How to incorporate custom toolbar buttons and cursors into your MapBasic application?

When you create your own MapBasic application, you might wish to create a toolbar with your own custom buttons. Of course, you can use one of the buttons which are provided with MapBasic (ICONS.DEF), but then chances are high that your application uses an icon which has been used by another developer in another application already…

You want to use your own unique toolbar buttons. It is quite easy to do so.

You just store some bitmaps in a DLL, and you call this DLL from your application. This document will explain – step by step – how to get there.

At the end of this document you will also find a description of how to incorporate your own custom cursor.

This document has been updated for MapInfo Professional 10.0.1

In this explanation a free development kit will be used to create and edit the DLL: **Pelles C**

So, if necessary: please feel free to download (http://www.smorgasbordet.com/pellesc/) and install this development kit to be able to follow the steps below.

You will see how easy it is to incorporate your own custom toolbar buttons and your own custom cursors into your MapBasic application.

Step 1: Arrange software to create a DLL

You will need to create a DLL (a Dynamic Link Library) to store your bitmaps. The MapBasic development environment does not offer you the possibility to create and edit a DLL file.

You have got to use third-party software for this.

If you do not have any experience with creating and editing DLL files, please have a look at this one: **Pelles C**

Pelles C is a complete development kit for Windows. And Pelles C is available for free.

You can download Pelles C free of charge from the following location: http://www.smorgasbordet.com/pellesc/

The installation of Pelles C is quite straightforward.

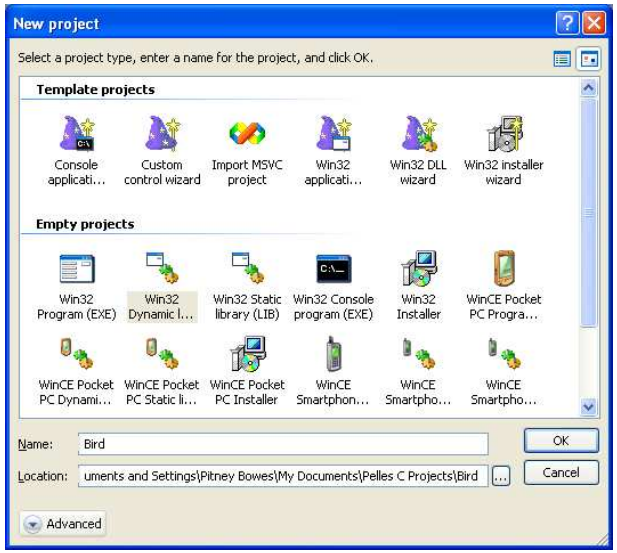
The first time you run **Pelles C IDE** you will be asked to create a System definitions database.

*(The following version of **Pelles C IDE** is used here: Version 6.00.4 from August 2, 2009.)*

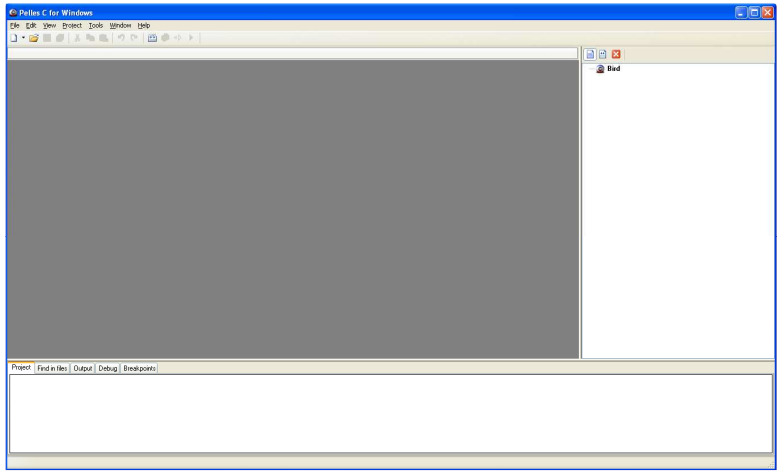
Step 2: Create a DLL

Start **Pelles C IDE** and go to **File > New > Project…**

In the **New project** dialog, select from **Empty projects** the **Win32 Dynamic library (DLL)**, enter a name and click OK.



