GETTING *The most* OF...

Multimedia Fusion 2

The Application Properties

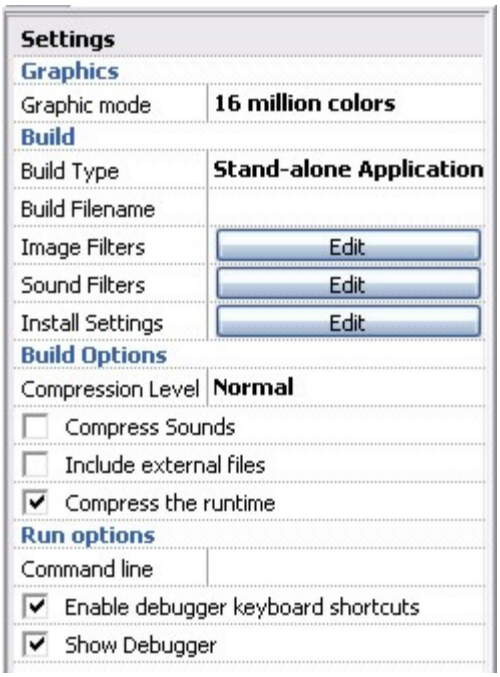
Clickteam has made several advancements with Multimedia Fusion 2 and none are more obvious than how the Properties Toolbar works. In this guide we will explore the various tabs of the Properties Toolbar.

To begin with.

We will explore a few of the application properties tabs (of which there are 6). Each tab contains a collection of functions that are grouped according to their commonality.

The Settings Tab

This property is rather straightforward: it defines the most generalized settings that will effect your entire application.



Graphics Mode.

256 colours.

A rather old-fashioned graphic mode, also called **palettized** mode, the colour of each pixel is stored in one byte. This means a 320x200 pixel sprite will need 64Kb o memory. The colour of each pixel is not directly stored in memory, but as the appropriate number from a list of 256 numbered colours, known as the palette. Under Windows, the first 10 and last 10 colours of the palette should be set to fixed values. (Some graphic programs like Paint Shop Pro allow you to include this system palette in the graphics.)

When you import an image with more than 256 colours into a 256 colour application, or an image that does not include the Windows system palette, Fusion has to **remap** the picture: that is, it must use the nearest match available in the colour palette to the actual colour used in the picture. This is the reason why you might seen some slight modifications in the appearance of an image when you insert it in a 256 colour application.

Multimedia Fusion comes with a default colour palette that has been carefully designed by graphic artists to match all the shade of colours without too many problems. We suggest you use it.

Pros :
- Smaller applications

- Faster display (less bytes to move)

Cons :

- Limited number of colours giving very bad results in gradients or flat areas
- Remapping pictures is tedious
- Not really worth doing given modern machines' capabilities.
- Bad, limited and possibly slow ink effects (fades)

Fusion is not a drawing package : for better results we suggest that you use the remap function of a true paint program, remapping the pictures to the Fusion palette (or the palette you use in your application), with the use of dithering to emulate the colours. You can also set the palette of each individual frame with the frame property: Frame / Properties / Palette. Just prepare a 256 colours BMP file with the palette you want to use, and enter it in this box.

Note for sub-applications: sub-applications are run in the middle of a the host-application frame. The palette of the host frame is enforced for all the frames of the sub-application. Having separate palettes in a sub-application will have no effect at all: all the graphics will be remapped to the host palette. Therefore, if you plan to have sub-applications, create all the graphics with the correct palette to avoid remapping problems and increase the loading speed (since, of course, when Fusion detects that the palettes are the same, the remapping process is skipped).

32768 colours.

This is the mode of choice for publishing an application. Each pixel is stored in two bytes: a 320x200 pixel sprite will use 128Kb of memory. This mode is the first **true colour** mode: the actual values of the red, green and blue components of the pixels are stored. Each one of these values is stored in 5 bits, which gives 2^5 possibilities.

Pros :

- True colours
- No need to remap
- Good compromise between size and quality
- Good speed
- The mode used on the majority of display cards

Cons :

- Bigger applications than 256 colours
- Number of colours sometimes limited for gradients

16 million colours.

The mode of choice for editing an application. Each pixel is stored in four bytes: a 320x200 pixel sprite will use192Kb of memory. The number of possible colours is far beyond the possibilities of the human eye to distinguish.

