



Making a CD-ROM interface with Multimedia Fusion 2





Making a CD-ROM interface with Multimedia Fusion 2

Table of contents

Table of contents	2
Welcome	2
What we want on our CD-ROM interface menu.....	3
Inserting graphics and text.....	3
Interactive buttons	4
Button settings	6
Making the button do something.....	7
Adding sounds	9
Making more buttons.....	11
Finalizing the CD-ROM window	11
Making an autorun file	12
That was it	12

Welcome

Welcome to the tutorial, explaining how to make a simple CD-ROM interface application. With Multimedia Fusion 2 you can with little knowledge about other development software, create personal or professional CD-ROM's with your very own content. You can add graphics, sounds and music after your own choice. Make cool welcome screens for your school projects; make a well-arranged CD browser for your CD compilations or just a simple autorun menu to start your game made in Multimedia Fusion 2.

What we want on our CD-ROM interface menu

In this first chapter, we will focus on creating a simple menu with simple effects.

We can start out planning what we want to make:

- A headline for the application
- Three or four clickable buttons
- Mouse-over effect on the buttons
- Sounds when the buttons are clicked
- The buttons executes programs from the CD

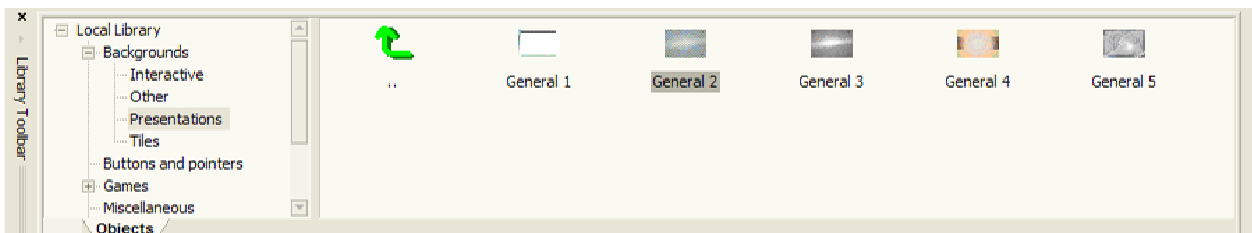
Other tasks:

- Making an autorun file so your application automatically starts

Inserting graphics and text

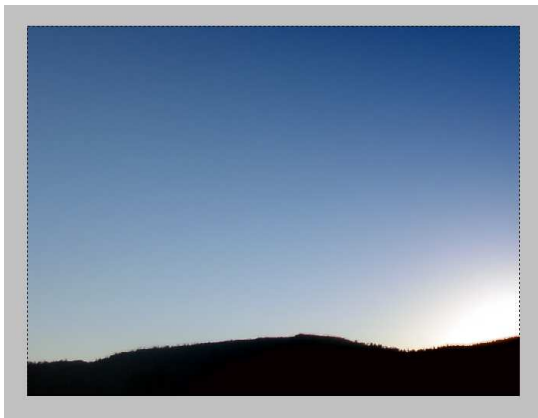
Multimedia Fusion 2 makes it straight forward to simply begin inserting everything we need. For our application we need a nice background. You can browse the Multimedia Fusion 2 libraries for nice backgrounds or you can search the internet for some.

Here I found an existing background from the MMF2 library and dragged it into the frame.



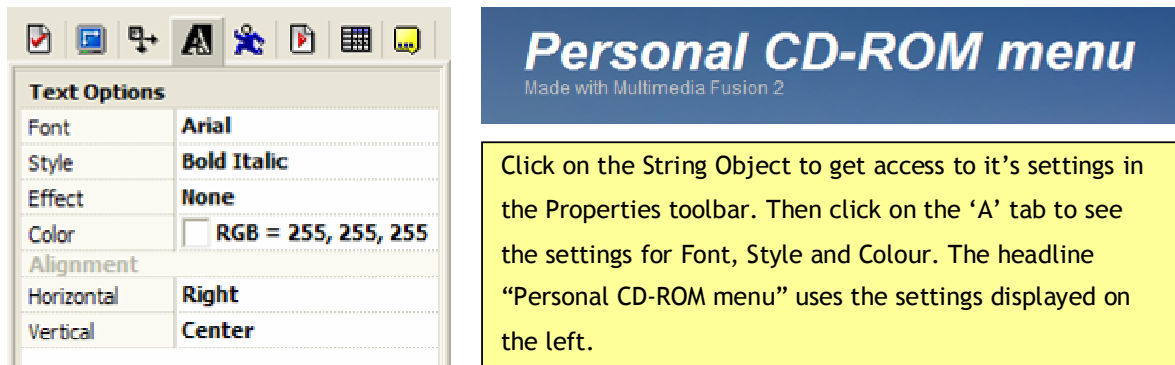
After you dropped it on the frame, you can resize it and drag it so it fits to the size of the window. We now have a nice background for our menu.

