechthoek. `outline` bepaald of alleen of de buitenlijn moet worden getekend (true) of dat het binnenste moet worden opgevuld (false).
draw_triangle(x1,y1,x2,y2,x3,y3,outline) Tekent een driehoek. `outline` bepaalt alleen of de buitenlijn moet worden getekend (true) of dat het binnenste moet worden opgevuld (false).
draw_circle(x,y,r,outline) Tekent een cirkel op positie (x,y) met straal r. `outline` bepaalt of alleen of de buitenlijn moet worden getekend(true) of dat het binnenste moet worden opgevuld (false).
draw_ellipse(x1,y1,x2,y2,outline) Tekent een ellips. `outline` bepaalt alleen of de buitenlijn moet worden getekend (true) of dat het binnenste moet worden opgevuld (false).
draw_arrow(x1,y1,x2,y2,size) Tekent een pijl vanaf (x1,y1) tot (x2,y2). size bepaalt de grootte van de pijl in pixels.
draw_button(x1,y1,x2,y2,up) Tekent een knop, `up` bepaalt op (1) of neer (0).

draw_path(path,x,y,absolute) Met deze functie kan je het geselecteerde pad in de room tekenen op positie (x,y). als absolute true is wordt het pad getekend op de positie waar hij was gedefinieerd en de waarden van x en y worden genegeerd.
draw_healthbar(x1,y1,x2,y2,amount,backcol,mincol,maxcol,direction,showback,showborder) Met deze functie kan je een gezondheidsbalk(of een andere balk dat een waarde laat zien, zoals bijv. de schade). De argumenten x1, y1, x2 en y2 selecteren het totale stuk om te tekenen. amount bepaald het percentage hoever de balk gevuld moet worden (moet liggen tussen de 0 en de 100). backcol is de kleur voor de achtergrond van de balk. mincol en maxcol bepalen de kleur wanneer de waarde 100 en 0 zijn. Het bedrag tussen de kleur wordt geïnterpoleerd. Dus je kan gemakkelijk een balk maken die bijv. van groen naar rood gaat. De richting is de richten welke kant de balk op wordt getekend. 0 wijst erop dat de balk naar de linkerzijde, 1 naar de rechterzijde, 2 naar de bovenkant en 3 naar de onderkant wordt getekend. Showback wijst tot slot erop of een achtergrond moet worden getoond en showborder wijst erop of de achtergrond en de balk een zwarte lijn eromheen zouden moeten hebben.

Meeste van de bovenstaande functies gebruiken de kleur en alpha eigenschap die kunnen worden veranderd met de volgende functies.

draw_set_color(col) Plaatst de tekeningskleur dat van nu af aan voor tekeningsprimitieven moet worden gebruikt.
draw_set_alpha(alpha) Plaatst de alpha- transparantiewaarde dat van nu af aan voor tekeningsprimitieven moet worden gebruikt. Tussen de 0-1 zou moeten liggen, 0 is volledig transparant, is 1 volledig ondoorzichtig.
draw_get_color() Geeft de tekeningskleur terug die voor tekeningsprimitieven wordt gebruikt.
draw_get_alpha() Geeft de alpha waarde terug die voor tekeningsprimitieven wordt gebruikt.

Een heleboel voorgedefinieerde kleuren:

c_aqua
c_black
c_blue
c_dkgray
c_fuchsia
c_gray
c_green

c_lime

c_ltgray

c_maroon

c_navy

c_olive

c_purple

c_red

c_silver

c_teal

c_white

c_yellow

De volgende functies helpen je kleuren te maken die je wil.

make_color_rgb(red,green,blue) Geeft een kleur terug met geselecteerde waarde rood, groen, en blauw, waar rood, groen en blauw moeten liggen tussen 0 en 255.
make_color_hsv(hue,saturation,value) Geeft een kleur terug met de geselecteerde waarde tint, verzadiging en waarde (elk tussen 0 en 255).
color_get_red(col) Geeft de rode waarde terug.
color_get_green(col) Geeft de groene waarde terug.
color_get_blue(col) Geeft de blauwe waarde terug.
color_get_hue(col) Geeft de tint waarde terug.
color_get_saturation(col) Geeft de verzadiging waarde terug.
color_get_value(col) Geeft de waarde terug van een kleur.
merge_color(col1,col2,amount) Geeft de kleur terug van een col1 en col2. Het samenvoegen wordt bepaald door bedrag. Een waarde van 0 beantwoordt aan col1, een waarde van 1 aan col2, en waarden ertussen aan samengevoegde waarden.

Deze overige functies bestaan ook.

draw_getpixel(x,y) Geeft de kleur van de pixel op positie (x,y) in de room terug. dit is niet erg snel, dus gebruikt het voorzichtig.
screen_save(fname) Bewaart een bmp afbeelding van het scherm in bestand fname. Nuttig om screenshots te maken.
screen_save_part(fname,x,y,w,h) Bewaart een deel van het scherm in bestand fname.

Lettertypes en tekst

In spellen is het soms nodig tekst te schrijven. Om een tekst te schrijven moet je het lettertype goed instellen. Lettertypes kunnen worden gemaakt met de font eigenschappen(ook in *Game Maker* of gebruikmakend van de functies of geïmporteerde lettertypes). er zijn verschillende functies om teksten te schrijven. In iedere functie kan je de positie van de tekst op je scherm bepalen. Er zijn twee verschillende functies om de horizontale en verticale groepering van de tekst met betrekking tot die positie.

Voor tekst schrijven bestaan de volgende functies:

`draw_set_font(font)` Selecteert het lettertype dat zal worden gebruikt tijdens het schrijven van tekst. Gebruik -1 om het standaardfont te plaatsen (Arial 12).
`draw_set_halign(halign)` Selecteert de horizontale groepering dat wordt gebruikt tijdens het schrijven van de tekst. Kies één van de volgende constanten als waarden:

`fa_left`
`fa_center`
`fa_right`

`draw_set_valign(valign)` Selecteert de verticale groepering dat wordt gebruikt tijdens het schrijven van de tekst. Kies één van de volgende constanten als waarden:

`fa_top`
`fa_middle`
`fa_bottom`

`draw_text(x,y,string)` Tekent de tekenreeks op positie (x,y), gebruikmakend van de tekenkleur en alpha. Een # symbool of interlinie chr (13) of linefeed chr (10) worden geïnterpreteerd als newline karakters. Op deze wijze kunt je meer lijnen teksten tekenen. (Gebruik \# om # te krijgen.)
`draw_text_ext(x,y,string,sep,w)` Gelijkaardig aan de vorige routine maar je kunt twee meer dingen specificeren. Eerst en vooral, wijst sep op de scheidingsafstand tussen de lijnen van tekst in een meer lijnen tekst. Gebruik -1 om de standaardafstand te krijgen. Gebruik w om op de breedte van de tekst in pixel te wijzen. De lijnen die langer zijn dan deze breedte zijn verdeelde - omhoog bij ruimten of - tekens. Gebruik -1 voor niet verdeelde lijnen.
`string_width(string)` Breedte van de tekenreeks in huidige font aangezien het

tekenen de `draw_text()` functie is. Kan voor het plaatsen van een grafiek worden gebruikt.

`string_height(string)` Hoogte van de tekenreeks in huidige font aangezien het tekenen de `draw_text()` functie is. Kan voor het plaatsen van een grafiek worden gebruikt.

`string_width_ext(string,sep,w)` Breedte van de tekenreeks in huidige font aangezien het tekenen de `draw_text_ext()` functie is. Kan voor het plaatsen van een grafiek worden gebruikt.

`string_height_ext(string,sep,w)` Hoogte van de tekenreeks in huidige font aangezien het tekenen de `draw_text_ext()` functie is.

De volgende routines staan je toe om vormen en geroteerde teksten en ook de kleuren van de gebruiksgradiënt op teksten te tekenen. ***Deze functies zijn alleen beschikbaar in de geregistreerde versie!***

`draw_text_transformed(x,y,string,xscale,yscale,angle)` Tekent de tekenreeks op positie (x,y) op dezelfde manier als hierboven, maar vormt het horizontaal en verticaal met de vermelde factoren en roteert het linksdraaiend over `angle` graden.

`draw_text_ext_transformed(x,y,string,sep,w,xscale,yscale,angle)` Combineert `draw_text_ext` en `draw_text_transformed`. Het maakt het mogelijk een meer lijnen tekst te tekenen (draaiend en op schaal).

`draw_text_color(x,y,string,c1,c2,c3,c4,alpha)` Tekent de tekenreeks op positie (x,y) op dezelfde manier als hierboven. De vier kleuren bepalen de kleuren van de linksboven-, rechtsboven-, rechtsonder-, en links-onderhoek van de tekst. `alpha` is de alpha transparantie dat wordt gebruikt(0-1).

`draw_text_ext_color(x,y,string,sep,w,c1,c2,c3,c4,alpha)` Gelijk aan `draw_text_ext()` maar met gekleurde toppen.

`draw_text_transformed_color(x,y,string,xscale,yscale,angle,c1,c2,c3,c4,alpha)` Gelijk aan `draw_text_transformed()` maar met gekleurde toppen.

`draw_text_ext_transformed_color(x,y,string,sep,w,xscale,yscale,angle,c1,c2,c3,c4,alpha)` Gelijk aan `draw_text_ext_transformed()` maar met gekleurde toppen.

Geavanceerde tekenfuncties

Deze functies zijn alleen beschikbaar in de geregistreerde versie van Game Maker.

Hierboven, zijn een aantal basis tekenfuncties beschreven. Hier vindt je een aantal extra functies die je veel meer mogelijkheden bieden. Eerst en vooral zijn er functies om vormen met gradiëntkleuren te tekenen. Ten tweede zijn er functies om meer algemene veelhoeken te tekenen, en definitief is er de mogelijkheid om textuur in kaart gebrachte veelhoeken te tekenen.

De volgende uitgebreide versies van de basis tekenfuncties bestaan, elk van hen krijgt extra kleurenparameters die de kleur bepalen bij verschillende toppen. De standaard tekenkleur wordt niet gebruikt in deze functies.

`draw_point_color(x,y,col1)` Tekent een punt op positie (x,y) in de gegeven kleur.

`draw_line_color(x1,y1,x2,y2,col1,col2)` Tekent een lijn vanaf positie (x1,y1) tot (x2,y2), met een vloeiende overgang tussen col1 en col2.

`draw_rectangle_color(x1,y1,x2,y2,col1,col2,col3,col4,outline)` Tekent een vierkant. De vier kleuren bepalen de kleuren in de linker-boven, rechter-boven, rechter-onder, en linker-onder hoek. outline bepaald of alleen de buitenste lijn moet worden getekend (true) of alles moet worden opgevuld(false).

`draw_roundrect_color(x1,y1,x2,y2,col1,col2,outline)` Tekent een vierkant met ronde hoeken. col1 is de kleur in het midden en col2 the kleur op de grens. outline bepaalt of alleen de buitenste lijn getekend moet worden (true) of dat het opgevuld moet worden (false).

`draw_triangle_color(x1,y1,x2,y2,x3,y3,col1,col2,col3,outline)` Tekent een driehoek. De drie kleuren zijn de kleuren van de drie hoeken welke worden geïnterpoleerd. outline bepaalt of alleen de buitenste lijn moet worden getekend (true) of dat het opgevuld moet worden (false).

`draw_circle_color(x,y,r,col1,col2,outline)` Tekent een cirkel op (x,y) met straal (radius) r. col1 is de kleur in het midden en col2 de kleur aan de buitenkant. outline bepaalt of alleen de buitenste lijn getekend moet worden (true) of moet worden opgevuld (false).

`draw_ellipse_color(x1,y1,x2,y2,col1,col2,outline)` Tekent een ellips. col1 is de kleur in het midden en col2 aan de buitenkant. outline bepaald of alleen de buitenste lijn moet worden getekend (true) of moet worden opgevuld (false).

Je kunt grotere primitieven ook tekenen. Dit is iets anders. Je begint door de primitief te specificeren die je wilt tekenen. Daarna specificeer je de hoeken, en als laatste definieert je de primitief, op welk ogenblik het wordt getekend. Er zijn zes types van primitieven:

`pr_pointlist` De hoeken zijn een setje punten.

`pr_linelist` De hoeken zijn een set van delen van lijnen. Elk paar van een deel

van een lijn. Dus Er moet een even aantal set van hoeken zijn.
pr_linestrip De hoeken vormen een polylijn met de eerste vastgemaakt aan de tweede enz. De laatste is niet vastgemaakt aan de eerste. Je moet een extra kopie maken van de eerste hoek voor dit.
pr_trianglelist De toppen zijn een reeks driehoeken. Elk drievoud toppen vormt een driehoek. Zo moet het aantal toppen een veelvoud van 3 zijn.
pr_trianglestrip De toppen vormen opnieuw driehoeken maar dit keer iets anders. De eerste drie vormen de eerste driehoek. De laatste twee van deze hoeken, samen met de volgende hoek, vormen de tweede driehoek enz. Zo specificeert je een nieuwe driehoek, die met vorige wordt verbonden.
pr_trianglefan Gelijkaardig aan een driehoekslijst maar dit keer maakt de eerste hoek deel uit alle driehoeken. Opnieuw, specificeert elke nieuwe hoek een nieuwe driehoek, die met de vorige hoek en de eerste hoek wordt verbonden.

De volgende functies bestaan om nieuwe primitieven te tekenen.

draw_primitive_begin(kind) Begint een primitief van het geselecteerde type.
draw_vertex(x,y) Voegt hoek(x,y) toe aan de primitief, gebruikmakend van kleur en alpha waarde die eerder moeten zijn geselecteerd.
draw_vertex_color(x,y,col,alpha) Voegt hoek (x,y) toe aan de primitief, met zijn eigen kleur en alpha waarde. Dit stelt je in staat om primitieven de maken waarvan de kleuren en alpha overlopen.
draw_primitive_end() Beëindigt de eigenschappen van de primitief. Deze functie tekent het.

Tot slot is het mogelijk om primitieven van sprites of achtergronden te tekenen die als vormen gebruikt kunnen worden. Tijdens het gebruiken van een vorm wordt het beeld geplaatst op de primitief, die het een nieuwe vorm geeft om de primitief te passen. De vormen worden gebruikt om detail aan primitieven toe te voegen, bijv. een bakstenen muur. Om vormen te gebruiken moet je eerst de id van de vorm verkrijgen die je wilt gebruiken. Voor dit bestaan de volgende functies:

sprite_get_texture(spr,subimg) geeft de id van de vorm terug corresponderend met subafbeelding subimg van de geselecteerde sprite.
background_get_texture(back) geeft de id van de vorm terug corresponderend met de geselecteerde achtergrond.

Een geselecteerde textuur zou nog niet in videogeheugen kunnen zijn. Het systeem zal het daar zetten zodra je het nodig hebt maar soms wil je dit zelf beslissen. Voor dit bestaan de volgende twee functies:

texture_preload(texid) Zet onmiddellijk de textuur in videogeheugen.
texture_set_priority(texid,prio) Wanneer er ook weinig videogeheugen is zullen sommigen tijdelijk worden verwijderd om ruimte voor anderen te maken die nodig zijn. Degenen met laagste prioriteit worden eerst verwijderd. Het gebrek, allen hebben prioriteit 0 maar je kunt de prioriteit hier veranderen. (Gebruik alleen positieve waarden!)

Om texturen aan primitieven toe te voegen moet je specificeren welke delen van de texturen waar moeten worden gezet op de primitief. De posities in de textuur zijn vermeld met waarden tussen 0 en 1 maar er is hier een probleem. De grootte van texturen moet bevoegdheden van 2 zijn(zo bijv. 32x32 of 64x64). Als je sprites of achtergrond als texturen wilt gebruiken zorg je beter ervoor zij een dergelijke grootte hebben. Zo niet, zal de test leeg zijn. Te weten komen wat van de textuur scheiden wordt eigenlijk gebruikt je kan de volgende twee functies gebruiken. Zij keren een waarde tussen 0 en 1 terug die op de breedte of de hoogte van het daadwerkelijke deel van de textuur wijst die wordt gebruikt. Het specificeren van deze waarde als textuurcoördinaat zal op de recht of onderkant van de textuur wijzen.

texture_get_width(texid) Geeft de waarde van de breedte van de textuur met gegeven id terug. De breedte ligt tussen 0-1.
texture_get_height(texid) Geeft de waarde van de hoogte van de textuur met gegeven id terug. De breedte ligt tussen 0-1.

Om getextureerde primitieven te tekenen gebruik je de volgende functies:

draw_primitive_begin_texture(kind,texid) Begin een primitief van de vermelde soort met de gegeven textuur.
draw_vertex_texture(x,y,xtex,ytex) Voegt hoek (x,y) toe op positie (xtex,ytex) in de textuur, mengend met de kleur en de alpha waarde alvorens xtex en ytex normaal tussen 0 en 1 zou moeten liggen maar ook de grotere waarden kunnen worden gebruikt, leidend tot een herhaling van de textuur (zie verder).
draw_vertex_texture_color(x,y,xtex,ytex,col,alpha) Voegt hoek (x,y) toe op positie (xtex,ytex) in de textuur, gemengd met zijn eigen kleur en alpha waarde.
draw_primitive_end() Beëindigt de beschrijving van de textuur. Deze functie tekent het.

Er zijn drie functies die beïnvloeden hoe de texturen worden getekend:

texture_set_interpolation(linear) Wijst erop of lineaire (true) te gebruiken interpolatie of het meest dichtbijgelegen (false) pixel. De lineaire

interpolatie geeft meer vlotte texturen maar kan ook een beetje onscherp zijn en kost soms extra tijd. Het plaatsen beïnvloedt ook de tekening van sprites en achtergrond. Het normale is `false`. (Dit kan ook in de spelopties worden veranderd.)

`texture_set_blending(blend)` Wijst erop of aan gebruik het mengen met kleuren en alpha- waarden. Schakelen van dit zou veel sneller op oude hardware kunnen zijn. Het plaatsen beïnvloedt ook de tekening van sprites en achtergrond. Het normale is `true`.

`texture_set_repeat(repeat)` Wijst erop of het gebruik van de textuur herhaald moet worden. Dit werk als volgt. Zoals hierboven vermeld liggen de coördinaten tussen 0-1. Als je een waarde groter dan 1 specificeert, wordt de rest niet getekend. Door te plaatsen herhaal waar de textuur wordt herhaald. Merk op dat sprites en de achtergronden altijd zonder het herhalen worden getekend. Zo zodra je de sprite van de achtergrond tekent wordt deze waarde teruggesteld aan `false`. Het normale is `false`.

Er zijn twee meer functie die niet alleen nuttig voor tekeningstexturen zijn. Normaal worden de primitieven gemengd met de achtergrond gebruikmakend de alpha waarde. Je kunt eigenlijk erop wijzen hoe dit moet gebeuren. Naast de normale wijze is het mogelijk om erop te wijzen, dat de nieuwe kleur moet aan de bestaande kleur worden toegevoegd of van de bestaande kleur worden afgetrokken. Dit kan worden gebruikt om bijv. vleklichten of schaduwen tot stand te brengen. Ook is het mogelijk om te sorteren van vergt het maximum van de nieuwe en bestaande kleur. Dit kan bepaalde verzadigingsgevolgen vermijden die je met het toevoegen kunt krijgen. Merk op dat zowel het aftrekken als het maximum niet met de alpha waarde volledig rekening houden. (DirectX staat dit niet toe.) Zo zorg je beter ervoor het buitengebied zwart is. Er zijn twee functies. Eerste geef je slechts de vier hierboven beschreven opties. De tweede functie geeft je meer mogelijkheden. Je zou een beetje met de montages moeten experimenteren. Indien effectief gebruikt kunnen zij worden gebruikt om bijv. interessante explosie of halogevolgen tot stand te brengen.

`draw_set_blend_mode(mode)` Selecteert de modus om te mengen. De volgende waardes zijn mogelijk: `bm_normal`, `bm_add`, `bm_subtract`, en `bm_max`. Vergeet niet om de modus te veranderen naar normaal na het gebruikt want anders worden sprites en zelfs achtergronden getekend met de meng modus.

`draw_set_blend_mode_ext(src,dest)` Selecteert welke modus er moet worden gebruikt voor de bron en bestemmings kleur. De nieuwe kleur is soms de bron en soms de bestemming. Deze factoren zijn bepaald met deze functie. Om dit te begrijpen, De bron en bestemming hebben allebei als rood, groen, blauw, en alpha componenten. Dus de bron is (Rs, Gs, Bs, As) en de bestemming is (Rd, Gd,

Bd, Ad). Alle getallen moeten tussen de 0 en 1 liggen. De meng factoren die je kan kiezen voor zijn:

- bm_zero: Meng factor is (0, 0, 0, 0).
- bm_one: Meng factor is (1, 1, 1, 1).
- bm_src_color: Meng factor is (Rs, Gs, Bs, As).
- bm_inv_src_color: Meng factor is (1–Rs, 1–Gs, 1–Bs, 1–As).
- bm_src_alpha: Meng factor is (As, As, As, As).
- bm_inv_src_alpha: Meng factor is (1–As, 1–As, 1–As, 1–As).
- bm_dest_alpha: Meng factor is (Ad, Ad, Ad, Ad).
- bm_inv_dest_alpha: Meng factor is (1–Ad, 1–Ad, 1–Ad, 1–Ad).
- bm_dest_color: Meng factor is (Rd, Gd, Bd, Ad).
- bm_inv_dest_color: Meng factor is (1–Rd, 1–Gd, 1–Bd, 1–Ad).
- bm_src_alpha_sat: Meng factor is (f, f, f, 1); f = min (As, 1–Ad).

Bijv. de normale meng modus verandert de bron naar bm_src_alpha en de bestemming naar bm_inv_src_alpha. Vergeet niet de normale modus terug te zetten omdat anders de sprites en zelfs de achtergronden worden getekend met de nieuwe meng modus.

De textuurprimitieven van de tekening zijn weinig werk maar kunnen tot grote resultaten leiden. Je kunt het zelfs gebruiken om nep 3D spelen te maken.

Oppervlakten tekenen

Deze functies zijn alleen beschikbaar in de geregistreerde versie van Game Maker!

In bepaalde situaties zou je niet direct op het scherm maar eerst op een canvas kunnen tekenen dat dan later kan worden gebruik om dingen op het scherm de tekenen. Een dergelijk canvas wordt een oppervlakte (surface) genoemd. Bijvoorbeeld, wil je de gebruiker op het scherm laten tekenen. Het moet dan niet meteen op het scherm verschijnen (omdat het elke volgende stap weer wordt gewist) , maar in plaats daarvan wil je het op een afzonderlijke oppervlakten tekenen die op het scherm in elke stap wordt gekopieerd. Of je wil een textuur gebruiken die steeds verandert.

De oppervlakten maken dit allen mogelijk. Zij zijn eigenlijk eerder eenvoudig te gebruiken. Je creëert eerst een oppervlakte. Daarna stel je in wat er met de verdere tekening op deze oppervlakte zou moeten gebeuren. Vanaf dat moment werkt de tekening functies uit op de oppervlakte. Zodra je dat gedaan hebt stel je het tekeningsdoel opnieuw in en de verdere tekening dat gebeurt op het scherm. Je kunt de oppervlakte tekenen op het scherm op vele verschillende

manieren of het gebruiken als textuur. Er zijn een paar dingen waar je van bewust moet zijn. Zie aan het eind de opmerkingen.

De volgende functies bestaan om oppervlakten te tekenen

surface_create(w,h) Creëert een oppervlakte van de vermelde breedte en de hoogte. Geeft id van de oppervlakte terug, die in alle verdere functies moet worden gebruikt. Merk op dat de oppervlakte niet ontruimd zal worden. Dit is de verantwoordelijkheid van de gebruiker. (Plaats het als doel en roep de aangewezen functie op.)
surface_free(id) Bevrijd geheugen die wordt gebruikt voor de oppervlakte.
surface_exists(id) geeft terug of de oppervlakte met de ingestelde id bestaat.

surface_get_width(id) Geeft de breedte van de oppervlakte terug.
surface_get_height(id) Geeft de hoogte van de oppervlakte terug.
surface_get_texture(id) Geeft de textuur terug dat behoord bij de oppervlakte. Dit kan worden gebruikt om getextureerde objecten met het plaatje van de oppervlakte te tekenen.

surface_set_target(id) Plaatst de vermelde oppervlakte als tekendoel. Al het verdere tekenen gebeurt op deze oppervlakte. Het stelt de projectie terug om de oppervlakte eenvoudig te overschrijven.
surface_reset_target() Stelt het tekeningsdoel naar het normale scherm in.

surface_getpixel(id,x,y) Geeft de kleur terug op positie (x,y) in de oppervlakte. Dit is niet erg snel, dus gebruik het zorgvuldig.
surface_save(id,fname) Slaat een bmp plaatjes van de oppervlakte op met de gegeven bestandsnaam. Handig om screenshots te maken.
surface_save_part(id,fname,x,y,w,h) Slaat een deel van de oppervlakte op in een bestand.

draw_surface(id,x,y) Tekent de oppervlakte op positie (x,y). (Zonder kleur mengen en geen alpha transparantie.)
draw_surface_stretched(id,x,y,w,h) Tekent de oppervlakte uitgerekt over de hele ingestelde regio.
draw_surface_tiled(id,x,y) Tekent de oppervlakte betegelt zodat het de hele room bedekt.
draw_surface_part(id,left,top,width,height,x,y) Tekent een deel van de oppervlakte met zijn oorsprong op positie (x,y).

draw_surface_ext(id,x,y,xscale,yscale,rot,color,alpha) Tekent de oppervlakte met de gemengde kleur aangepast en geroteerd (gebruik c_white voor geen kleur menging) en alpha transparantie (0-1).
draw_surface_stretched_ext(id,x,y,w,h,color,alpha) Tekent de oppervlakte uitgerekt over de hele ingestelde regio. color is de mengkleur and alpha bepaald de transparantie instelling.
draw_surface_tiled_ext(id,x,y,xscale,yscale,color,alpha) Tekent de oppervlakte betegelt zodat het de hele room bedekt maar nu met aangepaste factoren en transparantie instelling.
draw_surface_part_ext(id,left,top,width,height,x,y,xscale,yscale,color,alpha) Tekent een deel van de oppervlakte met zijn oorsprong op positie (x,y) maar nu met aangepaste factoren en transparantie instelling.
draw_surface_general(id,left,top,width,height,x,y,xscale,yscale,rot,c1,c2,c3,c4,alpha) De meest algemene tekenfunctie. Het tekent het ingestelde deel van de oppervlakte met zijn oorsprong op positie (x,y) maar nu met aangepaste factoren, een draaihoek, een kleur voor elk van de vier hoeken (linksboven, rechtsboven, rechtsonder, en linksonder), en een alpha transparantie waarde.

surface_copy(destination,x,y,source) Kopieert de inhoud van de oppervlakte op positie (x,y) in de voorbestemde oppervlakte. (Zonder kleurmenging.)
surface_copy_part(destination,x,y,source,xs,ys,ws,hs) Kopieert het ingestelde gedeelte van de inhoud van de oppervlakte op positie (x,y) in de voorbestemde oppervlakte. (Zonder kleurmenging.)

Merk op dat er geen functies zijn om een deel van het scherm naar een oppervlakte te kopiëren. (Dit is onmogelijk toe te schrijven aan mogelijke formaatverschillen tussen het scherm en de oppervlakten.) Als dit wordt vereist moet je een oppervlakte plaatsen dat als doel wordt teruggeven en daarna de room tekenen. je kunt de oppervlakte het kopieer routines gebruiken om delen van de oppervlakte te krijgen.

Merk op dat je ook sprites en achtergronden van oppervlakten kunt maken. Zie de sectie over veranderen van resources voor meer informatie.