Pros :

- More than enough available colours
- No need to remap

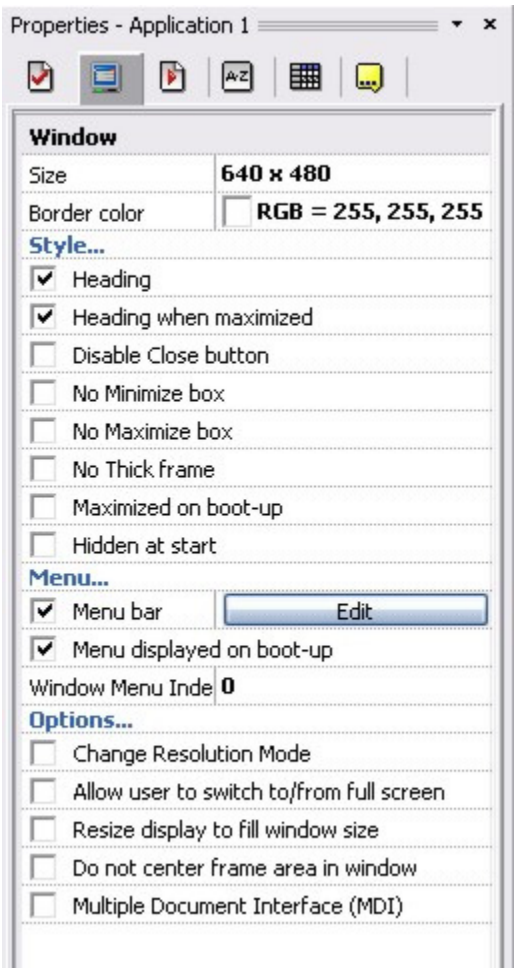
Cons :

- Very big applications
- Slower, three time as much memory to move than with 256 colours

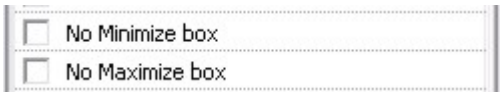
My advice :
- Always develop your application in 16 million colours: this will ensure you work in the best conditions.
- Before publishing or building your stand-alone application, reduce the number of colours to 32768 in the **Graphic Mode** property box. Warning One: this operation cannot be undone, so make sure you work on a copy of your application. Warning Two: if you have a project with multiple applications, you will have to manually reduce the number of colours in each application.
- If your intention is to produce a 256 colour application in the first place, **develop** in 256 colours: this will make sure you actually see the true results as you work (Otherwise, the subsequent remapping could reveal the colours you're currently working in to be deceptive...)

☑ The application window setup.

This tab of the Properties Toolbar contains many check marks, and a few new ones since Multimedia Fusion 1.5, owing to user suggestions. Most of the options are obvious, but some merit further explanations.

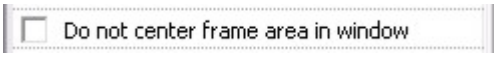


The novelties in Build 92 (and above).
Three new check marks to customize the look of your final application.

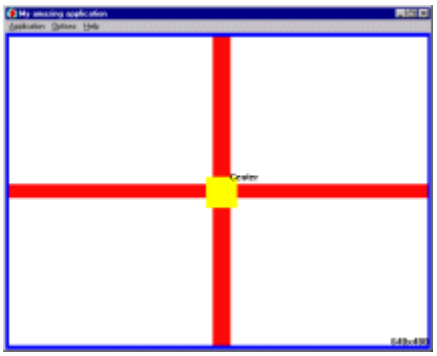


No Minimize box, No Maximize box. When checked, the relevant icons will disappear from the title bar of your application. With them will also disappear the possibility of reducing or zooming the window.

No thick frame. In the Windows world a **thick frame** border is a border that allow the window to be resized. Check this mark and the user will not be able to resize the window of your application.

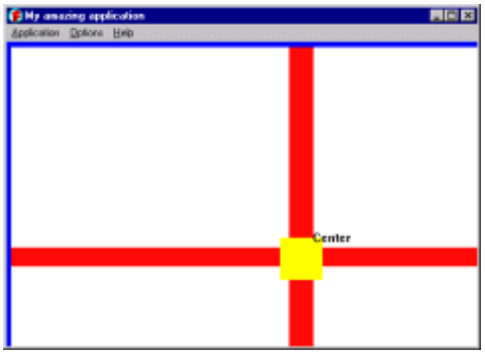


Do not center frame area in window. As a default the frame area (where the graphics are displayed) is always centered in the middle of the border window of the application. This can only be seen if you make the window borders smaller than the frame area. This was the solution we chose for the whole click range, and some users asked us if we could remove this option. Now, if you check this mark, the top left of the frame area will always be displayed in the top left of the window, the remaining graphics being clipped at the top right of the window.