Now we will use the 'String Object'. It makes it easy to display some simple text on your frame.



By default it uses a small and black text size, but for our menu we want something bigger and something that fits better with the colours of the background.

Here is an example on what the text we insert for example could be:



The screenshot shows the Multimedia Fusion 2 interface. On the left, the 'Text Options' panel is open, displaying settings for a text object. The settings are as follows:

Text Options	
Font	Arial
Style	Bold Italic
Effect	None
Color	<input type="checkbox"/> RGB = 255, 255, 255
Alignment	
Horizontal	Right
Vertical	Center

On the right, a blue header bar contains the text 'Personal CD-ROM menu' in a large, bold, italicized font, with 'Made with Multimedia Fusion 2' in a smaller font below it. Below the header, a yellow box contains the following text: 'Click on the String Object to get access to it's settings in the Properties toolbar. Then click on the 'A' tab to see the settings for Font, Style and Colour. The headline "Personal CD-ROM menu" uses the settings displayed on the left.'

You can easily add more string objects and give them different sizes and colours.

Interactive buttons

Next thing we insert is buttons. For the buttons we will in this example use an object called "Active System Box" as it has some features that makes our work faster and easier to do.

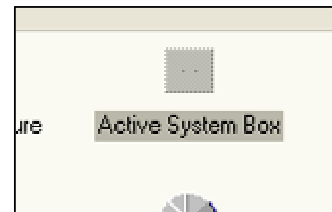
In this example we want it to:

- Be a menu item that we can click on
- Display some text of our choice
- Show a border when our mouse hovers over it.
-

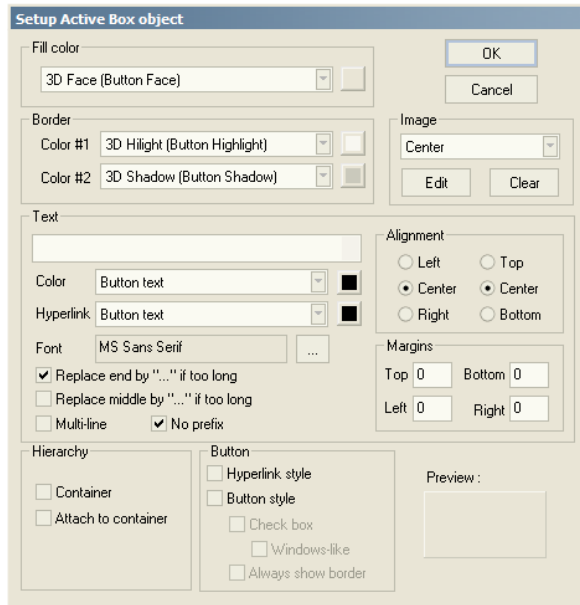
The Active System Box has a lot of other useful (but more advanced) features that we won't focus on in this tutorial.

Click 'Insert new object', find the object and click OK.

You will notice that there is also a 'Background System Box', but it doesn't have all the same features as the 'Active' version of it. For example it doesn't have inbuilt features to make it act like a clickable button which is what we want.



You will now be shown the Active System Box's setup dialog, which will look something like the picture to the right. By default the box will be rather boring to look at, its fill colour is grey and borders are also boring shades of grey. We want to spice it up a little. If we want to use a background image for the button, we click the 'Edit' button, close to the top right corner of the dialog. You can then either draw a background image there or you can paste in another image you created in another drawing application.



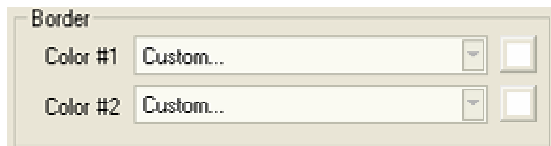
In this tutorial, the following image is used for the button:



If you don't want an image as a background you can simply use a colour by clicking on the box next to the 'Fill colour' drop-down box and select a colour in the colour selector that will pop up.

