Game Maker Tutorial

Your First Game

Written by Mark Overmars

Copyright © 2007-2009 YoYo Games Ltd
Last changed: December 23, 2009
Uses: Game Maker 8.0, Lite or Pro Edition, Simple Mode
Level: Beginner

Even though *Game Maker* is very easy to use, getting the hang of it might be a bit difficult at first. This tutorial is meant for those that have some difficulty getting started with *Game Maker*. It will lead you step by step through the process of making your first game. Realize that this is the most difficult part. To make your first game you have to understand a number of the basic aspects of *Game Maker*. So please read this tutorial carefully and try to understand all the steps. Once you finished your first game the second one is going to be a lot easier.

The Game Idea

It is important that we first write a brief description of the game we are going to make. Because this is going to be our first game we better design something simple. It should keep the player interested for just a short time. Our game is going to be a little action game that we will name *Catch the Clown*. (Always try to come up with a nice name for your game.) Here is our description of the game:

> **Catch the Clown**
> *Catch the Clown* is a little action game. In this game a clown moves around in a playing field. The goal of the player is to catch the clown by clicking with the mouse on him. If the player progresses through the game the clown starts moving faster and it becomes more difficult to catch him. For each catch the score is raised and the goal is to get the highest possible score. Expected playing time is just a few minutes.

Clearly, a game like this will have limited appeal. But we have to start simple. Later we can add some features to the game to make it more interesting.

A Design Document

The second step in creating a game is to write a more precise design document. You are recommended to always do this before making your game, even if it is very simple. Here is our design document for *Catch the Clown*. (I omitted the description that was already given above.)

Catch the Clown Design Document

Game objects

There will be just two game objects: the clown and the wall. The wall object has a square like image. The wall surrounding the playing area is made out of these objects. The wall object does nothing. It just sits there to stop the clown from moving out of the area. The clown object has the image of a clown face. It moves with a fixed speed. Whenever it hits a wall object it bounces. When the player clicks on the clown with the mouse the score is raised with 10 points. The clown jumps to a random place and the speed is increased with a small amount.

Sounds

We will use two sounds in this game. A bounce sound that is used when the clown hits a wall, and a click sound that is used when the player manages to click with the mouse on the clown.

Controls

The only control the player has is the mouse. Clicking with the left mouse button on the clown will catch it.

Game flow

At the start of the game the score is set to 0. The room with the moving clown is shown. The game immediately begins. When the player presses the <Esc> key the game ends.

Levels

There is just one level. The difficulty of the game increases because the speed of the clown increases after each successful catch.

That should be good enough for the moment. We can now start creating the game. So start up *Game Maker* and let's get going. Note that this tutorial uses version 8.0 of *Game Maker*. If you use a different version, the images look a bit different. It also assumes the program runs in simple mode. You can switch between simple and advanced mode by clicking on the menu item **Advanced Mode** in the **File** menu. In advanced mode there are many more options in the different menus and forms but we won't need these for our simple game.

The game we are going to create is already given in the folder `Example` that comes with this tutorial. You can load it from there but you are recommended to recreate it by following the steps described below. In this way you will better understand how a game is being made in *Game Maker*. All the sprites, images, and sounds we will use are provided in the folder `Resources`.

Adding Sprites and Sounds

As the game design document describes we will need two images for the two game objects. Such images are called sprites in *Game Maker*. There is a lot to know about sprites but for the moment, simple think

of them as little images. So we need to make or find such images. For making the images you can use any drawing program you like, for example the paint program that is part of any *Windows* system. But *Game Maker* also has a built-in drawing program for this purpose. Creating nice-looking sprites is an art that requires a lot of practice. But fortunately there are large collections of images of all sorts available for free. *Game Maker* includes a considerable number of these and on our *YoYo Games* web site (www.yoyogames.com) you can find many more. Alternatively, search the web and you are bound to find images in large quantities. For our little game we use the following two sprites, which can be found in the `Resources` folder that comes with this tutorial.

The clown: The wall:

To add these sprites to the game we proceed as follows:

Creating the clown sprite resource for the game:

1. From the **Resources** menu, choose **Create Sprite**. The Sprite Properties form appears, like the one shown in Figure 1.

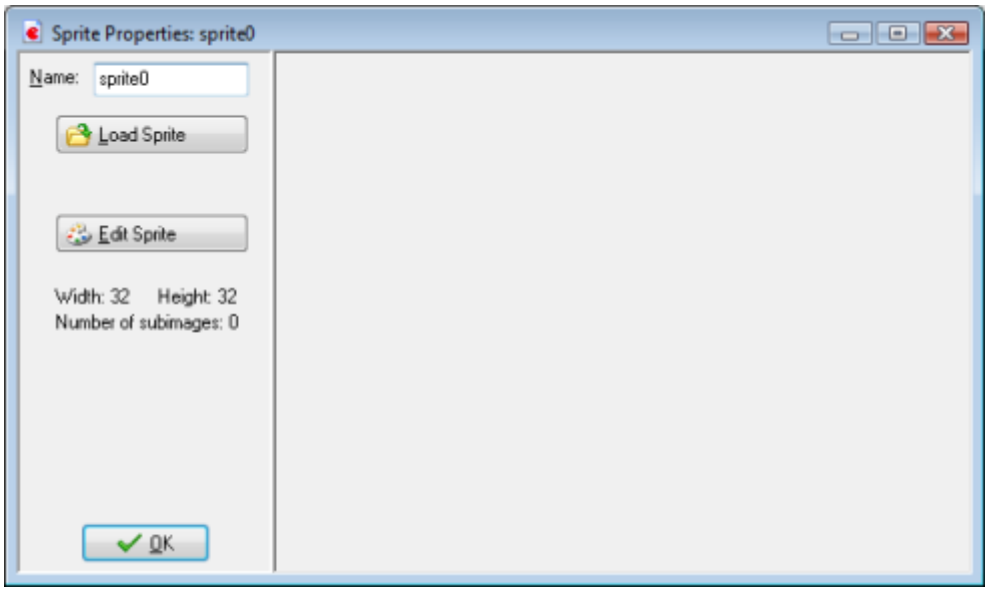

Figure 1. The empty Sprite Properties form.

2. Click on the **Name** field where currently is says `sprite0`. This is the default name for the sprite. Rename it to `spr_clown`.
3. Click on the **Load Sprite** button. This opens a file requester.
4. Navigate to the `Resources` folder that came with this tutorial and selected the image file `clown.png`. The Sprite Properties form should now look like Figure 2.
5. Press the **OK** button to close the form.

