

Lecture 1 - Getting to know Game Maker

Written by Carl Gustafsson

Goal of the lecture

The goal of this lecture is that the reader should be introduced to the program Game Maker. The reader should understand its basic building blocks and know how to add resources (e.g. images and sounds) to a game.

This lecture was revised for round 2 of the Game Maker programming course at www.gameuniv.net. Changes to the original document are shown with **slightly greenish background**. If you read this document for the first time, just ignore those markings and read it as if nothing was marked.

This lecture was also revised for round 3 of the Game Maker programming course at www.gameuniv.net. Changes to the previous revision of the document are shown with **slightly blueish background**. People reading this document for the first time could ignore the different background colors.

Introduction

Hi and welcome to this course in Game Maker programming!

The name of your teacher (that would be me) is Carl Gustafsson. Let me just tell you a little about who I am.

I am 30 years old and live in Sweden (somewhere in the northern Europe). I have been using Game Maker since late 2001. Since I have not attained any courses or so, all I have learned is from my own experience and from tutorials and examples.

I would like to remind you that English is not my native language, and therefore you might find strange words and constructs. Should you find anything that I have explained badly or in a too confusing way, please tell me and I will try to correct it.

When writing programs and games there are always multiple ways of solving a task. The solutions appearing in this course represent the way **I** prefer. There might very well be better ways, but it is up to you, the student, to find the way of your own preference. My "taste" of programming will probably show in the sense that I prefer writing code as text as opposed to using the drag-and-drop programming feature of Game Maker.

If you need to contact me, use either my [email at gameuniv.net](mailto:email@gameuniv.net) or use the course forum. If you have any questions, first try the forum and see if there is anyone else who can help you. I am afraid I can not put as much time as I would want into this, and therefore you might probably be helped faster if you use the forum.

About Game Maker

Game Maker is a program made for creating computer games. It was (and still is) written by Mark Overmars, a professor at the Institute of Information and Computing Sciences at Utrecht University. Game Maker is free in the sense that you are not required to pay anything for using it. **HOWEVER!** You are **strongly encouraged** to register your copy of it and pay a small registration fee (at the time of writing: \$15) in order to follow this course. This is in order to ensure the future development of Game Maker and to encourage Mark to continue his work on it. Also, there are parts of Game Maker that are available only to registered users, so registration is really a good thing to do.

To obtain Game Maker, please download it from its official homepage, www.gamemaker.nl. At the time of writing, the latest version is 6.0.

Also download the [manual](#), since you very much need it to complete the course.

The purpose (as I understand it) of Game Maker is to provide a program for writing two-dimensional games on a standard PC. Game Maker works on most Windows operating systems since Windows 98, except possibly Windows NT 4.0 (not sure). A note to be made here is that it IS possible to create three-dimensional games with Game Maker, but the 3D functions in Game Maker takes a bit getting used to, and they are quite new in the program. This course will probably not cover the 3D routines, at least not as it looks today.

What I find being the most remarkable about Game Maker is its ability to both provide a games creation environment that is easy to get into for the beginner, and, at the same time, providing enough power and versatility to create really good-looking and advanced games. This makes it interesting to a wide range of people of all ages and all kinds of educational levels. This is made possible through the user interface of the development environment, which includes both a drag-and-drop possibility, allowing the user to create a game with a minimum of keyboard input, as well as a text coding possibility, allowing the more experienced user to get into the deeper parts of the program and create more advanced "scripts".

Game Maker is a so-called sprite-based and event-driven games creator. It includes collision detection for sprites and event detection for all player events I can come up with (e.g. keyboard, mouse and joystick input).

The next chapters will be a tour of all the different entities of Game Maker. Some of them, the ones that I reckon are the most important ones, will be mentioned in more detail, while the ones I have not used that much (yet) is only touched on the surface. They will be delved into later on in the course.

Starting up Game Maker

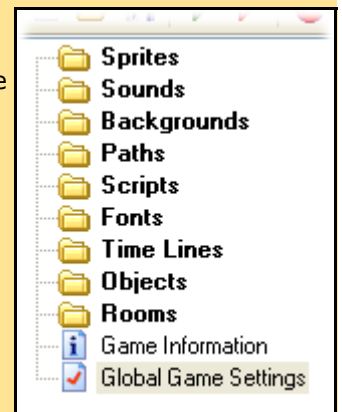
I assume that you have downloaded Game Maker from the site mentioned above (www.gamemaker.nl). If not, do so now. After downloading, you also need to install it. The installation is similar to other application installation programs.

The first time you run Game Maker, you are asked whether you want to run it in "**Simple**" or "**Advanced**" mode. Choose "**Advanced**" here. The difference is that some of the features of Game Maker are disabled in "**Simple**" mode, just so that beginners are not too confused. But since we want to go through all parts of Game Maker in this course, please choose "**Advanced**". Should you make any mistake, or if you already have installed it and selected "**Simple**" mode, it is easy to change the mode. Just select in the menu **File -> Advanced Mode**. There should be a little checkmark next to "**Advanced Mode**" in the menu when "**Advanced**" mode is active.

The entities of Game Maker

These are the entities of Game Maker. (Entity, what is that? Well, call it "thing", "stuff", "thingamabob", whatever... these are the "things" that games made with Game Maker are made of. Ah, look up "entity" in the dictionary :)). They can also be called "Resources".

If you start up Game Maker, you will see a list of all the different entity types on the left side of the Game Maker window. The entities are placed in a tree-like structure with "folders" that can be opened and closed, just like your common file explorer window. This part of the screen is called the Resource Explorer. The different entities are Sprites, Sounds, Backgrounds, Paths, Scripts, Fonts, Time Lines, Objects and Rooms. The additional two things listed in this view are the Game Information and the Game Options. We will take a look at those later.




Sprites

A sprite is basically an image or a series of images - an animation. It is called a sprite because, in addition to data about an image (a bitmap), it contains additional properties about e.g. width, height, transparency, bounding box and collision detection settings.

Sprites can be added to the game making environment either by loading image files, or by creating the sprite images using the built-in image editor. We will go through the steps to add a simple little sprite here.

