

Game Maker

version 1.4 (June 1, 2000)

by

Mark Overmars

Introduction

Writing computer games normally is a lot of work and requires skill in programming. So wouldn't it be nice if you could make your own games in an easy way? **Game Maker** was written to do just that. It allows you to make games like Pacman by just using some simply clicking and drag-and-drop with your mouse. It is suited for grown-ups and children from age 9 upwards (with a little help from a parent). But even though it is simple to use, **Game Maker** allows you to make appealing games, with animated graphics, backgrounds, sounds, etc.

Game Maker can be used free of charge. It requires a reasonable powerful computer (Pentium with 16 Mb of memory minimum; preferably Pentium 166Mhz and 24 Mb of memory or more) running Windows 95 or later. It requires at least 65000 colors (high color, 16-bits) and a screen resolution of 800x600 or more.

Game Maker comes with a collection of freeware images and sounds to get you started. These are not part of the game but were taken from public domain collections. Also there are a number of example games, in particular Pacman, Breakout and a Peg game. These games are mainly provided as examples and not as full-blown games, although they are actually quite a bit of fun to play. On the web-site

<http://www.cs.uu.nl/~markov/kids/gmaker/index.html>

more extensive games are provided. Send you own creations to markov+games@cs.uu.nl and I might add them to the site. (See below on how to do this.)

The global idea

Games created with **Game Maker** take place in one or more rooms. (Rooms are flat, not 3D (sorry).) In these rooms there are various objects. Some objects belong to the background and don't do anything, some form walls, or other static things, and others are moving around and/or act and react.

So the first thing to do is to make some objects. Below you find more information on how to do this but let me give a global description here. Objects first of all have an image such that you can see them. (You can give objects an empty image if you don't want to see them.) Objects also have a name for easy reference. Note that you can place multiple instances of the same object in a room. So if you have e.g. three monsters you need only to define one object (unless you want them to have a different image or different behavior). Objects can be solid, that means that they cannot occupy the same place, or not. Also, they can be active, that is, walk around and react to each other, or passive. For example, background objects will neither be solid, nor active. Other objects can walk over them and nothing happens. Walls are solid but not active, so other objects cannot run into them. Figures that move around in your game are active and might be solid or not, depending on their use.

Active objects can perform actions. There are different moments, called events, when actions are required. The most important ones are: creation, collision, and meeting events. When an instance of an object is created (either because it was placed in the initial room, or when it is created during the game) these actions are performed. Such actions for example put the object in motion. When an active object collides with a solid object, a collision event happens, and the object should take appropriate action (e.g. reverse direction

or stop, and make a sound). When an active object meets another object, a meeting event happens. You should take action, based on the object you meet. For example, if you meet a monster you might kill yourself, and if you meet a bonus object you might add something to the score (the bonus object should in this case probably destroy itself to avoid that you keep on walking over it). Such a bonus object would typically be a non-solid active object. There are also other events: each instance of an object has an alarm clock that can generate events, and there are also keyboard and mouse events to which objects can react.

As indicated, objects can perform actions in case of events. There are many different actions possible. Objects can start or stop moving in a direction, change their speed or position, kill themselves or other objects, change into something else, create new objects, or play sound files. There is actually a complete programming language incorporated in **Game Maker** in which you can fully program the actions. But for many games you only need the standard actions and there is no need to write any line of code.

After you created the objects and specified the required actions, it is time to define the rooms. Simple game will have just one room. More complicated games can have multiple rooms and there are actions to move from one room to another. Defining rooms is easy. You specify some properties like size and color, and then you place the objects in it.

Now you are ready to run the game. Objects will come to life because of their creation actions and start reacting with each other. The user can control reactions using keyboard or mouse events.