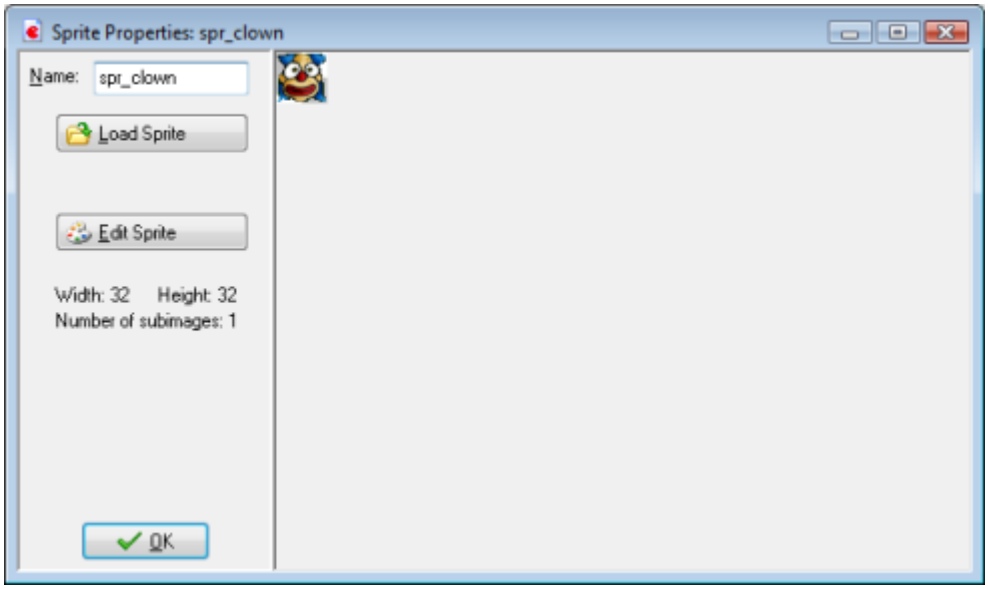

Figure 2. The clown sprite.

Next we will add the wall object in the same way.

Creating the wall sprite:

1. From the **Resources** menu, choose **Create Sprite**. Click on the **Name** field and rename it to `spr_wall`.
2. Click on the **Load Sprite** button and select the image file `wall.png`.
3. Press the **OK** button to close the form.

As you might have noticed, the clown and wall sprite have now appeared in the list of resources at the left of the *Game Maker* window. Here you will always find all the sprites, sounds, objects, rooms, etc. that you have created in your game. Together we call them the *resources* of the game. You can select a resource by clicking on its name. Now you can use the **Edit** menu to change the resource, duplicate it, or delete it. Right-clicking on the resource name will show the same menu. This overview of resources will become crucial when you are creating more complicated games.

Now that we created the sprites we will create two sound effects. One must play when the clown hits a wall and the other must play when the clown is successfully caught with the mouse. We will use two wave files for this. Wave files are excellent for short sound effects. A number of these sound effects are part of the installation of *Game Maker* and many more can be found on the web.

Create two sound resources:

1. From the **Resources** menu, choose **Create Sound**. The Sound Properties form appears. Click on the **Name** field and rename it to `snd_bounce`.
2. Click on the **Load Sound** button, navigate to the `Resources` folder that came with the tutorial, and select the sound file `bounce.wav`. The form should now look as shown in Figure 3.
3. Press the **OK** button to close the form.

Figure 3. The bounce sound resource.

4. Create another sound resource and name it `snd_click`.
5. Click the **Load Sound** button and select the sound file `click.wav`.
6. Close the form.

Within the sound properties form you can use the play button, with the green triangle pointing to the right, to listen to the sound (it is constantly repeated). Again, notice that the two sounds are shown in the list of all resources.

Objects and Actions

Having created the sprites and sounds does not mean that anything is happening. Sprites are only the images for game objects and we have not yet defined any game objects. Similar, sounds will only play if we tell them to be played. So we need to create our two game objects next.

But before we will do this you will have to understand the basic way in which *Game Maker* operates. As we have indicated before, in a game we have a number of different *game objects*. During the running of the game one or more *instances* of these game objects will be present on the screen or, more general, in the game world. Note that there can be multiple instances of the same game object. So for example, in out *Catch the Clown* game there will be a large number of instances of wall objects, which surround the playing field. There will be just one instance of the clown object.

Instances of game objects don't do anything unless you tell them how to act. You do this by indicating how the instances of the object must react to *events* that happen. There are many different events that can happen. The first important event is when the instance is created. This is the **Create Event**. Probably some action is required here. For example we must tell the instance of the clown object that it should start moving in a particular direction. Another important event happens when two instances collide with each other; a so-called **Collision Event**. For example, when the instance of the clown collides with an instance of the wall, the clown must react and change its direction of motion. Again other events happen when the player presses a key on the keyboard or clicks with mouse on an instance. For the clown we will use a **Mouse Event** to make it react to a press of the mouse on it.

To indicate what must happen in the case of an event, you specify *actions*. There are many useful actions for you to choose from. For example, there in an action that sets the instance in motion in a

particular direction, there is an action to change the score, and there is an action to play sounds. So defining a game object consists of a few aspects: we give the object a sprite as an image, we can set some properties, and we can indicate to which events instances of the object must react and what actions they must perform.

Note the distinction between *objects* and *instances* of those objects. An object defines a particular game object with its behavior (that is, reaction to events). Of this object there can be one or more instances in the game. These instances will act according to the defined behavior. Stated differently, an object is an abstract thing. Like in normal life, we can talk about a chair as an abstract object that you can sit on, but we can also talk about a particular chair, that is an instance of the chair object, which actually exists in our home.

So how does this work out for the game we are making? We will need two objects. Let us first create the very simple wall object. This object needs no behavior at all. It will not react to any events.