First, add a new sprite entity to the game. This can be done in one of these ways:

- Click the **Add Sprite** button: 
- Choose the menu alternative: **Add->Add Sprite**
- Press **CTRL-ALT-S** on the keyboard
- **Right-click** on the **Sprite** folder in the Resource Explorer and choose "**Add Sprite**".

Whichever way you chose, you should now have a new and empty sprite entity added to the game. Game Maker also displays to you the Sprite Properties window. The Sprite Properties window shows all the data available for this sprite. The first thing I usually do is to give the sprite a meaningful name. The name automatically given by Game Maker is "**sprite0**", which does not tell you anything about what the sprite contains. It is really helpful later on if all the sprites have names that immediately tell you what the sprite

depicts, instead of you having to look at the images of each sprite to determine which one it was you were going to use.

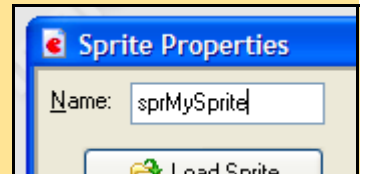
The name of the sprite can be changed through changing the contents of the textbox cleverly labeled "Name". Change the name to something better, e.g. "**sprMySprite**" (without the quotation marks). It might seem strange to you that I try to be this detailed about how everything is done. The reason for that is that I want everyone to be able to follow, even though you may not have seen Game Maker before, or maybe you are not that used to using computers. I promise I will not be as detailed about this in the following lectures.

Note that I chose to start the sprite name with the prefix "**spr**". That is a way of naming things that I recommend that you use in Game Maker. If you name every sprite **sprXXXX** you will not have any problem separating them from e.g. the sounds, which I call **sndXXXX**. Imagine for example that you have a sprite called "Bullet" and a sound called "Bullet". Now, when programming, which is which? It is much better to have a totally unique name to each entity. You will understand how I want you to name the entities later on.

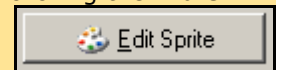


Another thing about names on entities. **Only** use normal characters like letters, numbers and underscores when defining a name. **Never** use spaces or weird characters like slashes, asterisks, pound signs, dollar signs, ampersands, and similar. Don't use language-specific characters such as the "å", "ä" and "ö", which we have in Sweden.


Also, don't **start** with a number, use it only later in the name.




Now that the sprite has a name, it is time to give it an image. We now have two choices; either load an existing image from a file, or create an image ourselves using the built-in image editor. Loading an image is just a matter of clicking the "**Load Sprite**" button, but creating our own image means clicking the "**Edit Sprite**" button. For this lesson, we could try and make our own image for the sprite. So, go ahead and click the "**Edit Sprite**" button.

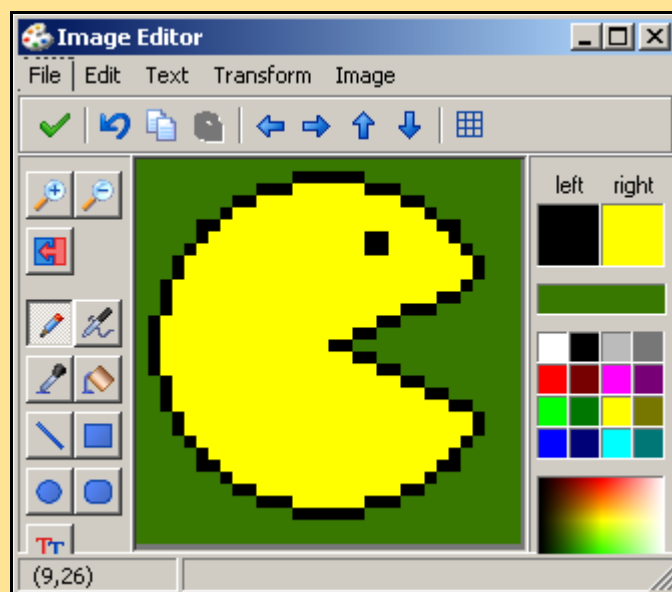


This brings us to the **Sprite Editor** window. Here we see one green, tinted image, and some tools on the toolbar. The green image, called "image 0" is tinted just because it is the currently selected image. Try clicking somewhere beside the image, in the empty area, in order to deselect it. The image will now clearly be green. In order to edit the image itself, select it again by clicking on it. We now have no less than four (4) ways of entering the image editor for this image. So, do **one** of the following:

- Click the "**Edit the Image**" button 
- Choose the menu alternative: **Edit->Edit ...**
- Press **CTRL-E** on the keyboard
- **Double-click image 0** in the image list.

Finally we have found the **Image Editor** window. This is where we can really edit the image. In order to make it a bit easier, click on the magnifying glass a few times so that the view of the image is larger. 

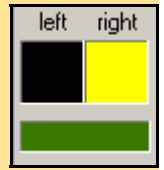
This is what I drew, using the circle tool, the line tool, the pen tool and the fill tool of the Image Editor.



I do not think I need to explain to you how to use the Image Editor. It is very similar to other image drawing tools. It contains the basic tools needed to make images. If you need to make more advanced art,

you probably need an external image editor, e.g. [Gimp](#), [Paintshop Pro](#), Photoshop or similar. If you want to, you can select another image editor as the default image editor in Game Maker. That is done in the Preferences (**File -> Preferences**, select the "Image Editor" tab).

Pay attention to the green bar just below the "Left" and "Right" colors. This bar tells you which color in the image that is transparent. That is right. If the **transparent** property of the sprite is checked (which it is by default), one of the colors of the image will be transparent. Which color? The color that is made transparent is determined by the color of the lowest, leftmost color in the image. That color is currently green, which is why the green color surrounding the pacman will be transparent when viewed in the game. Any color can be used for this. But why does one need transparency? Because otherwise all sprites would be rectangular. The pacman depicted above would have to move around with a green rectangle around it all the time. Not that pretty. So, one color is made transparent, and the sprite will then show up all the other colors just as needed.



Click on the green checkmark (**OK**) when you are done editing the image. You are now returned to the **Sprite Editor**. If you here check the checkmark "**Show Preview**", you will see a preview of how the sprite will look with transparency turned on. This preview will also show the animation of the sprite, if more than one image is added to the sprite. We will see this later in the course, though, if you feel like experimenting, try adding another image to the sprite (Hint: **Edit->Add Empty**).