A simple example

Let us look at a very simple example. We want to make a game in which an object jumps around on the screen. The player should try to catch it by pressing the mouse on it. The game is provided under the name **Catch the Dog**. Best open it and play it to understand what I mean. Now let us look at how it was created. Press the button **Create Objects**. You will see that there is just one object in the list: the dog. Click on it with the mouse. Suddenly a lot of information pops up. At the bottom left you see the name of the object (dog) and the image. Furthermore it is indicated that it is solid and active. At the right you see the events and after some of them some blocks that indicate actions. Only three events contain actions: the creation event, the alarm event, and the mouse event. The create event contains two actions. The first one moves the dog to a random position. The second one sets the alarm clock to 10 ticks (1 second). The alarm event does exactly the same things. The result is that every second the dog moves to a random position. Finally let's look at the mouse event. This is executed when the player manages to press the mouse on the object. There are four actions here. The first action adds one to the score. The second action plays a little sound. And the other two actions again move the dog to a random position and set the alarm clock.

Close the window and press the button **Create Rooms**. You see that there is just one boring room with the dog inside it. Just note that at the left it is indicated that the speed is 10 (so 10 ticks per second). You can change this to make the game go faster or slower.

Close the window and press the button **Create Sounds**. Here you see that there is just one sound defined for the game. This is the sound used when you manage to click on the dog.

I hope this convinces you how simple things are. You might want to play a bit with this game and change some aspects. E.g. start with two dogs in the room (select the dog from the list at the bottom of the Rooms window and click anywhere in the field). Or change the alarm clock setting in the Objects form (click with the right mouse button on the action to change the settings). You can also make the object move rather than stand still (drag the action with the 8 arrows to the alarm event and select all arrow buttons). Finally, you might want to change the sound that the dog makes.

But maybe it is better to first read on, to understand how to use **Game Maker**.

The main interface

When you start **Game Maker** you are asked for the name of the game that you want to play or edit. (If you want to start creating a new game, click on the button labeled **Cancel**.) Now a window appears that might look disappointingly simple. It contains just a menu and a few buttons. But don't be fooled; a lot more awaits you. From left to right you find the following buttons:

-  **New Game** Start making a new game. After this you can press the buttons to create objects and design rooms, described below.
-  **Open Game** Use this to open an existing game. After you opened a game you can play it or change it.
-  **Save Game** Only available when you changed a game. Use this to save your game.
-  **Run Game** Runs the game. Suddenly the window becomes larger and the current game is being played. (You can also press <Ctrl>-R.)
-  **Pause Game** Pauses the currently running game. Press **Run Game** to continue it. (You can also press <Ctrl>-P.)
-  **Stop Game** Only available when the game is running. Stops the game. (You can also press <Ctrl>-Q.)
-  **Step** Do only one step in the game (only available when paused). (You can also press <Ctrl>-S.)
-  **Game Info** Gives you some information about the game (if provided) that you can also edit. (You can also press <Ctrl>-I.) Pressing the <F1> key also displays the help.
-  **Create Objects** When you press this button a large window appears in which you can create objects for your game, or edit existing objects (see below). Here you can (should) set the name of the game.
-  **Create Rooms** If you press this button you can create the rooms for your game (see below).
-  **Create Sounds** here you can indicate the sounds you want to use in the game (see below).