Create the wall object:

1. From the **Resources** menu, choose **Create object**. The Object Properties form appears, as is shown in Figure 4.

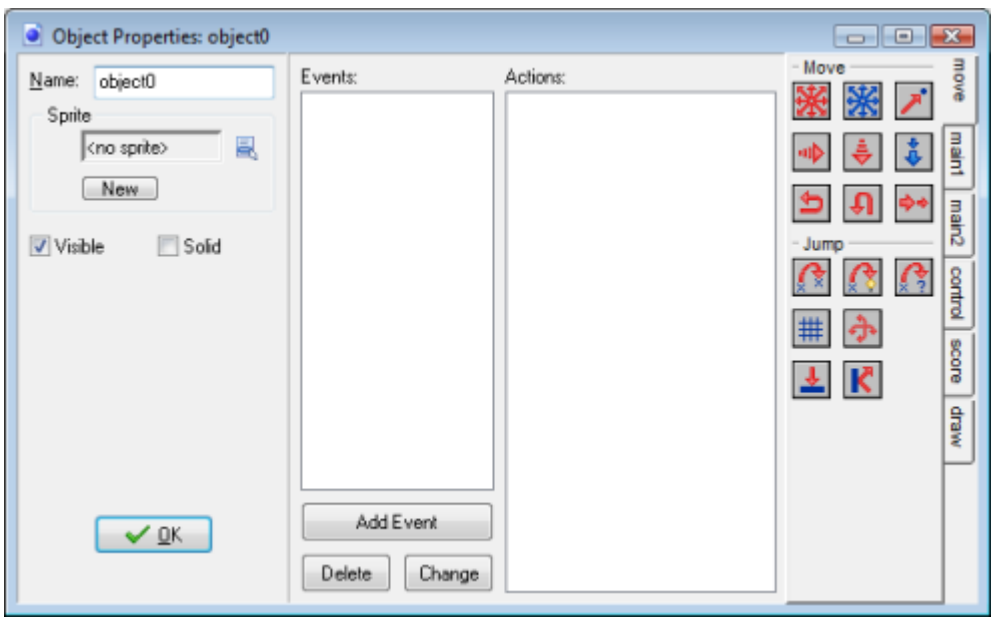

Figure 4. The empty Object Properties form.

2. Click on the **Name** field and rename the object to `obj_wall`.
3. Click on the menu icon at the end of the **Sprite** field and in the list of available sprites select the `spr_wall` sprite.
4. Instances of the wall object must be solid, that is, no other instances should be allowed to penetrate them. To this end click on the box next to the **Solid** property to enable it.
5. The filled-in form is shown in Figure 5. Press **OK** to close the form.

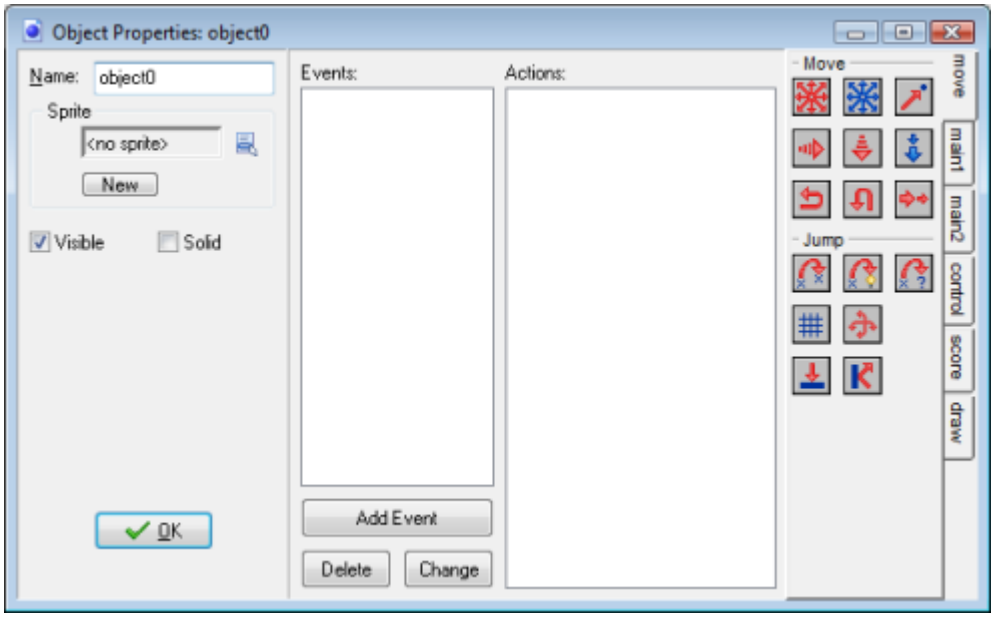

Figure 5. The filled-in properties form for the wall object.

For the clown object we start in the same way.

Create the clown object:

1. From the **Resources** menu, choose **Create object**.
2. Click on the **Name** field and rename the object to `obj_clown`.
3. Click on the icon at the end of the **Sprite** field and select the `spr_clown` sprite.

Note that we do not make the clown object solid. But for the clown there is a lot more that needs to be done. We have to specify its behavior. For this we need the rest of the form. In the middle you see an empty list with three buttons below it. This list will contain the different events that the object must respond to. With the buttons below it you can add events, delete events or change events. There are a large number of different events but you normally need just a few in your game.

Next to the events there is an empty list of actions that must be performed for the selected event (if any). And at the right of this list that are a number of tabbed pages with little icons. These icons represent the different actions. In total there are close to 100 different actions you can choose from. If you hold your mouse above one of the icons a short description of the corresponding action is given. You can drag actions from the tabbed pages at the right to the action list to make them happen when the event occurs.

We are first going to define what should happen when an instance of the clown object is created. In this case we want the clown to start moving in an arbitrary direction.

Let the clown object move:

4. Press the **Add Event** button. The Event Selector, as shown in Figure 6 will appear.

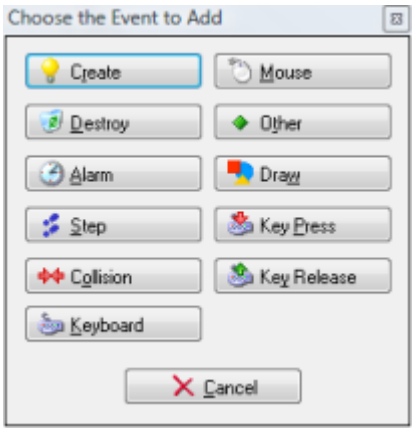

Figure 6. The Event Selector.

5. Click on the **Create** button. The create event is now added to the list of events. It is automatically selected (with a blue highlight).

6. Next you need to include a **Move Fixed** action in the list of actions. To this end, press and hold the mouse on the action image with the eight red arrows in the page at the right, drag it to the empty actions list, and release the mouse. An action form is shown asking for information about the action.
7. In the action form for the **Move Fixed** action you can indicate in which direction the instance should start moving. Select all eight directions (not the middle one; which corresponds to no motion). Note that the selected directions turn red. When multiple directions are selected one is chosen randomly. Also set the **Speed** to 4. See Figure 7 for the result. Press **OK** to indicate that we are ready with this action.