Dit moet wel voorzichtig gedaan worden. Merk deze dingen op:

- Je zou het tekendoel nooit moeten veranderen terwijl je eigenlijk op het scherm tekent, gebruikt het nooit in draw events. Dit zal ernstige problemen met de projectie en viewport veroorzaken.
- Oppervlakten werken niet goed in 3D modus. Je kan ze gebruiken als je niet in de 3D modus zit (door d3d_end() op te roepen voor ze te gebruiken, maar als je de 3D modus opnieuw start worden de oppervlakten vernietigt.
- Om snelheidsredenen, zit de oppervlakte alleen in het videogeheugen. Als een resultaat, kan je de oppervlakte verliezen door bijv. verandering van scherm resolutie of waneer de screensaver start.
- Oppervlakten worden niet opgeslagen als een spel wordt opgeslagen.

Tiles

In een room kan je tiles toevoegen, maar dat wist je al. Een tile is een deel van een background. Tiles zijn zichtbare afbeeldingen, ze reageren niet op events en generen geen aanrakingen. Daardoor werken ze veel sneller als objecten. Voor iets dat geen events of aanrakingen nodig heeft, kan je best tiles gebruiken. Soms kan het handig zijn om een tile te gebruiken als mooie afbeelding, en een simpel object om de aanrakingen te genereren.

Je hebt echter meer controle over tiles dan je zou denken. Je kan ze toevoegen tijdens het ontwerpen van de room, maar ook tijdens het spelen van het spel. Je kan niet alleen hun positie aanpassen, maar ook de schaal veranderen en ze zelf gedeeltelijk transparant maken. Een tile heeft de volgende eigenschappen:

- **`background`**. De achtergrond waar de tile van genomen is.
- **`left, top, width, height`**. Het deel van de achtergrond dat gebruikt is.
- **`x,y`**. De positie vanuit de bovenlinkerhoek van de room.
- **`depth`**. De diepte van de tile. Je kan kiezen welke diepte je gebruikt, zodat tiles bijvoorbeeld tussen objecten verschijnen.
- **`visible`**. Bepaald of de tile zichtbaar is.
- **`xscale, yscale`**. Elke tile kan met een schaal getekend worden (standaard is 1).
- **`blend`**. Een mengkleur, gebruikt om de tile te tekenen.
- **`alpha`**. Een alpha waarde wordt gebruikt om de transparantheid van een tile aan te geven. 1 = niet transparant, 0 = volledig transparant.

Om de eigenschappen van een bepaalde tile te weten moet je de id van de tile kennen. Wanneer je tiles toevoegt tijdens het maken van een room kan je de id onderaan in de informatiebalk zien. Er is ook een functie om de id van een tile op een specifieke plaats te vinden.

Voor tiles bestaan er de volgende functies:

tile_add(background,left,top,width,height,x,y,depth) Voegt een nieuwe tile toe in de room met de aangegeven warden (zie hierboven voor hun betekenis). Deze functie geeft het id van de tile terug, om later nog te gebruiken.
tile_delete(id) Verwijdert de tile met de gegeven id.
tile_exists(id) Geeft terug of er een tile bestaat met de gegeven id.

tile_get_x(id) Geeft de x-positie van de tile met de gegeven id terug.
tile_get_y(id) Geeft de y-positie van de tile met de gegeven id terug.
tile_get_left(id) Geeft de linker waarde van de tile met de gegeven id terug.
tile_get_top(id) Geeft de bovenste waarde van de tile met de gegeven id terug.
tile_get_width(id) Geeft de breedte van de tile met de gegeven id terug.
tile_get_height(id) Geeft de hoogte van de tile met de gegeven id terug.
tile_get_depth(id) Geeft de diepte van de tile met de gegeven id terug.
tile_get_visible(id) Geeft terug of de tile met de gegeven id zichtbaar is.
tile_get_xscale(id) Geeft de x-schaal van de tile met de gegeven id terug.
tile_get_yscale(id) Geeft de y-schaal van de tile met de gegeven id terug.
tile_get_background(id) Geeft de achtergrond van de tile met de gegeven id terug.
tile_get_blend(id) Geeft de mengkleur van de tile met de gegeven id terug.
tile_get_alpha(id) Geeft de alpha waarde van de tile met de gegeven id terug.

tile_set_position(id,x,y) Plaatst de tile met de gegeven id op de gegeven plaats.
tile_set_region(id,left,right,width,height) Stelt de regio van de tile met de gegeven id in op zijn achtergrond.
tile_set_background(id,background) Stelt de achtergrond voor de tile met de gegeven id in.
tile_set_visible(id,visible) Stelt in of de tile met de gegeven id zichtbaar is.
tile_set_depth(id,depth) Stelt de diepte van de tile met de gegeven id in.
tile_set_scale(id,xscale,yscale) Stelt de schaal van de tile met de gegeven id in.
tile_set_blend(id,color) Stelt de mengkleur in van de tile met de gegeven id in. ***Alleen mogelijk met de geregistreerde versie van Game Maker!***

`tile_set_alpha(id,alpha)` Stelt de alpha waarde van de tile met de gegeven id in.

De volgende functies hebben te maken met lagen van tiles, dat zijn verzamelingen van tiles op dezelfde diepte.

`tile_layer_hide(depth)` Verbergt alle tiles op de gegeven depth.
`tile_layer_show(depth)` Laat alle tiles op de gegeven depth zien.
`tile_layer_delete(depth)` Verwijdert alle tiles op de gegeven depth.
`tile_layer_shift(depth,x,y)` Verplaatst alle tiles op de gegeven diepte lag met de vector x,y. Dit kan gebruikt worden om scrollende lagen van tiles te maken.
`tile_layer_find(depth,x,y)` Geeft de id terug van de tile op de gegeven depth op de positie x,y. Wanneer er daar geen tile bestaat wordt –1 terug gegeven. Wanneer er verschillende tiles zijn wordt het id van de eerste terug gegeven.
`tile_layer_delete_at(depth,x,y)` Verwijdert de tile op de gegeven depth op de positie x,y. Wanneer er daar meerdere tiles zijn worden ze allemaal verwijderd.
`tile_layer_depth(depth,newdepth)` Verandert de diepte van alle tiles op de gegeven depth naar een nieuwe diepte. Met deze functie kan je hele lagen van tiles naar een nieuwe diepte verplaatsen.

Het scherm

Het scherm vertegenwoordigt het gehele gebied op de monitor. Het heeft een grootte (meestal 1024x768, of 1280x1024), een kleur diepte, dat is, het aantal bits dat wordt gebruikt om één enkel pixel te vertegenwoordigen (meestal 16 = hoge kleur of 32 = volledige kleur) en een ververs frequentie, dat is, het aantal keer per seconde dat de display wordt ververst (meestal tussen 60 en 120). Deze instellingen kunnen normaal worden veranderd hoewel de schermeigenschappen. Voor spellen niettemin, vooral als ze in volledig scherm worden gespeeld, het is belangrijk om deze instellingen te kunnen veranderen. Al deze instellingen kunnen worden ingesteld in de **Game Settings (Spel Instellingen)**. Voor het gebruik tijdens het spelen bestaan de volgende functies. Merk op hoewel dat die de instellingen verandert tijdens spelspel in een tijdvertraging zal resulteren aangezien de dingen moet worden heringesteld. ***De functies om de modus te veranderen zijn alleen beschikbaar in de geregistreerde versie.***

`display_get_width()` Geeft de waarde van de breedte van het scherm in pixels terug.
`display_get_height()` Geeft de waarde van de hoogte van het scherm in pixels terug.
`display_get_colordepth()` Geeft de kleur diepte in bits terug.
`display_get_frequency()` Geeft de ververs frequentie terug van het scherm.

display_set_size(w,h) Stelt de breedte en de hoogte van het scherm in. Geeft terug wanneer het gelukt is. (Realiseer dat slechts bepaalde combinaties worden toegestaan.)

display_set_colordepth(coldepth) Stelt de kleur diepte in. In het algemeen zijn slechts 16 en 32 toegestane waarden. Geeft terug wanneer het gelukt is.

display_set_frequency(frequency) Stelt de ververs frequentie in van het scherm. Er zijn maar een paar frequenties toegestaan. Normaal kan je dit instellen als 60 met eenzelfde room snelheid om een vloeiende 60 frames per seconde motion te krijgen. Geeft terug wanneer het gelukt is.

display_set_all(w,h,frequency,coldepth) Stelt alles in één keer in. Gebruik -1 voor waarden die je niet wilt veranderen. Geeft terug wanneer het gelukt is.

display_test_all(w,h,frequency,coldepth) Test of de vermelde instellingen worden toegestaan. het veranderd geen instellingen. Gebruik -1 voor waarden die je niet wilt veranderen. Geeft terug of de instellingen gebruikt mogen worden.

display_reset() Veranderd de scherm instellingen weer terug naar de instellingen van de start van het programma.

Soms is het nuttig om informatie over de positie van de muis op het scherm te krijgen of deze positie te veranderen. Voor dit bestaan de volgende functies:

display_mouse_get_x() Geeft de x-coördinaat van de muis op het scherm terug.

display_mouse_get_y() Geeft de y-coördinaat van de muis op het scherm terug.

display_mouse_set(x,y) Stelt de positie van de muis op het scherm in met de ingestelde waarden.

Het venster

Het actieve spel speelt zich af in een venster. Dit venster heeft een aantal eigenschappen, zoals of het een rand heeft, of het in volledig schermmodus draait, enz. Normaal is dit ingesteld in de Spel Instellingen (**Game Settings**). Maar je kunt dit wijzigen tijdens het spel. De volgende functies bestaan hier voor:

window_set_visible(visible) Stelt in of het spel venster zichtbaar is. Normaal wil je het venster tijdens het hele spel zien. Het programma zal geen toetsenbord- gebeurtenissen ontvangen als het venster onzichtbaar is.

window_get_visible() Geeft terug of het spel venster zichtbaar is.

window_set_fullscreen(full) Stelt in of het venster in volledig schermmodus wordt getoond.
window_get_fullscreen() Geeft terug of het venster in volledig schermmodus wordt getoond.
window_set_showborder(show) Stelt in of de rand om het venster wordt getoond. (Wordt in volledig schermmodus nooit getoond.)
window_get_showborder() Geeft terug of de rand om het venster in venstermodus wordt getoond.
window_set_showicons(show) Stelt in of de rand knoppen (minimaliseren, maximaliseren, sluiten) worden getoond. (In volledig schermmodus worden deze nooit getoond.)
window_get_showicons() Geeft terug of de rand knoppen worden getoond in venstermodus.
window_set_stayontop(stay) Stelt in of het venster altijd op de voorgrond moet staan.
window_get_stayontop() Geeft terug of het venster altijd op de voorgrond staat.
window_set_sizeable(sizeable) Stelt in of het vensterformaat door de speler kan worden gewijzigd. (De speler kan het formaat alleen wijzigen in de venstermodus en als de venster rand wordt getoond.)
window_get_sizeable() Geeft terug of het vensterformaat wijzigbaar is voor de speler.
window_set_caption(caption) Stelt de titelregel (een tekenreeks) van het venster in. Normaal stel je dit in bij het definiëren van de kamer (room). Deze titel is te wijzigen met behulp van de variabele room_caption. Deze functie is normaal gesproken dus niet handig, behalve als je de kamer zelf tekent, in plaats van dit aan *Game Maker* over te laten. De titel is alleen zichtbaar als het venster niet in volledig schermmodus draait en de vensterrand zichtbaar is.
window_get_caption() Geeft de venster titel terug.
window_set_cursor(curs) Stelt de muis cursor in die binnen het venster wordt gebruikt. Je kunt de volgende constanten gebruiken:

cr_default
cr_none
cr_arrow
cr_cross
cr_beam
cr_size_nesw
cr_size_ns

cr_size_nwse

cr_size_we

cr_uparrow

cr_hourglass

cr_drag

cr_nodrop

cr_hsplit

cr_vsplit

cr_multidrag

cr_sqlwait

cr_no

cr_appstart

cr_help

cr_handpoint

cr_size_all

Als je de muis cursor wilt verbergen, gebruik dan `cr_none` als waarde.
window_get_cursor() Geeft de cursor die in het venster wordt gebruikt terug.
window_set_color(color) Stelt de kleur in van dat deel van het venster dat niet wordt gebruikt om de kamer te tonen.
window_get_color() Geeft de venster kleur terug.
window_set_region_scale(scale,adaptwindow) Als het venster groter is dan de huidige kamer wordt de kamer normaal gesproken getoond in een vierkant gecentreerd in het venster. Het is ook mogelijk om het te laten verschalen en zo het hele venster te vullen. Een waarde van 1 betekend geen verschaling. Bij een waarde van 0 zal de vierkant zo worden geschaald dat het het hele venster vult. Als je de waarde negatief zet zal het worden opgeschaald zodat het zo groot mogelijk in het venster past maar toch de lengte/breedte verhouding van het vierkant hetzelfde blijft. (Wat je normaal gesproken wilt.) `adaptwindow` geeft aan of het vensterformaat moet worden gewijzigd als de geschaalde kamer er niet in past. Het wijzigen van het vensterformaat is alleen effectief als de schaalfactor positief is.
window_get_region_scale() Geeft de schaal van het getekende gebied terug.

Het venster heeft een positie op het scherm en een formaat. (Als we praten over positie en formaat bedoelen we altijd het gedeelte van het venster zonder de randen.) Je kunt deze waarden wijzigen hoewel je dit hoegenaamd altijd met je spel doet. Normaal gesproken worden ze automatisch vastgesteld of door de speler. De volgende functies kunnen worden gebruikt om de venster positie en het formaat te wijzigen. Onthoud dat deze functies een verband hebben met de venstermodus.

Als het venster in volledig schermmodus draait kunnen de waarden wel worden gewijzigd, maar zullen ze pas effect hebben als de volledig schermmodus wordt uitgeschakeld.

`window_set_position(x,y)` Stelt de positie van het venster in op de aangegeven positie.
`window_set_size(w,h)` Stelt het formaat van het venster in op de aangegeven grootte. Merk op dat als de aangegeven grootte te klein is om het tekengebied te kunnen bevatten, het (automatisch) groot genoeg wordt gehouden om het tekengebied te kunnen bevatten.
`window_set_rectangle(x,y,w,h)` Stelt de positie en het formaat van het venster in. (Verwerkt de twee bovenstaande routines in een.)
`window_center()` Centreert het venster op het scherm.
`window_default()` Geeft het venster het standaard formaat en positie (gecentreerd) op het scherm.
`window_get_x()` Geeft de huidige x-coördinaat van het venster terug.
`window_get_y()` Geeft de huidige y-coördinaat van het venster terug.
`window_get_width()` Geeft de huidige breedte van het venster terug.
`window_get_height()` Geeft de huidige hoogte van het venster terug.

Nog eens: je zult misschien nooit een van deze venster positioneringfuncties gebruiken omdat *Game Maker* hier automatisch voor zorgt.

In bijzondere gevallen wil je misschien de positie van de muis ten opzichte van het venster weten. (Normaal gesproken gebruik je altijd de muisposities in de room, bijv. met een 'view'.) De volgende functies bestaan hier voor:

`window_mouse_get_x()` Geeft de x-coördinaat van de muis in het venster terug.
`window_mouse_get_y()` Geeft de y-coördinaat van de muis in het venster terug.
`window_mouse_set(x,y)` Stelt de positie van de muis in het venster in op de aangegeven waardes.

Geluid en muziek

Geluid speelt een cruciale rol in computerspellen. Geluiden zijn toegevoegd in je spel in the vorm van geluid resources. Verzeker je er van dat de namen die je gebruik geldige variabel namen zijn. Zoals je zult zien kun je vier verschillende vormen van geluid aangeven: normale geluiden, achtergrond muziek, 3D geluiden en geluiden die moeten worden afgespeeld door de media speler.

Normale geluiden worden gebruikt voor geluidseffecten. Over het algemeen worden hier wave-bestanden voor gebruikt. Er kunnen meerdere van deze geluiden op hetzelfde moment worden afgespeeld (ook meerdere instanties van hetzelfde normale geluid). Je kunt alle soorten van effecten op normaal geluid toepassen.

Achtergrond muziek bestaat gewoonlijk uit midi-bestanden, maar daar kunnen ook wave-bestanden voor worden gebruikt. Geluidseffecten kunnen er ook voor worden aangewend. Het enige verschil met normale geluiden is dat slechts één achtergrond muziek tegelijk kan worden gespeeld. Als je er een start zal het lopende geluid worden gestopt.

3D geluiden laten 3D geluidseffecten toe welke hieronder zijn beschreven. Dit zijn 'mono' geluiden (wave of midi).

Tenslotte, als je een ander geluidstype wilt gebruiken, in het bijzonder mp3, kunnen deze niet worden afgespeeld door DirectX. Daarvoor moet de gewone media speler voor worden gebruikt. Dit is veel meer begrensd. Slechts één geluid kan tegelijkertijd worden gespeeld. Er kunnen geen effecten worden toegepast (ook geen volume veranderingen) en de timing voor bijv. geluste (looping) geluiden zijn slecht. Ook kunnen er pauzes en vertragingen in deze geluiden voorkomen. Je wordt dringend verzocht deze niet te gebruiken (ook zullen sommige computers dit niet ondersteunen).

Basis geluid functies

Er zijn vijf verschillende functies gerelateerd aan geluiden, twee om een geluid te spelen, één om the controleren of een geluid wordt gespeeld, en twee om geluiden te stoppen. Meestal wordt de geluidnaam gebruikt als index. Maar je kunt de geluidnaam ook opslaan in een variabele, en die gebruiken.

`sound_play(index)` Speelt het aangeduide bestaand één keer. Als het geluid achtergrond muziek is wordt de huidige achtergrond muziek gestopt.
`sound_loop(index)` Speelt het aangeduide geluid, voortdurend gelust. Als het geluid achtergrond muziek is wordt de huidige achtergrond muziek gestopt.
`sound_stop(index)` Stopt het aangeduide geluid. Als er meerdere geluiden met deze index gelijktijdig spelen, zullen ze alle worden gestopt.
`sound_stop_all()` Stopt alle geluiden.
`sound_isplaying(index)` Geeft (een kopie van) het aangegeven geluid dat wordt gespeeld terug. Merk op dat deze functies true teruggeven als het geluid daadwerkelijk speelt door de luidsprekers. Nadat je een functie hebt gevraagd een geluid te spelen zal het niet onmiddellijk de luidsprekers bereiken dus de functie

kan nog een tijdje false teruggeven. Hetzelfde gebeurt wanneer het geluid is gestopt hoor je het nog een poosje (bijv. door echo) en de functie zal nog 'true' terugkoppelen.

Het is mogelijk om meer geluidseffecten te gebruiken. In het bijzonder zijn het: het volume en de 'pan' (verdeling tussen de linker en rechter luidspreker). In al deze gevallen kan het volume alleen worden verlaagd. Deze functies werken niet voor bestanden die worden gespeeld door de media speler (zoals mp3 bestanden).

sound_volume(index,value) Wijzigt het volume voor het aangegeven geluid (0 = laag, 1 = hoog).
sound_global_volume(value) Wijzigt het globale volume voor alle geluiden (0 = laag, 1 = hoog).
sound_fade(index,value,time) Wijzigt het volume voor het aangegeven geluid in een nieuwe waarde (0 = laag, 1 = hoog) gedurende de aangegeven tijd (in duizendste seconden). Dit kan worden gebruikt om geluid te laten vervagen of versterken (fade in/ fade out).
sound_pan(index,value) Wijzigt de 'pan' voor het aangegeven geluid (-1 = links, 0 = centrum, 1 = rechts).
sound_background_tempo(factor) Wijzigt het tempo van de achtergrond muziek (als het een midi bestand is). `factor` geeft de factor waarmee het tempo mee moet worden vermenigvuldigd aan. Zo correspondeert een waarde van 1 met het normale tempo. Grotere waardes corresponderen met een sneller tempo, kleinere waarden met een langzamer tempo. Moet liggen tussen 0.01 and 100.

Naast midi en wave bestanden (en mp3 bestanden) is er een vierde type bestand dat kan worden gespeeld: direct-music bestanden. Deze hebben de extensie .sgt. Zulke bestanden verwijzen echter vaak naar ander bestanden, bijv. met band of stijl informatie. Om deze bestanden te vinden, moet het geluidssysteem weten waar deze staan. Tot zover kun je de volgende functies gebruiken om de zoek map van deze bestanden in te stellen. Merk op dat je deze bestanden zélf moet toevoegen. *Game Maker* voegt niet automatisch zulke bestanden in.

sound_set_search_directory(dir) Stelt de map in waarin direct-music bestanden zich bevinden. De dir tekenreeks mag niet de laatste backslash bevatten.

Geluidseffecten

Deze functies zijn alleen beschikbaar in de geregistreerde versie van Game Maker.

Geluidseffecten kunnen worden gebruikt om de manier waarop geluid en achtergrondmuziek klinkt te wijzigen. Realiseer je dat geluidseffecten alleen op wave en midi bestanden worden toegepast, niet op mp3 bestanden. Dit gedeelte beschrijft de bestaande functies om geluidseffecten te gebruiken en te wijzigen. Realiseer je dat je, om deze functies te gebruiken, goed moet begrijpen hoe geluid en synthesizers werken. Hier wordt geen uitleg over de verschillende parameters gegeven. Zoek op het web of in boeken voor meer informatie.

Om geluid effecten toe te passen voor een specifiek geluid kun je dit aangeven als je de geluid resource definieert of met de volgende functie.

sound_effect_set(snd,effect) Stelt een (combinatie van) geluidseffect(en) voor het aangegeven geluid in. `effect` kan één van de volgende waardes zijn:
se_none
se_chorus
se_echo
se_flanger
se_gargle
se_reverb
se_compressor
se_equalizer
Je kunt een combinatie van effecten instellen door de waardes toe te voegen. Zo kun je bijv. de volgende functies gebruiken:

```
sound_effect_set(snd,se_echo+se_reverb);
```

voor een combinatie van echo en resonantie effecten.

Alle effecten hebben enkele standaard instellingen. Je kunt deze instellingen wijzigen wanneer een effect is toegepast op een geluid. De volgorde is hier cruciaal. Als eerste pas je het effect van het geluid en daarna stel je de parameters er voor in. Wanneer je effecten op het geluid opnieuw toepast, zijn de instellingen kwijt en moet je ze opnieuw instellen. Merk op dat alle parameter moeten liggen in een bepaald bereik, welke hieronder is aangegeven. De volgende functies bestaan om effectparameters te wijzigen:

sound_effect_chorus(snd,wetdry,depth,feedback,frequency,wave,delay,phase) Stel de parameters voor het chorus effect in voor het aangegeven geluid. De volgende parameters kunnen worden gebruikt:

`wetdry` Verhouding van nat (voorbewerkt) signaal naar droog (niet voorbewerkt) signaal (bereik: 0 tot 100, standaard 50)
`depth` Percentage door welke de vertraag tijd is omgebogen door de laag-frequentie oscillator (slingergenerator), in honderdsten van een percentage punt. (bereik: 0 tot 100, standaard 25)
`feedback` Percentage van het output signaal om terug te koppelen in de effecteninput. (bereik: -99 tot 99, standaard 0)
`frequency` Frequentie van de laag-frequentie oscillator. (bereik: 0 tot 10, standaard 10)
`wave` Golfvorm van de LFO. (0 = driehoek, 1 = golf, standaard 1)
`delay` Aantal duizendste seconden dat de input is gepauzeerd voordat deze wordt afgespeeld (bereik: 0 tot 20, standaard 0)
`phase` Periode verschil tussen links en rechts LFO's (bereik: 0 tot 4, standaard 2)
`sound_effect_echo(snd,wetdry,feedback,leftdelay,rightdelay,pandelay)` Stel de parameters voor het echo effect voor het aangegeven geluid in. De volgende parameters kunnen worden gezet:
`wetdry` Verhouding tussen nat (voorbewerkt) signaal en droog (niet voorbewerkt) signaal. (bereik: 0 tot 100, standaard 50)
`feedback` Percentage terugkoppeling naar de input (bereik: 0 tot 100, standaard 0)
`leftdelay` Pauze voor het linker kanaal, in duizendste seconden. (bereik: 1 tot 2000, standaard 333)
`rightdelay` Pauze voor rechter kanaal, in duizendste seconden. (bereik: 1 tot 2000, standaard 333)
`pandelay` Of de linker en rechter pauzes met elke opeenvolgende echo moeten worden geruild. (0 = niet ruilen, 1 = ruil, standaard 0)
`sound_effect_flanger(snd,wetdry,depth,feedback,frequency,wave,delay,phase)` Zet de parameters voor het flanger effect voor het aangegeven geluid. De volgende parameters kunnen worden gezet:
`wetdry` Verhouding van nat (voorbewerkt) signaal naar droog (niet voorbewerkt) signaal. (bereik: 0 tot 100, standaard 50)
`depth` Percentage door welke de pauzetijd is verbogen door de laag-frequente slingergenerator, in honderdste van een procentpunt (bereik: 0 tot 100, standaard 25)
`feedback` Percentage van output signaal om terug te koppelen in de effectinput. (bereik: -99 tot 99, standaard 0)
`frequency` Frequentie van de LFO. (bereik: 0 tot 10, standaard 0)
`wave` Golfvorm van de LFO. (0 = driehoek, 1 = golf, standaard 1)

`delay` Aantal duizendste seconden waarmee de input is gepauzeerd alvorens deze wordt teruggekoppeld. (bereik: 0 tot 20, standaard 0)
`phase` Fase verschil tussen de linker en de rechter LFO's (bereik: 0 tot 4, standaard 2)
`sound_effect_gargle(snd,rate,wave)` Zet de parameters voor het gorgel effect voor het aangegeven geluid. De volgende parameters kunnen worden gebruikt:
`rate` Grootte van verbuiging, in Hertz (trillingen per seconde). (bereik: 1 tot 1000, standaard 1)
`wave` Vorm van de verbuigingsgolf. (0 = driehoek, 1 = vierkant, standaard 0)
`sound_effect_reverb(snd,gain,mix,time,ratio)` Zet de parameters voor het nagalm effect voor het aangegeven geluid. De volgende parameters kunnen worden gezet:
`gain` Input bereik van het signaal, in decibellen (dB). (bereik: -96 tot 0, standaard 0)
`mix` Nagalm mix, in dB. (bereik: -96 tot 0, standaard 0)
`time` Nagalm tijd, in duizendste seconden. (bereik: 0.001 tot 3000, standaard 1000)
`ratio` Frequentie verhouding. (bereik: 0.001 tot 0.999, standaard 0.001)
`sound_effect_compressor(snd,gain,attack,release,threshold,ratio,delay)` zet de parameters voor het compressor effect voor het aangegeven geluid. De volgende parameters kunnen worden gezet:
`gain` Output bereik van het signaal na compressie. (bereik: -60 tot 60, standaard 0)
`attack` Tijd voordat de compressie zijn volle waarde bereikt. (bereik: 0.01 tot 500, standaard 0.01)
`release` Snelheid waarmee compressie wordt gestopt nadat de input onder de treshold komt. (bereik: 50 tot 3000, standaard 50)
`threshold` Punt waar compressie beging, in decibellen. (bereik: -60 tot 0, standaard -10)
`ratio` Compressie verhouding. (bereik: 1 tot 100, standaard 10)
`delay` Tijd nadat treshold is bereikt voordat de attack-fase is gestart, in duizendste seconden. (bereik: 0 tot 4, standaard 0)
`sound_effect_equalizer(snd,center,bandwidth,gain)` Zet de parameters voor het equalizer effect voor het aangegeven geluid. De volgende parameters kunnen worden gezet:

`center` Centrum frequentie in Hertz. (bereik: 80 tot 16000)

`bandwidth` Bandbreedte, in semi-tonen. (bereik: 1 tot 36)

`gain` Gain. (bereik: -15 tot 15)

3D geluid

Deze functies zijn alleen beschikbaar in de geregistreerde versie van Game Maker.