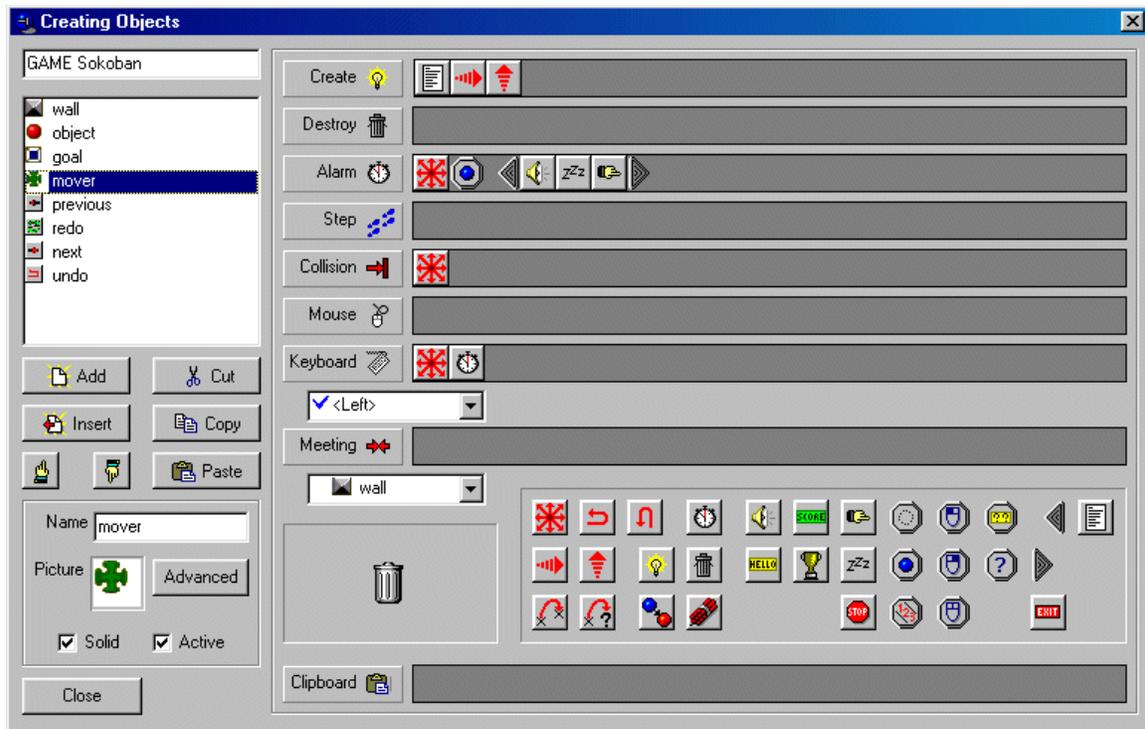
Some more commands are available in the menu. E.g. commands to rename the game, to delete the game or to save the game under a different name. For some of the other commands, see below.

So when you want to play a particular game you first press the button **Open Game**, select the game you want, and then press **Run Game**. You can press **Game Info** to get information on the game you play.

If you want to design a new game, press the button **New Game** and indicate the name of the game. Press **Create Sounds** if you want to use any sounds, and load and name them. Next press **Create Objects** to create the objects you need. When your objects are ready, choose **Create Rooms** to design the rooms for your game. Then press **Run Game** to test your game. Press **Game Info** to provide information about the game you made. Finally press **Save Game** to save your game.

Creating objects

The first step in designing a game is to create the objects that appear in the game. Typical objects you might need are walls, the figures that move, bonus items, etc. To create objects press the button with the blue ball on it. A form will open that looks as follows:



At the very top left of the form you see the name of the game. Change this into any name you like. Below that, you see a list of all objects currently defined. Below it there are buttons to add a new object at the end of the list or insert an object before the currently selected object. You can also change the order of the objects in the list by pressing the buttons with the up and down arrows. Finally, you can cut or copy an object to the clipboard and paste it back from the clipboard. You can use this to e.g. make copies of objects (by using copy and paste). You can even copy an object from one game to another. Some words of warning are required here though. First of all, don't move an object by using cut and paste. When you cut an object, all references to it in other objects (in particular meeting events) are removed. They don't reappear when you paste the object back. Also, when you copy an object from one game to another better don't have meeting events defined or sounds in the events because they might get mixed up completely.

Once you press **Add** or **Insert** or click on an object, some information about the object appears at the left bottom. First of all, there is the name. Make sure that all objects have different names. Although not strictly necessary, for advanced use it is better to only use letters, digits and the underscore ' _ ' symbol in the names. There is also a box that contains the object's image. Click on the box to load a different image (see below for more information on images). There is also a button labeled **Advanced**. When you press this button you can indicate more precisely how objects should be drawn. See the language guide for more information.

You also see two boxes labeled **Solid** and **Active**. As indicated before, solid means that the object will create collisions when other objects hit it. Set it for things like walls but not for other objects. Active means that the object reacts on certain events. Once you hit the **Active** box, at the right a large amount of information occurs. It shows all the different events and the possible actions you can use. See below for more information.

Finally there is the button labeled **Close**. Press this button to close the object form.

Object image

Objects have an image associated with them such that you can see them. An image can have any size. An image can either be an icon file (*.ico), a bitmap file (*.bmp), or a gif file (*.gif). To pick the image for an

object, click on the image at the left bottom of the object form. Use the file dialog to pick the correct file. Images are considered partially transparent such that they can move over a background. The color of the left bottom pixel of the image is the transparency color. So always make sure that this pixel is part of the background. Let me describe the possible image types in a bit more detail.