Borders

For the borders we click the small buttons and select a pure white colour for them both. In the colour dialog that pops up we click a white colour and click ok.



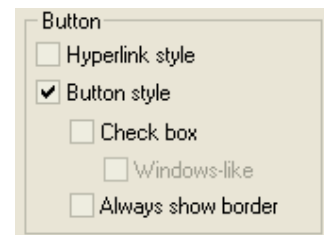
Text

We want our button to tell us which program it will open. But until we have decided which program it should open, we use the text 'Button 1' as a placeholder. Click on the '...' Button at the Font box setting to set the font and size. For the text colour we'll use white. The rest of the font settings aren't important at this point.

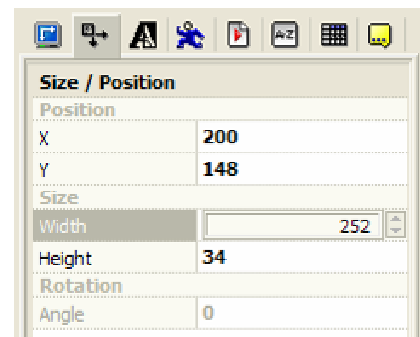


Button settings

The most important thing for our own custom button is that it can be clicked. At the button settings we check the 'Button style' checkbox so it looks like the picture. Now our object will display our text on our background image/colour and display a white border around the button when the mouse hovers over it. We could also check the 'Always show border' checkbox if we always want the borders to be displayed but in this example we want the white borders as a mouse-over effect only.

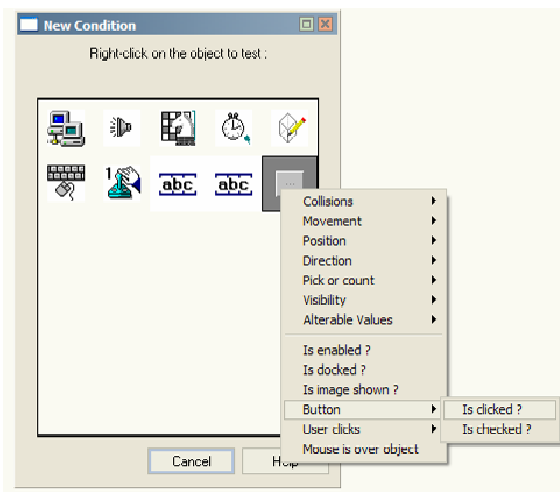
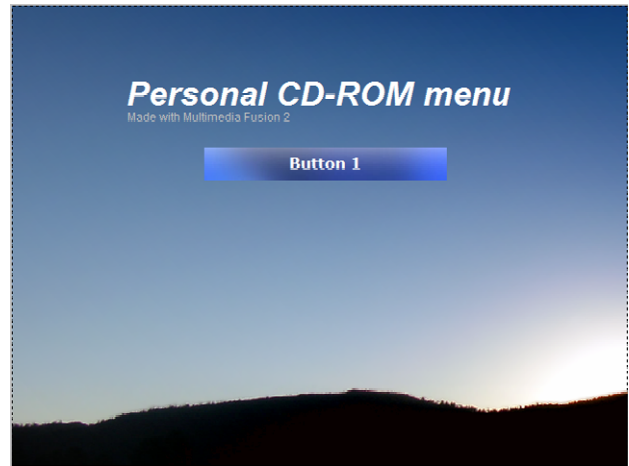


Now click OK to go back to the frame editor. You can now resize the button so it has a reasonable size. If you used a background image for the button and want your button to have the same size, you can either resize the object with the mouse or use the properties toolbar for more precision.



Making the button do something

So far our button is purely visual though it gives a 'clicked' effect if you click on it at runtime + you get a mouse-over effect. You can try to run the application and test it. Now we need to tell Multimedia Fusion 2 what we want our application to do if we click the button. Switch to the event editor and click on the 'New condition' line.