The original 640x480 application running in a 640x480 window

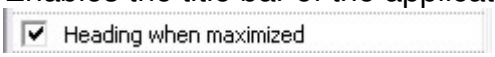
Window reduced, without the new checkmark



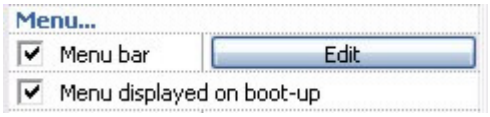
Window reduced, with the new checkmark

The obvious check marks.

Enables the title bar of the application



... ditto



if "**Menu displayed on boot-up**" unchecked, and if "**Menu bar**" is checked, the user will have the possibility of redisplaying the menu at any time in the application by pressing F8

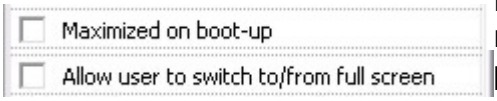
☑ Creating a full-screen application.

There are different ways of creating a full screen application: changing the resolution mode, or zooming the display.

Zooming the display

The simplest, as it does not depend on the machine or on the size of the frame: create your frame in 640x480, and see it run in every resolution. To create a full screen application like this, use the following settings :

Fun with the resize display option.



If you check the "**Maximized on boot-up**" option and leave the possibility of resizing your application to the user, you can end up with tiny applications running in as little as 32x32 pixels! Note that all the zone detection and controls are resized to match what was originally programmed in the application. For example if you program a detection zone from 100 to 200, and reduce the size of the application, the mouse coordinates will be computed to simulate the 100 to 200, and the application will keep on working. (Well, it *should* 8-)

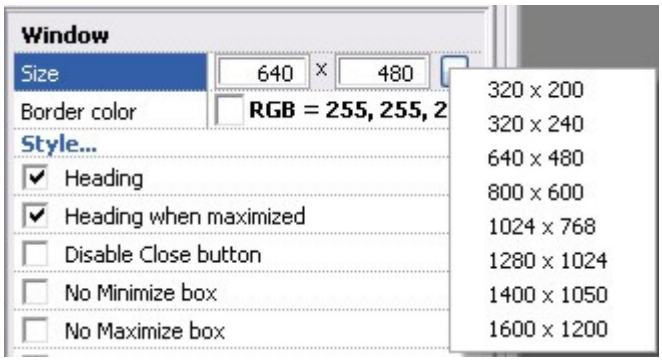
Resizing the display can be very slow if your computer has no hardware

acceleration for zooming. Very..., very..., very... slow. Use it only when you are sure about the machine on which the application is going to be run.

Changing the resolution mode.

This is the method of choice for creating a full screen application, but you have to be careful as it is hardware-dependent. Windows only works in a number of resolutions (find them in the desktop properties box).
- 320 x 200: the smallest and best for creating scrolling games. Hélas! as we say over here, this mode is not supported by every machine. It may work under DirectX6 better, but nothing is really sure in the wonderful world of Microsoft. If Fusion cannot change the graphic mode, it will open a tiny 320x200 window in the middle of the display, or switch to 640x480 with the window centered in the screen (which is better than nothing).
- 640 x 480: this mode works with almost every machine and is a good compromise.
- 800 x 600 , 1024 x 768 and above: will work on every machine. Beware of the monitor limitation: some machines' installations are wrong, and Windows may decide to raise the resolution to a graphic mode not accepted by the user's monitor. In this case the picture will be blurred or black. This ability to change resolution to a higher mode comes in useful for complex applications which have a lot of data to display.

You can specify in the Window Size boxes a size different from the above values by simply typing the size in the edit boxes.

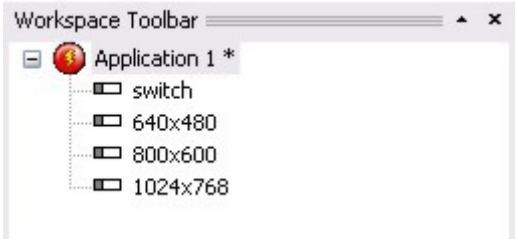


Programming a multi-resolution application.