Click once again on the green checkmark when you are done in the Sprite Editor. We have now made a first sprite for the game. The sprites are the basic building blocks of everything that is moving inside a game.

As I said earlier, in addition to editing our own images in this way, Game Maker is able to load an image from a file. All kinds of formats I can come up with are supported, including GIF animations, which turn into animated sprites.


Now, I suggest that you save the game file. Perhaps it does not look like anything much, and perhaps you do not think that you will have any use for the file later on, but I have grown the habit of saving as much as possible. So, go to the menu and choose **File->Save**. I chose to save my file as "**Lecture1**", you could choose any name you please.

Sounds

In addition to the graphics of a game, the sound effects and the music are important. Therefore there is an entity in Game Maker called "**Sounds**". Sounds can only be imported to Game Maker from other files. There is no built-in "sound editor" in Game Maker, so, in order to edit a sound, you will have to select an external sound editor. This can be done in the Preferences (File -> Preferences, select the "Sound Editor" tab).

There are two main uses of sounds. Either the sounds are used as sound effects, e.g. shooting, hitting, exploding, or they are used as background music to the game. Sound effects are loaded from "WAV" files, and music is loaded from "MID" (MIDI) files. There is also a possibility to use "MP3" files as background music.

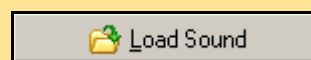
It is now time to add a sound, and as usual there are multiple ways of doing it. Do one of these:

- Click the "**Add a Sound**" button: 
- Choose the menu alternative: **Add -> Add Sound**
- Press **CTRL-ALT-U** on the keyboard
- **Right-click** on the **Sound** folder in the Resource Explorer and choose "**Add Sound**".

Now there should be a "**Sound Properties**" window with an empty **Sound** entity in the game. Just as with sprites, a sound is given a default name, like "**sound0**" or "**sound1**". Change it so that it depicts the sound it contains. I thought we were going to add an applause sound effect, so let us call the sound entity "**sndApplause**".




To load a sound file, click the button "**Load Sound**":



This opens a file selector dialog. Here we will try to find the Game Maker sound resources directory. It will be found in the directory where Game Maker was installed. Usually this is in **C:\Program Files\Game_Maker5\Sounds**. Find the directory and select the sound file "**applause.wav**" here and open it.


Note: If no resources are installed with Game Maker, they can be downloaded from [Game Maker's official web page](#).

Once the file is selected, it can be played in the "**Sound Properties**" window through clicking the "**Play**" button. Do that now. If your speakers are connected and turned on, you should here an applause sound. There are a few other settings here, but we will leave them for now. Click the "**OK**" button to store the changes made. 

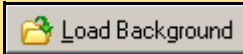
Backgrounds

The Backgrounds entity in Game Maker is used for the backdrops of the game. A background is basically a nice image that just sits there as the background for the rest of the game. The background can be made to move, resulting in a so-called scrolling background. Finally, a background can also be constructed from tiles. That is very useful if the background is supposed to be very large. It is possible to use multiple, partly transparent backgrounds on the same screen, and moving them with different speeds and thus creating some sort of parallax effect, giving a bit of depth to the game world. This will be utilised in a later lecture where we will create a background of moving stars.

Let us add a new background, in one of these ways:

- Click on the "**Add a Background**" button: 
- Choose the menu alternative: **Add -> Add Background**
- Press **CTRL-ALT-B** on the keyboard
- **Right-click** on the **Background folder** in the Resource Explorer and choose "**Add Background**".

Now we have a new Background entity, shown in the "Background Properties" window. A background is partially similar to a sprite in that there is a possibility to either load an image, or edit your own image using the image editor. We will here load in an image from a file.

Click the "Load Background" button: 


This brings up a file selector dialog. Once again, find the directory where Game Maker was installed (usually C:\Program Files\Game_Maker5). Also, go into the "Backgrounds" directory there. Here there should be a file called "sky.gif". Select it and open it.

Call the background "bgrSky". That is enough with backgrounds. There is not much more that can be done here. Click the "OK" button to accept the changes made to this Background entity.

Paths

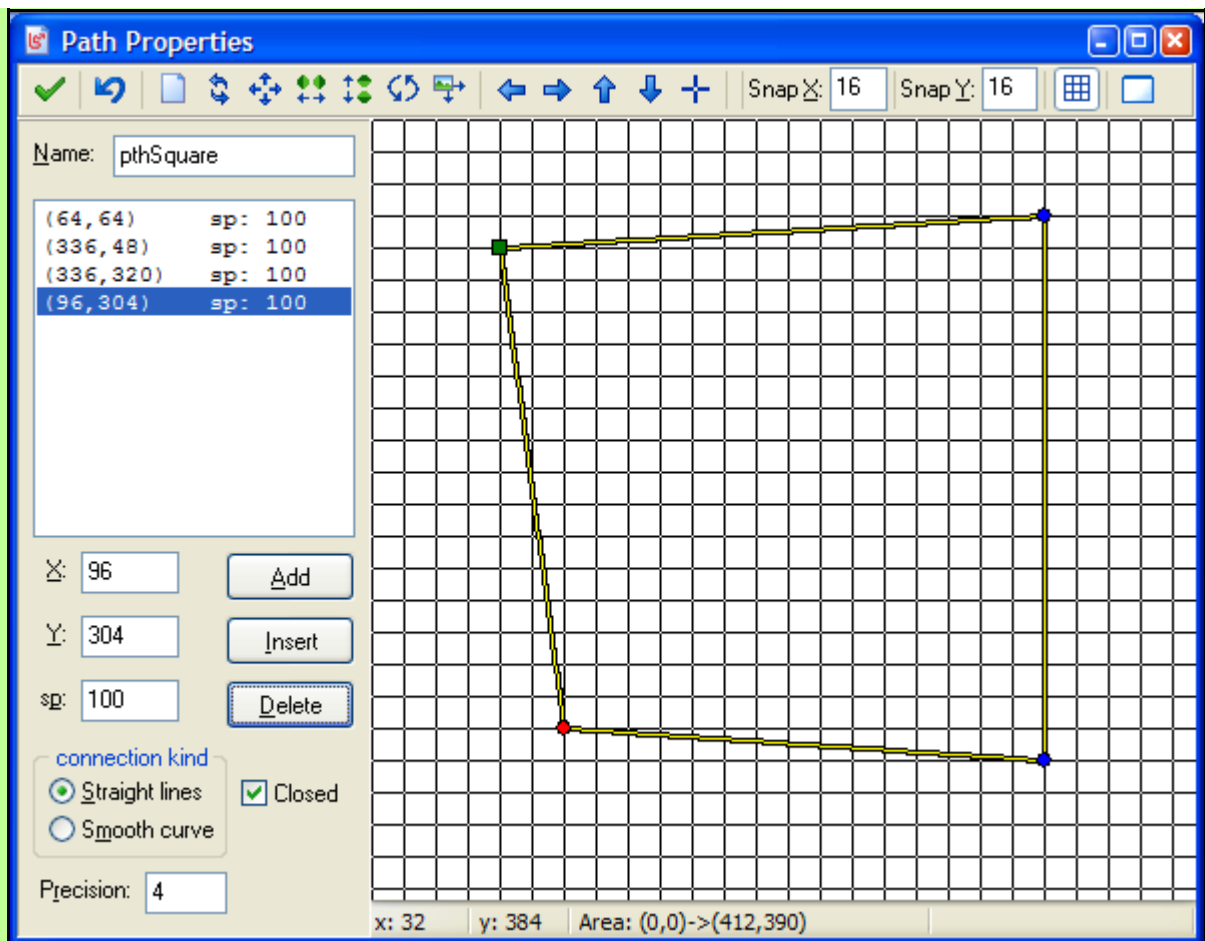
Paths are like tracks that a game object can follow. It defines how the object moves. Useful for example for patrol patterns for sentries, or attack patterns for spaceships. A path consists of control points that are bound together with either straight lines or so-called "Bezier curves".