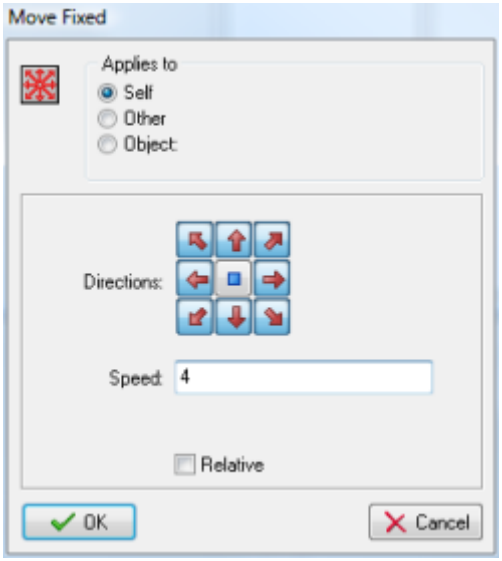

*Figure 7. Setting the directions for the **Move Fixed** action.*

You have now specified behavior that must be executed when an instance of the clown object is created, by adding the event, including an action, and setting the action properties. The object properties form for the clown object should now look as in Figure 8.

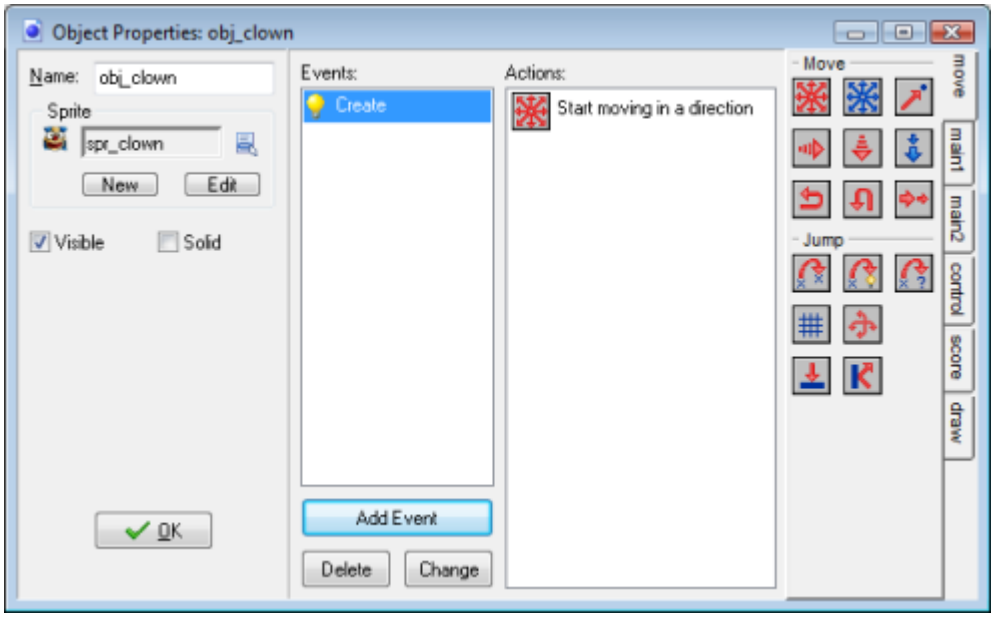

*Figure 8. The properties from for the clown object after specifying the **Create** event.*

The next event we will define is a collision with a wall. Here we will bounce the clown against the wall and we will play the bounce sound effect.

Handling a collision with the wall:

1. Press the **Add Event** button. In the Event Selector click on the **Collision** button and select `obj_wall`. The collision event is now added to the list of events.
2. Include a **Bounce** action by dragging it from the page at the right. The action form shown in Figure 9 will appear. There are two properties we can change but their default values are fine. We are not interested in precise bounces and we want to bounce against solid objects. (Remember that we made the wall object solid.) Press **OK** to close the action form.
3. Select the page with the tab **main1**. From it include the **Play Sound** action and drag it below the **Bounce** action already present. In the action from, click on the icon to the right of the **Sound** property and from the list select `snd_bounce`. Leave the **Loop** property to **false** as we want to play the sound only once. The form should look like in Figure 10. Press **OK** to close it.

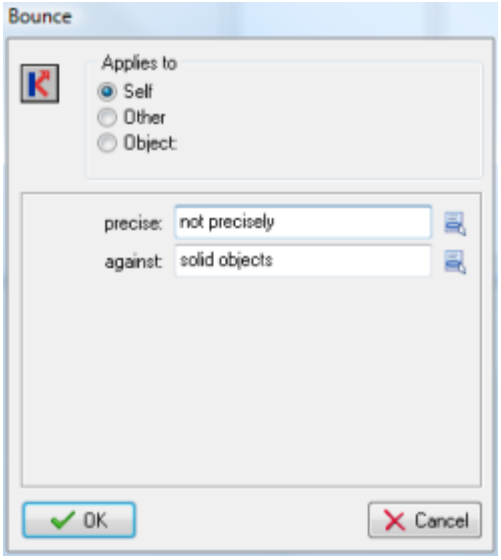

*Figure 9. The **Bounce** action form.*

Figure 10. Playing the bounce sound effect.

That concludes the collision event with the wall object. The object properties form should now look as in Figure 11.

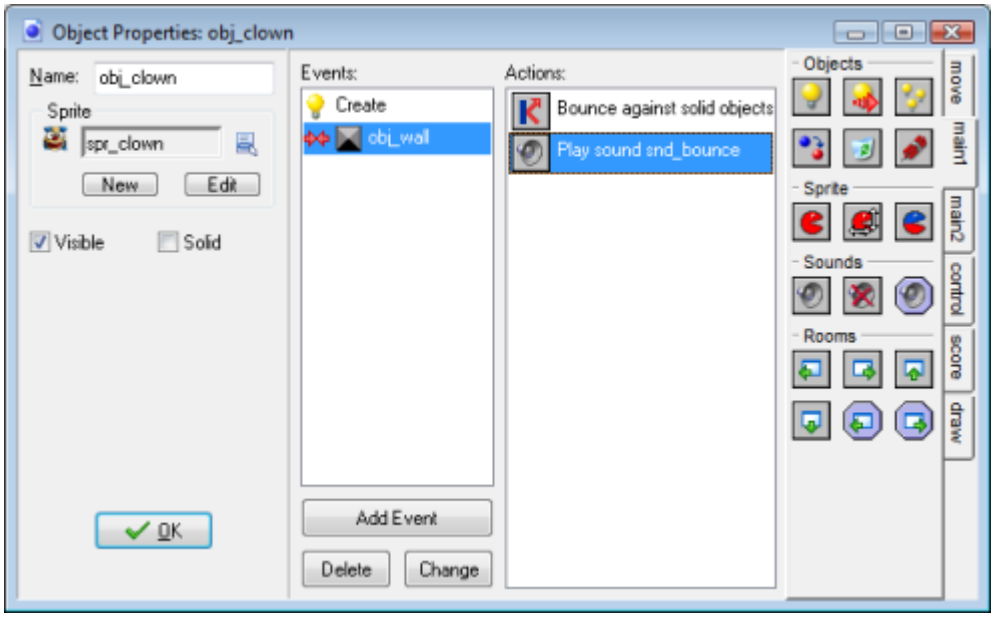

Figure 11. The collision event with the wall object.