- Icon files are normally 32x32 pixels. There are huge collections of them available on the web. Some of these free public domain icons are provided with this program, but they are not part of the program. They normally have a transparency color defined, which is used by **Game Maker**. B.t.w., **Game Maker** internally converts the icon file into a gif file and also stores it as a .gif file.
- You can also use a bitmap file. The left bottom pixel color is used as transparency color. (For compatibility with the past, bmp files of size 32 times a multiple of 32 are seen as animated 32x32 bitmaps.
- But the most powerful way is to use an (animated) gif file. During the running of the game the animation is played. **Game Maker** stores all images as animated gif files. Animated gif files are available everywhere on the web. A number of them are provided with this program. You can create your own animated gif files using any of the many available gif animator programs, e.g. the free Microsoft GIF Animator at

<http://msdownload.microsoft.com/msdownload/gifanimator/gifsetup.exe>

or by using **Image Maker** which is available from the **Game Maker** web site

<http://www.cs.uu.nl/~markov/kids/gmaker/index.html>

Events

When the game starts running, events occur. By specifying actions for some of the events, you determine what happens in the game. For example, when an instance of an object is created, a creation event occurs for this instance. You can e.g. indicate that at this moment the object should start moving to the left. When the object hits a wall, a collision event occurs. You can e.g. say that in the case of a collision the object should reverse its horizontal direction. If you now place the object in a room, with walls to the left and the right, the object will keep on moving left and right between the walls.

Events can only be specified for active objects. Once you make an object active, at the right of the form all possible events occur. Each event is followed by a gray rectangle. In this rectangle you can drag the actions that should happen when the event occurs (see below). The following events exist. (In most games you will indicate actions for only few of them.)

-  **Creation Event.** A creation event occurs whenever an object is created. All instances of active objects will get a creation event when the game starts. You typically use them to give the object a direction and a speed. Also when you create an instance of an object during the game or when you change an instance into a different object, a creation event happens.
-  **Destruction Event.** This event happens when an instance of an object is destroyed. Often you don't need to do anything in this case, but you might use it to do something with the score, to end the game, or to create a new object somewhere else.
-  **Step Event.** This event occurs every step in the game. For simple games you don't need to do anything here but for more complicated games it is one of the most crucial events. You can for example continuously change the speed or direction of motion.
-  **Alarm Event.** Each active object has an alarm clock that you can set (see the actions below). The alarm clock counts down and when it reaches 0 an alarm event occurs. You can use this to let certain things happen from time to time. For example, an object can change its direction of motion every 10 steps (in such a case the alarm event, as one of its actions, sets the alarm again). Or you can open a door for a short period of time after which you close it again.

-  **Collision Event.** A collision event happens when an active object bumps into a solid object. This is not allowed so the actions in this event should take care that the collision does not occur. (If the actions do not avoid the collision, the program will try a couple of times, after which the object will stop moving. If there are no actions defined, the object will not react to the collision and might go straight through the solid object.) A typical action here is to stop the motion, reverse the direction of motion, or choose a random new direction of motion. But also more complicated actions can occur, e.g., you can push the other object out of the way. (Collision events only occur when you hit a solid object. If you want to have an action happening when you hit a non-solid object, use the meeting event below. When you specify a meeting event for a solid object, a collision event won't occur. In this way you can specify special behavior for certain objects.)
-  **Meeting Event.** For each object type you can specify a different set of actions. Indicate the object in the drop-down list and then specify the actions. You can specify meeting events for both solid and non-solid objects. Meeting events play a crucial role. When your man meets a monster he might die. When he hits a piece of gold the piece of gold is destroyed and the score is raised. And when he meets a button a door can be opened (changed from a solid closed door into a non-solid open door).
-  **Mouse Event.** A mouse event occurs when the user clicks with the mouse on the object. This can be used for user interaction. For example, you can generate some objects that look like buttons, and when the user clicks on them, something can happen in the game.
-  **Keyboard Events.** For further interaction you can specify keyboard events. In the dropdown list specify the key on the keyboard and next indicate the actions that should take place when the key is pressed (when the user keeps the key pressed, the event occurs every step). You can specify actions for the arrow keys, for the numeric keypad (when <NumLock> is pressed) for the normal character keys, and for the function keys. (User interaction is also possible with the joystick. See the document *Tips and Tricks*.)