Since paths is a quite new feature to Game Maker, I have personally not used it much yet. Therefore I will only touch it slightly here. Perhaps we will look deeper into the subject in a later lecture. Let us add a new path to the game. Do one of the following:

- Click on the "**Add a Path**" button: 
- Choose the menu alternative: **Add -> Add Path**
- Press **CTRL-ALT-P** on the keyboard
- **Right-click** on the **Path** folder in the Resource Explorer and choose "**Add Path**".

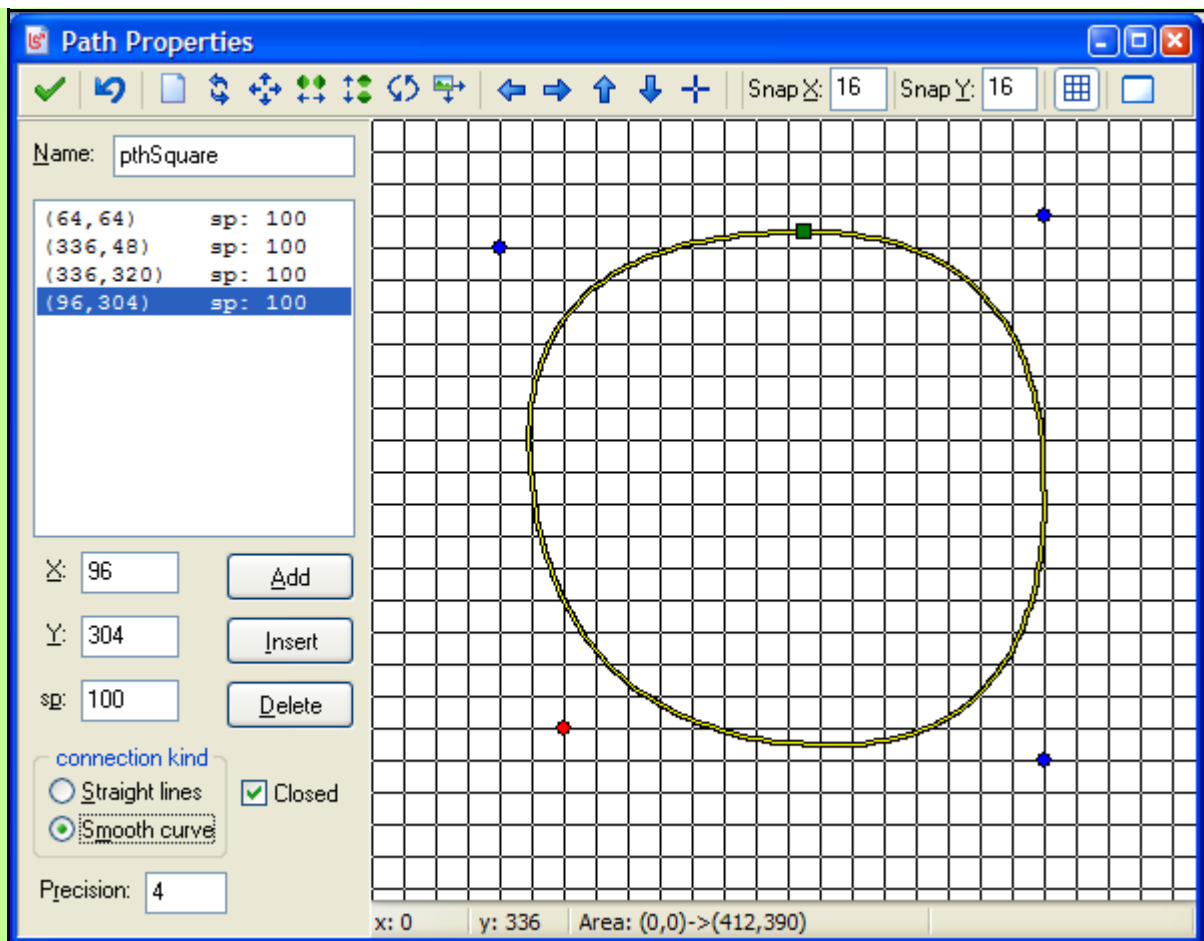
A new path should now have been added, and the "**Path Properties**" window shows. Immediately change the path name to "**pthSquare**".

It is now time to add some control points to the path. Try adding them in a square-like pattern, something like this:



Points are added by clicking in the white area on the right hand of the Path Properties window. As you can see, it is not important to put the points exactly right in this lecture. It is just to get the feeling of creating things in Game Maker. If you think a point is completely out of line, you can just click and drag it to move it to the correct position. Lines that connect the control points are automatically added by Game Maker.