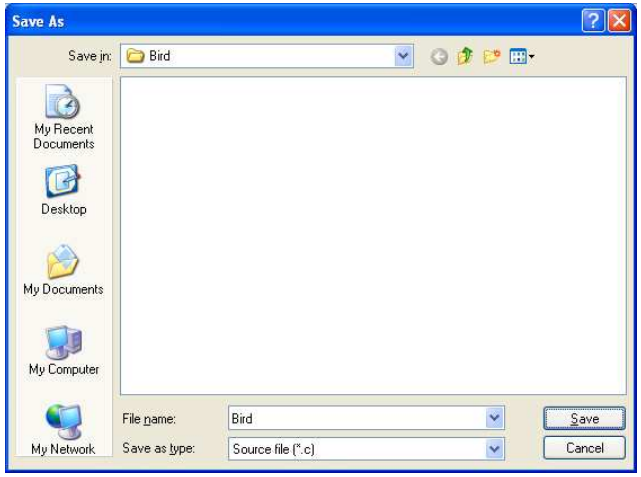
You now have an empty project.



Go to **File > New > Source Code**

You do need a source file to be able to build your DLL later on. But don't be afraid: you are not going to enter a single line of code here.

Save the empty source file (**File > Save**)



When asked to add this file to the current project, please answer **Yes**.

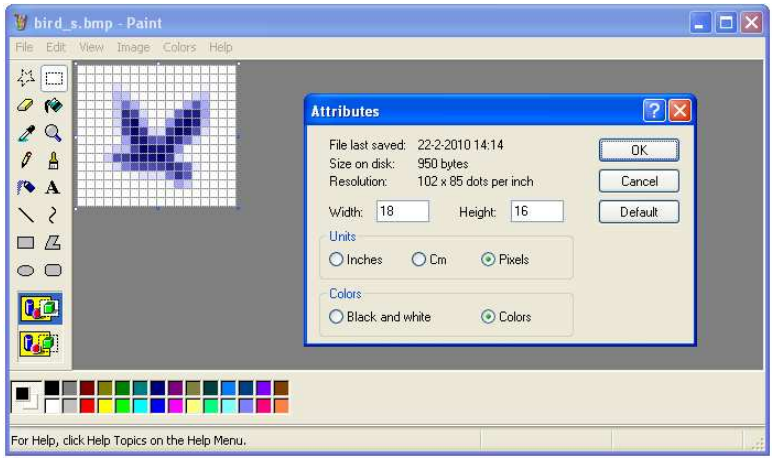
Step 3: Create bitmaps

You must create two bitmap resources for each custom icon.

The first bitmap resource must be 16 (or 18) pixels wide by 16 pixels high; this is the icon that will appear if the user does not check the **Large Buttons** check box in MapInfo Professional's Toolbar Options dialog box.

The second bitmap resource must be 24 (or 26) pixels wide by 24 pixels tall; this is the icon that will appear if the user does check the **Large Buttons** check box.

You must create both resources. (Note: you must create bitmap resources, not icon resources.)



MapInfo Professional 10:
New MapBasic Capability for Custom Icons

Before MapInfo Professional 10.0, custom icons were rectangular in size: 18x16 for small icons and 26x24 for large icons. New toolbar and menu features introduced in MapInfo Professional 10.0 require square icons: 16x16 for small icons and 24x24 for large icons.

When creating bitmap resources you've got to think with which version of MapInfo Professional your tool will be used:

For best results, recreate your icons to 16x16 for small icons and 24x24 for large icons. Icons at these sizes are not backwards compatible and generate an error message in versions earlier than 10.0.1.

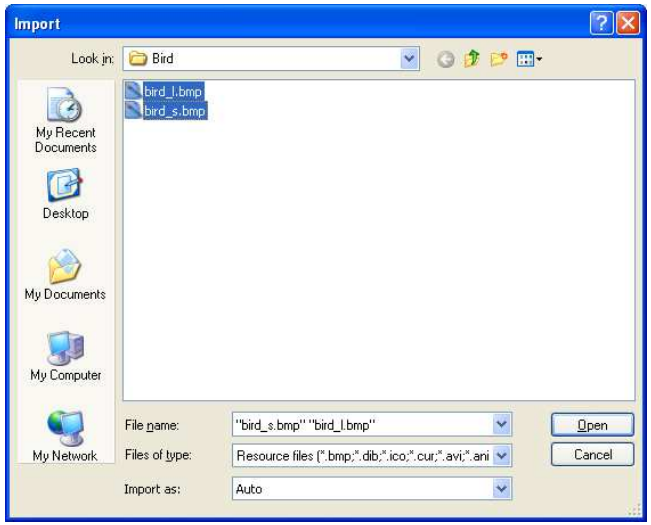
For backwards compatibility, use 18x16 for small icons and 26x24 for large icons. But if you do so, please make sure you do not use the first and last column of pixels, because from version 10.0.1 onwards these columns of pixels will be removed to not display them (cropping the image). In effect, the newest versions will just treat your bitmaps as square icons - but for backwards compatibility there has to be a blank column of pixels at the beginning and at the end.