3D geluiden refereren naar geluid dat een positie (en snelheid) heeft ten opzichte van de luisteraar. Hoewel het meest bedoelde gebruik is voor 3D spellen is het ook effectief voor 2D spellen. Het idee is dat een geluid een positie heeft in ruimte. In alle functies is de luisteraar verondersteld op positie (0,0,0). Het systeem berekend hoe de luisteraar het geluid zou horen en past het overeenkomstig aan. Het effect is in het bijzonder goed als je een goed luidspreker systeem hebt maar werkt ook op kleine luidsprekers.

Naast een positie, heeft het geluid ook een snelheid. Dit leidt tot welbekende Doppler effecten welke correct zijn verwerkt in het model. Ten slotte kan het geluid een orientatie hebben, die ook goed wordt weergegeven.

Game Maker ondersteunt 3D geluid opties met de onderstaande functies. Ze werken alleen voor geluid resources die zijn gemerkt als 3D. (Het nadeel is dat 3D geluiden mono zijn en niet stereo.)

`sound_3d_set_sound_position(snd,x,y,z)` Zet de positie van het aangegeven geluid ten opzichte van de luisteraar op de aangegeven positie in ruimte. Waardes op de x-as woorden groter van links naar rechts, op de y-as van beneden naar boven en op de z-as van dichtbij naar ver. De waardes hebben als eenheid meters. Het volume waarmee het geluid wordt gehoord hangt af van deze maten op dezelfde manier als in de echte wereld.

`sound_3d_set_sound_velocity(snd,x,y,z)` Zet de snelheid van het aangegeven geluid met de aangegeven vector (richting en snelheid) in ruimte. Merk op dat het instellen van snelheid niet betekent dat de positie veranderd. De snelheid is alleen gebruikt voor het berekenen van Doppler-effecten. Dus als je het geluid wilt verplaatsen moet je dit zelf wijzigen door middel van de positie van het geluid.

`sound_3d_set_sound_distance(snd,mindist,maxdist)` Zet de minimum afstand waarop het geluid niet meer sterker wordt en de maximum afstand waarop het geluid niet meer kan worden gehoord. Dus als de afstand ligt tussen 0 en de minimum afstand het geluid is op de maximale sterkte. Als tussen de minimale afstand en de maximale afstand de sterkte langzaam daalt tot de maximale afstand

is bereikt of het geluid niet meer hoorbaar is. Standaard is de minimum afstand 1 meter en de maximale afstand is 1 biljoen meter.

sound_3d_set_sound_cone(snd,x,y,z,anglein,angleout,voloutside) Normaal gesproken heeft geluid dezelfde sterkte in alle richtingen op dezelfde afstand van de bron. Je kunt de geluidskegel zetten om dit te veranderen en het geluid een richting te geven. x,y,z specificeren de richting van de geluidskegel. `anglein` specificeert de binnenwerkse hoek. Als de luisteraar binnen deze hoek is hoort hij het dit op zijn normale volume. `angleout` specificeert de buitenwerkse hoek. Als de luisteraar hier buiten is hoort hij het geluid op volume `voloutside`. Om precies te zijn, `voloutside` is een negatief getal dat aangeeft hoeveel honderden decibellen moeten worden verminderd van het binnenwerkse volume. Tussen de binnenwerkse en buitenwerkse hoek wordt het volume geleidelijk minder.

CD muziek

Deze functies zijn alleen beschikbaar in de geregistreerde versie van Game Maker.

Er zijn ook een aantal functies voor muziek van een CD.

cd_init() Moet eerst worden uitgevoerd voordat de andere functies worden gebruikt. Moet ook worden uitgevoerd wanneer een CD is gewijzigd (of simpel van tijd tot tijd).

cd_present() Geeft terug of er een CD in de standaard CD drive is.

cd_number() Geeft het aantal nummers op de CD terug.

cd_playing() Geeft terug of de CD speelt.

cd_paused() Geeft terug of de CD is gepauzeerd of gestopt.

cd_track() Geeft het nummer van de huidige track terug (het eerste nummer is 1).

cd_length() Geeft de lengte van de totale CD in duizendste seconden terug.

cd_track_length(n) Geeft de lengte van nummer n van de CD in duizendste seconden terug.

cd_position() Geeft de huidige positie op de CD in duizendste seconden terug.

cd_track_position() Geeft de huidige positie van het nummer wat wordt gespeeld terug in duizendste seconden.

cd_play(first,last) Zegt de CD om nummer first te spelen tot nummer last. Als je de hele CD wilt afspelen gebruik je 1 en 1000 als argumenten.

cd_stop() Stopt het afspelen.

cd_pause() Pauzeert het afspelen.

cd_resume() Hervat het afspelen.

cd_set_position(pos) Zet de positie op de CD in duizendste seconden.

cd_set_track_position(pos) Zet de positie in het huidige nummer in duizendste seconden.

cd_open_door() Opent de CD-la van de CD speler.

cd_close_door() Sluit de la van de CD speler.

Er is een heel algemene functie om toegang te krijgen tot de multimedia functionaliteit van Windows.

MCI_command(str) Deze functies zenden de commando tekenreeks naar het Windows multimedia systeem gebruik makend van de Media Control Interface (MCI). Het geeft de tekenreeks terug. Je kunt dit gebruiken om alle soorten van multimedia toepassingen te beheren. Zie de Windows documentatie voor informatie hoe die commando te gebruiken. Bijvoorbeeld `MCI_command('play cdaudio from 1')` speelt een CD (nadat je het correct hebt geïnitialiseerd, door gebruik te maken van andere commando's). Deze functie is alleen voor gevorderde gebruikers!

Splash screens, highscores, en andere pop-ups

In dit gedeelte beschrijven we een paar functies, die kunnen worden gebruikt om splash screens met video's, plaatjes, berichten en vragen aan de speler te laten zien, en om de highscore lijst te laten zien.

Splash beelden

Veel spellen hebben splash screens. Deze schermen laten: een video, een plaatje, of tekst zien. Meestal worden ze gebruikt aan het begin van een spel (als een intro), het begin van een level, of aan het eind van een spel (bijvoorbeeld de credits). In *Game Maker* kunnen zulke splash screens met tekst, plaatjes of video ieder moment van het spel worden getoond. Het spel is tijdelijk gepauzeerd terwijl het splash screen wordt getoond. Dit zijn de te gebruiken functies:

show_text(fname,full,backcol,delay) Laat een tekst splash beeld zien. fname is de naam van het tekst bestand (.txt of .rtf). Je moet dit bestand zelf in de map met het spel zetten. Ook wanneer je een stand-alone versie van je spel maakt, moet je niet vergeten het bestand erbij te doen . full verwijst of het in full-screen mode moet worden getoond. backcol is de achtergrond kleur, en delay is de

vertraging in seconden voordat de speler terugkeert naar het spel. (De speler kan altijd nog met de muis klikken in het scherm om terug te gaan naar het spel.)

show_image(fname,full,delay) Laat een splash beeld met een plaatje zien. fname is de naam van het afbeelding bestand (alleen .bmp, .jpg en .wmf bestanden). Je moet dit bestand zelf in de map van het spel plaatsen. full verwijst of het in full-screen mode moet worden getoond. delay is de vertraging in seconden voordat de speler terugkeert naar het spel.

show_video(fname,full,loop) Laat een splash beeld met een video zien. fname is de naam van het video bestand (.avi of .mpg). Je moet dit bestand zelf in de map met het spel zetten. full verwijst of het in full-screen mode moet worden getoond. loop verwijst of de video na het eind opnieuw moet beginnen.

show_info() Laat het spel informatie scherm zien.

load_info(fname) Laad de spel informatie uit het bestand fname genaamd. Dit moet een rtf-bestand zijn. Dit maakt het mogelijk verschillende help bestanden op verschillende momenten te laten zien.

Pop-up berichten en vragen

Een paar andere functies bestaan: pop-up berichten, vragen, een keuzemenu of een dialoog waar de speler een nummer, een tekenreeks of een verwijzing naar een kleur of bestandsnaam kan invoeren.

show_message(str) Laat een dialoog zien met str als een bericht.

show_message_ext(str,but1,but2,but3) Laat een dialoog zien met str als een bericht en maximaal drie knoppen. but1, but2 and but3 bevatten de tekst op de knoppen. Een lege tekenreeks betekent dat de knop niet word getoond. In de teksten kun je het & symbool gebruiken om te laten zien dat het volgende teken wordt gebruikt als de toetsenbord snelkoppeling voor deze knop. De functie geeft het getal van de knop waarop geklikt is terug (0 als de gebruiker op de Esc-toets drukt).

show_question(str) Laat een vraag zien; geeft true wanneer de gebruiker Yes klikt en false als de gebruiker op No klikt.

get_integer(str,def) Vraagt de speler voor een getal waarde. str is het bericht. def is het nummer dat al wordt getoond zonder dat de gebruiker iets intypt.

get_string(str,def) Vraagt de speler voor een tekenreeks. str is het bericht. def is de tekenreeks die al wordt getoond zonder dat de gebruiker iets intypt.

message_background(back) Selecteert het achtergrond plaatje voor de pop-ups van alle functies hierboven. back moet een achtergrond zijn die is geïmporteerd in

het spel. als back transparant is dan is de achtergrond ook transparant (alleen voor Windows 2000 of hoger).

message_alpha(alpha) Bepaalt hoe transparant een pop-up box moet zijn voor alle functies hierboven. alpha moet liggen tussen 0 (niet te zien(volledig transparant) en 1 (niet transparant) (alleen voor Windows 2000 of hoger).

message_button(spr) Bepaalt de afbeelding die gebruikt kan worden voor de knoppen in het pop-up bericht. spr moet een sprite zijn bestaande uit drie afbeeldingen, de eerste verwijst naar de knop wanneer er niet op is geklikt en de muis is ver weg, de tweede verwijst naar de knop wanneer de muis wel op de knop staat maar niet klikt, en de derde is de knop wanneer er op is geklikt.

message_text_font(name,size,color,style) Bepaalt het lettertype voor de tekst in het pop-up bericht.(Dit is een normaal Windows lettertype, niet een geimpoteerd lettertype!) style verwijst naar de stijl van het lettertype (0=normaal, 1=vet, 2=schuin, en 3=vet-schuin).

message_button_font(name,size,color,style) Bepaalt het lettertype voor de tekst op de knoppen in het pop-up bericht. style verwijst naar de stijl van het lettertype (0=normaal, 1=vet, 2=schuin, en 3=vet-schuin).

message_input_font(name,size,color,style) Bepaalt het lettertype voor het veld om tekst in te typen in het pop-up bericht. style verwijst naar de stijl van het lettertype (0=normaal, 1=vet, 2=schuin, en 3=vet-schuin).

message_mouse_color(col) Bepaalt de kleur van het lettertype in het pop-up bericht wanneer de muis op de knop staat.

message_input_color(col) Bepaalt de kleur voor de achtergrond van het veld om tekst in te typen in het pop-up bericht..

message_caption(show,str) Bepaalt de tekst voor de titelbalk. show verwijst of er een rand om moet worden getekend of niet wel (1) of niet (0) en str verwijst naar de inhoud wanneer de rand wel wordt getekend.

message_position(x,y) Bepaalt de positie van het pop-up bericht op het scherm.

message_size(w,h) Bepaalt de grootte van het pop-up bericht op het scherm. als je 0 kiest voor de breedte de breedte van de afbeelding wordt gebruikt. Als je 0 kiest voor de hoogte de hoogte wordt berekent op basis van de regels in het pop-up bericht.

show_menu(str,def) Laat een pop-up menu zien. str verwijst naar de tekst in het menu. Dit bestaat voor de verschillende menu items met een verticale streep ertussenin. Bijvoorbeeld, str = 'menu0|menu1|menu2'. Wanneer het eerste item is geselecteerd wordt er een 0 teruggegeven etc. wanneer de speler geen item selecteert, de voorgeprogrammeerde waarde def(de waarde kun je zelf bepalen)

wordt teruggegeven.

show_menu_pos(x,y,str,def) Laat een pop-up menu zien net als in de vorige functie maar nu op positie x,y op het scherm.

get_color(defcol) Vraagt de speler voor een kleur. defcol is de voorgeprogrammeerde kleur(deze kan je ook zelf bepalen). Als de gebruiker op Cancel drukt wordt de waarde -1 teruggegeven.

get_open_filename(filter,fname) Vraagt de speler een bestandsnaam te openen met de gegeven filter. de filter heeft heet formaat als bijv. 'name1|mask1|name2|mask2|...'. Een mask bevat de verschillende opties met een semicolon ertussen. * betekent iedere tekenreeks. Bijvoorbeeld: 'bitmaps|*.bmp;*.wmf'. Als de gebruiker op Cancel klikt wordt een lege tekenreeks teruggegeven.

get_save_filename(filter,fname) Vraagt de speler op te slaan met de gegeven filter. Als de gebruiker op Cancel klikt wordt een lege tekenreeks teruggegeven.

get_directory(dname) Vraagt om een pad. dname is de voorgeprogrammeerde padnaam. Als de gebruiker op Cancel drukt wordt een lege tekenreeks teruggegeven.

get_directory_alt(capt,root) Een alternatief om naar een pad te vragen. capt is inhoud dat wordt getoond. root is de root van het pad dat moet worden getoond. Gebruik een lege tekenreeks om het hele pad te laten zien. Als de gebruiker op Cancel klikt wordt een lege tekenreeks teruggegeven..

show_error(str,abort) Laat een standaard foutmelding(en/of schrijft het naar het log bestand). abort verwijst of het spel moet worden afgebroken.

Highscore lijst

Eén speciale pop-up is de highscore list dat wordt gehandhaafd door elk spel. De volgende functies bestaan:

highscore_show(numb) Laat de highscore tabel zien. numb is de nieuwe score. Als deze score goed genoeg is voor de lijst, dan kan de speler een naam invoeren. Gebruik -1 om alleen de highscore lijst te laten zien.

highscore_set_background(back) Selecteert de achtergrond afbeelding om te gebruiken. `back` moet een geïmporteerde achtergrond zijn.

highscore_set_border(show) Bepaald of er om de tabel een rand moet worden getekend.

highscore_set_font(name,size,style) Bepaald het lettertype dat wordt gebruikt in de tabel. (Dit is een normaal Windows lettertype, niet een geïmporteerd

lettertype.) Je moet de naam, grootte en stijl (0=normaal, 1=vet, 2=schuin, 3=vetschuin).

highscore_set_colors(back,new,other) Bepaald de kleuren die worden gebruikt voor de achtergrond, de nieuwe naam, en de overige namen in de highscore.

highscore_set_strings(caption,nobody,escape) Veranderd de verschillende normale tekenreeks wanneer de highscore wordt getoond. `caption` is de inhoud van de tabel. `nobody` is de tekenreeks die wordt gebruikt wanneer er een plaats niet wordt gebruikt. `escape` is de tekenreeks onderin de tabel verwijzend om op de Esc-toets te drukken om weg te gaan. Je kan dit bijv. gebruiken als het spel in een andere taal dan engels is.

highscore_show_ext(numb,back,border,col1,col2,name,size) Laat de highscore zien met een paar opties (kan ook worden bereikt met een paar van de vorige functies). numb is de nieuwe score. Als deze score goed genoeg is om aan de lijst te worden toegevoegd, de speler kan een naam invoeren. gebruik -1 om De lijst te laten zien. back is de achtergrond afbeelding, border verwijst of er een rand moet worden getekend of niet. col1 is de kleur voor de nieuwe score, col2 de kleur voor de andere scores. name is de naam van het lettertype dat wordt gebruikt, en size is de grootte van het lettertype.

highscore_clear() Leegt de highscore list.

highscore_add(str,numb) Voegt een speler met naam: `str` en score `numb` toe aan de lijst.

highscore_add_current() Voegt de score toe aan de lijst. De speler wordt gevraagd een naam op te geven.

highscore_value(place) Geeft de score terug van de persoon op de opgegeven plaats(1-10). Dit kan worden gebruikt om je eigen highscore list te maken.

highscore_name(place) Geeft de naam terug van de persoon op de opgegeven plaats(1-10).

draw_highscore(x1,y1,x2,y2) Tekent de highscore lijst in de room in De opgegeven rechthoek, gebruikmakend van het standaard lettertype.

Resources

In *Game Maker* kun je diverse types van resources, zoals sprites, geluiden, lettertypes, objects, enz. bepalen. In dit hoofdstuk zal je een aantal functies vinden die informatie over de recources geven. In het volgende hoofdstuk vind je informatie om recources te wijzigen en te creëren.

Sprites

De volgende functies zullen je informatie over sprites geven:

`sprite_exists(ind)` Geeft terug of de sprite in de betreffende index bestaat.

`sprite_get_name(ind)` Geeft de naam van de sprite terug met de opgegeven index.

`sprite_get_number(ind)` Geeft de nummer van de subimages van de sprite terug met de opgegeven index.

`sprite_get_width(ind)` Geeft de breedte van de sprite terug met de opgegeven index.

`sprite_get_height(ind)` Geeft de hoogte van de sprite met de opgegeven index.

`sprite_get_transparent(ind)` Geeft terug of de sprite in de betreffende index doorzichtig is.

`sprite_get_smooth(ind)` Geeft terug of de sprite in de betreffende index smoothed randen heeft.

`sprite_get_preload(ind)` Geeft terug of de sprite in de betreffende index moet preloaded zijn.

`sprite_get_xoffset(ind)` Geeft terug de x-offset van de sprite met de opgegeven index.

`sprite_get_yoffset(ind)` Geeft terug de y-offset van de sprite met de opgegeven index.

`sprite_get_bbox_left(ind)` Geeft terug de linker zijde van de bounding box van de sprite met de opgegeven index.

`sprite_get_bbox_right(ind)` Geeft terug de rechter zijde van de bounding box van de sprite met de opgegeven index.

`sprite_get_bbox_top(ind)` Geeft terug de top zijde van de bounding box van de sprite met de opgegeven index.

`sprite_get_bbox_bottom(ind)` Geeft terug de onder zijde van de bounding box van de sprite met de opgegeven index.

`sprite_get_bbox_mode(ind)` Geeft terug de bounding box mode (0=automatisch, 1=volledige afbeelding, 2=handmatig) van de sprite met de opgegeven index

`sprite_get_precise(ind)` Geeft terug of de sprite in de betreffende index met gebruik van het nauwkeurige botsing controleren.

Geluiden

De volgende functies zullen je informatie over geluid geven:

sound_exists(ind) Geeft terug of het geluid in de betreffende index bestaat.
sound_get_name(ind) Geeft de naam van het geluid terug met de opgegeven index.
sound_get_kind(ind) Geeft het soort van het geluid met de opgegeven index terug. (0=normaal, 1=achtergrond, 2=3d, 3=media speler).
sound_get_preload(ind) Geeft terug of het geluid met de gegeven index van te voren geladen word.

De geluiden gebruiken vele resources en de meeste systemen kunnen slechts een beperkt aantal geluiden opslaan en spelen. Als je een groot spel maakt zou je meer controle willen hebben over de geluiden die in het audio geheugen worden geladen. Je kunt de voorlaad optie gebruiken voor geluiden, om ervoor te zorgen de geluiden slechts worden geladen wanneer ze worden gebruikt. Dit heeft niettemin het probleem dat je een kleine fout zou kunnen krijgen wanneer het geluid voor het eerst wordt gebruikt. Ook worden de geluiden niet automatisch leeggemaakt wanneer je hen niet meer nodig hebt. Voor meer controle kun je de volgende functies gebruiken.

sound_discard(index) Bevriest het audio geheugen dat word gebruikt voor het vermelde geluid.
sound_restore(index) Herstelt het vermelde geluid in audio geheugen voor het directe spelen.

Achtergronden

De volgende functies zullen je informatie over achtergronden geven:

background_exists(ind) Geeft terug of de achtergrond in de betreffende index bestaat.
background_get_name(ind) Geeft de naam van de achtergrond terug met de opgegeven index.
background_get_width(ind) Geeft de breedte van de achtergrond terug met de opgegeven index.
background_get_height(ind) Geeft de hoogte van de achtergrond terug met de opgegeven index.
background_get_transparent(ind) Geeft terug of de achtergrond in de betreffende index doorzichtig is.
background_get_smooth(ind) Geeft terug of de achtergrond in de betreffende index smoothed randen heeft.

background_get_preload(ind) Geeft terug of de achtergrond in de betreffende index preloaded moet zijn.

Lettertypes

De volgende functies zullen je informatie over de lettertypes geven:

font_exists(ind) Geeft terug of het lettertype in de betreffende index bestaat.
font_get_name(ind) Geeft de naam van het lettertype resource terug met de opgegeven index.
font_get_fontname(ind) Geeft de naam van het lettertype terug met de opgegeven index.
font_get_bold(ind) Geeft terug of het lettertype met de gegeven index vet is.
font_get_italic(ind) Geeft terug of het lettertype met de gegeven index italic is.
font_get_first(ind) Geeft het eerste karakter terug van het lettertype met de gegeven index.
font_get_last(ind) Geeft het laatste karakter terug van het lettertype met de gegeven index.

Paden

De volgende functies zullen je informatie over de paden geven:

path_exists(ind) Geeft terug of het pad in de betreffende index bestaat.
path_get_name(ind) Geeft de naam van het pad terug met de opgegeven index.
path_get_length(ind) Geeft de lengte terug van het pad met de gegeven index.
path_get_kind(ind) Geeft het soort van connecties terug van het pad met de gegeven index. (0=recht, 1=afgerond).
path_get_closed(ind) Geeft terug of het pad met de gegeven index gesloten is.
path_get_precision(ind) Geeft de precisie van de afrondingen terug.
path_get_number(ind) Geeft het aantal punten terug van het pad .
path_get_point_x(ind,n) Geeft de x-coördinaat terug van het n-ste punt op het pad. 0 is het eerste punt.
path_get_point_y(ind,n) Geeft de y-coördinaat terug van het n-ste punt op het pad. 0 is het eerste punt.
path_get_point_speed(ind,n) Geeft de snelheids factor terug van het n-ste punt op het pad. 0 is het eerste punt.

path_get_x(ind,pos) Geeft de x-coördinaat terug van de positie pos op het pad. pos moet tussen 0 en 1 liggen.

path_get_y(ind,pos) Geeft de y-coördinaat terug van de positie pos op het pad. pos moet tussen 0 en 1 liggen.

path_get_speed(ind,pos) Geeft de snelheids factor terug van de positie pos op het pad. pos moet tussen 0 en 1 liggen.

Scripts

De volgende functies zullen je informatie over scripts geven:

script_exists(ind) Geeft terug of het script in de betreffende index bestaat.

script_get_name(ind) Geeft de naam van het script terug met de opgegeven index.

script_get_text(ind) Geeft de tekst terug van het script met de gegeven index.

Tijdlijnen

De volgende functies zullen je informatie over de tijdlijnen geven:

timeline_exists(ind) Geeft terug of de tijdlijn in de betreffende index bestaat.

timeline_get_name(ind) Geeft de naam van de tijdlijn terug met de opgegeven index.

Objecten

De volgende functies zullen je informatie over de objecten geven:

object_exists(ind) Geeft terug of het object in de betreffende index bestaat.

object_get_name(ind) Geeft de naam van het object terug met de opgegeven index.

object_get_sprite(ind) Geeft de index van het standaard sprite terug van het object met de gegeven index.

object_get_solid(ind) Geeft terug of het object met de gegeven index standaard solid is.

object_get_visible(ind) Geeft terug of het object met de gegeven index standaard zichtbaar is.

object_get_depth(ind) Geeft de diepte terug van het object met de gegeven index.

object_get_persistent(ind) Geeft terug of het object met de gegeven index

persistent is.

object_get_mask(ind) Geeft de index terug van het mask van het ocject met de gegeven index. (-1 als het object geen special mask heeft).

object_get_parent(ind) Geeft de index van het parent object terug van het object met de gegeven index. (-1 als het object geen parent heeft).

object_is_ancestor(ind1,ind2) Geeft terug of het object met de index `ind1` parent is van object `ind2`.

Rooms

De volgende functies zullen je informatie over de rooms geven:

room_exists(ind) Geeft terug of de room in de betreffende index bestaat.

room_get_name(ind) Geeft de naam van de room terug met de opgegeven index.

Merk op dat, omdat tijdens het spelen de room veranderd, er andere routines zijn om informatie over de inhoud van de huidige room te krijgen.

Resources veranderen

Deze functies zijn alleen beschikbaar in de geregistreerde versie van Game Maker.

Het is mogelijk nieuwe resources te creëren tijdens het spel. Ook kun je bestaande resources veranderen. Dit hoofdstuk beschrijft de mogelijkheden. Maar wees gewaarschuwd. **Het veranderen van resources kan leiden tot serieuze fouten in je spel!!!** Je moet de volgende regels volgen als je resources wil veranderen:

- Verander geen resources die in gebruik zijn. Dat zal tot fouten leiden! Verander bijvoorbeeld geen sprite dat wordt gebruikt door een instantie.
- Als je het spel opslaat tijdens het spelen, dan worden toegevoegde en veranderde resources niet mee opgeslagen. Dus als je je opgeslagen spel later laadt zullen deze er niet meer zo zijn. Over het algemeen kun je het ingebouwde systeem voor laden en opslaan niet meer gebruiken als je resources verandert.
- Als je je spel herstart tijdens het spelen dan worden de veranderde resources niet in hun originele staat teruggebracht. Over het algemeen kun je de herstart functie niet meer gebruiken als je resources verandert.
- Resources veranderen kan erg langzaam zijn. Bijvoorbeeld, het veranderen van sprites en backgrounds is relatief erg sloom. Dus gebruik dat niet tijdens het spelverloop.

- Resources creëren tijdens het spelverloop kan makkelijk een grote hoeveelheid geheugen gebruiken (vooral bij sprites en backgrounds). Dus wees daar heel voorzichtig mee. Als je bijvoorbeeld een sprite hebt met 32 frames en een grote van 128x128 pixels en je besluit tot het maken van 36 geroteerde kopieën dan gebruik je daarvoor 36x32x128x128x4 = 36 MB geheugen!
- Wees er zeker van dat resources worden verwijderd als ze niet langer nodig zijn, anders zal het systeem snel te veel geheugen gebruiken.

Over het algemeen kun je het best geen resources veranderen tijdens het spelverloop. Beter kun je ze maken en veranderen aan het begin van het spel of misschien het begin van een room.

Sprites

De volgende functies zijn beschikbaar voor het veranderen van de eigenschappen van een sprite:

sprite_set_offset(ind,xoff,yoff) Stelt de offset van het sprite met de gegeven index in.
sprite_set_bbox_mode(ind,mode) Stelt de bounding box mode van het sprite in. (0=automatisch, 1=volledige afbeelding, 2=manual).
sprite_set_bbox(ind,left,top,right,bottom) Stelt de bounding box van het sprite met de gegeven index in. Werkt alleen als de bounding box mode is ingesteld op manual.
sprite_set_precise(ind,mode) Stelt in of het sprite met de gegeven index precise collision checking gebruikt (true of false).