There are two actions that are both performed (in the given order) when the collision occurs. If you for some reason made a mistake, you can right-click with the mouse on an action you added and for example choose **Delete** to remove the action (or press the <Delete> key on the keyboard). You can also choose **Edit Values** to change the properties of the action. (Double-clicking on the action will do the same.) And you can drag them up and down to change the order in which they are executed.

Finally we need to define what to do when the user clicks with the left mouse on the clown. We are going to add four actions here: First we will add 10 points to the score. This is easy as *Game Maker* automatically keeps and displays a score. Next we will play the click sound. After this we will jump the clown to a random position, and we will set a new random direction of motion with a slightly increased speed. The last two actions are added to gradually increase the difficulty of the game.

Handling a mouse press:

1. Press the **Add Event** button. In the Event Selector click on the **Mouse** button and in the menu that appears select Left Pressed. This event happens when the user presses the left mouse button while the mouse cursor is on top of the instance.

2. From the tabbed page labeled **score** include the **Set Score** action. As **new score** indicate a value of 10. Also click on the box next to the property **Relative** to enable it. When **Relative** is enabled the value is added to the current score. Otherwise the score would be replaced by the value. The action from should look like in Figure 12.

Figure 12. Adding 10 to the current score.

3. From the page **main1** include a **Sound** action. As **Sound** indicate `snd_click`. Leave **Loop** to false.

4. From the page **move** include a **Jump to Random** action. This action places the instance in a random collision-free position. The parameters can be left unchanged. See Figure 13.

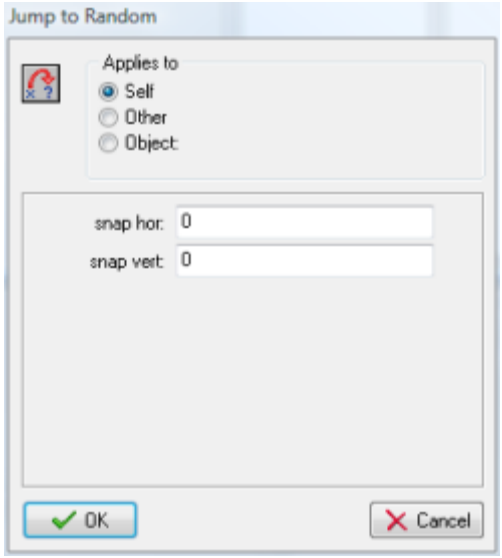

Figure 13. Jumping to a random position.

5. Finally we include a **Move Fixed** action. Again select all eight arrows (and not the center square). As **Speed** indicate a value of 0.5 and enable the **Relative** property to add 0.5 to the current speed.

We are now ready with the clown object. We have included actions for the three events that are important. It should now look as in Figure 14. Press the **OK** button to close the form.

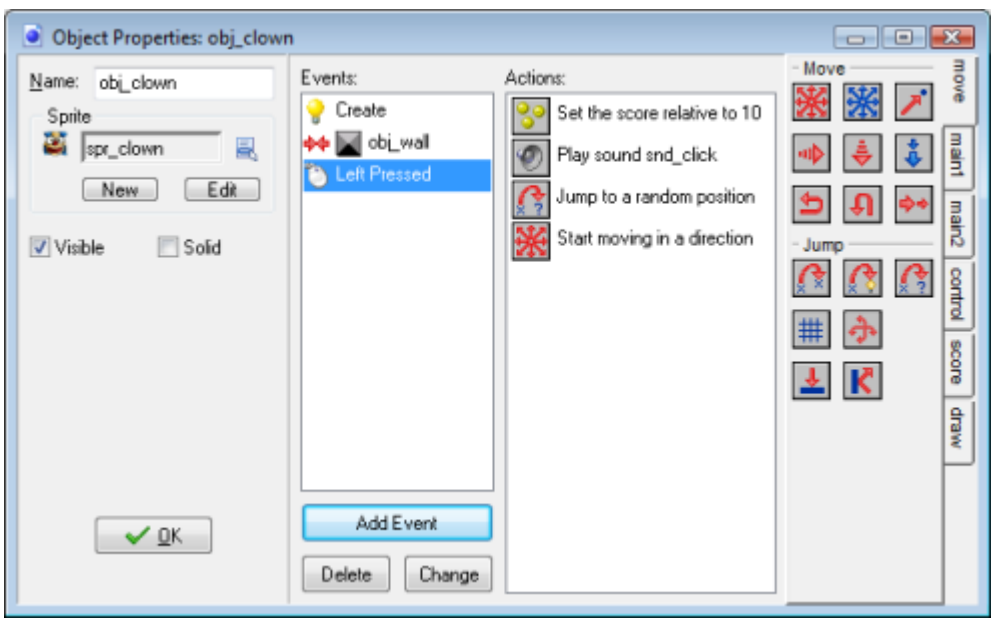

Figure 14. The clown object with all actions included.

Creating the Room

Now that we have created the game objects there is one more thing to do. We need to create the room in which the game takes place. For most games, designing effective rooms (often also called levels) is a time-consuming task because here we must find the right balance and progression in the game. But for *Catch the Clown* the room is very simple: a walled area with one instance of the clown object inside it.