If you now select the "Smooth Curve" instead of "Straight lines" in the "connection kind" panel, you will see that the path is changed into a not-completely-unlike-a-circle pattern.




The green square defines the starting position of the path.

This is all we need to know about paths so far.

Scripts

A script in Game Maker is a piece of program code. You can say that there are two ways of programming in Game Maker; the drag-and-drop approach, and the script coding approach. As you will see later, the script coding approach is the one I personally prefer. A script is a self-contained, free collection of programmatical statements that can be executed from any game object or any script, including itself, which brings forth the concept of recursion. Sorry, got a little ahead of myself there. We will start using scripts in lecture 5.

Just for the sake of doing it, add a script to the game in either of these ways:

- Click on the "Add a Script" button: 
- Choose the menu alternative: **Add -> Add Script**
- Press **CTRL-ALT-C** on the keyboard
- **Right-click** on the **Script** folder in the Resource Explorer and choose "**Add Script**".

This brings up the "**Script Properties**" window, which, practically speaking, is a little text editor. Here it is possible to write the scripts that later are used by the game objects. Give the script a name, just like all other entities, but this time we skip the prefix. Call the script "**CreateEnemy**". The reason for skipping any prefix (like e.g. "scrCreateEnemy") is that names of scripts tend to be like verbs, in contrary to all other entities, whose names look more like nouns. Therefore it is so unlikely that a script gets the same name as e.g. a sprite that I choose to not use a prefix for the scripts. You can of course do as you please with this and develop your own taste.


If you like, you could try to write something in the script editor. It does not matter what, as long as it does not contain foul language, in which case Game Maker will instantaneously uninstall from your system (Yes, that is right, that WAS a joke! ;)).

We will be using scripts a lot later on, so do not worry.

Fonts

The Fonts entity is new to Game Maker 6.0. It can be used if you want to include fonts in the game. This is necessary if you want to use the text drawing functions of Game Maker. In previous versions of Game Maker, fonts could be added as "Data Files".

Add a new Font entity to the game by doing one of these actions:

- Click the "**Add a Font**" button: 
- Choose the menu alternative: **Add -> Add Font**
- Press **CTRL-ALT-D** on the keyboard
- **Right-click** on the **Fonts** folder in the Resource Explorer and Choose "**Add Font**".

We will add the Arial font to this font entity, so name the entity "**fntArial**". Then select the **Arial** font in the font listbox. You can leave the font size at 12 points. Then it is possible to determine which characters of the font that should be added. The numbers are from the ASCII code table (www.asciitable.com). If you click on the "Normal" button, all the "normal" letters and numbers (a-z and 0-9).


Then click **OK**.

Now the Arial font is copied to the game and ready to be used for drawing text on the screen.

Time Lines

If you want an object to perform certain actions at certain specified times in the game, **Time Lines** is a great property to use. In a **Time Line**, certain times are specified and actions are added to those times. This, too, is a new feature of Game Maker, but I have some theories that it can be used for AI constructs, though I have not tried that yet.

We will now add a Time Line to our game. This is done in one of these ways:

- Click the **Add a Time Line** button: 
- Choose the menu alternative: **Add -> Add Time Line**
- Press **CTRL-ALT-T** on the keyboard
- **Right-click** on the **Time Line** folder in the Resource Explorer and choose "**Add Time Line**".

A time line has now been added to our game, and the window "**Time Line Properties**" shows. Change the name of the Time Line to "**timStart**".

In the first white space, called "**Moments**" it is possible to add points in time ("moments") when something should happen. Add a new moment by clicking on the "**Add**" button. When adding a moment, the time, at which the moment should occur, needs to be entered in a small pop-up window. Enter "**50**" there and click "**OK**". This means that a new moment has been added, that will occur at time **50**.

Perhaps it would now be a good time to mention a bit about timing in Game Maker. In Game Maker, as in most games I know of, time is split up into small chunks, called "**steps**". In each step, all game objects are given the possibility to respond to events that have occurred in the game, and at the end of each step the screen is redrawn, displaying the new status of each game object. It is possible to change the speed at which all this happens. The default speed is **30 steps per second**, which is also known as **FPS, Frames Per Second**.

So, since the speed setting has not been altered yet in our game, the moment 50, that we entered in our time line, will occur at about $50 / 30 = 1.67$ seconds after game start.

Once a moment is defined, it is possible to add actions to it. This is done by dragging actions from the action collection to the far right in the Time Line Properties window and into the Action list just left of the collection. Since this is about the same as adding actions to objects, which will be looked into very soon, the concept is not delved into more here.

The buttons to the left of the Moments list are used to **Add, Change** or **Delete** a moment. It is also possible to **Shift** a range of moments, which means moving all the moments in the range a certain number of steps, it is possible to **Merge** two moments, and, finally, to **Clear** the entire Moments list.

Objects


The objects are the single most important entity in Game Maker. These are the "living", acting objects in the game. An object is given a behaviour through defining which events it should react to, and how it should react. An object is usually represented by a sprite on the screen. There are other ways of displaying an object, but using a sprite is the fastest and easiest way.

The events that an object can react to can be e.g. input events (keyboard, joystick, mouse), collision events (collision with other objects), alarm events (timers) and a few other event types.

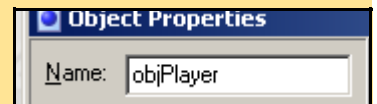
When an event occurs, a certain number of actions are taken. These actions are chosen from an action list. The actions can be e.g. movement, creation of other object instances, playing a sound, changing the sprite, etc.

It is the objects that "make" the game. Without objects, the game would be pretty much lifeless. Now, since objects is the most complex type of entity, they may be the hardest one to grasp. So, we will try to take it all easy.

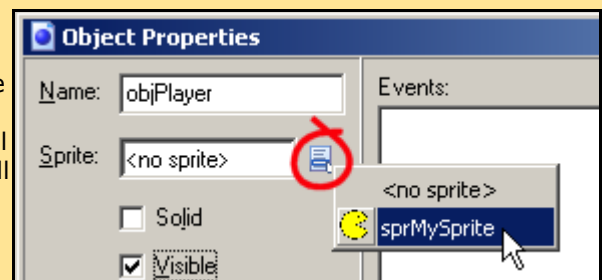
Add a new object through one of the following actions:

- Click the **Add an Object** button: 
- Choose the menu alternative: **Add -> Add Object**
- Press **CTRL-ALT-O** on the keyboard
- **Right-click** on the **Objects** folder in the Resource Explorer and choose "**Add Object**".