De volgende functies kunnen worden gebruikt om sprites te creëren en te verwijderen.

sprite_duplicate(ind) Creëert een duplicaat van het sprite met de gegeven index. De functie resulteert in de index van het nieuwe sprite. Het resultaat is -1 als er een fout is.
sprite_assign(ind,spr) Kopieert het sprite met de gegeven index naar sprite spr. Dus dit maakt een kopie van het sprite. Op deze manier kun je makkelijk een sprite op een andere over zetten.
sprite_merge(ind1,ind2) Voegt de images van sprite ind2 samen in sprite ind1, voegt ze toe aan het eind. Als de formaten niet overeenkomen dan worden de images uit gerekt tot het past (stretched to fit). Sprite ind2 wordt niet verwijdert!
sprite_add(fname,imgnumb,precise,transparent,smooth,preload,xorig,yorig) Voegt de afbeelding opgeslagen in het bestand fname toe aan de sprite resources. Alleen bmp, jpg and gif afbeeldingen kunnen worden toegevoegd. Als de afbeelding een bmp of jpg is dan kan het een strip zijn met een aantal subimages

voor het sprite. Gebruik imgnumb om aan te geven hoeveel sub images gebruikt worden(1 voor een enkele afbeelding). Voor (geanimeerde) gif afbeeldingen wordt dit argument niet gebruikt de afbeeldingen in de gif worden gebruikt als sub images. precise geeft aan of precise collision checking gebruikt moet worden. `transparent` geeft aan of het sprite deels transparant moet zijn. `smooth` geeft aan of de randen verzacht moeten worden. `preload` geeft aan of het sprite van tevoren geladen moet worden in het texture geheugen. `xorig` en `yorig` geven de positie van de oorsprong van het sprite. De functie resulteert in de index van het nieuwe sprite dat je dan kunt gebruiken om het sprite te tekenen of te verbinden aan een instantie met sprite_index. Het resultaat is -1 als er een fout is.

sprite_replace(ind,fname,imgnumb,precise,transparent,smooth,preload,xorig,yorig) Hetzelfde als bovenstaande alleen wordt in dit geval de sprite met index ind vervangen. Het resultaat van de functie is true wanneer de functie succesvol is uitgevoerd.

sprite_create_from_screen(x,y,w,h,precise,transparent,smooth,preload,xorig,yorig) Creëert een sprite door een deel van het venster te kopiëren. Dat maakt het mogelijk om alle sprites te maken die je wilt. Teken de afbeelding op het scherm met de draw functies en maak dan een sprite daarvan. (Als je dit niet in de draw event doet kun je het zelfs op zo'n manier doen dat het niet zichtbaar is op het scherm.) De overige parameters zijn zoals boven beschreven. De functie resulteert in de index van de nieuw gemaakte sprite. Een waarschuwing is hier op zijn plaats. Ook al spreken we over het scherm, het is eigenlijk het 'tekengebied' (drawing region) dat wordt gebruikt om te kopiëren. Het feit dat er een venster is op het scherm en dat de afbeelding vergroot kan zijn in het venster maakt niet uit.

sprite_add_from_screen(ind,x,y,w,h) Voegt een deel van het venster toe als subimage van het de sprite met index ind.

sprite_create_from_surface(id,x,y,w,h,precise,transparent,smooth,preload,xorig,yorig) Creëert een sprite door een deel van de surface met index id te kopiëren. Dat maakt het mogelijk om alle sprites te maken die je wilt. Teken de afbeelding op de surface met de draw functies en maak dan een sprite daarvan. De functie resulteert in de index van de nieuw gemaakte sprite. Merk op dat de alpha waarde behouden blijven in de sprite.

sprite_add_from_surface(ind,id,x,y,w,h) Voegt een deel van de surface id toe als subimage van sprite met de index ind.

sprite_delete(ind) Wist de sprite uit het geheugen.

De volgende functies zijn beschikbaar om de weergave van een sprite te veranderen.

sprite_set_alpha_from_sprite(ind,spr) Verandert de alpha (transparantie) waarden in de sprite met index `ind` door gebruik te maken van de hue waarden in de sprite `spr`. Dit kan niet ongedaan worden gemaakt.

Geluiden

De volgende functies kunnen gebruikt worden om nieuwe sounds te creëren en te verwijderen.

sound_add(fname,kind,preload) Voegt een sound resource toe aan het spel. fname is de naam van het geluidsbestand. kind definieert het soort geluid (0=normaal, 1=achtergrond, 2=3d, 3=media speler) preload geeft aan of het geluid meteen in het audio geheugen moet worden geladen (true of false). De functie geeft de index van het nieuwe sound als resultaat, dat gebruikt kan worden om het geluid af te spelen. (-1 als er een fout is, bijv. als het bestand niet bestaat).
sound_replace(index,fname,kind,loadonuse) Hetzelfde als bovenstaande functie alleen deze keer wordt het sound met index index vervangen door het nieuwe sound. Resulteert true als de functie succesvol is beëindigd.
sound_delete(index) Wist het sound met index index, en geeft al het geassocieerde geheugen vrij. Het geluid kan niet meer terug gekregen worden.

Achtergronden

De volgende functies kunnen worden gebruikt om nieuwe achtergronden te creëren en te verwijderen.

background_duplicate(ind) Creëert een duplicaat van de achtergrond met de gegeven index. Het resultaat is de index van de nieuwe achtergrond. Als er een fout is is het resultaat -1.
background_assign(ind,back) Vervangt achtergrond met index `ind` door de achtergrond met index `back`.
background_add(fname,transparent,smooth,preload) Voegt de afbeelding in het bestand fname toe als achtergrond resources. Alleen bmp end jpg afbeeldingen kunnen hiervoor worden gebruikt. `transparent` geeft aan of de afbeelding deels transparant is. `smooth` houdt in of de randen smooth moeten zijn. `preload` geeft aan of de afbeelding in het textuur geheugen geladen wordt. Het resultaat is de index van de nieuwe achtergrond dat je kunt gebruiken om de afbeelding te tekenen of toe te wijzen aan de variabele background_index[0] om hem als achtergrond van de room te gebruiken. Als er een fout is is het resultaat -1.
background_replace(ind,fname,transparent,smooth,preload) Zelfde

als bovenstaande alleen wordt in dit geval de achtergrond met index ind overschreven. De functie geeft true als het gelukt is. Als de achtergrond op het moment van uitvoeren zichtbaar is in de room, dan wordt die ook vervangen.

background_create_color(w,h,col,preload) Creëert een achtergrond met de gegeven afmetingen en kleur. Het resultaat is de index van de nieuwe achtergrond. Als er een fout is is het resultaat -1.

background_create_gradient(w,h,col1,col2,kind,preload) Creëert een achtergrond met kleurverloop met de gegeven afmetingen. col1 en col2 geven de 2 kleuren aan. kind is een nummer tussen 0 en 5 dat de soort kleuroverloop aangeeft: 0=horizontaal 1=verticaal, 2= rechthoek, 3=ovaal, 4=dubbel horizontaal, 5=dubbel verticaal. Het resultaat is de index van de nieuwe achtergrond. Als er een fout is is het resultaat -1.

background_create_from_screen(x,y,w,h,transparent,smooth,preload) Creëert een achtergrond door een gegeven gebied van het scherm te kopiëren. Dit maakt het mogelijk om elke achtergrond te maken die je wilt. Teken de afbeelding op het scherm met de draw functies en maak er dan een achtergrond van. (Als je het niet in de draw event doet kan je het zo maken dat het onzichtbaar is op het scherm.) De overige parameters zijn zoals boven. Het resultaat is de index van de nieuwe achtergrond. Een waarschuwing is hert op zijn plaats: Ook al spreken we over het scherm, het is eigenlijk het tekengebied. Het feit dat er een venster is op het scherm en dat the afbeelding vergroot of verkleind wordt maakt niet uit.

background_create_from_surface(id,x,y,w,h,transparent,smooth,preload) Creëert een achtergrond door een gegeven gebied van de surface met de gegeven id te kopiëren. Dit maakt het mogelijk om elke achtergrond te maken die je wilt. Teken de afbeelding op de surface met de draw functies en maak er dan een achtergrond van. Merk op dat alpha waardes behouden blijven in de achtergrond.

background_delete(ind) Wist de achtergrond uit het geheugen.

De volgende functie is bedoeld voor het veranderen van de weergave van een achtergrond.

background_set_alpha_from_background(ind,back) Verandert de alpha (transparantie) waarden in de achtergrond met de index `ind` gebruik makend van de hue waarden in de achtergrond `back`. Dit kan niet ongedaan gemaakt worden.

Lettertypes

Het is mogelijk om lettertypes (fonts) the creëren, vervangen, en te verwijderen tijdens het spel met de volgende functies. (Vervang niet het actieve lettertype of maak het na het vervangen op nieuw actief.)

font_add(name,size,bold,italic,first,last) Voegt een lettertype toe en geeft de index terug, geef aan de name, de size, of het vet of italic moet zijn, en het eerste en laatste karakter.

font_add_sprite(spr,first,prop,sep) Voegt een nieuw lettertype toe en geeft de index terug. Het lettertype wordt gemaakt van een sprite. Het sprite moet een subimage hebben voor elk karakter. `first` geeft de index van het eerste karakter in het sprite aan. Voorbeeld: gebruik `ord('0')` als je sprite alleen bestaat uit cijfers. `prop` geeft aan of het lettertype proportioneel is. In een proportioneel lettertype wordt de breedte van de bounding box gebruikt als de breedte van het karakter. `sep` geeft de hoeveelheid ruimte tussen de karakters aan. Een normale waarde is tussen de 2 en 8, maar het is per lettertype verschillend.

font_replace(ind,name,size,bold,italic,first,last) Vervangt het lettertype `ind` door een nieuw lettertype, geef aan de name, de size, of het vet of italic moet zijn, en het eerste en laatste karakter.

font_replace_sprite(ind,spr,first,prop,sep) Vervangt het lettertype `ind` door een nieuw lettertype gebaseerd op een sprite.

font_delete(ind) Verwijdert een lettertype met de gegeven index.

Paden

Het is mogelijk om paden te creëren en om punten toe te voegen aan paden. Maar verander nooit een pad dat wordt gebruikt door een object. Dat kan zorgen voor onverwachte resultaten. De volgende functies zijn beschikbaar:

path_set_kind(ind,val) Zet de soort van connectie van het pad met de gegeven index (0=recht, 1=gerond).

path_set_closed(ind,closed) Maakt het pad gesloten (true) of open (false).

path_set_precision(ind,prec) Wijzigt de precisie waarmee de ronding van het pad berekend wordt (moet tussen 1 en 8 liggen).

path_add() Voegt een nieuw leeg pad toe. Het resultaat is de index van het pad.

path_delete(ind) Verwijdert het pad met de gegeven index.

path_duplicate(ind) Maakt een kopie van het pad met de gegeven index. Het resultaat is de index van het gekopieerde pad.

path_assign(ind,path) Kopieert het pad met de gegeven index naar pad path. Dus dit maakt een kopie van het pad. Op deze manier kun je makkelijk een pad op

een andere over zetten.

path_append(ind,path) Voegt pad path toe aan path ind.

path_add_point(ind,x,y,speed) Voegt een punt toe aan het pad met de gegeven index, op positie (x,y) en met de gegeven speed factor. Onthoud dat een factor van 100 staat voor de huidige speed. Lagere waarden zorgen voor vertraging en hogere waarden zorgen voor versnelling.

path_insert_point(ind,n,x,y,speed) Voegt een punt in het pad met de gegeven index voor punt n, op positie (x,y) en met de gegeven speed factor.

path_change_point(ind,n,x,y,speed) Verandert punt n in het pad met de gegeven index, met positie (x,y) en de gegeven speed factor.

path_delete_point(ind,n) Verwijdert punt n in het pad met de gegeven index.

path_clear_points(ind) Verwijdert alle punten in het pad. Maakt het pad leeg.

path_reverse(ind) Draait het pad om in tegenovergestelde richting.

path_mirror(ind) Spiegelt het pad horizontaal (het midden blijft het midden).

path_flip(ind) Spiegelt het pad verticaal (het midden blijft het midden).

path_rotate(ind,angle) Draait het pad tegen de klok in met angle graden (draait rond het midden).

path_scale(ind,xscale,yscale) Vergroot/verkleint het pad met gegeven factoren (vanuit het midden).

path_shift(ind,xshift,yshift) Verschuift het pad met de gegeven afstanden.

Scripts

Scripts kunnen niet gewijzigd worden tijdens het uitvoeren van het spel. Scripts zijn deel van de game logica. Het veranderen van scripts zorgt voor zichzelf veranderende code dat erg fout gevoelig is. Er zijn andere manieren om deze doelen te bereiken. Als het echt nodig is om een stukje code uit te voeren dat nog niet bekend is tijdens het ontwerpen (bijv. vanuit een bestand) kunnen de volgende functies gebruikt worden:

execute_string(str) Voert een stukje code in de tekenreeks str uit.

execute_file(fname) Voert een stukje code in een bestand uit.

Soms moet een script index opgeslagen worden in een variabel en gestart worden. Daarvoor gebruik je de volgende functie:

script_execute(scr,arg0,arg1,...) Voert het script met de gegeven index uit met de gegeven argumenten.

Tijdlijnen

De volgende functies zijn beschikbaar voor het maken en veranderen van tijdlijnen. Verander geen tijdlijnen die in gebruik zijn!

`timeline_add()` Voegt een nieuwe tijdlijnen toe. Het resultaat is de index van de gemaakte tijdlijnen.

`timeline_delete(ind)` Verwijdert de tijdlijnen met de gegeven index. Zorg ervoor dat de tijdlijnen niet gebruikt wordt door een object in een room.

`timeline_moment_add(ind,step,codestr)` Voegt een code actie toe aan de tijdlijnen op moment step. codestr is de code voor de acties. Als het moment nog niet bestaat dan wordt die toegevoegd. Zo kun je meerdere code acties aan een moment toevoegen.

`timeline_moment_clear(ind,step)` Deze functie wist alle acties op moment step in tijdlijnen ind.

Objecten

Ook objecten kunnen gemaakt en veranderd worden tijdens het spel. Verander nooit objecten die in gebruik zijn in een room. Dat kan zorgen voor onverwachte effecten.

`object_set_sprite(ind,spr)` Verandert het sprite van een object met de gegeven index. Gebruik -1 om het huidige sprite te verwijderen uit het object.

`object_set_solid(ind,solid)` Geeft aan of het object standaard solid moet zijn of niet (true of false).

`object_set_visible(ind,vis)` Geeft aan of het object standaard visible (zichtbaar) moet zijn of niet (true of false).

`object_set_depth(ind,depth)` Zet de standaard depth voor een object.

`object_set_persistent(ind,pers)` Geeft aan of het object standaard persistent moet zijn of niet (true of false).

`object_set_mask(ind,spr)` Stelt de mask sprite in voor het object met de gegeven index. Gebruik -1 voor de mask van het sprite van het object.

`object_set_parent(ind,obj)` Stelt de parent in van het object `ind`. Gebruik -1 voor geen parent. Het veranderen van de parent verandert het gedrag van het object.

De volgende functies zijn handig voor het creëren van objecten. Zoals bij het veranderen van alle resource, wees erg voorzichtig dat je niet constant nieuwe objecten maakt.

object_add() Voegt een nieuw object toe. Het resultaat is de index van het nieuwe object. Die je kunt gebruiken in de bovenstaande functies en voor het plaatsen van het object in de room.
object_delete(ind) Verwijdert het object met de gegeven index. Zorg ervoor dat het object op dat moment niet in een van de rooms staat.
object_event_add(ind,evtype,evnumb,codestr) Om een object iets te laten doen moeten er events gedefinieerd worden. Je kan alleen code acties toevoegen aan events. Je moet het object het event type en event nummer specificeren. (gebruik de constanten die staan bij de event_perform() functie). Vervolgens specificeer je de code tekenreeks die uitgevoerd moet worden. Je kan meerdere code acties toevoegen aan elke event.
object_event_clear(ind,evtype,evnumb) Je kunt deze functie gebruiken voor het wissen van alle acties in een event van een object.

Het maken van objecten is vooral handig als je scripts of actie bibliotheken maakt. Bijvoorbeeld, een initialisatie script kan een object maken om een tekst weer te geven en een ander script kan dat object gebruiken voor het weergeven van een tekst. Zo heb je een simpel mechanisme om teksten weer te geven zonder voor elke tekst een object te maken.

Rooms

Het veranderen van rooms tijdens het spel is iets gevaarlijks. Je moet je realiseren dat rooms telkens veranderen naar mate er wat gebeurt in het spel. Normaal gebeurt dat alleen in de actieve room en er zijn veel functies beschreven in de vorige secties voor het veranderen van instanties van objecten, achtergronden, en tiles in de actieve room. Maar veranderingen in de actieve room gaan verloren als de room niet persistent is. Je moet nooit dingen van een room veranderen als die actief is of persistent en al eens bezocht is. Zulke wijzigingen gaan meestal verloren maar soms geeft het onverwachte foutmeldingen. Dat komt door het feit dat rooms op een speciale manier met elkaar gekoppeld zijn daarom is er ook geen functie voor het verwijderen van een room.

De volgende functies zijn beschikbaar

room_set_width(ind,w) Stelt de breedte in van de room met de gegeven index.
room_set_height(ind,h) Stelt de hoogte in van de room met de gegeven index.
room_set_caption(ind,str) Stelt de caption (titel) in van de room met de gegeven index.
room_set_persistent(ind,val) Stelt in of de room persistent is of niet.
room_set_code(ind,str) Stelt de initialisatie code in voor de room met de

gegeven index.

room_set_background_color(ind,col,show) Stelt de kleur eigenschappen in van de room met de gegeven index als het geen achtergrondafbeelding heeft. col geeft de kleur aan en show geeft aan of de kleur zichtbaar moet zijn of niet.

room_set_background(ind,bind,vis,fore,back,x,y,htiled,vtiled,hspeed,vspeed,alpha) Stelt de achtergrond in met index bind (0-7) voor de room met de gegeven index. vis geeft aan of de background zichtbaar moet zijn en fore geeft aan of het een voorgrond moet zijn. back is de index van de background afbeelding. x,y geven de positie van de afbeelding aan. htiled en vtiled geven aan of afbeelding horizontaal en vertikaal herhaald moet worden. hspeed en vspeed geven de speed van de background aan. alpha geeft een transparantie waarde aan (1 = niet transparant, en is het snelst).

room_set_view(ind,vind,vis,xview,yview,wview,hview,xport,yport,wport,hport,hborder,vborder,hspeed,vspeed,obj) Stelt de view met index vind (0-7) in voor de room met de gegeven index. vis geeft aan of de view zichtbaar moet zijn. xview, yview, wview, en hview geven de positie van de view in de room aan. xport, yport, wport, en hport geven de positie van de view aan in het venster. Als de view een object moet volgen geven hborder en vborder de minimaal zichtbare rand aan om het object heen. hspeed en vspeed geven de maximale speed aan waarmee de view kan verplaatsen. obj is de index van het object of the index van de instantie van het object.

room_set_view_enabled(ind,val) Stelt in of de view gebruikt moet worden voor de room met de gegeven index.

room_add() Voegt een nieuwe room toe. Het resultaat is de index van de room. Let op dat de room niet op genomen wordt in de room volgorde, dus heeft de room geen volgende en vorige room. als je naar een nieuw gemaakte room wilt moet je de index opgeven.

room_duplicate(ind) Voegt een kopie van de room met de gegeven index toe. Het resultaat is de index van de nieuwe room.

room_assign(ind,room) Maakt room `ind` gelijk aan room `room`, dus maakt een kopie van room `room`.

room_instance_add(ind,x,y,obj) Voegt een instantie van object obj toe aan de room, geplaatst op de aangegeven positie. Het resultaat is de index van de instantie.

room_instance_clear(ind) Verwijdert alle instanties uit de room.

room_tile_add(ind,back,left,top,width,height,x,y,depth) Voegt een nieuw tile toe aan de room op de opgegeven positie. Het resultaat is de index van de nieuwe tile. back is de background waarvan de tile genomen moet worden. left,

top, width en height geven het deel van de background aan dat de tile vormt. x,y geven de positie aan waar de tile geplaatst moet worden en depth geeft de depth aan voor de tile.

`room_tile_add_ext(ind,back,left,top,width,height,x,y,depth,xscale,yscale,alpha)` Zelfde als de bovenstaande functie alleen kun je nu de scaling factor specificeren in x en y richting, en de alpha transparantie voor de tile.

`room_tile_clear(ind)` Verwijdert alle tiles uit de aangegeven room.

Bestanden, register en programma's uitvoeren

In meer geavanceerde spellen wil je waarschijnlijk gebruik maken van het lezen van data in je spel. Of je wilt informatie opslaan tijdens het spelen van het spel. In sommige situaties is het nodig om programma's uit te voeren.

Bestanden

Het is handig om externe bestanden te gebruiken in spellen. Je kan bijvoorbeeld een bestand maken dat beschrijft wat er gaat gebeuren. Ook wil je misschien informatie opslaan voor de volgende keer dat het spel word gespeeld (bijvoorbeeld de huidige room). Met de volgende functies kun je data lezen en schrijven in tekstbestanden:

`file_text_open_read(fname)` Opent het bestand met de aangeduide naam om te lezen. De functie geeft de id van het bestand terug dat in de andere functies gebruikt moet worden. Je kan veelvoudige bestanden tegelijkertijd (max 32) openen. Vergeet ze niet te sluiten als je ze niet meer gebruikt.

`file_text_open_write(fname)` Opent het aangeduide bestand voor schrijven en creëert dat indien het niet bestaat. De functie geeft de id van het bestand terug dat in de andere functies gebruikt moet worden.

`file_text_open_append(fname)` Opent het aangeduide bestand voor het bijvoegen van gegevens aan het einde en creëert dat indien het niet bestaat. De functie geeft de id van het bestand terug dat in de andere functies gebruikt worden moet.

`file_text_close(fileid)` Sluit het bestand met het gegeven bestand id.

`file_text_write_string(fileid,str)` Schrijft de reeks naar het bestand met het gegeven bestand id.

`file_text_write_real(fileid,x)` Schrijf de echte waarde naar het bestand met het gegeven bestand id.

`file_text_writeln(fileid)` Schrijf een 'enter' (teken voor een nieuwe lijn)

naar het bestand.

file_text_read_string(fileid) Leest een reeks van het bestand met het gegeven bestand id en geeft deze reeks terug. Een reeks einden aan het einde van lijn.

file_text_read_real(fileid) Leest een echte waarde van het bestand en keert deze waarde terug.

file_text_readln(fileid) Leest een 'enter' (teken voor een nieuwe lijn) uit het bestand.

file_text_eof(fileid) Geeft een waarde terug als het eind van het bestand is bereikt.

Om bestanden in het bestandssysteem te manipuleren kan de volgende functie gebruiken:

file_exists(fname) Geeft aan of het bestand bestaat (true) of niet bestaat (false).

file_delete(fname) Verwijderd het bestand met de gegeven naam.

file_rename(oldname,newname) Hernoemt het bestand met naam oldname in newname.

file_copy(fname,newname) Kopieert de bestand fname naar de newname.

directory_exists(dname) Geeft aan of de directory bestaat (true) of niet bestaat (false)

directory_create(dname) Creëert een directory met de gegeven naam (inclusief het pad naar het) indien het niet bestaat.

file_find_first(mask,attr) Geeft de naam van het eerste bestand terug dat met het id en de eigenschappen overeenkomt. Indien zo'n bestand niet bestaat, word de lege reeks teruggegeven. Het id kan een pad en wildcards bevatten, bijvoorbeeld van 'C:\temp*.doc. De eigenschappen geven de bijkomende bestanden die je zien wilt. (Zo dat de normale bestanden altijd teruggekeerd worden wanneer zij met het id overeenkomen.) Je kunt de volgende constanten optellen om de soorten bestanden te zien die je wilt:

fa_readonly alleen-lezen bestanden

fa_hidden verborgen bestanden

fa_sysfile systeem bestanden

fa_volumeid volume-id bestanden

fa_directory directories

fa_archive archief bestanden

file_find_next() Geeft de naam van het volgende bestand terug dat met het eerder gegeven id en de eigenschappen overeenkomt. Indien dit bestand niet bestaat, wordt er een lege reeks teruggekeerd.

file_find_close() Moet na het behandelen van alle bestanden geroepen worden om geheugen vrij te maken.
file_attributes(fname,attr) Geeft terug of het bestand alle eigenschappen heeft, gegeven in attr. Gebruik een combinatie van de boven aangeduide constanten.

De volgende functies kunnen gebruikt worden bestandsnamen te veranderen. Merk op dat deze functies aan de bestanden zelf niets veranderen, zij veranderen enkel de tekenreeksen:

filename_name(fname) Geeft de bestandsnaam terug van het opgegeven bestand, met extensie, maar zonder pad.
filename_path(fname) Geeft de url terug van het opgegeven bestand, inclusief de laatste backslash.
filename_dir(fname) Geeft de directory van de opgegeven bestandsnaam terug, dat normaal hetzelfde is als het pad voor het uiteindelijke backslash teken.
filename_drive(fname) Geeft de stations informatie terug.
filename_ext(fname) Geeft de extensie van het bestand terug, inclusief de scheidingspunt.
filename_change_ext(fname,newext) Geeft de opgegeven bestandsnaam terug met de nieuwe extensie inclusief de scheidingspunt.

In zeldzame situaties is het nodig om data te lezen van binaire bestanden. De volgende functies zijn hiervoor beschikbaar:

file_bin_open(fname,mod) Opent een bestand met de opgegeven naam. De mode identificeert wat er gedaan kan worden met het bestand: 0 = lezen, 1 = schrijven, 2 = lezen en schrijven). De functie keert de id van het bestand terug dat gebruikt moet worden in andere functies. Je kan meerde bestanden tegelijk openen (max 32). Vergeet ze niet te sluiten als je ze niet meer gebruikt.
file_bin_rewrite(fileid) Herschrijft het bestand met de opgegeven fileid, wist de inhoud en begint met schrijven aan het begin.
file_bin_close(fileid) Sluit het bestand met de opgegeven fileid.
file_bin_size(fileid) Geeft de grootte van het bestand met het opgegeven fileid terug in bytes.
file_bin_position(fileid) Geeft de huidige positie terug (in bytes; 0 is de eerste positie) van het bestand met de opgegeven fileid.
file_bin_seek(fileid,pos) Verplaatst de huidige positie van het bestand naar de opgegeven plaats. Om een bestand bij te voegen dien je de positie te veranderen naar de grootte van het bestand.

file_bin_write_byte(fileid,byte) Schrijft een byte met data naar het bestand met het opgegeven fileid.
file_bin_read_byte(fileid) Leest een byte van het bestand met het opgegeven fileid.

Indien de speler veilige modus in zijn voorkeuren heeft staan, voor een aantal van deze routines, mag je geen pad specificeren en mogen er alleen bestanden in de toepassingsfolder geschreven worden.

De volgende alleen-lezen functies kunnen handig zijn:

game_id* De unieke vaststeller voor het spel. Je kunt dit gebruiken indien je een unieke bestandsnaam nodig hebt.
working_directory* De werkdirectory. (Zonder de laatste backslash.)
temp_directory* De tijdelijke directory die is aangemaakt voor het spel. Je kan hier tijdelijke bestanden plaatsen. Ze zullen worden verwijderd aan het einde van het spel.

In zekere situaties wil je spelers de mogelijkheid geven om commando lijn argumenten te gebruiken (om bijvoorbeeld cheats of speciale modes te creëren). De volgende functies kun je hiervoor gebruiken.

parameter_count() Geeft het aantal commando lijn parameters terug (begrijp dat de naam van het spel ook een van hen is.)
parameter_string(n) Geeft de commando lijn parameters n terug. De eerste parameter heeft de index 0. Dit is de naam van het spel.

Je kan de waardes van de omgeving van de variabelen met de volgende functie lezen:

environment_get_variable(name) Geeft de waarde (een reeks) van de omgeving variabele met de gegeven naam terug.