Sounds like a huge list of possibilities. But for most games you need to specify actions for only a few events. Looking at the examples provided helps a lot in understanding the possibilities.

Actions

For each event you can specify the actions that must happen when this event occurs. Typical actions set the direction of the motion, the speed, etc. Each action is represented by an icon in the area at the bottom right of the form. (Because only active objects can have events, this is only shown when the current object is active.) When you let your mouse pointer rest on an icon, a description is given. You add actions to events by dragging them to the dark gray bars next to the names of the events (for keyboard and meeting events, make sure you first selected the right key or object in the dropdown lists below them). You can also drag actions from one event to another. If you hold the left <Ctrl> key, the action is copied. To remove an action, drag it to the trashcan. At the bottom of the form you see another dark gray box named Clipboard. You can also drag and copy actions here. They will stay on the clipboard as long as you don't stop **Game Maker**. In this way you can easily move or copy actions between different objects or between different keyboard or meeting events.

Most actions have some parameters that you have to provide. When you place an action, a form will pop up asking you for these values. If you want to change the parameters later, click on the action with your right mouse button. This form will look similar for most actions. At the top you can indicate to whom the action must apply. The default is self, which means that the action applies to the object that created the event. But you can also indicate that the action should apply to all instances of a particular object. So, for example, when your man meets a switch object you can make all doors disappear, or you can (temporarily) stop all monsters. For collision and meeting events you can also specify that the action should be applied to the other object involved. So, for example, when you hit a coin you can indicate that the other object (the coin) should disappear. Also, for many actions, you can indicate whether the change should be relative or not. Relative means that certain values, like the position or speed or timer, are increased with the amount you specify. If you uncheck relative, the values are set to the value you provide.

Movement

A number of actions deal with the movement of the objects.

-  **Set the direction of motion.** Click on the arrow indicating the direction you want, or click on the square in the middle to make the movement stop. You can press multiple directions. In this case a random choice is made.
-  **Reverse horizontal direction.** Reverses the horizontal direction of motion. Can e.g. be used when hitting a vertical wall.
-  **Reverse vertical direction.** Reverses the vertical direction of motion. Can e.g. be used when hitting a horizontal wall.
-  **Set horizontal speed.** Sets the horizontal speed. The number you give is the number of pixels the object moves in each step. Default, the speed is 8. Note that you set the speed relative to the current speed, unless you unmark the box labeled **Relative**. If you, e.g., want to make an object move faster all the time you can add a small value (e.g. 0.1) to the speed in every step.
-  **Set vertical speed.** Sets the vertical speed of the object.
-  **Move to position (x,y).** Moves the object to the specified position. Normally the position is relative to the current position. So e.g. (-32,0) means that the object is moved 32 pixels to the left. (0,32) means that the object is moves 32 pixels down. If you unmark the box labeled **Relative** you can specify an absolute position. (0,0) is the top left position in the room.
-  **Move to a random empty cell.** Moves the object to a random empty cell.

Creating and destroying objects

-  **Create a new object at (x,y).** Create a new object at position (x,y) (relative to the current object or absolute).
-  **Destroy object.** Use this action to destroy yourself or other objects. The destroy event will be generated.
-  **Change into another object.** Use this action to change the instance (or all instances of a particular type) into a different object. This is largely the same as destroying yourself and creating a new object of the new type at the current spot except that no new instance is created and things like speed, timers, etc. stay the same. But for the current object the destroy event is generated and for the new one the create event. In Pacman this event is constantly used to change the pacman object and to change the monsters into scared monsters.
-  **Destroy all objects at a position.** Destroys all the objects that exist at the indicated position. For example, if you have a bomb in the game and it explodes, you can kill all the object left, right, above and below it.

Other actions

-  **Set the alarm clock.** Here you can set the alarm clock such that after the indicated number of steps the alarm event happens. This can be used to let things happen after certain intervals.