This method will give the best results, but is the hardest to program. Here too, different choices are possible :

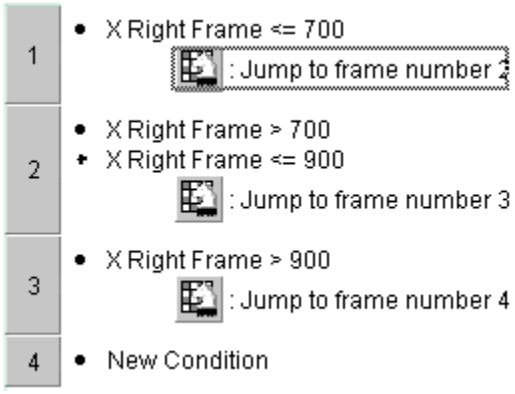
Program multiple frames.

1. Create one frame for each resolution you want to handle, and a preleminary switch frame.



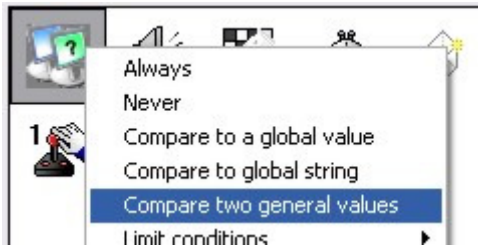
2. Program each resolution-dependant frame the way you want.

3. Size of the first frame should be big, at least equal to the size of our biggest frame (here 1024x768). Leave this frame empty : it won't eat any memory space in the runtime application. Maybe, change the background colour to black to avoid a white flash...
The events of the switch frame look at the coordinate of the right edge of the frame, and switch to the proper frame.

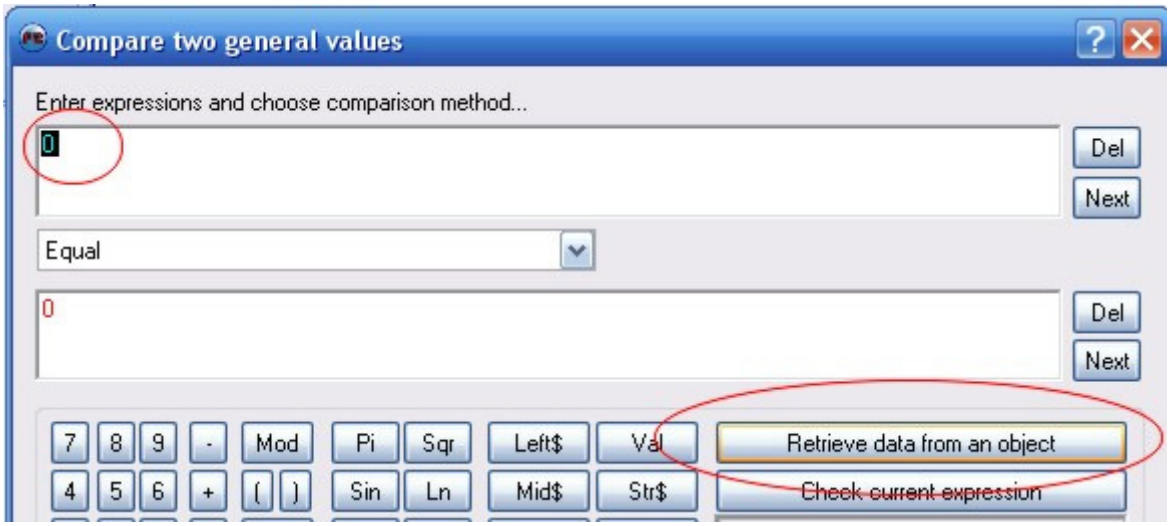


You will find thie **X Right Frame** expression by following these easy instructions:

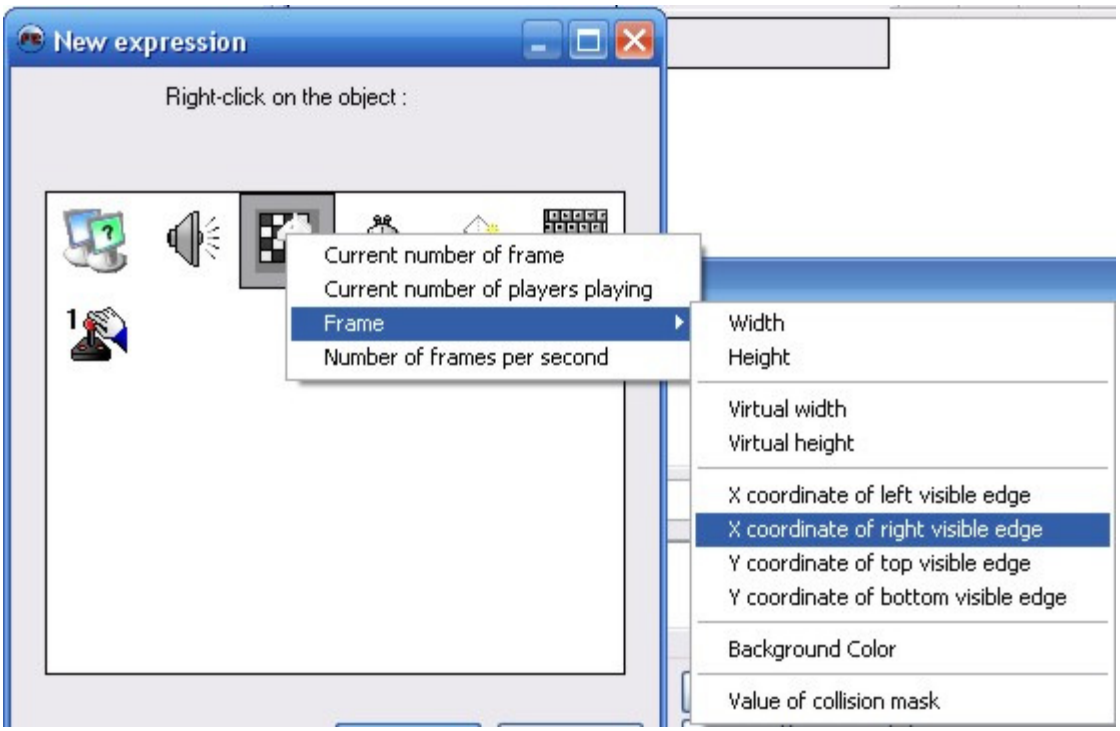
3a. Create a **New Condition** in the event editor and right click on the **Specials** icon and select the "**Compare two general values**" option.



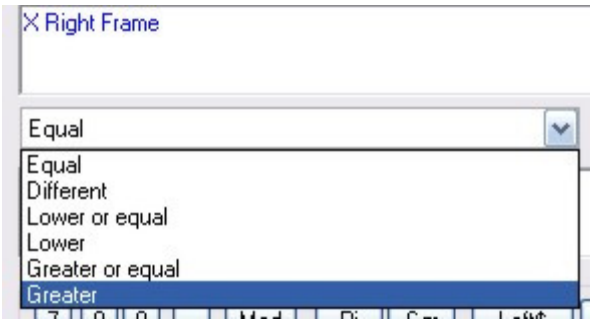
3b. This will bring up the something much like the Expression Editor but you can select two items (properties) to compare. With the first expression highlighted (circled in red below) click on the "**Retrieve data from an object**" button. (Just like the Expression Editor)



3c. This will disply all of the available object on the frame. Right click on the **Frame** icon and choose **Frame / X coordinate from the right visible edge**.



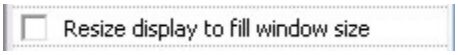
3d. Now change the type of comparison you are making using the pulldown menu (this reads Equals initially). For this exercise we will use Greater.



3e. Now just type in the horizontal size you are trying to determine. Once you have completed the different sizes to check for you can have the application jump to the appropriate frame tailored for that users resolution.

4. Programming one frame that handles multiple resolutions.

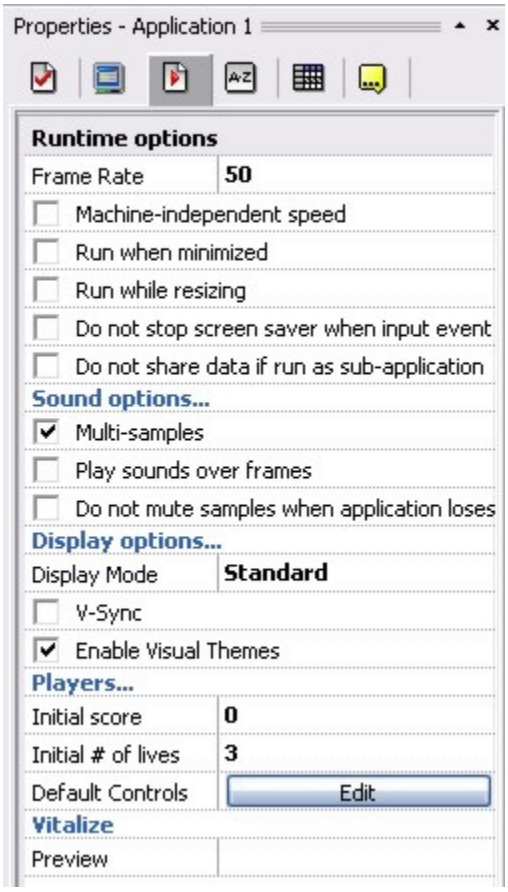
One neat new option in Multimedia Fusion 2 is a new checkmark in the **Properties Toolbar** / **Window Tab**