Register

Als je een kleine vorm van informatie wilt opslaan tijdens het spelen van het spel is er een simpeler mechanisme dan het gebruik maken van een bestand. Je kan het register gebruiken. Het register is een grote database dat Windows gebruikt om allerlei instellingen op te slaan van programma's. Een sleutel heeft een naam en een waarde. Je kan gebruik maken van tekenreeksen en getallen waardes. De volgende functies zijn beschikbaar:

registry_write_string(name,str) Maakt een sleutel aan in het register met de opgegeven naam en tekenreeks.
registry_write_real(name,x) Maakt een sleutel aan in het register met de opgegeven naam en getal.
registry_read_string(name) Geeft de tekenreeks terug van de opgegeven sleutel. (De sleutel moet aanwezig zijn. Anders word er een lege tekenreeks teruggegeven.)
registry_read_real(name) Geeft de getal waarde terug van de opgegeven sleutel. (De sleutel moet aanwezig zijn. Anders word er een lege getal waarde teruggegeven.)
registry_exists(name) Geeft een waarde terug als de sleutel aanwezig is.

Eigenlijk zijn de waardes in sleutels gegroepeerd. De bovenstaande functies werken allemaal met waardes die in de sleutels staan die voor de bijbehorende applicatie zijn. Je spel kan dit gebruiken om informatie te verkrijgen van het systeem waar het spel op draait. Je kunt ook waardes lezen uit andere sleutels. Ook kun je ze overschrijven, maar wees daar voorzichtig mee. JE KUNT ER MAKKELIJK JE SYSTEEM MEE VERNIETIGEN op deze manier. (Schrijven is niet toegestaan in de veilige mode.) Begrijp dat de sleutels opnieuw zijn geplaatst in groepen. De volgende functies werken in de groep HKEY_CURRENT_USER. Maar je kunt de standaard groep veranderen. Dus bijvoorbeeld, als je uit wilt vinden wat de huidige tijdelijke directory is, gebruik:

```
path = registry_read_string_ext('\Environment','TEMP');
```

De volgende functies zijn aanwezig.

registry_write_string_ext(key,name,str) Schrijft een waarde in de opgegeven sleutel in het register met de opgegeven naam en tekenreeks waarde.
registry_write_real_ext(key,name,x) Schrijft een waarde in de opgegeven sleutel in het register met de opgegeven naam en getal waarde.
registry_read_string_ext(key,name) Geeft de tekenreeks terug van de sleutel met de opgegeven naam. (De naam moet aanwezig zijn anders wordt er een lege tekenreeks teruggekeerd)
registry_read_real_ext(key,name) Geeft de getal waarde terug van de sleutel met de opgegeven naam. (De naam moet aanwezig zijn anders wordt er een lege getal waarde teruggekeerd)
registry_exists_ext(key,name) Geeft een waarde terug als de opgegeven sleutel met de opgegeven naam aanwezig is.

registry_set_root(root) Veranderd de standaard sleutel voor meer mogelijkheden. Gebruik de volgende waardes:

0 = HKEY_CURRENT_USER

1 = HKEY_LOCAL_MACHINE

2 = HKEY_CLASSES_ROOT

3 = HKEY_USERS

INI bestanden

Een goede manier om instellingen op te slaan in een bestand is het gebruik van INI bestanden. INI bestanden bevatten onderdelen en elk onderdeel bevat een aantal naam-waarde paren. Hier een voorbeeld van een INI bestand:

```
[Form]
Top=100
Left=100
Caption=Het beste spel ooit
[Game]
MaxScore=12324
```

Het bovenstaande INI bestand bevat twee onderdelen. Het eerste onderdeel bevat 3 paren en de eerste 2 daarvan bevatten een getal waarde en de derde een tekenreeks waarde. De meeste INI bestanden zijn gemakkelijk aan te maken en te wijzigen. De volgende functies in *Game Maker* kan je gebruiken om data te lezen en te wijzigen van INI bestanden:

ini_open(name) Opent het INI bestand met de opgegeven naam. Het INI bestand moet in dezelfde directory staat als het spel!

ini_close() Sluit het huidige INI bestand.

ini_read_string(section,key,default) Leest de tekenreeks waarde van de opgegeven sleutel. Wanneer de sleutel of het onderdeel niet aanwezig is word de default waarde teruggekeerd.

ini_read_real(section,key,default) Leest de getal waarde van de opgegeven sleutel. Wanneer de sleutel of het onderdeel niet aanwezig is word de default waarde teruggekeerd.

ini_write_string(section,key,value) Schrijft de waarde als een tekenreeks waarde in de opgegeven sleutel die in het opgegeven onderdeel staat.

ini_write_real(section,key,value) Schrijft de waarde als een getal waarde in de opgegeven sleutel die in het opgegeven onderdeel staat.

ini_key_exists(section,key) Geeft een waarde terug als het opgegeven sleutel aanwezig is in het onderdeel.
ini_section_exists(section) Geeft een waarde terug als het opgegeven onderdeel aanwezig is.
ini_key_delete(section,key) Verwijdert de opgegeven sleutel uit het opgegeven onderdeel.
ini_section_delete(section) Verwijdert het hele onderdeel dat is opgegeven.

Uitvoeren van externe bestanden

Game Maker heeft ook de mogelijkheid om externe programma's uit te voeren. Er zijn twee functies beschikbaar voor dit: execute_program and execute_shell. De functie execute_program start een programma, mogelijk met wat argumenten. Het kan wachten op het programma tot het klaar is (pauzeert het spel) of verder gaan met het spel. De functie execute_shell open een bestand. Dit kan elk bestandstype zijn waarvoor een programma is ingesteld waarmee het moet worden ingesteld. Bijvoorbeeld een html of word bestand. Of het kan ook een programma zijn. Het kan niet wachten op het voltooien dus het spel zal doorgaan.

execute_program(prog,arg,wait) Voert het programma prog uit met de argumenten arg. wait identificeert of het moet wachten tot het programma is voltooid (true) of dat het spel door moet gaan (false).
execute_shell(prog,arg) Voert het programma of bestand uit in de shell.

Beide functies zullen niet werken als de speler de veilige modus heeft ingesteld in het bestand of programma dat word geopend. Je kan dit controleren door de volgende alleen-lezen variabele te gebruiken:

secure_mode* Wanneer het spel draait in de veilige modus.

Data structuren

Deze functies zijn alleen beschikbaar in de geregistreerde versie van Game Maker.

In spellen moet je vaak informatie opslaan. Je moet bijvoorbeeld lijsten van voorwerpen opslaan die een persoon draagt of je wilt plaatsen die nog moeten worden bezocht opslaan. Je kunt de series voor dit gebruiken. Maar als je ingewikkeldere verrichtingen wilt doen, zoals het sorteren van de gegevens of het zoeken naar een bepaald voorwerp, moet je grote stukken van GML code schrijven die langzaam kunnen zijn om uit te voeren.

Om dit te verhelpen, heeft Game Maker een aantal ingebouwde gegevensstructuren die door functies kunnen worden betreden. Op het ogenblik zijn er zes verschillende types van beschikbare gegevensstructuur: de stapels (stacks), rijen (queues), lijsten (lists), mappen (maps), prioritaire rijen (priority queues), en rasters (grids). Elk van deze gegevensstructuren is bestemd voor een bepaald type van gebruik (zie verder).

Alle gegevensstructuren werken globaal op dezelfde manier. Je kunt een gegevensstructuur met een functie maken die een id voor de structuren teruggeeft. Je gebruikt deze id om handelingen op de gegevensstructuren uit te voeren. Zodra je dat gedaan hebt vernietigt je opnieuw de gegevensstructuur om opslagruimte te besparen. Je kunt zo veel van deze structuren op hetzelfde ogenblik gebruiken als je wilt. Alle structuren kunnen zowel tekenreeks als echte waarden opslaan.

Gelieve op te merken dat de gegevensstructuren en hun inhoud niet worden bewaard met de acties of de functies wanneer je het spel opslaat. Als je gegevensstructuren gebruikt en wilt toestaan voor het opslaan ervan moet je jouw eigen mechanisme voor dit tot stand brengen.

Wanneer er waarden worden vergeleken, bijvoorbeeld bij het zoeken in een map of het sorteren van een lijst (list), moet *Game Maker* besluiten wanneer twee waarden gelijk zijn. Voor tekenreeks en getal waarden is dit duidelijk maar voor echte aantallen, wegens afrondingsfouten, kunnen gelijke aantal gemakkelijk ongelijk worden. Bijvoorbeeld (5/3)*3 zal niet gelijk aan 5 zijn. Om dit te vermijden, wordt een precisie gebruikt. Wanneer het verschil tussen twee aantallen kleiner is dan deze precisie worden zij beschouwd als gelijk. In Game Maker heb je een precisie van 0.0000001 . Je kunt deze precisie veranderen met behulp van de volgende functies:

`ds_set_precision(prec)` Plaatst de precisie die voor vergelijkingen wordt gebruikt.

Deze precisie wordt gebruikt in alle gegevensstructuren maar niet in andere vergelijkingen in GML!

Stapels (stacks)

Een structuur van stapelgegevens is een zogenaamde LIFO (Last-In First-out of laatste in, eerste uit). Je kunt waarden op een stapel duwen en hen opnieuw verwijderen door hen van de stapel te halen. De waarde die het laatst op de stapel werd geduwd is de eerste die er dan van de stapel afgehaald moet worden. De stapels worden vaak gebruikt wanneer er onderbrekingen zijn om te behandelen of bij recursieve functies. De volgende functies bestaan voor stapels:

`ds_stack_create()` Creëert een nieuwe stapel. Deze functie geeft een geheel als een id terug die in alle andere functies moet worden gebruikt om tot deze stapel toegang te verkrijgen. Je kunt veelvoudige stapels creëren.

ds_stack_destroy(id) Vernietigt de stapel met bepaalde id, waardoor het gebruikte geheugen weer vrij komt. Vergeet niet om deze functie te gebruiken wanneer je met de structuur klaar bent.

ds_stack_clear(id) Ontruimt de stapel met bepaalde id die alle gegevens uit het verwijdert maar het niet vernietigt.

ds_stack_size(id) Geeft het aantal waarden terug die in de stapel zijn opgeslagen.

ds_stack_empty(id) Geeft terug of de stapel leeg is. Dit is hetzelfde als testen of het aantal waarden die in de stapel zijn opgeslagen gelijk zijn aan 0.

ds_stack_push(id,val) Duwt de waarde op de stapel.

ds_stack_pop(id) Geeft de waarde op de bovenkant van de stapel terug en verwijdert deze uit de stapel.

ds_stack_top(id) Geeft de waarde op de bovenkant van de stapel terug maar verwijdert deze niet uit de stapel.

Rijen (Queues)

Een rij is enigszins gelijkwaardig aan een stapel maar het werkt op FIFO (First-in, first-out of eerste-in eerste-uit) basis. De waarde die in de rij eerst wordt gezet is ook de eerste dat uit de rij moet worden verwijderd. Het werkt als een rij in een winkel. De persoon die eerste in een rij is wordt eerst gediend. De rijen worden gebruikt om dingen op te slaan die nog moeten worden gedaan maar er zijn verscheidene toepassingen. De volgende functies bestaan (merk op dat de eerste vijf functies voor stapels gelijkwaardig zijn; alle gegevensstructuren hebben deze vijf functies).

ds_queue_create() Creëert een nieuwe rij. De functie geeft een geheel als een id terug die in alle andere functies moet worden gebruikt om tot deze rij toegang te hebben. Je kunt veelvoudige rijen creëren.

ds_queue_destroy(id) Vernietigt de rij met bepaalde id, waardoor het gebruikte geheugen weer vrij komt. Vergeet niet om deze functie te gebruiken wanneer je met de structuur klaar bent.

ds_queue_clear(id) Ontruimt de rij met het bepaalde id, waardoor alle gegevens verwijderd worden uit de rij zonder deze te vernietigen.

ds_queue_size(id) Geeft het aantal opgeslagen waarden in de rij terug.

ds_queue_empty(id) Geeft terug of de rij leeg is. Dit is hetzelfde als testen of de grootte van de rij 0 is.

ds_queue_enqueue(id,val) Maakt een waarde in de rij aan.

ds_queue_dequeue(id) Geeft de waarde die langst in de rij is terug en verwijdert het uit de rij.

ds_queue_head(id) Geeft de waarde aan het hoofd van de rij terug, namelijk de waarde die het langst in de rij is geweest. (Het verwijdert de waarde niet uit de rij.)
ds_queue_tail(id) Geeft de waarde aan de staart van de rij terug, namelijk de waarde die het laatste aan de rij is toegevoegd. (Het verwijdert het niet uit de rij.)

Lijsten (Lists)

Een lijst slaat een verzameling van waarden in een bepaalde volgorde op. Je kunt waarden aan het eind of ergens in het midden van de lijst toevoegen. Je kunt de waarden met behulp van een index opzoeken. Ook kun je de elementen in een oplopende of aflopende volgorde sorteren. De lijsten kunnen in veel opzichten worden gebruikt, bijvoorbeeld om veranderende verzamelingen van waarden op te slaan. Zij worden uitgevoerd door eenvoudige series maar als dit in een samengestelde code wordt gedaan gaat het sneller dan wanner jezelf een serie maakt. De volgende functies zijn beschikbaar:

ds_list_create() Creëert een nieuwe lijst. De functie geeft een geheel als een id terug die in alle andere functies moet worden gebruikt om tot deze lijst toegang te hebben.
ds_list_destroy(id) Vernietigt de lijst met bepaalde id, waardoor het gebruikte geheugen weer vrij komt. Vergeet niet om deze functie te gebruiken wanneer je met de structuur klaar bent.
ds_list_clear(id) Ontruimt de lijst met bepaalde id, die alle gegevens verwijdert uit de lijst maar deze niet vernietigt.
ds_list_size(id) Geeft het aantal waarden terug die in de lijst zijn opgeslagen.
ds_list_empty(id) Geeft terug of de lijst leeg is. Dit is hetzelfde als testen of het aantal waarden in de lijst 0 is.
ds_list_add(id,val) Voegt een waarde aan het eind van de lijst toe.
ds_list_insert(id,pos,val) Voegt een waarde bij positie pos aan de lijst toe. De eerste positie is positie 0, laatste is de grootte minus 1.
ds_list_replace(id,pos,val) Vervangt de waarde bij positie pos in de lijst met de nieuwe waarde.
ds_list_delete(id,pos) Schrapt de waarde bij positie pos in de lijst. (Positie 0 is het eerste element.)
ds_list_find_index(id,val) Vind de positie waar de vermelde waarde is opgeslagen. Als de waarde niet in lijst staat, word -1 teruggegeven.
ds_list_find_value(id,pos) Geeft de waarde die bij de vermelde positie in de lijst staat terug.
ds_list_sort(id,ascend) Sorteert de waarden in de lijst. Wanneer ascend is

true, worden de waarden gerangschikt in oplopende volgorde, anders in aflopende volgorde.

Mappen (Maps)

In heel wat situaties moet je paren opslaan die uit een sleutel en een waarde bestaan. Bijvoorbeeld, een karakter kan een aantal verschillende voorwerpen hebben en voor elk voorwerp heeft het er een bepaald aantal van. In dit geval is het voorwerp de sleutel en het aantal is de waarde. De mappen handhaven dergelijke paren, die door sleutel worden gesorteerd. Je kunt paren aan de map toevoegen en zoeken naar de waarde die aan een bepaalde sleutels is gekoppeld. Omdat de sleutels worden gesorteerd kun je de vorige en volgende sleutels ook vinden. Soms is het ook nuttig om een map te gebruiken om sleutels zonder een overeenkomstige waarde enkel op te slaan. In dat geval kun je een waarde van 0 eenvoudig gebruiken. De volgende functies bestaan:

`ds_map_create()` Creëert een nieuwe kaart. De functie geeft een geheel terug als een id die in alle andere functies moet worden gebruikt om tot deze map toegang te verkrijgen.

`ds_map_destroy(id)` Vernietigt de kaart met bepaalde id, waardoor het gebruikte geheugen vrij komt. Vergeet niet om deze functie te gebruiken wanneer je met de structuur klaar bent.

`ds_map_clear(id)` Ontruimt de kaart met bepaalde id, die alle gegevens verwijdert uit de map maar het niet vernietigt.

`ds_map_size(id)` Geeft het aantal sleutel-waarde paren die in de kaart zijn opgeslagen terug.

`ds_map_empty(id)` Geeft terug of de kaart leeg is. Dit is hetzelfde als het testen of de grootte van de map 0 is.

`ds_map_add(id,key,val)` Voegt een sleutel-waarde paar aan de map toe.

`ds_map_replace(id,key,val)` Vervangt de waarde van de sleutel met een nieuwe waarde.

`ds_map_delete(id,key)` Verwijdert de sleutel en de overeenkomstige waarde van de kaart. (Als er veelvoudige ingangen met dezelfde sleutel zijn, wordt er slechts een verwijderd.)

`ds_map_exists(id,key)` Geeft terug of de sleutel in de kaart bestaat.

`ds_map_find_value(id,key)` Geeft de waarde terug die aan de sleutel is gekoppeld.

`ds_map_find_previous(id,key)` Geeft de grootste sleutel in de kaart terug die kleiner dan de vermelde sleutel is. (Merk op dat de sleutel wordt teruggegeven, niet de waarde. Je kunt de vorige routine gebruiken om de waarde te vinden.)

ds_map_find_next(id,key) Geeft de kleinste sleutel in de kaart terug die groter is dan de vermelde sleutel.

ds_map_find_first(id) Geeft de kleinste sleutel in de kaart terug.

ds_map_find_last(id) Geeft de grootste sleutel in de kaart terug.

Prioritaire rijen (Priority queues)

In een prioritaire rij worden een aantal waarden opgeslagen, elk met een prioriteit. Je kunt de waarden met minimum en maximumprioriteit snel vinden. Met behulp van deze gegevensstructuur kan je bepaalde dingen in de orde van prioriteit behandelen. De volgende functies bestaan:

ds_priority_create() Creëert een nieuwe prioritaire rij. De functie geeft een geheel als een id terug die in alle andere functies moet worden gebruikt om tot deze prioritaire rij toegang te hebben.

ds_priority_destroy(id) Vernietigt de prioritaire rij met bepaalde id, waardoor het gebruikte geheugen weer vrij komt. Vergeet niet om deze functie te gebruiken wanneer je met de structuur klaar bent.

ds_priority_clear(id) Ontruimt de prioritaire rij met bepaalde id, die alle gegevens verwijdert uit de rij maar deze vernietigt.

ds_priority_size(id) Geeft het aantal waarden terug die in de prioritaire rij zijn opgeslagen.

ds_priority_empty(id) Geeft terug of de prioritaire rij leeg is. Dit is het zelfde als testen of het aantal in de prioritaire rij 0 is.

ds_priority_add(id,val,prio) Voegt een waarde met de bepaalde prioriteit aan de prioritaire rij toe.

ds_priority_change_priority(id,val,prio) Verandert de prioriteit van een bepaalde waarde in de prioritaire rij.

ds_priority_find_priority(id,val) Geeft de prioriteit van de bepaalde waarde in de prioritaire rij terug.

ds_priority_delete_value(id,val) Verwijdert de bepaalde waarde (met zijn prioriteit) van de prioritaire rij.

ds_priority_delete_min(id) Geeft de waarde met de kleinste prioriteit terug en verwijdert deze van de prioritaire rij.

ds_priority_find_min(id) Geeft de waarde met de kleinste prioriteit terug maar verwijdert deze niet van de prioritaire rij.

ds_priority_delete_max(id) Geeft de waarde met de grootste prioriteit terug en verwijdert deze van de prioritaire rij.

ds_priority_find_max(id) Geeft de waarde met de grootste prioriteit terug. maar schrapt deze niet van de prioritaire rij.

Rasters (grids)

Een raster is eenvoudigweg een tweedimensionale serie. Een raster heeft een breedte en hoogte geheel. De structuur staat je toe om de waarde van cellen in het raster te plaatsen en terug te ontvangen door de index van deze te geven (die met 0 in zowel de x als de y richting begint. Maar je kunt de waarde ook in gebieden plaatsen, toevoegen en de som, maximum, min, en gemiddelde waarde over een gebied ontvangen. De structuur is nuttig om bijv. een speelgebied te vertegenwoordigen. Alhoewel de functionaliteit ook kan worden bereikt met behulp van tweedimensionale series, de verrichtingen op gebieden zijn sneller. De volgende functies bestaan:

`ds_grid_create(w,h)` Creëert een nieuw raster met de vermelde breedte en de hoogte. De functie geeft een geheel als een id terug die in alle andere functies moet worden gebruikt om tot dit raster toegang te hebben.

`ds_grid_destroy(id)` Vernietigt het raster met bepaalde id, waardoor het gebruikte geheugen vrij komt. Vergeet niet om deze functie te gebruiken wanneer je met de structuur klaar bent.

`ds_grid_resize(id,w,h)` Stelt opnieuw de breedte en hoogte van het raster in. De bestaande cellen houden hun originele waarde.

`ds_grid_width(id)` Geeft de breedte van het raster met vermelde id terug.

`ds_grid_height(id)` Geeft de hoogte van het raster met vermelde id terug.

`ds_grid_clear(id,val)` Ontruimt het raster met bepaalde id, aan de vermelde waarde (kan zowel een aantal als een tekenreeks zijn).

`ds_grid_set(id,x,y,val)` Plaatst de vermelde cel in het raster met bepaalde id, aan de vermelde waarde (kan zowel een aantal als een tekenreeks zijn).

`ds_grid_add(id,x,y,val)` Voegt de waarde aan de vermelde cel in het raster met een bepaalde id toe. Voor tekenreeksen beantwoordt dit als een aaneenschakeling.

`ds_grid_multiply(id,x,y,val)` Vermenigvuldigt de waarde van de vermelde cel in het raster met bepaalde id. Is slechts geldig voor aantallen.

`ds_grid_set_region(id,x1,y1,x2,y2,val)` Plaatst alle cellen in het gebied in het raster met bepaalde id, aan de vermelde waarde (kan zowel een aantal als een tekenreeks zijn).

`ds_grid_add_region(id,x1,y1,x2,y2,val)` Voegt de waarde aan de cel in het gebied in het raster met bepaalde id toe. Voor tekenreeksen beantwoordt dit aan een aaneenschakeling.

`ds_grid_multiply_region(id,x1,y1,x2,y2,val)` Vermenigvuldigt de waarde van de cellen in het gebied in het raster met bepaalde id. Is slechts geldig voor aantallen.

`ds_grid_set_disk(id,xm,ym,r,val)` Stelt alle cellen in de schijf met centrum

(xm,ym) en straal r.

ds_grid_add_disk(id,xm,ym,r,val) Voegt een waarde aan alle cellen in de schijf met centrum (xm,ym) en straal r toe.

ds_grid_multiply_disk(id,xm,ym,r,val) Vermenigvuldigt de waarde van alle cellen in de schijf met centrum (xm,ym) en de straal r.

ds_grid_get(id,x,y) Geeft de waarde van de vermelde cel in het raster met bepaalde id terug.

ds_grid_get_sum(id,x1,y1,x2,y2) Geeft de som van de waarden van de cellen in het gebied in het raster met bepaalde id terug. Werkt slechts wanneer de cellen aantallen bevatten.

ds_grid_get_max(id,x1,y1,x2,y2) Geeft het maximum van de waarden van de cellen in het gebied in het raster met bepaalde id terug. Werkt slechts wanneer de cellen aantallen bevatten.

ds_grid_get_min(id,x1,y1,x2,y2) Geeft het minimum van de waarden van de cellen in het gebied in het raster met bepaalde id terug. Werkt slechts wanneer de cellen aantallen bevatten.

ds_grid_get_mean(id,x1,y1,x2,y2) Geeft het gemiddelde van de waarden van de cellen in het gebied in het raster met bepaalde id terug. Werkt slechts wanneer de cellen aantallen bevatten.

ds_grid_get_disk_sum(id,xm,ym,r) Geeft de som van de waarden van de cellen in de schijf.

ds_grid_get_disk_min(id,xm,ym,r) Geeft het minimum van de waarden van de cellen in de schijf terug.

ds_grid_get_disk_max(id,xm,ym,r) Geeft het maximum van de waarden van de cellen in de schijf terug.

ds_grid_get_disk_mean(id,xm,ym,r) Geeft het gemiddelde van de waarden van de cellen van de schijf terug.

ds_grid_value_exists(id,x1,y1,x2,y2,val) Geeft terug of de waarde ergens in het gebied bestaat.

ds_grid_value_x(id,x1,y1,x2,y2,val) Geeft de x-coördinaat van de cel terug waarvan de waarde in het gebied bestaat.

ds_grid_value_y(id,x1,y1,x2,y2,val) Geeft de y-coördinaat van de cel terug waarvan de waarde in het gebied bestaat.

ds_grid_value_disk_exists(id,xm,ym,r,val) Geeft terug of de waarde ergens in de schijf bestaat.

ds_grid_value_disk_x(id,xm,ym,r,val) Geeft de x-coördinaat van de cel terug waarvan de waarde in de schijf bestaat.

ds_grid_value_disk_y(id,xm,ym,r,val) Geeft de y-coördinaat van de cel terug waarvan de waarde in de schijf bestaat.

Particles maken

Deze functies zijn alleen beschikbaar in de geregistreerde versie van Game Maker.

Particles zijn bedoeld om speciale effecten te maken. Particles zijn kleine elementen, vertegenwoordigd door een kleine sprite. Zulke particles bewegen rond overeenkomstig met voorgedefinieerde regels en kunnen terwijl ze bewegen van formaat, richting, kleur, etc. veranderen. Veel van zulke particles bij elkaar kunnen bijvoorbeeld vuurwerk, vlammen, explosies, regen, sneeuw, sterren, rondvliegend puin etc. vormen.

Game Maker bevat een uitgebreid particle systeem dat gebruikt kan worden om geweldige effecten te maken. Door zijn algemeenheid is het niet gemakkelijk in gebruik, je kunt dus beter eerst dit hoofdstuk zorgvuldig doornemen voordat je het probeert.

Als het te moeilijk voor je is, is er ook een heel simpel mechanisme om verschillende typen explosies, regen en zelfs vuurwerk te maken.

Particle systemen hebben veel parameters en het is niet altijd even gemakkelijk te begrijpen hoe je de effecten moet maken die je wilt hebben. Als eerste zijn er particle types. Een particle type definieert een specifiek type particle. Zulke types hebben veel parameters die de vorm, formaat, kleur en beweging instellen. Particles hoeven slechts één keer gebruikt te worden en kunnen dan overal in het spel gebruikt worden.

Ten tweede zijn er particle systemen. Er kunnen verschillende particle systemen in het spel zijn. Een particle systeem kan particles van verschillende typen in zich hebben. Een particle systeem heeft emiters die particles creëren, of continu of in uitbarstingen. Het kan ook atractors hebben die particles aantrekken. Als laatste, kan het destroyers hebben die particles vernietigingen. Als particles eenmaal zijn gemaakt in een systeem, worden ze automatisch gecontroleerd (geüpdatet en getekend) door het systeem.

Makkelijke Effecten

De gemakkelijkste manier om particles te gebruiken is met het effectmechanisme. Effecten worden gemaakt met behulp van het particle systeem, maar je hoeft je geen zorgen te maken over de details. Je selecteert simpelweg het type effect, de locatie waar het gemaakt moet worden, het formaat en de kleur. Dat is alles.

Er zijn een aantal verschillende soorten effecten:

- ef_explosion
- ef_ring
- ef_ellipse
- ef_firework
- ef_smoke
- ef_smokeup
- ef_star
- ef_spark
- ef_flare
- ef_cloud
- ef_rain
- ef_snow

Sommige wil je slechts één keer maken (zoals een explosie) en andere iedere step (zoals rook of regen). Merk op dat regen en sneeuw altijd worden gemaakt bovenaan de room dus de positie is niet relevant in dit geval.