Step 4: Add bitmaps to DLL

Now that you have your bitmaps, you have got to add them to your DLL project.

Return to **Pelles C IDE** and choose **File > New > Resources**

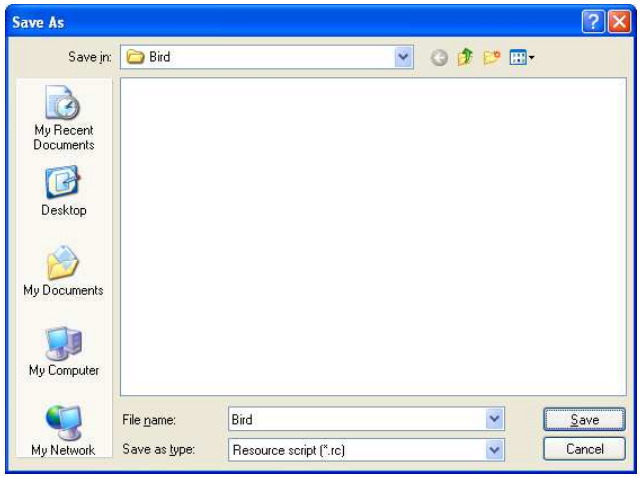
Next, choose **Resource > Import…** to import your bitmaps.



Please make sure that sequential ID numbers are assigned to the two bitmap resources.

In this example, an ID of 8001 was assigned to the small bitmap, and an ID of 8002 to the large bitmap. (This is done automatically by **Pelles C IDE**.)

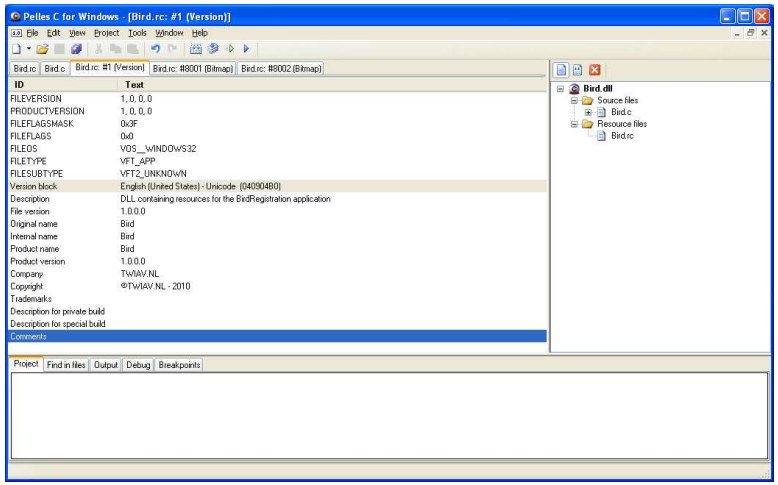
Next, choose **File > Save** to save the Resource script.



And again, when asked to add this file to the current project, please answer **Yes**.

You are ready to build your DLL now. No! Wait! Let's enter a Version to be able to document your DLL a little bit.

Choose **Resource > New > Version** and enter the relevant information.



Save your Resource script again.

Step 5: Build your DLL

Choose **Project > Build <My own>.dll** to build your own DLL.

You will get a warning – something about an empty input file (your empty source file) – but the DLL will be created anyhow.

Only five simple steps! Now you have got your DLL with the bitmaps to use as custom toolbar buttons.

Step 6: Call the DLL from your application

Once you have created the pair of bitmap resources, you can incorporate your custom bitmaps into your MapBasic application using either a **Create ButtonPad** or an **Alter ButtonPad** statement.