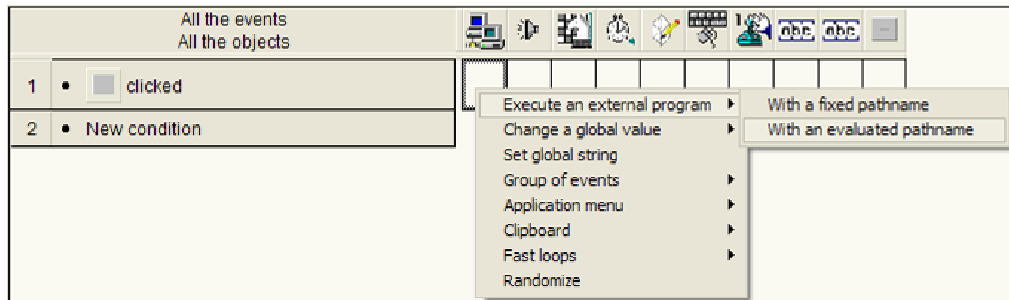


In the 'New Condition' dialog, you right click the icon for your Active System Box. At the button subgroup you select the 'Is clicked?' condition.

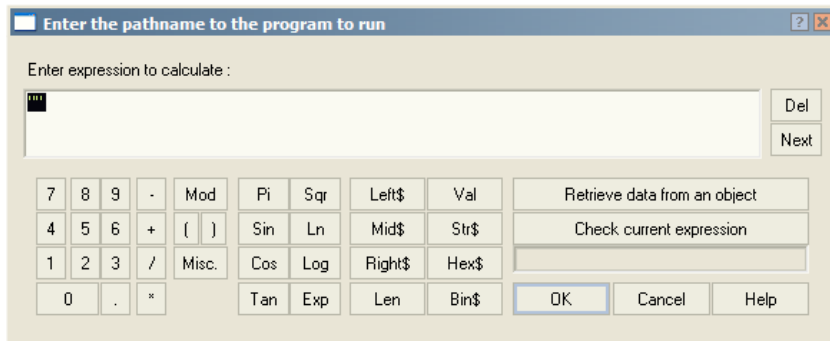
Now we have told MMF2 that we want to know when the button is clicked and we can now begin to add some functionality into our program.

Say you want to burn a CD-ROM with a program called "Setup.exe" on it and you want our application to run that application as soon as the user clicked the button.

In the event that appeared after you created the condition, we now wish use a feature of the 'Special object' (the very first object you see listed at the top of the event editor) called "Execute an external program -> with an evaluated pathname"

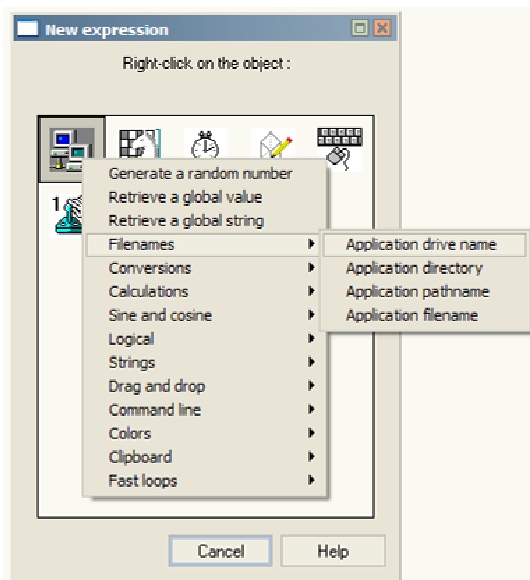


MMF2 will now show you the **Expression Evaluator** dialog.



Here you should enter the path to the program you want to launch. It would be easy to enter “D:\Setup.exe” assuming that the D:\ drive is the user’s CD-ROM drive but that is not always the cause! Many people have extra hard drives in their computer and their CD-ROM drive would then be E:\ or G:\.

So to make MMF2 know the correct path to the application you want to execute we can assume that the application is on the exact same drive letter as our own CD-ROM interface application. We then use an inbuilt **expression** from MMF2 to get this drive letter. Click ‘Retrieve data from an object’ to open this dialog:



From the ‘Special Object’ we can get the current drive letter by selecting “Application drive name”. This will be shown in the expression editor as **Appdrive\$**.

So the final expression to load the ‘Setup.exe’ application looks like this:

Appdrive\$ + “Setup.exe”

No matter which CD-ROM drive we put our CD into, MMF2 will use the correct pathname.