Ook al lijkt dit misschien weinig, toch kunnen ze gebruikt worden om geweldige effecten te maken. Bijvoorbeeld, door elke stap een kleine puf rode rook te maken onder een bewegend ruimteschip, wordt een spoor van vuur gemaakt. De volgende twee functies bestaan om effecten te maken:

effect_create_below(kind,x,y,size,color) Maakt een effect van het opgegeven soort (zie hierboven) op de aangegeven positie. size geeft het formaat op de volgende manier op: 0 = klein, 1 = medium, 2 = groot. color geeft de te gebruiken kleur. De effecten worden gemaakt onder de objecten, dat wil zeggen, met een depth van 100000.
effect_create_above(kind,x,y,size,color) Hetzelfde als de bovenstaande functie alleen dan boven de objecten, dat wil zeggen: met een depth van -100000.

Indien je alle effecten wilt verwijderen gebruik je de functie:

effect_clear() Verwijderd alle effecten.

Particle types

Een particle type beschrijft de vorm, kleur, beweging, etc. van een specifiek type particle. Je hoeft een particle type slechts één keer te definiëren. Daarna kun je het in elk particle systeem in het

spel gebruiken. Particle types hebben veel parameters die gebruikt kunnen worden om elk aspect van het type te veranderen. Waneer deze op de juiste manier worden ingesteld kun je elk effect creëren dat je wilt. We zullen deze instellingen hieronder doornemen.

Een aantal functies kunnen gebruikt worden om nieuwe particle types te creëren en weer te verwijderen:

part_type_create() Maakt een nieuw particle type en geeft de index van het type terug. Deze index moet in alle onderstaande functies gebruikt worden om de eigenschappen van het type in te stellen. Je zult ze dus vaak opslaan in een globaal variabele.
part_type_destroy(ind) Vernietigt het particle type waarvan de index is opgeslagen in het variabele ind. Gebruik deze functie als je het type niet meer nodig hebt om ruimte te sparen.
part_type_exists(ind) Geeft terug of het aangegeven particle type bestaat.
part_type_clear(ind) Verandert de index van het aangegeven type naar de standaard waarde.

De vorm van een particle

Een particle heeft een vorm. Deze vorm word aangegeven door een sprite. Je kunt elke willekeurige sprite gebruiken voor je particles maar er zijn voor particles 15 ingebouwde sprites. Deze zijn allemaal 64X64 in formaat en hebben alpha waardes zodat ze mooi vermengen met de achtergrond. Ze zijn aangegeven door de volgende constanten:

- `pt_shape_pixel`
- `pt_shape_disk`
- `pt_shape_square`
- `pt_shape_line`
- `pt_shape_star`
- `pt_shape_circle`
- `pt_shape_ring`
- `pt_shape_sphere`
- `pt_shape_flare`
- `pt_shape_spark`
- `pt_shape_explosion`
- `pt_shape_cloud`
- `pt_shape_smoke`
- `pt_shape_snow`

Je stelt de vorm in door de volgende functie:

> **part_type_shape(ind,shape)** Stelt de vorm van het particle in op één van de bovenstaande constanten (standaard is pt_shape_pixel).

Je kunt ook je eigen sprites gebruiken voor het particle. Als de sprite meerdere subimages heeft kun je aangeven wat daarmee moet gebeuren. je kunt een willekeurige gebruiken, de sprite animeren, Starten op het begin van de animatie of op een willekeurige plek, etc. Je gebruikt hiervoor de volgende functie.

> **part_type_sprite(ind,sprite,animat,stretch,random)** Stelt je eigen sprite in voor het particle type. Met animatie stel je in of de sprite wel (1) of niet (0) geanimeerd moet worden. Met stretch (1 of 0) geef je aan of de animatie moet worden uitgestrekt over de levensduur van de particle. En met random (1 or 0) stel je in of een willekeurige sub-sprite moet worden gekozen als start

Als je eenmaal een sprite voor het particle type hebt gekozen (een standaard of je eigen) kun je het formaat ervan instellen. Een formaat van 1 gebruikt het normale formaat van de sprite. Een particle type kan gedefinieerd worden zodat alle particles hetzelfde formaat hebben. Je kunt een reeks van formaten definiëren. Ook kun je aangeven of het formaat zou veranderen over de levensduur van het particle en of er fluctuaties moeten plaatsvinden in formaat, wat zorgt voor een glinsterend effect.

> **part_type_size(ind,size_min,size_max,size_incr,size_wiggle)** Stelt de formaat parameters in voor het particle type. Je specificeert het minimale start formaat, het maximale start formaat, De formaat toename elke step (gebruik een negatief nummer voor verkleining) en de hoeveelheid schommeling. (Het standaard formaat is 1 en veranderd niet.)
> **part_type_scale(ind,xscale,yscale)** Stelt de horizontale en verticale schaal in. deze factor word vermenigvuldigt met het formaat. Dit is in het bijzonder handig waneer je verschillende schalen nodig hebt in de x- en y-richting.

De particles hebben ook een oriëntatie. Ook hier kan de oriëntatie hetzelfde zijn voor alle particles, kan verschillen, of kan veranderen over de levensduur van de particle. De hoeken specificeren de rotaties tegen de klok in, in graden.

> **part_type_orientation(ind,ang_min,ang_max,ang_incr,ang_wiggle,ang_relative)** Stelt de oriëntatie, graden opties in voor het particle type. Je kunt de minimale hoek opgeven, de maximale hoek, de toename elke step en de hoeveelheid schelling in de hoek. (Standaard zijn alle waardes 0.) Je kunt ook

opgeven of de opgegeven hoek relatief (1) tot de huidige richting of absoluut (0). Bijv.: Door alle waardes op 0 te zetten, maar ang_relative op 1, de particle oriëntatie zal precies het pad van de particle volgen.

Kleur en vermenging

Particles zullen een kleur hebben. Er zijn verschillende manieren waarop je de kleur van een particle kunt instellen. De gemakkelijkste manier is één enkele kleur gebruiken. Je kunt ook twee of 3 kleuren instellen waartussen de particle veranderd tijdens zijn levensduur. Bijvoorbeeld, de particle veranderd van wit langzaam naar zwart. Nog een mogelijkheid is dat je instelt dat de kleur van elke particle verschillend moet zijn, gekozen uit een reeks van kleuren. Je kunt óf een reeks van rood, groen en blauw opgeven, óf een reeks van hue, saturion en waarde.

de standaard kleur is wit. Als je een sprite met z'n eigen kleuren gebruikt, is dit wat je normaal wilt en hoef je geen kleur in te stellen.

part_type_color1(ind,color1) Stelt een enkele kleur in die gebruikt moet worden voor de particles.
part_type_color2(ind,color1,color2) Stelt 2 kleuren in waartussen de kleur word berekend.
part_type_color3(ind,color1,color2,color3) hetzelfde maar dan word de kleur berekend tussen de start kleur, halverwege, en het einde.
part_type_color_mix(ind,color1,color2) Met deze functie geef je aan dat de particle een kleur moet krijgen die een willekeurige mix is uit de twee aangegeven kleuren. Deze kleur zal gelijk blijven over de gehele levensduur van de particle.
part_type_color_rgb(ind,rmin,rmax,gmin,gmax,bmin,bmax) Kan gebruikt worden om aan te geven dat de particle een vaste kleur moet hebben maar dan uit de aangegeven reeks. Je specificeert een reeks in rood, groen, en blauw component van de kleur (elk tussen 0 en 255).
part_type_color_hsv(ind,hmin,hmax,smin,smax,vmin,vmax) Kan gebruikt worden om aan te geven dat de particle een vaste kleur moet hebben maar dan uit de aangegeven reeks. Je specificeert een reeks in hue saturation en waarde component van de kleur (elk tussen 0 en 255).

Naast de kleur kun je ook en alpha transparante waarde opgeven. De ingebouwde particle vormen hebben al een alpha transparantie maar je kunt deze instellingen gebruiken om bijvoorbeeld te laten verdwijnen over zijn levensduur.

part_type_alpha1(ind,alpha1) Stelt één enkele alpha transparantie parameter (0-1) in voor het particle type.
part_type_alpha2(ind,alpha1,alpha2) Hetzelfde maar deze keer zijn start en eind waarde gegeven en word de alpha waarde ertussenin berekend.
part_type_alpha3(ind,alpha1,alpha2,alpha3) Deze keer zijn drie waardes ingesteld waartussen de alpha waarde word berekend .

Normaal gesproken worden particles vermengt met de achtergrond op dezelfde manier als sprites. Maar het is ook mogelijk om auditieve vermenging te gebruiken. Dit specifiek geeft een geweldig effect voor explosies.

part_type_blend(ind,additive) Stelt in of auditieve vermenging (1) of normale vermenging (0) moet worden gebruikt voor het particle systeem.

leven en dood

Particles bestaan voor een beperkte hoeveelheid tijd, hun levensduur. Hierna verdwijnen ze. Levensduur word gemeten in steps. Je kunt de levensduur opgeven (of een reeks van levensduren) voor elk particle type. Particles kunnen nieuwe particles maken van verschillende types. Er zijn hiervoor twee manieren. Ze kunnen nieuwe particles maken in elke step of ze kunnen nieuwe particles maken als ze verdwijnen. Ben voorzichtig dat de hoeveelheid particles niet te hoog word.

part_type_life(ind,life_min,life_max) Stelt de levensduur beperkingen in voor het particle type. (standaard zijn beide 100.)
part_type_step(ind,step_number,step_type) Stelt het aan tal en type particles die in elke step gemaakt moeten worden door het ingestelde particle type. Als je een negatief nummer gebruikt, dan word in elke step een particle gemaakt met een kans van -1/step. Bijvoorbeeld bij een waarde van -5 word er iedere 5 steps 1 particle gemaakt.
part_type_death(ind,death_number,death_type) Stelt het nummer en type particle in die gemaakt moet worden als een particle van het aangegeven typ (ind) doodgaat. Ook hier kun je negatieve nummers gebruiken om particles met een bepaalde kans te laten maken. Merk op dat deze particles alleen gemaakt worden op het eind van de levensduur van de particle, niet als hij word vernietigt door een destroyer (zie hieronder).

Particle beweging

particles kunnen alleen bewegen tijdens hun levensduur. Ze kunnen een eerste snelheid krijgen (of reeks snelheden) en richting, en snelheid en richting kunnen veranderen tijdens de levensduur.

Ook zwaartekracht die de particle in een bepaalde richting trekt kan worden opgegeven. de volgende functies bestaan hiervoor:

part_type_speed(ind,speed_min,speed_max,speed_incr,speed_wiggle) Stelt de snelheids eigenschappen van het particle type in. (standaard zijn alle waarden 0.) Je specificeert de minimale en maximale snelheid. Een willekeurige waarde tussen de opgegeven limieten word gekozen als het particle is gemaakt. Je kunt een snelheidsverhoging in elke step instellen, gebruik een negatief nummer om de particle af te remmen (de snelheid zal nooit kleiner als 0 worden). Als laatste kun je schommeling in de snelheid instellen.

part_type_direction(ind,dir_min,dir_max,dir_incr,dir_wiggle) Stelt de richting eigenschappen van het particle in. (Standaard zijn alle waarden 0.) Ook hier kun je een reeks richtingen opgeven (graden tegen de klok in; 0 geeft een beweging naar rechts aan). Bijvoorbeeld, om de particle te laten bewegen in een willekeurige richting kies je 0 en 360 als waarden. je kunt een toename in richting elke step opgeven, en een hoeveelheid schommeling.

part_type_gravity(ind,grav_amount,grav_dir) Stelt de zwaartekracht eigenschappen voor het particle type in. (Standaard is er geen zwaartekracht.) Je geeft de hoeveelheid zwaartekracht op die in elke step erbij gedaan moet worden en de richting van de zwaartekracht. Bijvoorbeeld: gebruik 270 voor een zwaartekracht naar beneden.

Particle systemen

particles bestaan in particle systemen. Dus om particles in je game te gebruiken moet je één of meerdere systemen maken. Er kunnen verschillende systemen zijn (maar gebruik er bij voorkeur niet te veel). Bijvoorbeeld, als je spel een aantal ballen heeft en elke bal moet een staart van particles hebben, dan heeft waarschijnlijk elke bal zijn eigen particle systeem. De makkelijkste manier om systemen te gebruiken is door er een te maken en dan de particles erin te zetten, die je ervoor gespecificeerd hebt. Maar, zoals we hieronder zullen zien, particle systemen kunnen emitters die automatisch particles produceren, attractors die particles aantrekken, en destroyers die particles vernietigen.

Als particles eenmaal aan een systeem zijn toegevoegd worden ze elke step automatisch geüpdatet en getekend. Geen verdere actie is dan nodig. Om het mogelijk te maken dat particles worden getekend, achter, voor, of tussen objecten in, heeft elk particle systeem zijn eigen depth, net zoals instances en tiles.

particle systemen leven oneindig nadat ze gemaakt zijn. Dus zelfs als je naar een andere room gaat of de game restart, de systemen en de particles blijven. Ben er dus zeker van dat je ze vernietigt nadat je ze niet meer nodig hebt.

De volgende basis functies zorgen voor particle systemen:

part_system_create() Maakt een nieuw particle systeem. het geeft de index van het systeem terug. Deze index moet gebruikt worden in alle onderstaande functies om de eigenschappen van het systeem in te stellen.
part_system_destroy(ind) Vernietigt het particle systeem ind. Gebruik deze functie als je het niet meer nodig hebt, het spaart geheugen.
part_system_exists(ind) Geeft terug of het aangegeven particle systeem bestaat.
part_system_clear(ind) Zet particle systeem ind terug naar zijn standaard instellingen, dit verwijderd alle particles, emitters en atractors erin.
part_system_draw_order(ind,oldtonew) Stelt de volgorde in waarin het particle systeem de particles tekent. als oldtonew true is word de oudste particle eerst getekend en ligt de nieuwste erboven (standaard). Anders worden de nieuwe particles eerst getekend. Dit kan een tamelijk ander effect geven.
part_system_depth(ind,depth) Stelt de depth van het particle systeem in. Dit kan gebruikt worden om de particles achter, voor, of tussen de instances te laten verschijnen.
part_system_position(ind,x,y) Stelt de positie in waar het particle systeem word getekend. Dit is normaal gesproken niet nodig maar als je wilt dat de particles een relatieve positie tot een bewegend object hebben, kun je de positie tot dat object instellen.

zoals hierboven aangegeven, word het particle systeem automatisch geüpdatet en getekend. Maar soms is dit niet wat je wilt. om dit te vergemakkelijken, je kunt automatisch updaten en tekenen uitzetten en zelf beslissen waneer je het systeem updatet. Hiervoor kun je de volgende functies gebruiken:

part_system_automatic_update(ind,automatic) Geeft aan of het particle systeem automatisch moet worden geüpdatet (1) of niet (0). standaard is 1.
part_system_automatic_draw(ind,automatic) Geeft aan of het particle systeem automatisch moet worden getekend (1) of niet (0). standaard is 1.
part_system_update(ind) Deze functie updatet alle particles in het systeem en laat de emitters particles maken. Je hoeft dit alleen te gebruiken als updaten niet automatisch gaat. (ook al is het handig om deze functie een aantal keer te

gebruiken om het systeem op gang te krijgen.)

part_system_drawit(ind) Deze functie tekent de particles in het systeem. Je hoeft dit allen te gebruiken als tekenen niet automatisch gaat. Het moet gebruikt worden in het draw event van een object.

De volgende functies behandelen particles in een particles systeem:

part_particles_create(ind,x,y,parttype,aantal) Deze functie maakt `aantal` particles van het aangegeven type op de positie (x,y) in het systeem.

part_particles_create_color(ind,x,y,parttype,color,aantal) deze functie maakt `aantal` particles van het aangegeven type op positie (x,y) in het systeem met de aangegeven kleur. Dit is alleen bruikbaar als het particle type slechts één kleur heeft gedefinieerd (of geen enkele kleur heeft gedefinieerd).

part_particles_clear(ind) Deze functies verwijderd alle particles uit het systeem.

part_particles_count(ind) Deze functie geeft het aantal particles terug die zich in het systeem bevinden.

Emitters

Emitters maken particles. Ze kunnen of continu een stroom van particles uitstoten of ze kunnen een aantal particles uitstoten bij gebruik van de juiste functie. een particle systeem kan een onbepaald aantal emitters hebben. Een emitter heeft de volgende eigenschappen:

- **xmin, xmax, ymin, ymax** Geeft de regio aan waarbinnen de emitter de particles maakt.
- **shape** Geeft de vorm van de regio aan. Het kan de volgende waarden hebben:
  - `ps_shape_rectangle`
  - `ps_shape_ellipse`
  - `ps_shape_diamond`
  - `ps_shape_line`
- **distribution** Geeft de distributie aan gebruikt om particles te maken. Het kan de volgende waarden hebben:
  - `ps_distr_linear` geeft een lineaire distributie aan, overal in de regio is een even grote kans dat particles gemaakt worden.
  - `ps_distr_gaussian` Geeft een Gaussian waarbij meer particles in het midden als aan de rand van de regio gemaakt worden.
- **particle type** Geeft het particle type aan dat gemaakt word

- **number** Geeft het aantal particles aan dat elke step gemaakt word. Als het kleiner als 0 is, dan word in elke step een particle gemaakt met een kans van -1/aantal. Bijvoorbeeld: bij een waarde van -5 word elke 5 steps 1 particle gemaakt.

De volgende functies zijn beschikbaar om emitters in te stellen en particles te laten maken. Merk op dat elk van deze functies de index van het particle systeem dat gebruikt word nodig heeft als eerste argument.

part_emitter_create(ps) Maakt een nieuwe emitter in het particle systeem. Het geeft de index van de emitter terug. De index van de emitter moet gebruik worden in alle volgende functies om de eigenschappen van de emitter in te stellen.
part_emitter_destroy(ps,ind) Vernietigd de emitter ind in het particle systeem. Gebruik deze functie als je de emitter niet meer nodig hebt om ruimte te besparen.
part_emitter_destroy_all(ps) Vernietigd alle emitters die in het particle systeem gemaakt zijn.
part_emitter_exists(ps,ind) Geeft terug of de aangegeven emitter in het particle systeem bestaat.
part_emitter_clear(ps,ind) Zet de aangegeven emitter terug naar de standaard instellingen.
part_emitter_region(ps,ind,xmin,xmax,ymin,ymax,shape,distribution) Stelt de regio en de distributie in voor de emitter.
part_emitter_burst(ps,ind,parttype,aantal) stoot een keer `aantal` particles van het aangegeven type uit de emitter .
part_emitter_stream(ps,ind,parttype,aantal) vanaf het moment dat deze functie gebruikt word elke step `aantal` particles van het aangegeven type uitgestoten uit de emitter. Als je een nummer kleiner als 0 gebruikt word elke step met een kans van -1/aantal gemaakt. Bijvoorbeeld: met een waarde van -5 word elke 5 steps 1 nieuwe particle gemaakt.

Attractors

Naast emitters kan een particle systeem ook attractors bevatten. Een attractor trekt particles aan (of stoot ze af). Een particle systeem kan meerdere attractors bevatten. Je word wel aangeraden het aantal attractors laag te houden omdat dit het proces van berekenen van particles afremt. Een attractor heeft de volgende eigenschappen:

- **x,y** geeft de positie van de attractor aan.

- **force** geeft de kracht van de attractor aan. Hoe de kracht werkt op de particles ligt aan de volgende parameters.
- **dist** geeft aan vanaf welke afstand de attractor effect heeft op particles. Alleen particles dichter bij de attractor dan deze afstand zullen worden aangetrokken.
- **kind** geeft het type attractor aan. de volgende waarden bestaan
    - `ps_force_constant` geeft aan dat de kracht onafhankelijk van de afstand is.
    - `ps_force_linear` geeft aan dat er een lineaire groei in de kracht is. Op de maximale afstand is de kracht 0 terwijl op de positie van de attractor de kracht de opgegeven waarde bezit.
    - `ps_force_quadratic` geeft aan dat de kracht kwadratisch groeit.
- **additive** geeft aan of de kracht bij de snelheid en richting worde geteld in elke step (true) of alleen word toegepast op de huidige positie van de particle (false). met additieve aan accelereert richting de attractor terwijl met auditieve kracht uit beweegt de particle richting de attractor met een constante snelheid.

De volgende functies definiëren attractors. merk op dat elke functie de index van het particle systeem nodig heeft als eerste argument om te werken .

part_attractor_create(ps) maakt een nieuwe attractor in het gegeven particle systeem. het geeft de index van de attractor terug. Deze index is in elk van de onderstaande functies nodig om de eigenschappen van de emmiter in te stellen.
part_attractor_destroy(ps,ind) Vernietigt de attractor ind in het particle systeem. Gebruik dit als je de attractor niet meer nodig hebt om ruimte te sparen.
part_attractor_destroy_all(ps) vernietigd alle attractors in het systeem die gemaakt zijn.
part_attractor_exists(ps,ind) Geeft terug of de aangegeven attractor bestaat in het aangegeven particle systeem.
part_attractor_clear(ps,ind) Zet de aangegeven atractor terug naar de beginwaarden.
part_attractor_position(ps,ind,x,y) Stelt de positie van de attractor ind in (x,y).
part_attractor_force(ps,ind,force,dist,kind,aditive) Stelt de kracht instellingen van de aangegeven attractor in.

Destroyers

Destroyers vernietigen particles waneer die zich in de regio van de destroyer bevinden. Een particle systeem kan een onbepaald aantal destroyers bevatten. Een destroyers heeft de volgende eigenschappen:

- **xmin, xmax, ymin, ymax** geeft de regio van de destroyer aan waarbinnen de particles worden vernietigd.
- **shape** geeft de vorm van de regio aan. Het kan de volgende waarden hebben:
    - `ps_shape_rectangle`
    - `ps_shape_ellipse`
    - `ps_shape_diamond`

De volgende functies zijn beschikbaar om de eigenschappen van de destroyers in te stellen. merk op dat elke van deze functies de index van het particle systeem waarbinnen de destroyer werkt nodig heeft.

part_destroyer_create(ps) maakt een nieuwe destroyer in het aangegeven particle systeem. Het geeft de index van deze destroyer terug. Deze index is nodig in alle onderstaande functies om de eigenschappen van de destroyer in te stellen.
part_destroyer_destroy(ps,ind) Vernietigd de destroyer ind in het particle systeem. gebruik dit als je de destroyer niet meer nodig hebt om ruimte te besparen.
part_destroyer_destroy_all(ps) Vernietigd alle destroyers in het aangegeven particle systeem.
part_destroyer_exists(ps,ind) Geeft terug of de aangegeven destroyer bestaat in het particle systeem.
part_destroyer_clear(ps,ind) Zet de destroyer terug naar de standaard instellingen.
part_destroyer_region(ps,ind,xmin,xmax,ymin,ymax,shape) Stelt de regio in voor de destroyer.

Deflectors

Deflectors buigen particles af waneer ze in hun regio komen. Merk op dat er alleen rekening word gehouden met de positie van de particles, niet met kleur of formaat. Een particle systeem kan een onbepaald aantal deflectors bevatten. Een deflector heeft de volgende eigenschappen:

- **xmin, xmax, ymin, ymax** geeft de regio van de deflectors aan waarbinnen particles worden afgebogen.
- **kind** geeft de soort deflector aan. Het kan de volgende waarden hebben:
    - `ps_deflect_horizontal` buigt de particles horizontaal af; word normaal gesproken gebruikt voor verticale muren
    - `ps_deflect_vertical` buigt de particles verticaal af; word normaal gesproken gebruikt voor horizontale muren

- **friction** de hoeveelheid frictie als gevolg van inslag met de deflector. Hoe hoger deze frictie, hoe meer de particle word afgeremd.

De volgende functies zijn beschikbaar om de eigenschappen van de deflectors in te stellen. Merk op dat elke van deze functies de index van het particle systeem waarbinnen de deflector werkt.

part_deflector_create(ps) Maakt een nieuwe deflector binnen het opgegeven particle systeem. Het geeft de index van de deflector terug. Deze index moet in elke van de onderstaande functies gebruikt worden om de eigenschappen van de deflector in te stellen.
part_deflector_destroy(ps,ind) Vernietigd destroyer ind in het particle systeem ps. gebruik dit als je de deflector niet meer nodig hebt om ruimte te besparen.
part_deflector_destroy_all(ps) Vernietigd alle deflectors die gemaakt zijn in particle systeem ps.
part_deflector_exists(ps,ind) Geeft terug of de deflector ind bestaat in particle systeem ps.
part_deflector_clear(ps,ind) Zet de aangegeven deflector terug naar de beginstand.
part_deflector_region(ps,ind,xmin,xmax,ymin,ymax) Stelt de regio in voor de deflector.
part_deflector_kind(ps,ind,kind) Stelt de soort in voor de deflector.
part_deflector_friction(ps,ind,friction) Stelt de frictie in voor de deflector.

Changers

Changers veranderen sommige particle types waneer ze in hun regio komen. Een particle systeem kan een onbepaald aantal changers bevatten. Een changer heeft de volgende eigenschappen:

- **xmin, xmax, ymin, ymax** Geeft de regio aan waarin de changer de particles veranderd.
- **shape** Geeft de vorm van de regio aan. Het kan de volgende waarden bevatten:
  - ps_shape_rectangle
  - ps_shape_ellipse
  - ps_shape_diamond
- **parttype1** Geeft het particle type aan dat veranderd moet worden.
- **parttype2** Geeft het particle type aan waarin de ander veranderd moet worden.
- **kind** Geeft het type changer aan. het kan de volgende waarden bevatten:

- `ps_change_motion` Veranderd alleen de beweging parameter van het particle, niet de kleur, vorm of levensduur instellingen
- `ps_change_shape` veranderd alleen de vorm parameters zoals formaat, kleur en vorm
- `ps_change_all` veranderd alle parameters, dit betekend eigenlijk dat de particle word vernietigd en een nieuwe van een ander type word gemaakt.

De volgende functies zijn beschikbaar om de eigenschappen van de changer te veranderen. Merk op dat elke van deze functies de index van het particle systeem waarbinnen de changer werkt nodig heeft.

`part_changer_create(ps)` Maakt een nieuwe changer binnen het aangegeven particle systeem. Het geeft de index van de changer terug. Deze index moet in alle onderstaande functies worden gebruikt om de eigenschappen van de changer te veranderen.
`part_changer_destroy(ps,ind)` Vernietigt changer ind in het aangegeven particle systeem. Gebruik deze functie als je het niet meer nodig hebt om ruimte te besparen.
`part_changer_destroy_all(ps)` Vernietigt alle changers in het aangegeven particle systeem die zijn gemaakt.
`part_changer_exists(ps,ind)` Geeft terug of de aangegeven changer bestaat binnen het aangegeven particle systeem.
`part_changer_clear(ps,ind)` Zet de changer ind terug naar de standaard instellingen.
`part_changer_region(ps,ind,xmin,xmax,ymin,ymax,shape)` Stelt de regio van de changer in.
`part_changer_types(ps,ind,parttype1,parttype2)` Stelt in welk particle type de changer moet veranderen in welk ander type.
`part_changer_kind(ps,ind,kind)` Stelt de soort van de changer in.