Often the alarm event sets the alarm clock again to let the thing happen again some steps later. You can also set the alarm clock of another object.

-  **Set the score.** Here you can set the score or add to the score (relative). The score will be displayed once you set it. (So if you want the score to be displayed from the beginning, set it to 0 in the create event of some instance.)
-  **Show the highscore.** This action shows the highscore table. If the current score is higher, the player can fill in his name. Best use when the player dies in the game, that is, just before stopping the game. The highscore table is saved.
-  **Play a sound.** You are asked for the name of the sound. See below how to add sounds to your games. You can also select <stop sound> to stop the sound that is currently playing.
-  **Show a message.** Displays a message box containing the message string. Game play will be interrupted until the player presses **OK**. Can be used to give certain instructions during the game. Use # in the string to indicate a new line.
-  **Go to another room.** Indicate the number of the room (relative to the current room number or absolute). This can be used to create games with multiple rooms or levels. For example, associate such an action with the meeting event of the man with a door. To restart the current room, move relative to room 0.
-  **Sleep a while.** Specify the time in milliseconds.
-  **End the game.** Ends the game.

Conditionals

Conditionals might be bit more complicated to understand at first. They ask a question. When the answer is yes (true), the next action is performed. Otherwise the next action is skipped. You can also perform or skip a number of actions by putting them in a block. (Note that conditional actions have a different shape to distinguish them.)

-  **If there is no collision at (x,y).** You specify a position (relative to the current position or absolute). If moving there would cause no collision, the next action or block of actions is performed. This can e.g. be use to check whether a position is collision-free before going there.
-  **If there is a particular object at (x,y).** Again you specify a position, but also an object. If that object occurs at that position (that is, the current object would meet it if moved there) the next action is performed.
-  **If the number of instances is a value.** You provide the number and the object. If the number of instances of that object is equal to the number the next action is performed. For example, if there are no coins left (number is 0) you can do something.
-  **If question to player.** You specify a question for the player. If the player answers yes, the next action is performed.
-  **If left mouse button pressed.** If the left mouse button is pressed the next action is performed.

-  **If right mouse button pressed.** If the right mouse button is pressed the next action is performed.
-  **If no mouse button pressed.** If no mouse button is pressed the next action is performed.
-  **If expression.** Here you can type in an arbitrary expression (see below). If it is true, the next action is performed.
-  **Begin block.** Use this symbol immediately after a conditional to create a block of actions.
-  **End block.** Place this at the end of the block.
-  **Exit event.** This action exits the event, that is, no further actions in the event are performed. This can be useful after a conditional.

Using expressions

In many actions you need to provide values. Rather than just typing a number, you can also type a formula, e.g. $32*12$. But you can actually type much more complicated expressions. For example, if you want to double the horizontal speed, you could set it to $2*hspeed$. Here *hspeed* indicated the current horizontal speed. There are a number of other values you can use. The most important ones are:

- **x** the x-coordinate of the instance
- **y** the y-coordinate of the instance
- **hdir** the horizontal direction (-1 = left, 0 = no motion, 1 = right)
- **vdir** the vertical direction (-1 = upwards, 0 = no motion, 1 = downwards)
- **hspeed** the horizontal speed (in pixels per step)
- **vspeed** the vertical speed (in pixels per step)
- **alarm** the value of the alarm clock (in steps)