Creating the room:

1. From the **Resources** menu choose **Create Room**. The Room Properties form as shown in Figure 15 will show.

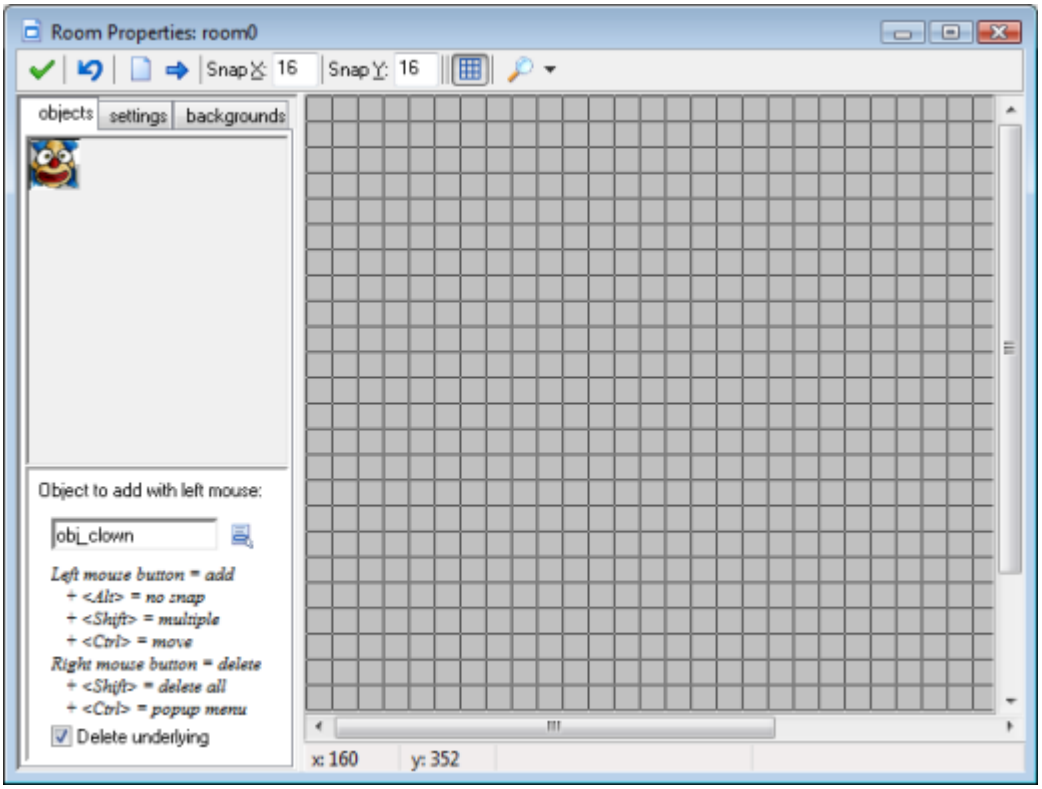

Figure 15. The Room Properties form.

2. On the left you see three tabbed pages. Select the page labeled **settings**. In the **Name** field type in `rm_main`. In the **Caption for the room** field type 'Catch the Clown'.
3. Select the **objects** tab. Enlarge the window somewhat such that you can see the complete room area at the right. At the top, change the value for **Snap X** and **Snap Y** to 32. As the size of our sprites is 32, this makes it easier to place the sprites at the correct locations.
4. At the left you see the image of the clown object. This is the currently selected object. Place one instance of it in the room by clicking with the mouse somewhere in the centre of the grey area.
5. Click on the icon with the menu symbol next to the field `obj_clown`. Here you can select which object to add. Select `obj_wall`. Click on the different cells bordering the room to put

instances there. To speed this up, press and hold the <Shift> key on the keyboard and drag the mouse with the mouse button pressed. You can remove instances using the right mouse button.

6. Press the button with the green V sign at the left top to close the form.

Saving and Testing

You might not have realized it but our game is ready now. The sprites and sounds have been added, the game objects have been designed and the first (and only) room in which the game takes place has been created. Now it is time to save the game and to test it.

Saving the games works as in almost any other Windows program. Choose the command **Save** from the **File** menu, select a location and type in a name. *Game Maker* games get a file extension `.gmk`. Note that you cannot directly play such game files. You can only load them in *Game Maker*. Below we will see how to make stand-alone game executables.

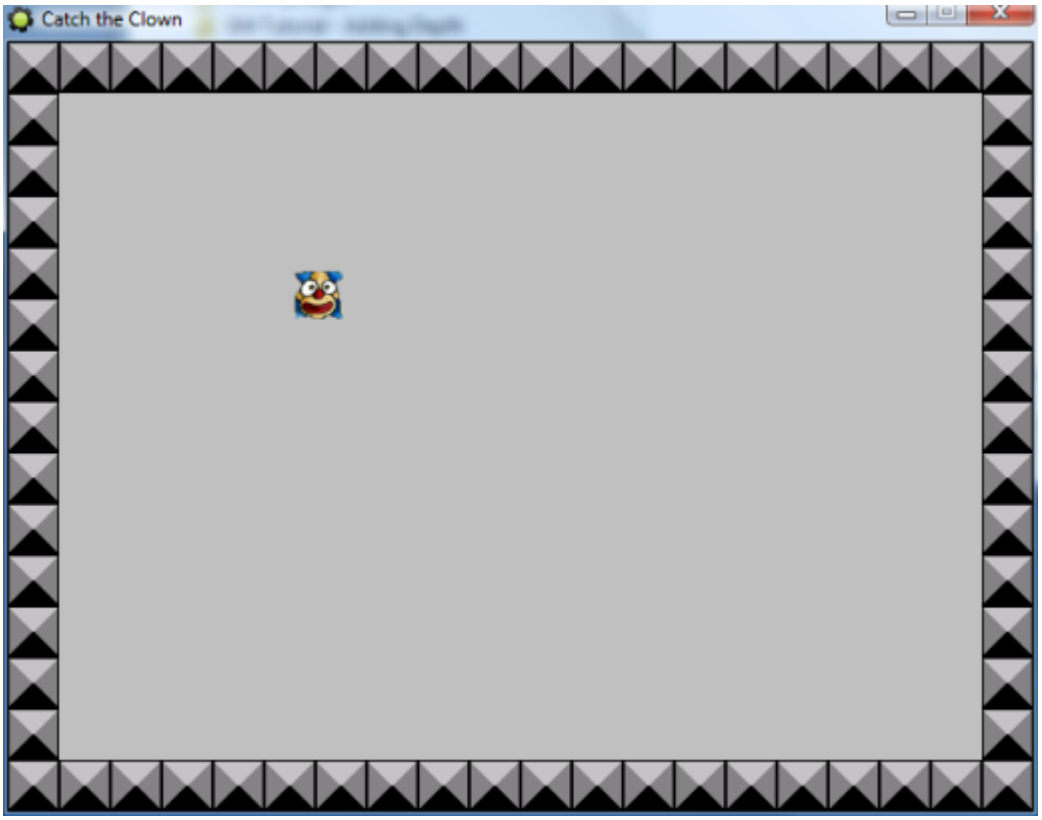

Figure 16. Playing the game.