Vuurwerk Voorbeeld

Hier is een voorbeeld van een particle systeem dat vuurwerk maakt. het vuurwerk gebruikt twee particle types: een dat de raket vormt en een die het eigenlijke vuurwerk vormt. De raket maakt de vuurwerk particles als het doodgaat. We maken ook een emitter die regelmatig raket particles maakt op de bodem van het scherm. Om dit te laten werken hebben we een object nodig. In het create event zetten we de volgende code die de particle types maakt, het particle systeem en de emitter:

```
{
  // make the particle system
  ps = part_system_create();

  // the firework particles
  pt1 = part_type_create();
  part_type_shape(pt1,pt_shape_flare);
  part_type_size(pt1,0.1,0.2,0,0);
  part_type_speed(pt1,0.5,4,0,0);
  part_type_direction(pt1,0,360,0,0);
  part_type_color1(pt1,c_red);
  part_type_alpha2(pt1,1,0.4);
  part_type_life(pt1,20,30);
  part_type_gravity(pt1,0.2,270);

  // the rocket
  pt2 = part_type_create();
  part_type_shape(pt2,pt_shape_sphere);
  part_type_size(pt2,0.2,0.2,0,0);
  part_type_speed(pt2,10,14,0,0);
  part_type_direction(pt2,80,100,0,0);
  part_type_color2(pt2,c_white,c_gray);
  part_type_life(pt2,30,60);
  part_type_gravity(pt2,0.2,270);
  part_type_death(pt2,150,pt1);    // maak het vuurwerk als het
doodgaat

  // create the emitter
  em = part_emitter_create(ps);

part_emitter_region(ps,em,100,540,480,490,ps_shape_rectangle,ps
_distr_linear);
  part_emitter_stream(ps,em,pt2,-4);   // maak één elke vier
steps
}
```

Dat moet werken. Misschien wil je zeker weten dat het particle systeem (en misschien de particle types) worden vernietigd als je naar een andere room gaat, anders gaat het vuurwerk voor altijd door.

Multiplayer spellen

Deze functies zijn alleen beschikbaar in de geregistreerde versie van Game Maker.

Spellen tegen de computer spelen is leuk. Maar een spel tegen andere mensen spelen kan nog leuker zijn. Het is relatief makkelijk om deze spellen te maken omdat je geen moeilijke kunstmatige intelligentie hoeft te ontwerpen. Natuurlijk kun je met 2 verschillende spelers voor hetzelfde scherm zitten en verschillende spelbesturingen gebruiken, maar het is leuker als elke speler met zijn eigen computer kan spelen. Of nog beter, een speler zit aan de andere kant van de oceaan. *Game Maker* heeft multiplayer ondersteuning. Realiseer alsjeblieft dat het maken van effectieve multiplayer spellen die synchroon lopen en geen hoge ping hebben een moeilijke taak is. Dit hoofdstuk geeft een korte omschrijving van de mogelijkheden. Op de website is een tutorial beschikbaar met meer informatie.

Een verbinding maken

Om 2 computers te laten communiceren hebben ze een protocol nodig. Zoals de meeste spellen, biedt *Game Maker* 4 verschillende types van verbindingen: IPX, TCP/IP, Modem, and Serial. De IPX verbinding (om meer precies te zijn, het is een protocol) is bijna volledig transparant. Het kan gebruikt worden om spellen te spelen met mensen op hetzelfde LAN netwerk te spelen. Het moet op je computer zijn geïnstalleerd om gebruikt te worden. (Als het niet werkt, kijk dan in de handleiding van Windows. Of ga naar het netwerk pictogram in het configuratiescherm van Windows en voeg het protocol toe.) TCP/IP is het internet protocol. Het kan gebruikt worden om games te spelen met andere spelers op het gehele internet, ervan uitgaande dat je het IP-adres van de speler weet. Op een LAN netwerk kun je het gebruiken zonder een IP-adres in te vullen. Een modem verbinding word gemaakt door een modem. Je moet een paar instellingen invullen (een initialisatie regel en een telefoonnummer) om het te gebruiken. Als laatste, als je een serial verbinding gebruikt (een directe verbinding tussen de computers) moet je een aantal poortinstellingen opgeven. Er zijn vier GML functies die gebruikt kunnen worden om deze verbindingen te initialiseren:

`mplay_init_ipx()` Initialiseert de IPX verbinding.

`mplay_init_tcpip(addr)` Initialiseerd de TCP/IP verbinding. addr is een regel die het internet adres of IP-adres bevat bijv.: 'www.gameplay.com' of '123.123.123.12', mogelijk gevolgd door een poortnummer (bijv.:'12'). Alleen als

je een sessie 'joint' (erbij komt) moet je dit adres invullen. Op een LAN netwerk hoef je geen adressen in te vullen.

mplay_init_modem(initstr,phonenr) Initialiseert de modem verbinding. initstr is de initialisatie regel voor het modem (kan leeg zijn). phonenr is een regel die het nummer dat gebeld moet worden bevat (bijv. '0201234567'). Alleen als je een sessie 'joint' (zie beneden) moet je een telefoonnummer invoeren.

mplay_init_serial(portno,baudrate,stopbits,parity,flow) Initialiseert een seriële verbinding. portno is het poortnummer (1-4). baudrate is het bautrate dat gebruikt moet worden (100-256K). stopbits bepaalt het aantal stopbits dat gebruikt moet worden (0 = 1 bit, 1 = 1.5 bit, 2 = 2 bits). parity bepaalt even of oneven (0=geen, 1=oneven, 2=even, 3=mark). En flow stelt het type flow control in (0=none, 1=xon/xoff, 2=rts, 3=dtr, 4=rts and dtr). Geeft terug of het succesvol was. Een kenmerkende invulling is: mplay_init_serial(1,57600,0,0,4). Schrijf 0 als eerste argument om een dialoog met de speler te openen om deze instellingen te veranderen.

Je spel moet één van deze functies één keer gebruiken. Al deze functies geven terug of ze succesvol zijn uitgevoerd. Ze zijn onsuccesvol als het gebruikte protocol niet geïnstalleerd is, of niet ondersteund wordt door de computer. Om te controleren of er een succesvolle verbinding beschikbaar is kun je de volgende functie gebruiken.

mplay_connect_status() Geeft de status van de huidige verbinding weer. 0 = geen verbinding, 1 = IPX verbinding, 2 = TCP/IP verbinding, 3 = modem verbinding, and 4 = seriële verbinding.

Om de huidige verbinding te verbreken gebruik je deze functie.

mplay_end() Beëindigt de huidige verbinding.

Als je een TCP/IP verbinding gebruikt wil je het persoon waarmee je het spel wilt spelen waarschijnlijk je IP-adres vertellen. De volgende functie kan je hierbij helpen:

mplay_ipaddress() Geeft het IP-adres van je computer terug (bijv. '123.123.123.12') als een regel. Je kunt dit bijv. laten zien ergens op het scherm. Merk op dat dit een trage functie is dus gebruik het niet iedere stap.

Opzetten en joinen van sessies

Als je verbinding hebt met een netwerk, kunnen daar meerdere spellenbezig zijn. Deze spellen noemen we sessies. Deze verschillende sessies kunnen van hetzelfde spel of van verschillende

spellen zijn. Een spelmoet zichzelf uniek kunnen identificeren. Gelukkig doet *Game Maker* dit voor je. Het enige dat je moet weten is dat als je de game-id in het opties scherm veranderd, ook de identificatie word veranderd. Op deze manier kun je voorkomen dat mensen met oude versies tegen mensen met nieuwe versies gaan spelen.

Als je een nieuw multiplayer spel start, moet je een sessie maken. Hiervoor gebruik je de volgende functie:

mplay_session_create(sesname,playnumb,playername) Maakt een nieuwe sessie op de huidige verbinding. sesname is een regel die de naam van de sessie bevat. playnumb is het maximale aantal spelers (gebruik 0 voor onbeperkt). playername is je naam als speler. Geeft terug of het succesvol was.

Eén speler moet een spel opzetten. De andere speler(s) van het spelmoeten bij deze sessie komen (joinen). Dit is iets gecompliceerder. Je moet eerst kijken welke sessies er beschikbaar zijn, en er dan één kiezen om te joinen. Er zijn hiervoor drie belangrijke functies:

mplay_session_find() Zoekt alle sessies die nog steeds spelers accepteren en geeft het aantal sessies terug.
mplay_session_name(numb) Geeft de naam terug van het sessie nummer opgeslagen in 'numb' (0 is de eerste sessie). Deze functie kan alleen gebruikt worden nadat de vorige functie is gebruikt.
mplay_session_join(numb,playername) Laat je de sessie 'numb' joinen (0 is de eerste sessie). playername is je naam als speler. Geeft terug of het succesvol was.

Er is nog één functie die de sessie modus kan veranderen. Deze moet gebruikt worden voordat een sessie gemaakt wordt:

mplay_session_mode(move) Stelt in of de sessie naar een andere 'host' moet worden verplaatst als de eerste 'host' stopt. move moet true of false zijn (false is normaal).

Om de status van je huidige sessie te checken kun je de volgende functie gebruiken

mplay_session_status() Geeft de status van de huidige sessie terug. 0 = geen sessie, 1 = sessie gemaakt, 2 = sessie gejoint.

Een speler kan stoppen met de huidige sessie met de volgende functie:

mplay_session_end() Beëindigt de sessie voor deze speler.

Spelers

Elk kopie van het spel die joint is een speler. Zoals eerder aangegeven, hebben spelers namen. Er zijn drie functies die spelers kunnen behandelen.

`mplay_player_find()` Zoekt alle spelers in de huidige sessie en geeft het aantal gevonden spelers terug.

`mplay_player_name(numb)` Geeft de naam van een speler terug (0 is de eerste speler, die je altijd zelf bent). Deze functie kan alleen gebruik worden na de vorige functie.

`mplay_player_id(numb)` Geeft het unieke id van een speler terug (0 is de eerste speler, die je altijd zelf bent). Deze functie kan alleen gebruikt worden na de eerste functie. Dit id wordt gebruikt om berichten te zenden en ontvangen tussen individuele spelers.

Data delen

Het delen van data is waarschijnlijk de gemakkelijkste manier om je spel te synchroniseren. Alle communicatie is van je afgeschermd. Er is een set van 1000000 waardes die elke speler in het spel kan herkennen (liever alleen de eerste paar waardes om geheugen te besparen). Elke speler kan waardes schrijven én lezen. *Game Maker* zorgt ervoor dat elke speler de juiste waardes ziet. Een waarde kan een regel of een nummer zijn. Er zijn slechts twee functies:

`mplay_data_write(ind,val)` Schrijft de waarde val (tekenreeks of getal waarde) in nummer ind (ind tussen 0 en 1000000).

`mplay_data_read(ind)` Geeft de waarde terug die zich in ind bevind (ind tussen 0 en 1000000). Bij het starten van het spel zijn alle waardes 0.

Om de data op de verschillende computers te synchroniseren kun je de gegarandeerde modus gebruiken die ervoor zorgt dat alle data gegarandeerd aankomt aan de andere kant (maar langzaam is) of niet gegarandeerd. Om dit te veranderen gebruik je de volgende functie:

`mplay_data_mode(guar)` Stelt in of er wel of geen gegarandeerde verbinding gebruikt moet worden. guar moet true (het standaard) of false zijn.

Berichten

Het tweede communicatie mechanisme dat *Game Maker* ondersteund is het sturen en ontvangen van berichten. Een speler kan berichten sturen naar één of alle andere spelers. Spelers kunnen zien of berichten zijn aangekomen en actie ondernemen aan de hand van dat bericht. Berichten

kunnen worden verstuurd in gegarandeerde modus of in een niet gegarandeerde modus, wat sneller is.

De volgende functies voor berichten bestaan in *Game Maker*:

`mplay_message_send(player,id,val)` Stuurt een bericht naar de aangegeven speler (gebruik of de naam of het id van een speler; gebruik 0 om het bericht aan alle spelers te sturen). id is een geheel getal dat het id van het bericht aangeeft en val is de waarde in het bericht (of een getal of een regel). Het bericht word verzonden in een niet gegarandeerde modus. Als val een regel bevat is het maximale aantal karakters 30000.

`mplay_message_send_guaranteed(player,id,val)` Stuurt een bericht naar de aangegeven speler (gebruik of de naam of het id van een speler; gebruik 0 om het bericht aan alle spelers te sturen). id is een geheel getal dat het id van het bericht aangeeft en val is de waarde in het bericht (of een getal of een regel). Het bericht word verzonden in een gegarandeerde modus. Als val een regel bevat is het maximale aantal karakters 30000..

`mplay_message_receive(player)` Ontvangt het volgende bericht uit de wacht rij van de aangegeven speler (gebruik de naam of het id van een speler). Gebruik 0 voor berichten van iedere speler te ontvangen. De functie geeft terug of er inderdaad een nieuw bericht was. Als dat het geval is kun je de volgende functies gebruiken om de inhoud van de berichten te krijgen:

`mplay_message_id()` Geeft het id van het laatst ontvangen bericht terug.

`mplay_message_value()` Geeft de waarde van het laatst ontvangen bericht terug.

`mplay_message_player()` Geeft de speler terug die het laatst ontvangen bericht stuurde.

`mplay_message_name()` Geeft de naam van de speler terug die het laatst ontvangen bericht stuurde.

`mplay_message_count(player)` Geeft het aantal berichten terug dat nog steeds in de wacht rij staat van een bepaalde speler (gebruik 0 om alle berichten te tellen).

`mplay_message_clear(player)` Verwijdert alle berichten die nog steeds in de wacht rij van een bepaalde speler staan (gebruik 0 om alle berichten te verwijderen).

Een paar opmerkingen zijn hier op zijn plaats. Ten eerste, als je een bericht wilt verzenden naar één bepaalde speler, moet je zijn unieke id weten. Zoals eerder aangegeven kun je dit te weten komen met de functie: `mplay_player_id()`. Deze speler id word ook gebruikt om berichten van

één bepaalde speler te ontvangen. Een alternatieve manier om dit te doen is de naam van de speler geven als tekenreeks. Als meerdere spelers dezelfde naam hebben krijgt alleen de eerste het bericht.

Ten tweede vraag je je misschien af waarom ieder bericht een eigen id heeft. De reden hiervoor is dat het je spel helpt om verschillende typen berichten te sturen. De ontvanger kan het type bericht checken en passende actie ondernemen. (Omdat berichten niet gegarandeerd aankomen zou het sturen van id en waarde in verschillende berichten serieuze problemen opleveren.)

DLL's gebruiken

Deze functies zijn alleen beschikbaar in de geregistreerde versie van Game Maker.

In de gevallen waar de mogelijkheden van GML niet genoeg zijn voor jouw wensen, kun je de mogelijkheden uitbreiden door plug-ins te gebruiken. Een plug-in in de vorm van een DLL bestand (een Dynamic Link Library). In zo'n DLL bestand kun je functies definiëren. Zulke functies kunnen worden geprogrammeerd in elke programmeertaal die het maken van DLL's ondersteunt (bijv. Delphi, C, C++, enz.) Je moet wel een goede programmeur zijn om dit te kunnen doen. Plug-in functies moeten een bepaald formaat hebben. Ze kunnen tussen de 0 en 11 argumenten hebben, elk kan een getal (double in C) of een tekenreeks. (Bij meer dan 4 argumenten, worden op het moment alleen getallen ondersteund.) Ze moeten ook een getal of een tekenreeks teruggeven.

In Delphi kun je een DLL maken door eerst **New** in het menu **File** te kiezen en dan DLL te selecteren. Hier is een voorbeeld van een DLL die je kunt gebruiken in *Game Maker*, geschreven in Delphi. (Merk op dat dit Delphi code is, geen GML code!)

```
library MyDLL;

uses SysUtils, Classes;

function MyMin(x,y:double):double;  cdecl;
begin
  if x<y then Result := x else Result := y;
end;

var res : array[0..1024] of char;

function DoubleString(str:PChar):PChar; cdecl;
```

```
begin
  StrCopy(res,str);
  StrCat(res,str);
  Result := res;
end;

exports MyMin, DoubleString;

begin
end.
```

Deze DLL definieert twee functies: `MyMin` die twee getalargumenten neemt en het minimum van de twee teruggeeft, en `DoubleString` die de tekenreeks verdubbelt. Merk op dat je voorzichtig moet zijn met het beheren van geheugen. Daarom heb ik de resulterende tekenreeks globaal gedeclareerd. Let ook op het gebruik van de cdecl calling convention. Je kunt kiezen uit de cdecl en de stdcall calling convention. Als je eenmaal de DLL in Delphi hebt gebouwd zul je een bestand `MyDLL.DLL` krijgen. Dit bestand moet worden geplaatst in de werkmap van je spel. (Of elke andere plaats waar Windows het kan vinden.)

Om deze DLL in *Game Maker* te gebruiken moet je eerst de externe functies specificeren die je wilt gebruiken en welke typen argumenten ze gebruiken. Hiervoor is de volgende functie in GML:

> **external_define(dll,naam,calltype,restype,argaant,arg1type,arg2type, ...)** Definieert een externe functie. dll is de naam van het dll bestand. naam is de naam van de functies. calltype is de gebruikte calling convention. Gebruik hiervoor dll_cdecl of dll_stdcall. restype is het type resultaat. Gebruik hiervoor ty_real of ty_string. argaant is het aantal argumenten (0-11). Vervolgens moet je voor elk argument het type specificeren. Gebruik hiervoor opnieuw ty_real of ty_string. Als er meer dan 4 argumenten zijn moeten ze allemaal van het type ty_real zijn.

Deze functie geeft het id van de externe functie, die moet worden gebruikt om het op te roepen. Dus in bovenstaande voorbeeld, zou je aan het begin van het spel de volgende GML code willen gebruiken:

```
{
  global.mmm = external_define('MyDLL.DLL','MyMin',dll_cdecl,
```

```
ty_real,2,ty_real,ty_real);
  global.ddd =
external_define('MyDLL.DLL','DoubleString',dll_cdecl,
                                    ty_string,1,ty_string);
}
```

Nu gebruik je voor elke situatie waarin je de functies moet oproepen de volgende functie:

> **external_call(id,arg1,arg2,...)** Roept de externe functie op met het gegeven id, en de gegeven argumenten. Je moet het correcte aantal argumenten van het correcte type (real of string) geven. De functie geeft het resultaat van de externe functie.

Dus zou je bijvoorbeeld schrijven:

```
{
  aaa = external_call(global.mmm,x,y);
  sss = external_call(global.ddd,'Hallo');
}
```

Als je de DLL niet meer hoeft te gebruiken, kun je hem beter uit het werkgeheugen halen.

> **external_free(dll)** Haalt de DLL met de gegeven naam uit het werkgeheugen. Dit is onder andere noodzakelijk als het spel de DLL zou moeten verwijderen. Zolang de DLL niet uit het werkgeheugen is gehaald kan hij niet worden verwijderd. Je kunt dit bijv. het beste doen in een Game End event.

Je vraagt je misschien af hoe je een functie in een DLL maakt die iets doet in het spel. Bijvoorbeeld, misschien wil je een DLL maken die object instanties toevoegt aan het spel. De eenvoudigste manier is om je DLL functie een tekenreeks terug te laten geven die een stukje GML code bevat. De tekenreeks die het stukje GML bevat kan worden uitgevoerd door de volgende GML functie te gebruiken

> **execute_string(tkr)** Voert het stukje code in de tekenreeks `tkr` uit.

Als alternatief kun je je DLL een bestand laten maken met een script dat kan worden uitgevoerd (deze functie kan ook worden gebruikt om later het gedrag van een spel aan te passen).

execute_file(bnaam) Voert het stukje code uit in het bestand.

Nu kun je een externe functie oproepen en dan de resulterende tekenreeks uitvoeren, bijv. als volgt:

```
{
  ccc = external_call(global.ddd,x,y);
  execute_string(ccc);
}
```

In sommige rare situaties moet je DLL misschien de handle van het algemene graphics venster van het spel. Dit kan worden verkregen met de volgende functie en kan dan worden doorgegeven aan de DLL:

window_handle() Geeft het venster handle van het algemene venster.

Merk op dat DLL's niet kunnen worden gebruikt in secure mode.

Het gebruik van externe DLL's is een extreem krachtig mechanisme. Maar gebruik het alsjeblieft alleen als je weet wat je aan het doen bent.

3D Graphics

Deze functies zijn alleen beschikbaar in de geregistreerde versie van Game Maker.

Game Maker is een programma bedoeld om 2-dimensionale en isometrische spellen te maken. Toch zijn er wat mogelijkheden om 3-dimensionale graphics te creëren. Voordat je hiermee begint zijn er een aantal dingen die je moet weten.

- De 3D mogelijkheid in *Game Maker* is beperkt tot de graphics. Er is geen ondersteuning voor andere 3D functionaliteit. Als je eenmaal begint 3D graphics te gebruiken krijg je mogelijk problemen met andere aspecten van *Game Maker*, zoals de views, dieptes, enz. De mogelijkheden zijn beperkt en hebben een lage prioriteit om te worden uitgebreid.
- Als je de 3D mogelijkheid gebruikt, zijn er een aantal andere dingen die niet langer kunnen worden gebruikt.
    - Je kunt geen achtergronden en voorgronden meer in je rooms gebruiken. (De reden is dat ze getiled worden om het plaatje te vullen maar met perspectieven projecties werkt dit niet langer correct).

    - Je kunt de muispositie niet langer gebruiken. De muis zal niet worden getransformeerd naar de 3D coördinaten. Je kunt nog steeds de positie van de muis op het scherm (in de view) krijgen maar je zult er zelf berekeningen mee moeten uitvoeren (of de muis helemaal niet gebruiken).
    - Je kunt geen tiles meer gebruiken. Tiles zullen hoogst waarschijnlijk niet meer correct verschijnen.
    - Collision controle gebruikt nog steeds de 2D posities van de instanties in de room. Dus is er geen collision detectie in 3D. Soms kun je dit nog steeds gebruiken (als je de room als een herpresentatie van een platte wereld gebruikt (bijv. voor race- of FPS spellen), maar in andere situaties moet je zelf dingen doen).
- Alle 3D mogelijkheden zijn door code. Je moet de GML taal behoorlijk vloeiend kennen. Ook moet je echt een hoop begrijpen over hoe *Game Maker* werkt, anders zul je in de problemen raken.
- Je moet wat basiskennis hebben van 3D graphics. In het algemeen zal ik termen als perspectieven projecties, hidden surface removal, lighting, and fog zonder veel uitleg gebruiken.
- Je moet voorzichtig werken om een redelijke snelheid te houden. Ook zijn dingen niet echt geoptimaliseerd voor snelheid.

Als dit je niet ontmoedigd heeft, lees verder.

Naar 3D modus gaan

Als je 3D modus wilt gebruiken moet je eerst *Game Maker* in 3D modus zetten. Later kun je terugschakelen naar 2D modus als je dat wilt. De volgende functies zijn hiervoor beschikbaar.

`d3d_start()` Start het gebruik van 3D modus.
`d3d_end()` Stop het gebruik van 3D modus.

Merk op dat alle functies gerelateerd aan 3D modus beginnen met `d3d_`.

Het starten van 3D modus zal in de volgende veranderingen resulteren. Ten eerste wordt hidden surface removal ingeschakeld (gebruikmakend van een 16-bit z-buffer). Dit betekent dat voor elke pixel op het scherm alleen het tekenen met de kleinste z-waarde (= dieptewaarde) wordt getekend. Als instanties dezelfde diepte hebben is het onduidelijk wat er zal gebeuren en kun je lelijke effecten krijgen. Weet zeker dat instanties die misschien elkaar kunnen overlappen niet dezelfde dieptewaarde hebben!

Ten tweede, de normale orthografische projectie wordt vervangen door een perspectieven. Dit betekent het volgende. Normaal gesproken is de grootte van instanties op het scherm

onafhankelijk van zijn diepte. Met een perspectieven projectie verschijnen instanties met een grotere diepte kleiner. Als de diepte gelijk is aan 0 is het gelijk aan de oude grootte (tenzij je de projectie veranderd; zie hieronder). Het gezichtspunt van de camera wordt op een afstand boven de room geplaatst. (Deze afstand is gelijk aan de breedte van de room; dat geeft een redelijke standaardprojectie.) Alleen instanties voor de camera worden getekend. Dus gebruik geen instanties met een diepte kleiner dan 0 (of tenminste niet kleiner dan -b waar b de breedte van de room of de view is).

Ten derde wordt de verticale y-coördinaat omgekeerd. Terwijl normaal positie (0,0) de linkerbovenhoek van de view is, is in 3D modus positie (0,0) de linker onderhoek, zoals normaal is voor 3-dimensionale views.

Je kunt hidden surface remove en perspectieven projectie in- of uitschakelen door de volgende functies te gebruiken.

`d3d_set_hidden(inschakelen)` Schakelt hidden surface removal in (true) of uit (false).
`d3d_set_perspective(inschakelen)` Schakelt het gebruik van een perspectieven projectie in (true) of uit (false).

Simpel tekenen

Als 3D modus eenmaal is geactiveerd kun je *Game Maker* gebruiken zoals je het gewend bent (opmerkingen aan het begin uitgezonderd). Alleen zullen objecten in andere groottes verschijnen gebaseerd op hun diepte-instelling. Je kunt zelfs views gebruiken. Eén extra functie kan handig zijn. Als je een aantal dingen tekent in een stukje code wil je misschien de dieptewaarde wijzigen tussen de primitieven die je tekent. Hiervoor gebruik je:

`d3d_set_depth(diepte)` Stelt de diepte in die wordt gebruikt bij het tekenen.

Merk op dat op het moment dat een nieuwe instanties wordt getekend, de diepte opnieuw wordt ingesteld op de diepte van die instantie.

Polygons in 3D tekenen

Het probleem met op de oude manier tekenen is dat een sprite of polygon altijd in de xy-plane ligt, dat betekent dat alle hoeken dezelfde diepte hebben. Voor echt 3D wil je vertices op verschillende diepten kunnen hebben. Vanaf dit moment zullen we alleen over z-coördinaten praten in plaats van diepte. Dus willen we coördinaten specificeren in (x,y,z) formaat. Hiervoor zijn er speciale versies van de geavanceerde tekenfuncties:

d3d_primitive_begin(type) Begin een 3D primitive van het aangegeven type: `pr_pointlist`, `pr_linelist`, `pr_linestrip`, `pr_trianglelist`, `pr_trianglestrip` of `pr_trianglefan`.

d3d_vertex(x,y,z) Voeg vertex (x,y,z) toe aan de primitive, gebruikmakend van de vooraf ingestelde kleur en doorzichtigheid.

d3d_vertex_color(x,y,z,klr,doorzichtigheid) Voeg vertex (x,y,z) toe aan de primitive, met zijn eigen kleur en doorzichtigheid. Dit maakt het mogelijk om primitieven te maken met een vloeiend veranderende kleur en doorzichtigheid.

d3d_primitive_end() Beëindig de beschrijving van de primitive. Deze functie tekent hem.

Om bijvoorbeeld een tetrahedron (driezijdige piramide) te tekenen die op de z=0 plane staat met zijn top op z = 200, kun je de volgende code gebruiken:

```
{
  d3d_primitive_begin(pr_trianglelist);
    d3d_vertex(100,100,0);
    d3d_vertex(100,200,0);
    d3d_vertex(150,150,200);
    d3d_vertex(100,200,0);
    d3d_vertex(200,200,0);
    d3d_vertex(150,150,200);
    d3d_vertex(200,200,0);
    d3d_vertex(100,100,0);
    d3d_vertex(150,150,200);
    d3d_vertex(100,100,0);
    d3d_vertex(100,200,0);
    d3d_vertex(200,200,0);
  d3d_primitive_end();
}
```

Als je dit nu wilt gebruiken, zie je hoogst waarschijnlijk alleen een driehoek op het scherm omdat de top van de tetrahedron erachter zal zijn van het gezichtspunt. Ook, slechts één kleur gebruikend, zou het moeilijk zijn om de verschillende oppervlakken te zien. Hieronder zullen we manieren zien om het gezichtspunt te veranderen. Kleuren toewijzen kan worden gedaan zoals hiervoor door de functie `draw_set_color(col)` op te roepen tussen de vertices.