Now we have a new object and the "**Object Properties**" window shows. Immediately name the object. My suggestion for a name is "objPlayer". This name will be used later on when referencing the object, so it is best to choose a name that is as descriptive as possible. Also, note the prefix added even here, "obj", before the actual name of the object. This is, as before, because we later need to be able to determine from the name what kind of entity is referenced.



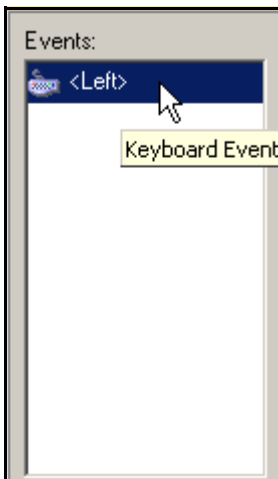
Once the name for the object is set, it is time to assign a sprite to it. This is the normal case; that an object is represented by a sprite. In order to select a sprite, click the little selector button to the right of the "**Sprite**" textbox on the "**Object Properties**" window. This brings up a list of all the sprites that are available in the game along with a small picture of the sprite. So far we have only added one sprite to the game; "**sprMySprite**". Select the sprite in the list to assign it to the object.



Another thing that ought to be checked is that the "**Visible**" checkbox has a checkmark in it. If not, the object will not be visible in the game. The other settings in the leftmost part of the "**Object Properties**" window can be left for now. They have a bit more advanced uses.

Now, the next two parts of the "**Object Properties**" window are the "**Events**" list and the "**Actions**" list. For each event that is added in the "**Events**" list, there is a list of actions ("Actions" list) that are executed every time the event occurs. In this way the so-called "behaviour" of the object is defined.

So, in order for our object to have some kind of behaviour, let us add an event that the object should "listen" to. Click the "**Add**" button below the "**Events**" list. This brings up the "**Event Selector**". The "**Event Selector**" contains each and every event that an object can be set to respond to. Now, say that we want our object to respond to the user pressing down the **LEFT** cursor key on the keyboard. In the "**Event Selector**", click on the "**Keyboard**" button. This brings up a list of all keyboard events. Select **<LEFT>** from this list.



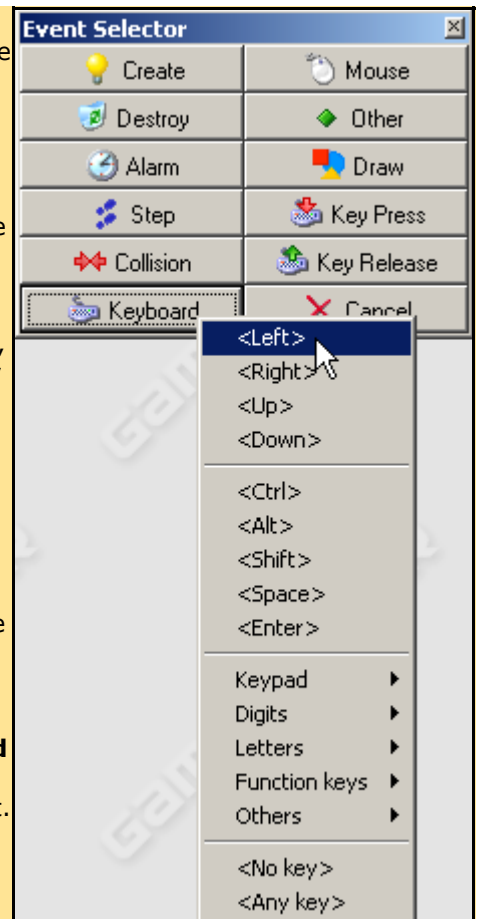
A **<Left>** key event appears in the event list. This means that the object now responds to the left cursor key being pressed. There are more than one type of keyboard events. If you look in the event selector, you will see that there is one category that is called "**Keyboard**", one that is called "**Key Press**" and one that is called "**Key Release**". They are different in the sense that "**Keyboard**" events will react as long as a key is **being held down** on the keyboard, "**Key Press**" will react only **once every time a key is actually pressed down**, and "**Key Release**" will react only **once every time a key is released**.

For this lecture we will use the "**Keyboard**"

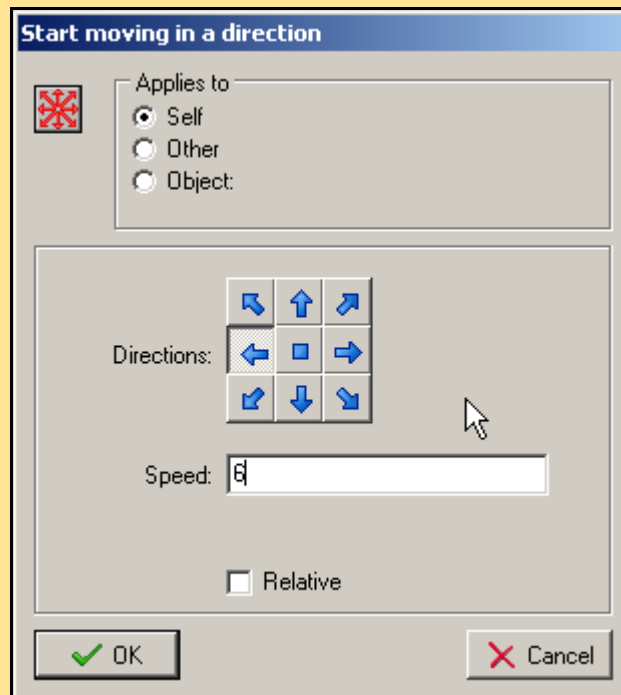
category of events.

So, as long as the **LEFT** cursor key on the keyboard is being held down, the actions in the action list on the right will be executed. In order for anything to happen we thus have to add some actions to the list. The actions that are available are placed under **seven** tabs along the right edge of the "**Object Properties**" window.