Next we need to test the game. Testing is crucial. You can test it yourself but you should also ask others to test it. Testing (or running the game in general) is simple; choose the command **Run normally** from the **Run** menu. The design window will disappear, the game will be loaded and, if you did not make any mistakes, the room will appear on the screen with the clown moving inside it, as in Figure 16. Try clicking on it and see whether the game behaves as expected. You should hear the correct sounds and the speed

of the clown should increase. To end the game, press the <Esc> key or click on the close button at the top right of the window. The design window will reappear.

Now it is time to fine tune the game. You should ask yourself for example the following questions: Is the initial speed correct? Is the increase in speed correct? Is the room size correct? Did we pick effective sprites and sounds for the game? If you are not happy, change these aspects in the game and test again. Remember that you should also let somebody else test the game. Because you designed the game it might be easier for you than for other people.

Once you are happy with your game you should create a stand-alone executable for the game. This is a version of the game that can run without the need for *Game Maker*. This is very simple. In the **File** menu choose the command **Create Executable**. You have to indicate the place and name of the stand-alone executable and you are done. You can now close *Game Maker*, and run the executable game. You can also give the executable to your friends and let them play it, or you can publish it on the YoYo Games website http://www.yoyogames.com for people to download. (Of course we do not recommend you to place this exact copy of the *Catch the Clown* game there. Better create your own original game.) For this you will need some screenshots of the game, which you can easily make by pressing <F9> while running the game.

Finishing touches

Our first game is ready but it needs some finishing touches to make it a bit nicer. First of all we are going to add some background music. This is very easy.

Create background music:

1. From the **Resources** menu, choose **Create Sound**. The Sound Properties form appears. Click on the **Name** field and rename it to `snd_music`.
2. Click on the **Load Sound** button, navigate to the `Resources` folder and select the sound file `music.mid`. Note that this is a midi file. Midi files are often used for background music as they use less memory.
3. Press the **OK** button to close the form.
4. Reopen the clown object by double clicking on it in the resource list at the left of the window.
5. Select the **Create** event. From the **main1** page include a **Play Sound** action. As **Sound** indicate `snd_music` and set **Loop** to true because we want the music to repeat itself forever.

Secondly we are going to add a background image. The grey background of the room is rather boring. To this end we use a new type of resource, the background resource.

Add a background image:

1. From the **Resources** menu, choose **Create Background**. The Background Properties form appears. Click on the **Name** field and rename it to `back_main`.
2. Click on the **Load Background** button, navigate to the `Resources` folder and select the image file `background.png`. The form should now look like Figure 17.

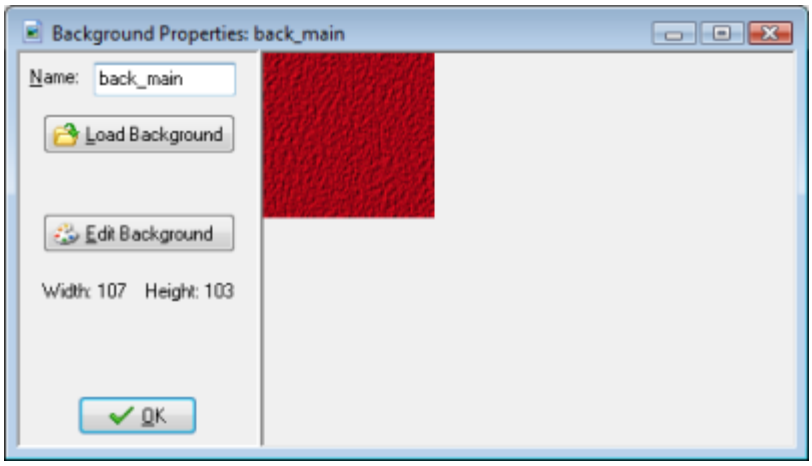

Figure 17. The Background Properties form.

3. Press **OK** to close the form.
4. Reopen the room by double clicking on it in the resource list.
5. Select the **backgrounds** tab. Deselect the property **Draw background color**.
6. Click on the little menu icon in the middle and pick the `back_main` in the popup menu. As you will see, in the room we suddenly have a nice background, As in Figure 18. Note the two properties **Tile Hor.** and **Tile Vert.** They indicate that the background must be tiled horizontally and vertically, that is, repeated to fill the whole room. For this to work correctly the background image must be made such that it nicely fits against itself without showing cracks.

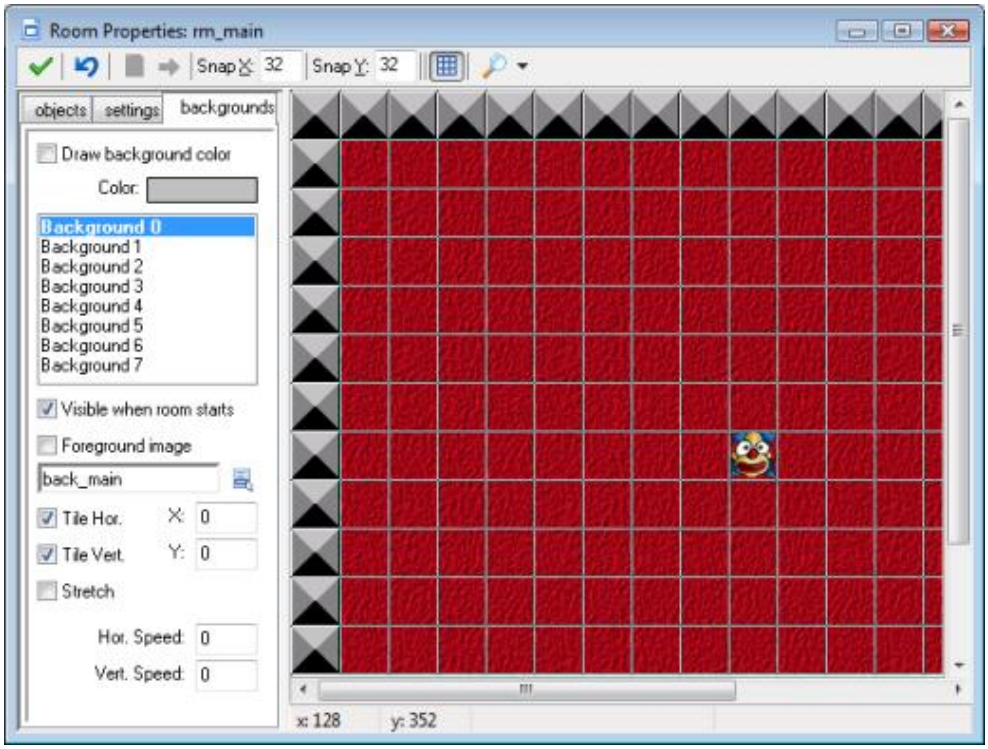

Figure 18. The room with a background.