In your program, refer to the ID of the smaller bitmap resource. In this example, the IDs 8001 and 8002 where assigned to the bitmap resources, so the program should refer to ID 8001, as shown in the following statement:

```
Create ButtonPad "Bird Registration" as
     ToolButton
          Icon 8001 File "Bird.dll"
          HelpMsg "Register a bird\nBird"
          DrawMode DM_CUSTOM_POINT
          Cursor MI_CURSOR_FINGER_LEFT
          Calling Registration
     Show
```

The DLL file where you store your custom icons (in this example, Bird.dll) must be installed on your user's system, along with the .MBX file. The DLL file can be installed in any of the following locations:

- The directory where the .MBX file is located;
- the directory where the MapInfo Professional software is installed;
- the user's Windows directory;
- the system directory within the Windows directory;
- or anywhere along the user's search path.

If you place the DLL in any other location, your MapBasic program must specify the directory path explicitly (for example, Icon 100 File "C:\GIS\MBICONS1.DLL").

Note that the **ProgramDirectory$( )** and **ApplicationDirectory$( )** functions can help you build directory paths relative to the MapInfo Professional directory or relative to the directory path where your MBX is installed.

Step 7: Run your application

Here you are! Your own custom tool button.

Disabling toolbar buttons

And this what your button looks like in MapInfo Professional 10 when it is disabled ("grayed out")

Disabling toolbar buttons in older versions of MapInfo Professional

If you want your tool buttons to be backwards compatible, please note that graying out works a little different in older versions of MapInfo Professional. See the example below:

If you want your application to be able to disable ("gray out") your toolbar button – e.g. if you do not want it to be available under certain conditions – then please make sure that you use a dark boundary line around your picture.

In the example a maroon line was used. And this is how it looks like when "grayed out":

At least the following colors can be used for this purpose:

Black (R=0, G=0, B=0)
Navy (R=0, G=0, B=128)
Maroon (R=128, G=0, B=0)

What happens if you do not use one of the colors mentioned above?

Let's test this. Here you can see the same button as before, but now with an olive border (R=128, G=128, B=0).

It looks great when enabled…

…but gives a very disappointing result when "grayed out".

Step 8: Let's also add a custom cursor!

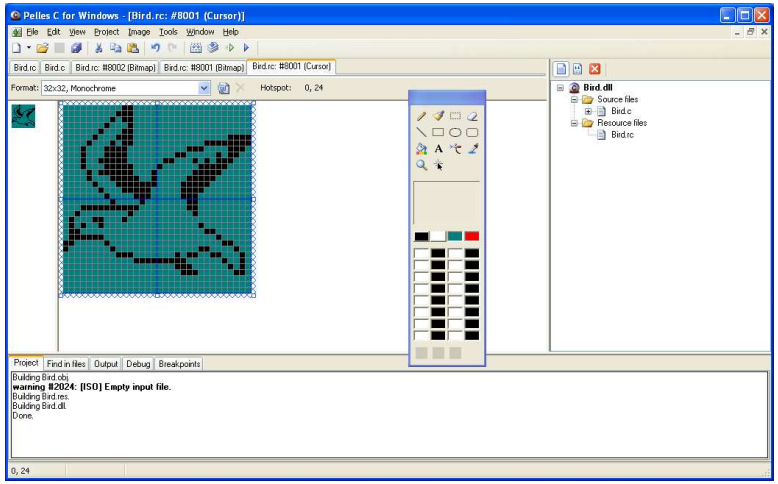
Pelles C also offers the possibility to create and edit cursor files (*.cur).

So, let us also add a custom cursor:

Choose **Resource > New > Cursor** and draw your own cursor.

(As you will see the cursor also gets the ID 8001 – but that's not a problem, because a cursor is not a bitmap)

Please make sure to make the beak the hotspot and to use the screen color for the background of your cursor.



After finishing your cursor design, save your Resource script again and rebuild your DLL.

Now you can refer to the cursor in the DLL from your application (see code below):

```
Create ButtonPad "Bird Registration" as
     ToolButton
          Icon 8001 File "Bird.dll"
          HelpMsg "Register a bird\nBird"
          DrawMode DM_CUSTOM_POINT
          Cursor 8001 File "Bird.dll"
          Calling Registration
     Show
```

Step 9: Run your application again

Here you are! Your own custom cursor

Happy coding!