In order to add an action to the list, the action is clicked and **dragged** from the actions panel and onto the actions list. The most obvious action for a **LEFT** cursor key is that the object should move to the left. Select the first tab, labeled "**move**". In this tab there is an action called "**Start moving in a direction**". It is represented by an icon with eight arrows pointing in different directions:



Drag that icon from the actions panel and onto the actions list, just left of the panel. This should add a "**Start moving in a direction**" action to the actions list. In addition, a new window is opened where different settings can be made for the action that is added. Here, enter a number in the "**Speed**" setting, e.g. **6**, and select the arrow pointing to the **left**, so that it looks like this:



The "**Relative**" box is used if the entered value (6) should be **added** to the existing value. This would here cause some kind of acceleration, which is not what we want, so leave it unchecked.

Click the "**OK**" button. Now we have added an action, "**Start moving in a direction**" and set the direction to **left** and speed to **6** in our action list for the event **LEFT** cursor key. If you want to alter the settings for the action that we have already added, **double-click** on the action in the action list to once again view the "**Start moving in a direction**" settings window.

We will not do any more settings for this object for now, so click the "**OK**" button in the "**Object Properties**" window too, in order to accept the changes made and close the object.


Here I would like to mention one important aspect of Game Maker and the object-oriented approach. The objects that we define in this way could be looked upon as blue-prints or casts for the actual thing that later appears in the game. When an object is supposed to appear in the game, the object is "**instantiated**". This means that a dynamic copy of the object is put into the game. This copy is then called an **instance** of the original object. It is possible to have any number of instances copied from the same object in the game (limited by memory and processing power of the computer of course). It is important to distinguish between an **object** and an **instance of an object**, especially when there are multiple instances of the same object in the game. (People used to the object-orientation implemented in C++ will know this as the distinction between a class and its instantiated object).

We will **instantiate** the object in a moment, first we just need to add a place for it to be instantiated in, namely a **room**.

Rooms

Rooms is the last entity type in Game Maker (phew, it surely has taken time explaining the previous ones ;)). A room can be looked upon as a place for the instances (instantiated objects) to act. For some games, a room can also be called a level, or a screen. When playing the game, one room at a time is displayed on the computer screen. It is possible to change rooms at will, but only one room can be displayed at any one time.

To add a new room, do one of these:

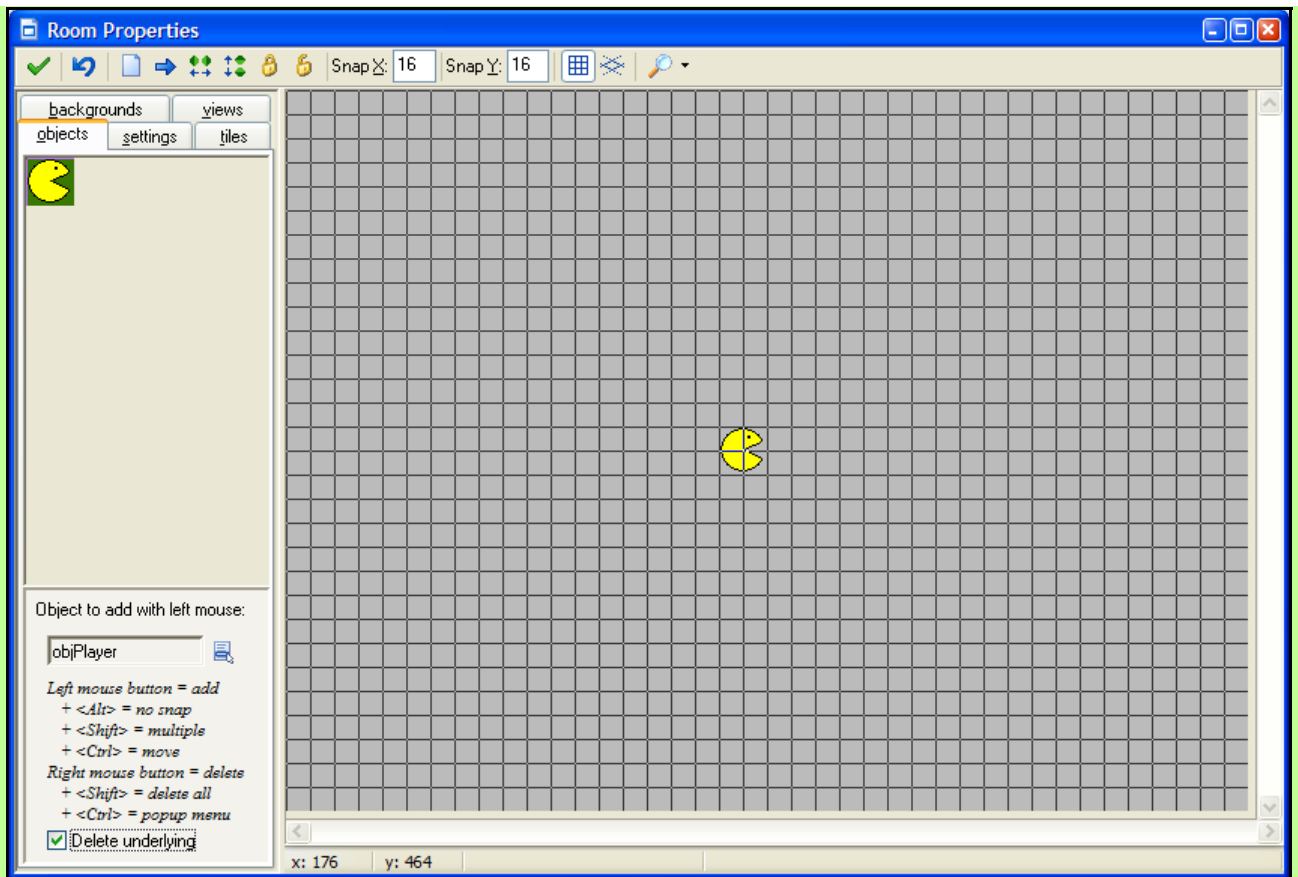
- Click the "**Add a Room**" button: 
- Choose the menu alternative: **Add -> Add Room**
- Press **CTRL-ALT-R** on the keyboard
- **Right-click** on the **Rooms** folder in the Resource Explorer and choose "**Add Room**".