Je kunt ook textured polygons in 3D gebruiken. Het werkt precies hetzelfde als beschreven in de geavanceerde tekenfuncties in de documentatie. Maar deze keer heb je 3D varianten nodig van de basisfuncties. Eén ding moet je je realiseren. In een textuur is positie (0,0) de linkerbovenhoek. Maar wanneer je projecties gebruikt (zoals hieronder aangegeven), is de linker onderhoek vaak (0,0). In zulke situaties moet je mogelijk de textuur verticaal spiegelen.

d3d_primitive_begin_texture(type,texid) Begin een 3D primitive van het aangegeven type met de gegeven textuur.
d3d_vertex_texture(x,y,z,xtex,ytex) Voeg vertex (x,y,z) toe aan de primitive met positie (xtex,ytex) in de textuur, verkleurd met vooraf ingestelde kleur en doorzichtigheid.
d3d_vertex_texture_color(x,y,z,xtex,ytex,klr,doorzichtigheid) Voeg vertex (x,y,z) toe aan de primitive met positie (xtex,ytex) in de textuur, verkleurd met zijn eigen kleur en doorzichtigheid.
d3d_primitive_end() Beëindig de beschrijving van de primitive. Deze functie tekent hem.

Dus kun je bijvoorbeeld de volgende code gebruiken om een achtergrond te tekenen die verdwijnt in de verte.

```
{
  var ttt;
  ttt = background_get_texture(back);
  d3d_primitive_begin_texture(pr_trianglefan,ttt);
    d3d_vertex_texture(0,480,0,0,0);
    d3d_vertex_texture(640,480,0,1,0);
    d3d_vertex_texture(640,480,1000,1,1);
    d3d_vertex_texture(0,480,1000,0,1);
  d3d_primitive_end();
}
```

Een driehoek heeft een voor- en achterkant. De voorkant wordt gedefinieerd als de zijde waar de vertices tegen de klok in worden gedefinieerd. Normaal worden beide zijden getekend. Maar als je een gesloten vorm maakt is dit verspilling omdat de achterkant van de driehoek nooit kan worden gezien. In zulke gevallen kun je backface culling aanzetten. Dit bespaart ongeveer de helft van de tekentijd, maar laat aan jou de taak om je polygons op de goede manier te definiëren. De volgende functie bestaat:

d3d_set_culling(cull) Geeft het starten van backface culling (true) of het stoppen van backface culling (false) aan.

Basisvormen tekenen

Een aantal functies bestaat om basisvormen te tekenen, zoals blokken en muren. Merk op dat deze vormen ook correct werken met backface culling ingeschakeld.

d3d_draw_block(x1,y1,z1,x2,y2,z2,texid,hherhaling,vherhaling) Tekent een blok in de huidige kleur met de aangegeven tegenoverliggende hoeken gebruikmakend van de aangegeven textuur. Gebruik -1 om geen textuur te gebruiken. `hherhaling` geeft aan hoe vaak de textuur moet worden herhaald langs de horizontale rand van elk oppervlak `vherhaling` doet hetzelfde voor de verticale rand.

d3d_draw_cylinder(x1,y1,z1,x2,y2,z2,texid,hherhaling,vherhaling,gesloten,stappen) Tekent een verticale cilinder in de huidige kleur in de aangegeven bounding box gebruikmakend van de aangegeven textuur. Gebruik -1 om geen textuur te gebruiken `hherhaling` geeft aan hoe vaak de textuur moet worden herhaald langs de horizontale rand van elk oppervlak `vherhaling` doet hetzelfde voor de verticale rand. `gesloten` geeft aan of de cilinder een gesloten onder- en bovenkant moet hebben. `stappen` geeft aan hoeveel draaistappen moeten worden genomen. Een typische waarde is 24.

d3d_draw_cone(x1,y1,z1,x2,y2,z2,texid,hherhaling,vherhaling,gesloten,stappen) Tekent een verticale conus in de huidige kleur in de aangegeven bounding box gebruikmakend van de aangegeven textuur. Gebruik -1 om geen textuur te gebruiken `hherhaling` geeft aan hoe vaak de textuur moet worden herhaald langs de horizontale rand van elk oppervlak `vherhaling` doet hetzelfde voor de verticale rand. `gesloten` geeft aan of de conus een gesloten onder- en bovenkant moet hebben. `stappen` geeft aan hoeveel draaistappen moeten worden genomen. Een typische waarde is 24.

d3d_draw_ellipsoid(x1,y1,z1,x2,y2,z2,texid,hherhaling,vherhaling,stappen) Tekent een ellipsoïde in de huidige kleur in de aangegeven bounding box gebruikmakend van de aangegeven textuur. Gebruik -1 om geen textuur te gebruiken `hherhaling` geeft aan hoe vaak de textuur moet worden herhaald langs de horizontale rand van elk oppervlak `vherhaling` doet hetzelfde voor de verticale rand. `stappen` geeft aan hoeveel draaistappen moeten worden genomen. Een typische waarde is 24.

d3d_draw_wall(x1,y1,z1,x2,y2,z2,texid,hherhaling,vherhaling) Tekent een verticale muur in de huidige kleur met de gegeven hoeken

gebruikmakend van de aangegeven textuur. Gebruik -1 om geen textuur te gebruiken. `hherhaling` geeft aan hoe vaak de textuur moet worden herhaald langs de horizontale rand van elk oppervlak `vherhaling` doet hetzelfde voor de verticale rand.

d3d_draw_floor(x1,y1,z1,x2,y2,z2,texid,hherhaling,vherhaling)
Tekent een (schuine) vloer in de huidige kleur met de gegeven hoeken gebruikmakend van de aangegeven textuur. Gebruik -1 om geen textuur te gebruiken. `hherhaling` geeft aan hoe vaak de textuur moet worden herhaald langs de horizontale rand van elk oppervlak `vherhaling` doet hetzelfde voor de verticale rand.

Het volgende stukje code tekent twee blokken:

```
{
  var ttt;
  ttt = background_get_texture(back);
  d3d_draw_block(20,20,20,80,40,200,ttt,1,1);
  d3d_draw_block(200,300,-10,240,340,100,ttt,1,1);
}
```

De wereld bekijken

Standaard kijk je langs de negatieve z-as naar het midden van de room. Vaak wil je in 3D spellen veranderen hoe je naar de wereld kijkt. Bijvoorbeeld, in een first person shooter wil je waarschijnlijk hebben dat de camera van een positie een beetje boven de xy-oppervlakte over de xy-oppervlakte kijkt. In grafische termen stel je de correcte projectie in. Om de manier waarop je kijkt te veranderen bestaan de volgende twee functies.

d3d_set_projection(xvan,yvan,zvan,xnaar,ynaar,znaar,xomhoog,yomhoog,zomhoog) Definieert hoe in de wereld moet worden gekeken. Je specificeert het punt waar vanaf te kijken, het punt waarnaar te kijken en de omhoog vector.

Deze functie vereist enige uitleg. Om de projectie in te stellen heb je eerst de positie nodig waarvandaan je kijkt. Dit wordt aangegeven door de parameters `(xvan,yvan,zvan)`. Daarna moet je de richting aangeven waarin je kijkt. Dit wordt gedaan door een tweede punt te geven om naar te kijken. Dit is het punt `(xnaar,ynaar,znaar)`. Tenslotte kun je nog steeds de camera rondom de lijn van het gezichtspunt naar het kijkpunt draaien. Om dit te specificeren moeten we een omhoog vector, dat is de richting die omhoog is vanuit de camera. Dit is gegeven door de

laatste drie parameters (xomhoog,yomhoog,zomhoog). Laat me een voorbeeld geven. Om langs de xy-oppervlakte te kijken, zoals in een first person shooter, kun je gebruiken

```
{
  d3d_set_projection(100,100,10,200,100,10,0,0,1);
}
```

Dus je kijkt vanuit punt (100,100) en 10 boven de oppervlakte in de richting van (200,100). De omhoog vector punten is de z-richting zoals verplicht. Om dit een beetje gecompliceerder te maken, stel je voor dat je een instantie in je room hebt die de positie van de camera specificeert. Het zal een huidige (x,y) positie hebben en een richting (en misschien zelfs een snelheid). Je kunt nu dit als je camera specificeren door de volgende code te gebruiken:

```
{
  with (obj_camera)
    d3d_set_projection(x,y,10,
                x+cos(direction*pi/180),y-
sin(direction*pi/180),10,
                0,0,1);
}
```

Dit ziet er misschien een beetje ingewikkeld uit. We kijken vanaf de camerapositie (x,y), 10 boven de grond. Om een punt in de correcte richting vast te stellen moeten we een beetje meetkunde doen. Dit punt is aangegeven door de volgende drie parameters. Tenslotte gebruiken we bovenstaande omhoog vector.

Eén belangrijke opmerking! Zodra *Game Maker* een room begint te tekenen zal het gezichtspunt terugzetten naar de standaardpositie. Dus het eerste wat je moet doen bij het tekenen van de scène is de projectie die je wilt instellen. Dit moet worden gedaan in een Draw Event!

Er is ook een uitgebreide versie van de bovenstaande functie:

> **d3d_set_projection_ext(xvan,yvan,zvan,xnaar,ynaar,znaar,xomhoog,yomhoog,zomhoog,gezichtsveld,verhouding,zdichtbij,zverweg)** Een uitgebreide versie van deze functie in welke je ook de breedte van het gezichtsveld, de verhouding tussen de horizontale en verticale zijde van de view, en de dichtbij en ver weg clipping planes.

De toegevoegde parameters werken als volgt. Als je de camerapositie specificeert, het punt om naar te kijken, en de omhoog vector, kun je nog steeds veranderen hoe breed de lens van de camera is. Dit wordt het gezichtsveld genoemd. Een redelijke waarde is 45 graden en dit is ook de standaardwaarde. Maar je kunt dit veranderen als je dat wilt. Vervolgens kun je de verhouding tussen de horizontale en verticale projectie specificeren. Normaal gesproken wil je dezelfde verhouding gebruiken als die van de room of view, bijv. 640/480. Tenslotte kun je de clipping planes aangeven. Objecten die dichterbij de camera zijn dan `zdichtbij` worden niet getekend. Hetzelfde voor objecten verder dan `zverweg`. Het kan belangrijk zijn om deze parameters op een redelijke waarde in te stellen omdat ze ook de precisie van de z-comparisons beïnvloeden. Als je range te groot maakt wordt de precisie slechter. Standaard gebruiken we 1 en 32000. `zdichtbij` moet groter zijn dan 0!

Soms heb je tijdelijk een normale orthografische projectie nodig zoals gebruikt als er geen 3D is. Of je wilt terugkeren naar de standaard perspectieven projectie. Hiervoor kun je de volgende functies gebruiken:

`d3d_set_projection_ortho(x,y,b,h,draaiing)` Stelt een normale orthografische projectie in het aangegeven gebied in de room, gedraaid over het aangegeven aantal graden.
`d3d_set_projection_perspective(x,y,b,h,draaiing)` Stelt een normale perspectieven projectie in het aangegeven gebied in de room, gedraaid over het aangegeven aantal graden.

Een standaardgebruik hiervan is het tekenen van een overlay om bijv. de score of andere aspecten weer te geven. Om dit te doen stellen we een orthografische projectie in. We moeten ook tijdelijk hidden surface removal uitschakelen omdat we willen dat de informatie onafhankelijk van de huidige dieptewaarde wordt getekend. Het volgende voorbeeld geeft weer hoe je een overlay met de score kunt creëren.

```
{
  draw_set_color(c_black);
  d3d_set_projection_ortho(0,0,room_width,room_height,0);
  d3d_set_hidden(false);
  draw_text(10,10,'Score: ' + string(score));
  d3d_set_hidden(true);
}
```

Transformaties

Transformatie maakt het mogelijk om de plaats waar dingen in de wereld worden getekend te veranderen. Bijvoorbeeld, de functie om blokken te tekenen kan alleen blokken parallel aan de assen tekenen. Door eerst een draaitransformatie in te stellen kun je gedraaide blokken creëren. Ook worden sprites altijd parallel aan de xy-oppervlakte getekend. Door eerst een transformatie in te stellen kun je dit veranderen. Er zijn twee typen functies: functies die een transformatie instellen en functies die transformaties toevoegen.

`d3d_transform_set_identity()` Stelt de transformatie in op de identiteit (geen transformatie).

`d3d_transform_set_translation(xo,yo,zo)` Stelt de transformatie in op een overgang over de aangegeven vector.

`d3d_transform_set_scaling(xv,yv,zv)` Stelt de transformatie in op een vergroting of verkleining met de aangegeven hoeveelheden.

`d3d_transform_set_rotation_x(richting)` Stelt de transformatie in op een draaiing over de x-as met de aangegeven hoeveelheid.

`d3d_transform_set_rotation_y(richting)` Stelt de transformatie in op een draaiing over de y-as met de aangegeven hoeveelheid.

`d3d_transform_set_rotation_z(richting)` Stelt de transformatie in op een draaiing over de z-as met de aangegeven hoeveelheid.

`d3d_transform_set_rotation_axis(xa,ya,za,richting)` Stelt de transformatie in op een draaiing over de as aangegeven door de vector met de aangegeven hoeveelheid.

`d3d_transform_add_translation(xo,yo,zo)` Voegt een overgang toe over de aangegeven vector.

`d3d_transform_add_scaling(xv,yv,zv)` Voegt een vergroting of verkleining toe met de aangegeven hoeveelheden.

`d3d_transform_add_rotation_x(richting)` Voegt een draaiing over de x-as toe met de aangegeven hoeveelheid.

`d3d_transform_add_rotation_y(richting)` Voegt een draaiing over de y-as toe met de aangegeven hoeveelheid.

`d3d_transform_add_rotation_z(richting)` Voegt een draaiing over de z-as toe met de aangegeven hoeveelheid.

`d3d_transform_add_rotation_axis(xa,ya,za,richting)` Voegt een draaiing toe over de as aangegeven door de vector met de aangegeven hoeveelheid.

Realiseer je dat draaiing, vergroting en verkleining relatief zijn aan de origin van de wereld, niet relatief aan het object dat wordt getekend. Als het object niet op de origin is zal het ook verplaatsen naar een andere plaats, wat we niet willen. Dus om bijv. een object te draaien over

zijn eigen x-as, moeten we eerst de origin overzetten, het dan draaien, en tenslotte terugzetten naar zijn eigen positie. Dit is waarvoor de functies om transformaties toe te voegen zijn.

Het volgende voorbeeld legt dit misschien beter uit. Ga er van uit dat we een sprite `spr` hebben die we op positie (100,100,10) willen tekenen. We kunnen de volgende code gebruiken om dit te doen

```
{
  d3d_transform_set_translation(100,100,10);
  draw_sprite(spr,0,0,0);
  d3d_transform_set_identity();
}
```

Merk op dat omdat we een transformatie gebruiken we de sprite nu op positie (0,0) moeten tekenen. (Dit gaat er van uit dat de huidige instantie een diepte van 0 heeft! Als je het niet zeker weet, eerst de diepte instellen.) Als we dit zouden gebruiken in onze first person shooter zouden we de sprite niet zien. De reden is dat het nog steeds parallel aan de xy-oppervlakte is. We willen het 90 graden draaien langs de x-as (of y-as). Dus moeten we een draaiing toevoegen. Onthoud de volgorde: we moeten eerst de sprite draaien en dan overzetten. Dus kunnen we de volgende code gebruiken.

```
{
  d3d_transform_set_identity();
  d3d_transform_add_rotation_x(90);
  d3d_transform_add_translation(100,100,10);
  draw_sprite(spr,0,0,0);
  d3d_transform_set_identity();
}
```

Soms wil je tijdelijk de huidige transformatie opslaan, bijvoorbeeld om een extra transformatie toe te voegen en dan de oude herstellen (dit gebeurt vaak als men hierarchische modellen tekent). Hiervoor kun je de huidige transformatie op een stack duwen en later hem er van af trekken om het opnieuw de huidige transformatie te maken. De volgende functies bestaan hiervoor:

d3d_transform_stack_clear() Wist de stack van transformaties.

d3d_transform_stack_empty() Geeft aan of de transformatie stack leeg is.

d3d_transform_stack_push() Duwt de huidige transformatie op de stack.
Geeft aan of er ruimte op de stack was om het daarop te duwen (als je vergeet om

de transformatie te trekken zul je op een gegeven moment geen ruimte op de stack meer over hebben).

d3d_transform_stack_pop() Trekt de bovenste transformatie van de stack en maakt het de huidige. Geeft aan of er een transformatie op de stack was.

d3d_transform_stack_top() Maakt de bovenste transformatie de huidige, maar verwijdert hem niet van de stack. Geeft aan of er een transformatie op de stack was.

d3d_transform_stack_discard() Verwijdert de bovenste transformatie van de stack maar maakt het niet de huidige. Geeft aan of er een transformatie op de stack was.

Het gebruik van transformaties is een krachtig mechanisme. Maar wees voorzichtig en zet de transformatie altijd terug op de identiteit zodra je ermee klaar bent.

Fog

Fog kan in 3D spellen worden gebruikt om objecten in de verte vaag eruit te laten zien of zelfs te laten verdwijnen. Dit helpt bij het creëren van atmosfeer en het maakt het mogelijk om objecten die ver weg zijn niet te tekenen. Gebruik de volgende functie om fog in- of uit te schakelen:

d3d_set_fog(inschakelen,kleur,begin,eind) Schakelt het gebruik van fog in of uit. `kleur` geeft de kleur van de fog aan. `begin` geeft de afstand aan waarop de fog moet beginnen. `eind` geeft de afstand aan waarop de fog maximaal is en niets meer kan worden gezien.

Om beter te begrijpen wat er gebeurt, zijn er twee typen fog, table based fog en vertex based fog. Het eerste type berekent fogwaarden op een pixelbasis. Het tweede type berekent de fogwaarde voor elke vertex en interpoleert deze dan. Het eerste type is beter maar niet altijd ondersteund. *Game Maker* probeert table based fog te gebruiken als dit ondersteund wordt en anders gebruikt het vertex based fog (tenzij geen fog wordt ondersteund). Merk op dat sommige grafische kaarten aangeven dat ze met table based fog kunnen werken maar bieden de gebruiker de mogelijkheid om dit uit te schakelen in de geavanceerde weergave instellingen. In deze situatie is het resultaat mogelijk een zwart scherm!

Belichting

Scènes die je wilt tekenen met bovenstaande functies zien er nogal plat uit omdat er geen licht is. De kleur van de oppervlakken is gelijk, onafhankelijk van hun oriëntatie. Om scènes er realistischer te maken moet je belichting inschakelen en lichten op de correcte plaatsen plaatsen. Goede lichtscènes maken is niet eenvoudig maar het effect is erg goed.

Om belichting in te schakelen kun je de volgende functie gebruiken;

d3d_set_lighting(inschakelen) Schakelt het gebruik van belichting in (true) of uit (false).

Wanneer je belichting gebruikt wordt voor elke vertex van een polygon de kleur vastgesteld. Vervolgens wordt de kleur van interne pixels gebaseerd op de kleur van deze vertices. Er zijn twee manieren waarop dit kan worden gedaan: of de hele polygon krijgt dezelfde kleur, of de kleur is vloeiend geïnterpoleerd over de polygon. Standaard wordt smooth shading gebruikt. Dit kan worden veranderd gebruikmakend van de volgende functie:

d3d_set_shading(smooth) Schakelt het gebruik van smooth shading in (true) of uit (false).

Om belichting te gebruiken moet je natuurlijk lichten definiëren. Twee verschillende lichten bestaan: directionele lichten (zoals de zon), en positionele lichten. Lichten hebben een kleur. (We ondersteunen alleen diffuus licht, geen spectaculaire reflectie.) De volgende functies bestaan om lichten te definiëren en te gebruiken:

d3d_light_define_direction(ind,rx,ry,rz,klr) Definieert een directioneel licht. `ind` is de index van het licht (gebruik een klein positief getal). (rx,ry,rz) is de richting van het licht. `klr` is de kleur van het licht (vaak wil je `c_white` gebruiken). Deze functie zet het licht niet aan.
d3d_light_define_point(ind,x,y,z,bereik,klr) Definiëert een positioneel licht. `ind` is de index van het licht (gebruik een klein positief getal). (x,y,z) is de positie van het licht. `bereik` geeft aan tot hoever het licht schijnt. De intensiteit van het licht zal over dit bereik afnemen. `klr` is de kleur van het licht. Deze functie zet het licht niet aan.
d3d_light_enable(ind,aan) Zet licht nummer `ind` aan (true) of uit (false).

De manier waarop een object het licht reflecteert hangt af van de rotatie tussen de lichtrichting en de normaal (loodlijn) van de oppervlakte, dat is de vector die loodrecht op de oppervlakte staat. Om belichte objecten te maken moet je niet alleen de positie van de vertices verschaffen, maar ook hun loodlijnen. Hiervoor zijn vier extra functies beschikbaar om de vertices van primitieven te definiëren:

d3d_vertex_normal(x,y,z,nx,ny,nz) Voeg vertex (x,y,z) toe aan de primitive, met normaal vector (nx,ny,nz).
d3d_vertex_normal_color(x,y,z,nx,ny,nz,klr,doorzichtigheid) Voeg vertex (x,y,z) toe aan de primitive, met normaal vector (nx,ny,nz), en met zijn

eigen kleur en doorzichtigheid.

d3d_vertex_normal_texture(x,y,z,nx,ny,nz,xtex,ytex) Voeg vertex (x,y,z) toe aan de primitive, met normaal vector (nx,ny,nz), en met positie (xtex,ytex) in de texture, verkleurd met vooraf ingestelde kleur en doorzichtigheid.

d3d_vertex_normal_texture_color(x,y,z,nx,ny,nz,xtex,ytex,klr,doorzichtigheid) Voeg vertex (x,y,z) toe aan de primitive, met normaal vector (nx,ny,nz), en met positie (xtex,ytex) in de texture, verkleurd met zijn eigen kleur en doorzichtigheid.

Merk op dat voor basisvormen die je kunt tekenen de loodlijnen automatisch goed worden ingesteld.

Modellen creëren

Als je grote modellen moet tekenen is het nogal duur om alle verschillende tekenfuncties opnieuw en opnieuw op te roepen in elke step. Om dit te voorkomen kun je modellen creëren. Een model bestaat uit een aantal getekende primitieven en vormen. Zodra een model is gecreëerd kun je het op verschillende plaatsen tekenen met slechts één oproep. Modellen kunnen ook worden geladen van een bestand en opgeslagen worden in een bestand.

Voordat ik de verschillende beschikbare functies geef is er één belangrijk punt: het handelen met texturen. Als eerder beschreven worden texturen genomen van sprites en achtergronden. De indexen van de texturen kunnen verschillen op verschillende momenten. Met gevolg dat modellen geen textuurinformatie bevatten. Alleen als je een model tekent geef je een texture op. Dus kun je slechts één texture in een model gebruiken. Als je meerdere texturen moet gebruiken moet je ze of erin combineren (en handel voorzichtig met textuurcoördinaten), of moet je meerdere modellen gebruiken. Het voordeel hiervan is dat je eenvoudig hetzelfde model met verschillende texturen kunt tekenen.

Om modellen te creëren, te laden, op te slaan en te tekenen bestaan de volgende functies:

d3d_model_create() Creëert een nieuw model en geeft zijn index. Deze index wordt in alle andere functies die met modellen te maken hebben gebruikt.

d3d_model_destroy(ind) Vernietigt het model met de gegeven index en maakt zijn geheugen vrij.

d3d_model_clear(ind) Wist het model met de gegeven index, d.w.z. verwijdert al zijn primitieven.

d3d_model_save(ind,bnaam) Slaat het model op met de gegeven

bestandsnaam.

d3d_model_load(ind,bnaam) Laadt het model van de gegeven bestandsnaam.

d3d_model_draw(ind,x,y,z,texid) Tekent het model op positie (x,y,z). `texid` is het texture dat moet worden gebruikt. Gebruik -1 als je geen texture wilt gebruiken. Als je het model wilt draaien, vergroten of verkleinen kun je de eerder beschreven transformatieroutines gebruiken.

Voor elke primitive functie is een gelijke die hem toevoegt aan een model. De functies hebben dezelfde argumenten als eerder beschreven uitgezonderd dat elk een eerste argument heeft, de index van het model, en dat er geen textuurinformatie wordt gegeven.

d3d_model_primitive_begin(ind,type) Voeg een 3D primitive aan het model toe van het aangegeven type: `pr_pointlist`, `pr_linelist`, `pr_linestrip`, `pr_trianglelist`, `pr_trianglestrip` of `pr_trianglefan`.

d3d_model_vertex(ind,x,y,z) Voeg vertex (x,y,z) toe aan het model.

d3d_model_vertex_color(ind,x,y,z,klr,doorzichtigheid) Voeg vertex (x,y,z) toe aan het model, met zijn eigen kleur en doorzichtigheid.

d3d_model_vertex_texture(ind,x,y,z,xtex,ytex) Voeg vertex (x,y,z) toe aan het model met positie (xtex,ytex) in de texture.

d3d_model_vertex_texture_color(ind,x,y,z,xtex,ytex,klr,doorzichtigheid) Voeg vertex (x,y,z) toe aan het model met textuur- en kleurwaarden.

d3d_model_vertex_normal(ind,x,y,z,nx,ny,nz) Voeg vertex (x,y,z) toe aan het model, met normaal vector (nx,ny,nz).

d3d_model_vertex_normal_color(ind,x,y,z,nx,ny,nz,col,alpha) Voeg vertex (x,y,z) toe aan het model, met normaal vector (nx,ny,nz), en met zijn eigen kleur en doorzichtigheid.

d3d_model_vertex_normal_texture(ind,x,y,z,nx,ny,nz,xtex,ytex) Voeg vertex (x,y,z) toe aan het model, met normaal vector (nx,ny,nz), met texture positie.

d3d_model_vertex_normal_texture_color(ind,x,y,z,nx,ny,nz,xtex,ytex,klr,doorzichtigheid) Voeg vertex (x,y,z) toe aan het model, met normaal vector (nx,ny,nz), met textuur- en kleurwaarden.

d3d_model_primitive_end(ind) Beëindig de beschrijving van de primitive in het model.

Behalve primitieven kun je ook basisvormen aan de modellen toevoegen. Opnieuw zien de functies er bijna hetzelfde uit, maar met een model index en zonder textuurinformatie:

d3d_model_block(ind,x1,y1,z1,x2,y2,z2,hherhaling,vherhaling)
Voegt een blokvorm toe aan het model.
d3d_model_cylinder(ind,x1,y1,z1,x2,y2,z2,hherhaling,vherhaling,gesloten,stappen) Voegt een cilindervorm toe aan het model.
d3d_model_cone(ind,x1,y1,z1,x2,y2,z2,hherhaling,vherhaling,gesloten,stappen) Voegt een conusvorm toe aan het model.
d3d_model_ellipsoid(ind,x1,y1,z1,x2,y2,z2,hherhaling,vherhaling,stappen) Voegt een ellipse vorm toe aan het model.
d3d_model_wall(ind,x1,y1,z1,x2,y2,z2,hherhaling,vherhaling)
Voegt een muurvorm toe aan het model.
d3d_model_floor(ind,x1,y1,z1,x2,y2,z2,hherhaling,vherhaling)
Voegt een vloervorm toe aan het model.

Het gebruik van modellen kan de snelheid van de graphics in je 3D spellen bespreekbaar verhogen en je zou ze zoveel mogelijk moeten gebruiken.

Slotwoord

De 3D functies in *Game Maker* kunnen worden gebruikt om leuke 3D spellen te maken. Ze zijn echter gelimiteerd in functionaliteit en laten nog steeds behoorlijk veel werk aan jou over. Verwacht niet dat je er je eigen *Quake* mee kunt maken. *Game Maker* is en blijft in de eerste plaats een pakket om 2-dimensionale spellen mee te maken.