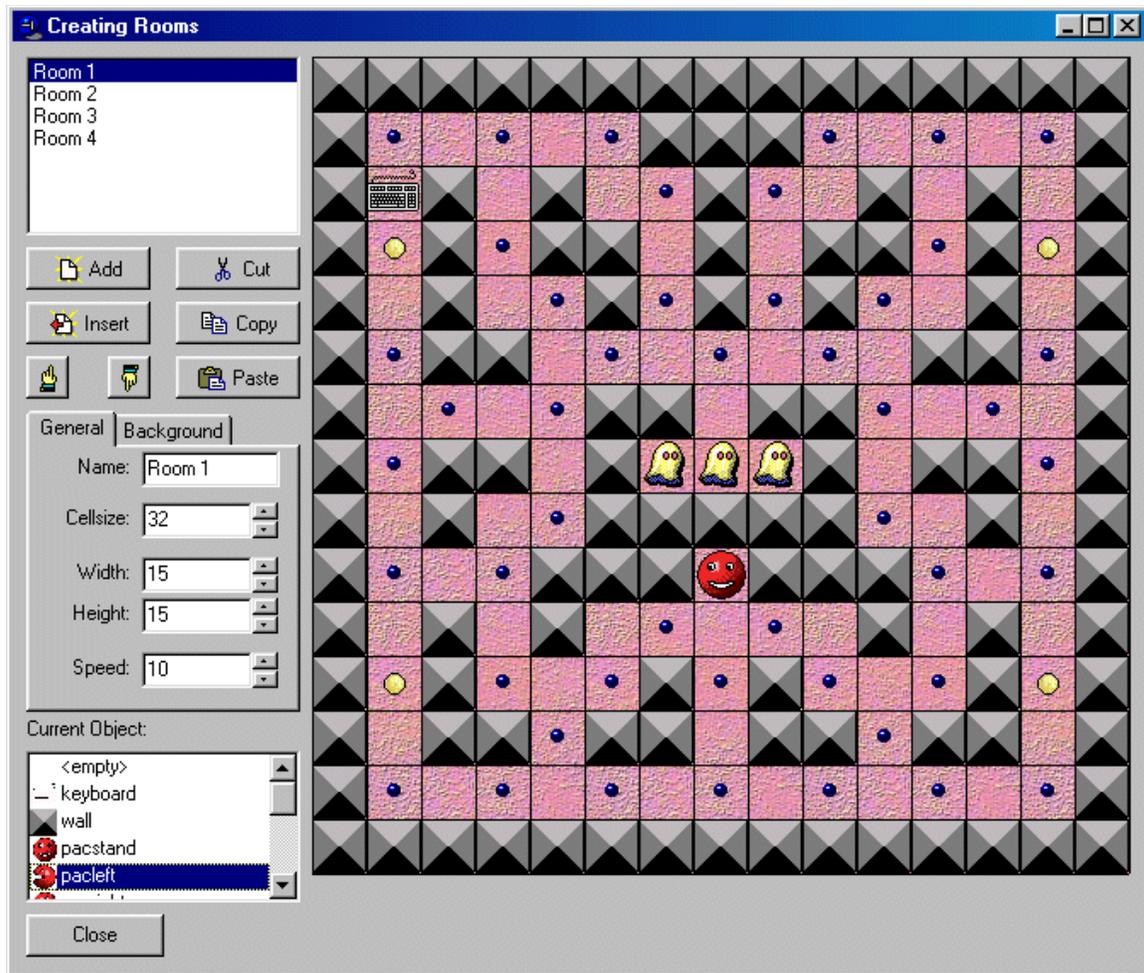
You can also refer to these values for other objects by putting the object name and a dot in front of it. So e.g. if you want a ball to move to the place where the coin is you can set the position to (coin.x,coin.y). In the case of a collision or meeting event you can refer to the x-coordinate of the other object as other.x. In conditional expressions you can use comparisons like < (smaller than), >, etc. For complete information on expressions see the *Game Maker Language Documentation* provided.

Using code

Even though you can create quite elaborate games using the standard actions, at some stage you might want further control. To this end **Game Maker** has a complete built-in programming language and interpreter. You can create actions that contain pieces of code. Within this code you can actually do almost everything you can do in the actions, but you can do a lot more. For a detailed description of the language, see the enclosed *Game Maker Language Documentation*.

Creating rooms

After you defined the objects you need it is time to create the room(s). Click on the **Create Rooms** button and the following form will pop up.



At the top left you find the list of rooms. Below that there are buttons to add or insert a room, change the order of the rooms, or cut or copy a room to the clipboard and paste it back. (You can copy rooms between games, but this only works if they use the same objects!) Once you added a room or selected a room, more information is shown. In the middle left you see the following information:

- **Name.** The name of the room will be shown when running the game.
- **Cellsize.** The size of a cell in the room in pixels. Normally this is 32 but if you want to use smaller images you might want to change this (some of the provided examples use 24).
- **Width.** The horizontal number of cells. Change it using the up and down arrow buttons.
- **Height.** The vertical number of cells.
- **Speed.** The game speed, that is, the number of steps per second. (If your machine is slow this number might not really be achieved.)