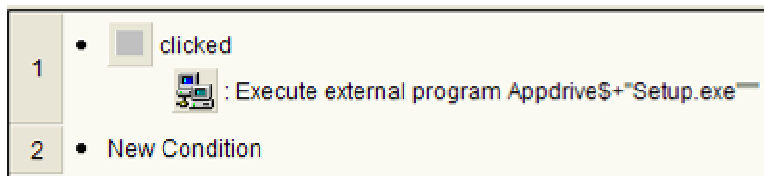
For example if you run our application from drive D:\, MMF2 would see our expression as

“D:\Setup.exe” and as “E:\Setup.exe” if ran from drive E:\.

MMF2 also asks for a ‘Command line’ but we can skip it in this tutorial. Finally MMF2 also asks you if it should ‘Wait’ for the Setup.exe application to finish. This means that the

application will be paused until the executed application has finished running. The ‘Hide’ option means that your application will hide itself until the executed application has finished running.

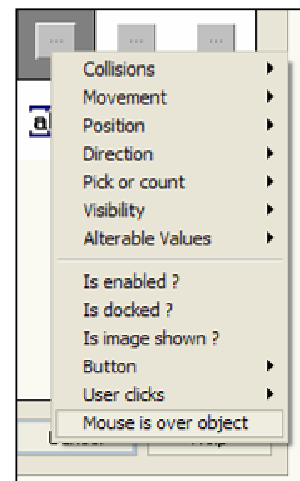
Our event as it looks from the event editor:



As it looks from the event list editor:

Adding sounds

We could leave the menu as it is now but that would be a bit boring. We want a sound when the mouse moves over the button. We then once more click ‘New condition’ at the bottom of the event list. Right click the Active System Box’s icon and select ‘Mouse is over object’ from the bottom of the popup menu. We can then in the new event insert an action to play a sample from the MMF2 library or one of your own sounds. I won’t go into much detail here as it is described in the help files how to play a simple wave sample.



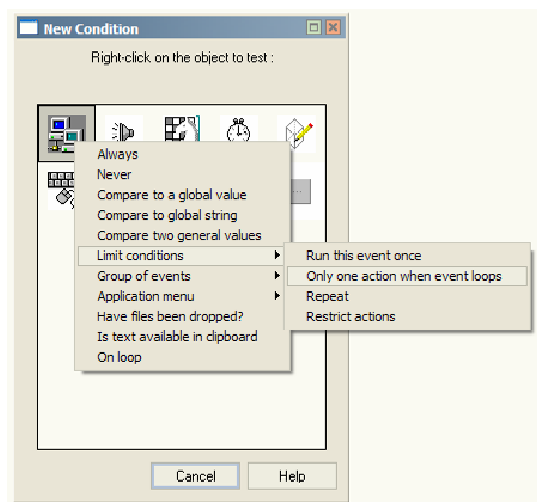
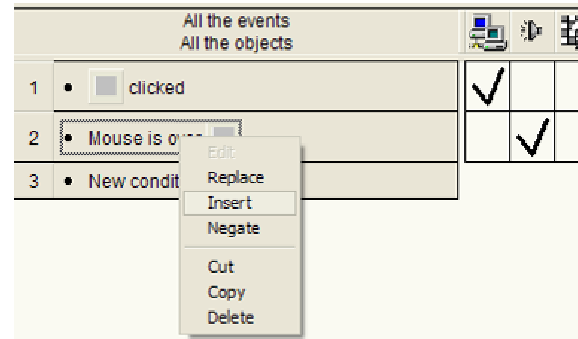
There is one problem with this event though. If you run your application and moves the mouse over the button, the sound will play continuously and messy. Why? Since ‘Mouse is over object’ is true all the time, the sample will be played all the time. We only want the sound to play once per time the mouse moves over a button, so we now have to

insert an additional condition into the same event. This may sound complex but is quite simple:

Right click on the first condition as shown on the image and then select 'Insert'. You will be shown the 'New condition' dialog as if you were about to create a new event, but this time it will insert the new condition into the same event as the 'Mouse is over' condition.

When you see the 'New Condition' dialog you (again) right click the 'Special object' and select 'Only one action when event loops' from the 'Limit conditions' subgroup.

Now "drag" this condition to the bottom of your event so it looks like the image below.