A new room appears in the game and the "**Room Properties**" window shows. This view displays the room and all instances that are placed in it. At first it is, of course, empty. It just looks like a large grey rectangle with a black grid on top. The grid is only there to assist in the placement of instances. It will not be visible later, when the game is run.

Start with naming the room. This is done through selecting the tab "**settings**" in the "**Room Properties**" window. Name the room "**FirstRoom**". Below the name of the room, a caption for the room can be entered. The caption will be shown in the window border, just like most other Windows windows (hehe). Enter "**My First Game**" for caption.

Next, click on the "**objects**" tab. Now it is time to add an **instance** of the "**objPlayer**" object to the room. Click the list selector button to the right of the textbox that now says "**<delete>**". This opens the object selector. Select the only object available, "**objPlayer**". You will see an image of the object above the textbox. Now, **left-click** somewhere in the room view to place an instance of the object there. It is not important exactly where you click, just as long as it is somewhere near the middle of the room. If you think that you have made a mistake and want to delete an instance, just right-click on it in the room.

When you are done, it should look something like this:



Once the game starts, this room will be displayed, and the instance of the object **"objPlayer"** is displayed in the middle of the room.

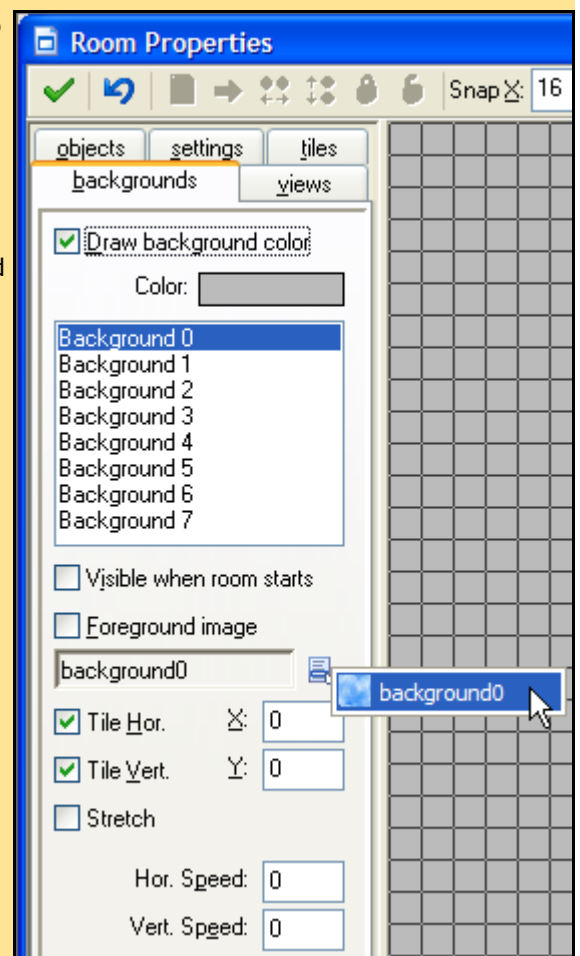
Click the **"backgrounds"** tab. Time to add our background to the game. Make sure that **"Background 0"** is selected in the backgrounds list, then click the list selector to the right of **"<no image>"**. This opens the background selector list. Select the background we loaded earlier.

This will display that background image as the background of the room. Note that it does not only display the small portion of clouds that we loaded into the background, but it is instead **"tiling"** the image so that it covers the entire room. This is because the **"Tile Hor."** and **"Tile Vert."** checkboxes are marked. Try unmarking them to see the difference. Mark them again so that the whole room is filled with clouds. The **"Visible when room starts"** checkbox should also be marked.

The list of backgrounds in the **"Room Properties"** window depicts the eight backgrounds (0 - 7) that are possible to assign to a room. How come then that you can have multiple backgrounds in the same room? That is because at times it could be convenient to use partially transparent backgrounds in multiple **"layers"** for some effects. This will be useful later on.


Click **"OK"** to store the changes to the room.

Now we have had a look at all entities of Game Maker. We have added a sprite, a sound, a background, a path, a script, a **font**, a time line, an object and a room. It is now time to save the game. Actually, I really would like to stress that it is important to save your game file often! At least for me it happens a lot that I make some kind of mistake that is undoable and I have to go back to the last saved copy of the game. Then I feel really bad if the last copy of the game is 5 hours old.



Remember: Save often! This goes for anything you do on the computer.

Running the game

It is now time for the big moment - running the game to see if it all works! Remember: To quit the game once it is running, press the **ESC** key. To run the game, press **F5** (or click the "**Run Game**" button ()).

You should now see a cloudy sky and a pacman (or whatever you drew as a sprite) in the middle of it all. Nothing else happens - because nothing else is **supposed** to happen.

Press **LEFT** cursor key. The pacman instance now moves to the left and out of the room. Cool! (Or what? ;))

Well, it maybe does not look like much, but we have now taken a brief tour through all things there are in Game Maker. Of course, some of them have only been touched on the surface (especially objects and scripts), but the main thing is there. And a game has been done!

This is the end of the first lecture. It is now time for you to continue on to the assignments and to answer any quiz or survey that I might add to this weeks work.

Good luck!

Carl

Assignments

Due date: Friday, 20 May 2005, 08:00 AM
Maximum grade: 100

Hi and welcome to the first assignment!

What you need to do here is to make the pacman (or whatever sprite you made in lecture 1) move up, right and down too when pressing the respective cursor key.
If you want to go even further, you can make it stop when no key is being pressed.

When you are finished, please zip the GMD file and upload it here. Only the .gmd file is needed. Good luck! Carl

PS. If you need a free zip program, I can recommend either www.filzip.com or www.7-zip.org. Filzip has a better interface, but I think that 7-zip packs better. DS.

Due date: Friday, 20 May 2005, 08:00 AM

Read the following sections of the Game Maker 6.0 manual:

- SO YOU WANT TO CREATE YOUR OWN COMPUTER GAMES
- INSTALLATION
- REGISTRATION
- THE GLOBAL IDEA
- LET US LOOK AT AN EXAMPLE
- THE GLOBAL USER INTERFACE
- DEFINING SPRITES
- SOUNDS AND MUSIC
- BACKGROUNDS
- DEFINING OBJECTS

There is nothing to submit to this assignment... of course.



Regards
Carl