If you play the game a bit you will see that it is very easy because you know exactly where the clown is going. To make it more difficult we let the clown change its direction of motion from time to time. To this end we are going to use an alarm clock. Each instance can have multiple alarm clocks. These clocks tick down and at the moment they reach 0 an **Alarm** event happens. In the creation event of the clown we will set the alarm clock. And in the alarm event we change the direction of motion and set the alarm again.

Adding the alarm clock:

1. Reopen the clown object by double clicking on it in the resource list at the left of the window.
2. Select the **Create** event. From the **main2** page include a **Set Alarm** action. As **Number of steps** indicate 50. The alarm number we keep as Alarm 0. See Figure 19.

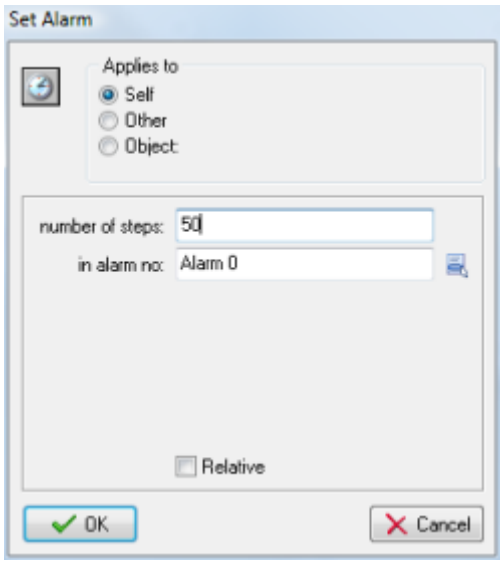

Figure 19. Setting the alarm clock to 50 steps.

3. Click on **Add Event**. Choose the button **Alarm** and in the popup menu select **Alarm 0**.
4. In the event include the **Move Fixed** action (from the **move** tab). Select all eight arrows. Set the **Speed** to 0 and enable the **Relative** property. In this way 0 is added to the speed, that is, it does not change.
5. To set the alarm clock again, include a **Set Alarm** action. As **Number of steps** again indicate 50.

We set the alarm clock to 50 steps but you might wonder what a step is. Default *Game Maker* takes 30 steps per second. So 50 steps is slightly more than 1.5 seconds. (You can change the game speed in the **settings** tab in the room properties.)

Finally, each game must tell the players what the goal is and how the user plays the game. So some help is required. *Game Maker* has a standard mechanism for this.

Adding a help text:

1. From the **Resources** menu select **Change Game Information**. A simple text editor will appear.

2. Type in some useful information for the player, in particular about the goal of the game and the way to control it. You can use different fonts, sizes, and colors. See for example Figure 20.

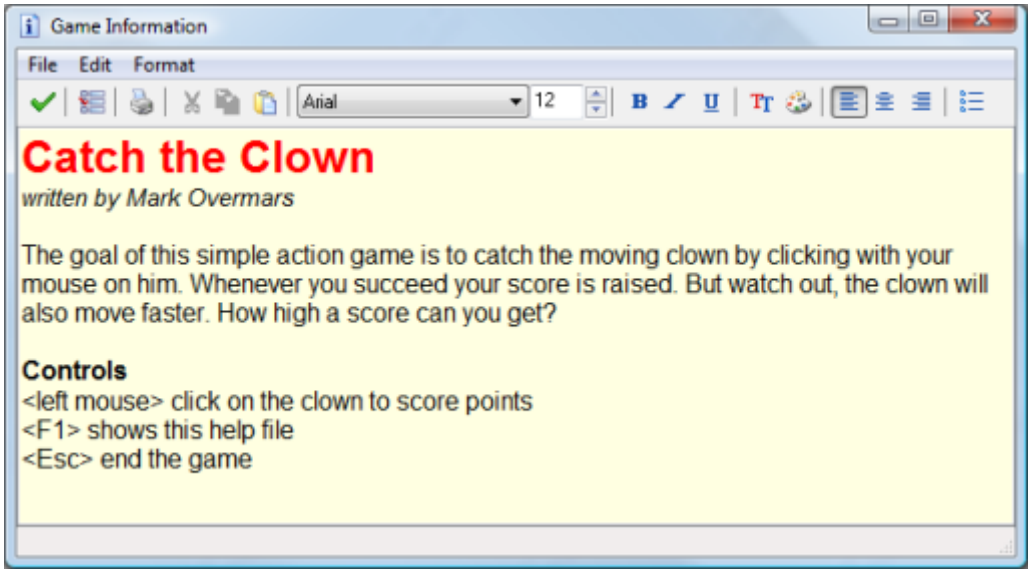

Figure 20. Adding some help for the player.

During the game this text is automatically shown when the player presses the F1 key (like in most other programs).

Your First Game is Ready

Congratulations. You finished your first game. And the first game is always the most difficult one. You also learned about the most important aspects of *Game Maker*: sprites, images and sounds, the game objects, events and actions, and the rooms.

Before continuing with a new game you might want to play a bit more with the *Catch the Clown* game. Here are some things you might want to try to add:

- Have two clowns moving around. (This is extremely easy because you can place multiple instances of the same object in a room.)
- Have a different dark clown that you should not catch because it will cost you part of your score.

But you surely can come up with other creative ideas.

In this tutorial we have only covered some of the most basic aspects of *Game Maker*. We discussed only a few events and actions. There are many more for you to explore. You can try to do so yourself or you can download and read one of the other tutorials which you can download from

http://www.yoyogames.com

Further Reading

For further reading on creating games using *Game Maker* you are recommended to buy our book:

> Jacob Habgood and Mark Overmars, *The Game Maker's Apprentice: Game Development for Beginners*, Apress, 2006, ISBN 1-59059-615-3.

The book gives a step-by-step introduction into many aspects of *Game Maker* and in the process you create nine beautiful games that are also fun to play.