If you click on the tab labeled **Background** you get further information:

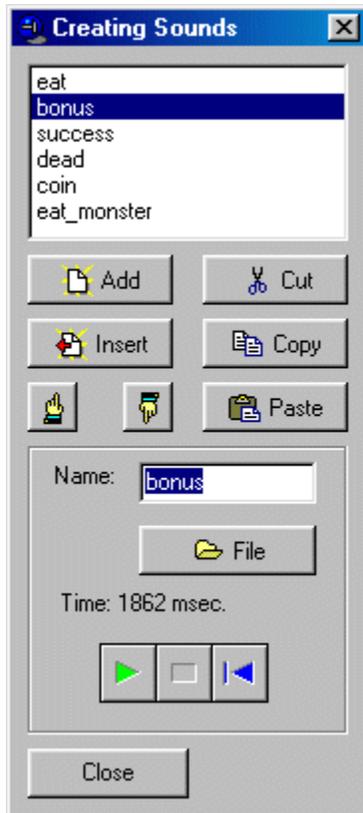
- **Color.** Click here to have a colored background. Click on the box to change the color.
- **Image.** Click here to have a background image. Click on the box to indicate the filename for the image. Click **Tiled** to have copies of the image placed next to each other. Click **Stretched** to have the image stretched over the background. If you click **Scrolling** you can make a scrolling background. In this case further fields appear in which you can set the horizontal and vertical scrolling speed.

At the right there is the room. You place objects in the room by choosing the object in the list at the bottom left, and then clicking with your mouse in the correct cell. (Each cell can contain only one object.) With the right mouse button you can clear cells.

When you are ready, click on the button labeled **Close**.

Creating sounds

A nice game should definitely have some sounds in it. To add sounds to your game, press the button **Create Sounds**. The following form will show:



At the top you see the list of sounds. Below it you find buttons to add or insert a sound, change the order of the sounds, or cut or copy a sound to the clipboard and paste it back. (Realize that if you cut a sound all sound actions that refer to it are removed.) When you click on a sound or add a sound, you can change its name and choose the sound file. You can also test the sound using the buttons. When you are done click the button labeled **Close**.

There are two types of sound files that can be used: midi files (extension .mid) or wave files (extension .wav). Midi files are typically used for background music. They will loop forever. Wave files are for short sound effects. To add a sound effect to a particular event, use the sound action. (To create background music, use a sound action in the creation event in some object.)

Creating game information

A good game provides the player with some information on how to play the game. This information is displayed when you press the button **Game Help**. A little build-in editor is opened where you can see and edit the game information. You can use different fonts, different colors, and styles. A good advice is to

make the information short but precise. Of course you should add your name because you created the game. All example games provided have an information file about the game and how it was created.

If you want to make a bit more fancy help, use e.g. Word. Then Select the part you want and use copy and paste to move it from word to the game maker editor.

Distributing your game

Of course you would like others to be able to play your games as well. This is very easy. Load the game you want to distribute. In the **File** menu there are three important items: **Import**, **Export**, and **Create stand-alone**. To distribute a game, click on the item **Export**. You are asked for a filename where to store the game. (The file name will end with .zip, because the games are stored as compressed zip files.) Best store it at a place where you can find it back. Now give this file to your friend. To put the game into **Game Maker**, such that you can play it, use the menu item **Import**, choose the correct zip file, and you are done. The game can now be played and is added to your list of games (so you don't have to import it again later).

Please mail your creations to markov+games@cs.uu.nl such that I can place them on the web site.

If you want to create a version of the game that does not require **Game Maker** to be installed on your computer, open the game and choose the menu item **Create stand-alone**. You are asked for the place where to create the stand-alone game. Indicate the directory and you are done. If your game was named XXX, at the indicated place a new directory XXX has been created in which you will find a program called XXX.exe (plus a directory called Games that stores the games information). Executing XXX.exe will play your game. Note that in the stand-alone version the menu and toolbar are not shown. The player can view the help file by using <F1>. If the game ends, the program ends as well.