This magic option will automatically set the size of the frame at runtime to the current size of the screen. All you have to do is to set the window of your application to full screen with no heading or menu (see above), and check this option. You should design your frames so that all the active objects are created in the top left corner of the area (within the 640x480 limit), or uses the **Storyboard object** / **Frame** / **Width** and **Storyboard Object** / **Frame** / **Height** expressions in the expression editor to get the actual size of the frame...

☑ The runtime properties

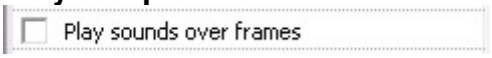
This **Properties Toolbar Tab** contains the remaining options for a fine tuning the resulting application.



Multi-Sample

Gives you the possibility of having more than one sound at the same time in the speaker, up to four sounds. Please note that enabling this option introduces a small delay when starting each sound : some users prefer to leave it off because of this delay. If your application can produce many sounds in short periods of time (like space battles for example), we suggest that you don't use this option, or use it cleverly: have some un-interruptible samples looping in the background that use the first 2 or 3 voices to create a nice background of sounds, and play the sound effects on the remaining channels. Otherwise you will end up in a sound-mess!

Play samples over frames.

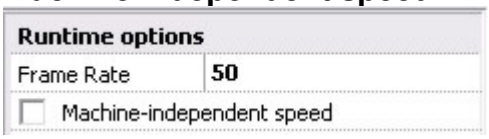


If checked, a sample playing at the end of a frame will keep on playing while going to the next frame. If not checked, all the sounds will be stopped prior to going to the next frame. Note that this option is set as default when you create a new Fusion application, and **unset** as default when you import a Games Factory application, to maintain compatibility.

Do not stop samples when application loses focus.

If you plan to create an application that continues playing background sounds, you have to check this option.

Machine independent speed.



A very cool option that compensates for the speed differences between machines. How does it work?

Without this option.
Fusion runs at the maximum possible speed for this application on this computer, with a top speed of 50 loops (or other values, 50 max) per second. If your application only moves one tiny sprites one pixel per loop, it will move at exactly 50 pixels per second. But if your application moves large chunks of screen that take more than 1/50 of second to draw, then the speed of the animation will be reduced. All the objects in the application are affected by this, so if we have our little sprite in the middle of a big, heavy graphics application, its speed can go down to 10 pixels per second, or even less.

With this option.
When you check this option, Fusion does its best to keep your application running at 50 loops per seconds. How does it do that? By skipping the screen drawing for certain loops. The most time-consuming task in a multimedia application is to draw all the objects on the screen. In Machine Independent speed mode, Fusion keeps an internal speed counter running at exactly 50 counts per seconds, and compares it to the actual number of loops of the application per second.
If the number of loops matches the counter, then everything is fine, Fusion sends the graphics to the monitor, and one can see the sprites in the new position.
If the number of loops is lower than the counter, then the application is too slow! Therefore, Fusion **skips** the drawing of the graphics for this loop, hoping that this will allow to recover the delay. On the next loop, hopefully the two values match and Fusion can send the graphics. Fusion allows up to 5 consecutive skips of the display, which may result in a jerky animation. But the main advantage of the machine independent mode is that the overall speed of the animation is preserved: our tiny sprite moves at exactly 50 pixels per second, even on a slower machine, but it moves 5 pixels at a time on the display.
This option is totally transparent to the programmer of the application, as all of the calculations, collision detections, etc. are still done internally.

That's it. I hope that this tutorial has made things a little clearer for you. One thing is sure: you should experiment with all the check marks! Please send us some feedback: we try to write tutorials for both beginners and advanced users, with both general information and advanced hints and tips, as well as explanations of how it all works internally. Are we doing things right? Please tell us!