This special condition makes MMF ignore your event if the event was 'true' the moment just before. Note that not all conditions always are true. Other conditions (for example collision conditions) will only tell MMF2 that two objects collided once even though they are still overlapping each other. The condition called 'Object is overlapping another object' is always true and conditions like 'Only one action when event loops' are therefore quite handy in many situations.

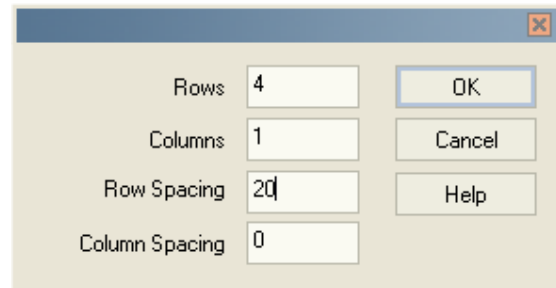
Test your application and you should hear a single sound when the mouse moves over the button.

This was the mouse-over sounds. We also want to hear a sound when the user presses the button. In the very first event you made to execute another application, you simply make a sound action there. There you don't need any special new conditions or anything. The button is only clicked once, so the sound will only be played once.



Making more buttons

To make more buttons you can simply go to the Frame Editor again, right click your button object and click the 'Clone' item. Enter '4' as the amount of rows you want to create (vertically) and the amount of columns you want (we'll leave it as 1 here).



We don't want our buttons to be extremely close together so we enter '20' as the Row Spacing value. This is the distance in pixels that the rows will get.

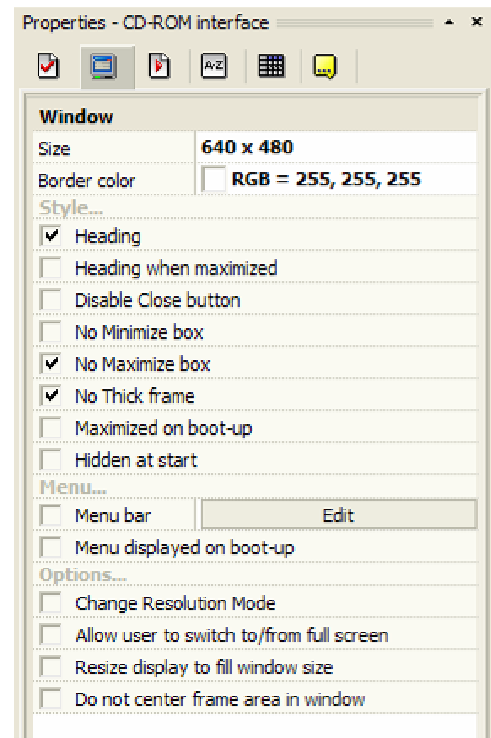
After you press OK you have four buttons in total. Now it's simply up to you to change the text in each button and make new events that execute different programs.

Finalizing the CD-ROM window

When you run your application, you have probably noticed the menu bar at the top of the window, you can resize your window and you can maximize the window. This has no real purpose in our application so we want to remove it.

Click your application in the workspace toolbar.

There you get access to most options regarding your how your application behaves and looks. Click the 'Window' tab icon to see the window options. If you change your window settings to look just like those to the left, the window won't be able to be resized anymore (No Thick Frame), you have no maximize button (No Maximize button) and the checkbox "Menu bar" is unchecked.



Making an autorun file

Many CD-ROMS automatically start a program like ours when we insert a CD. To make an application do this, we need to create a little text file containing some information. This has nothing really to do with Multimedia Fusion 2 as it's a feature of windows.

When Windows detects a CD it will look for a file called "autorun.inf". If it exists, windows will read it and then launch the program we told it to inside the file.

A very simple example of an autorun.inf file (you can write this in notepad for example):

```
[autorun]
open=cdrommenu.exe
```

If you build your application as cdrommenu.exe and burn it down to the CD-ROM (in the same path as the autorun file) your application will automatically be started when you put your CD into the drive.

This article cannot help you burn your CD-ROM, you need to read your burners manual book.

That was it

This is the end of the article, I hope you learned much about Multimedia Fusion 2 and how it could be used to create simple but useful things.

If you have any questions regarding this article or run into any problems, feel free to post a question on the helpful Clickteam forums.

www.clickteam.com