How to Show Multiple Colors on a Continuous Type Form

For Microsoft Access 2002/2003 or later.

By Peter Hibbs.

Version 1.0 June 2011.

Introduction.

It is sometimes useful to be able to highlight individual fields (or complete rows) in different colors on a Continuous type form and this demo database describes one method. It is possible to use Conditional Formatting to do this but this is limited to three or less different colors and can be quite slow to update the forms. This method uses the **Binary Large Object Bitmap** (BLOB) in an **OLE Object** control on the Continuous form to display the background color of a field/s and since the field is bound to the form's query, any number of colors can be used on the form. Also, no VBA code is required to implement this system.

The attached Access 2003 database (**Continuous Colors.mdb**) uses a couple of tables (**tblProducts** and **tblCategories**) from the Northwind database to show a list of products with various fields, including the **Category** field. The whole row for each product is highlighted in a different color for each category, i.e. 'Seafood' in yellow, 'Dairy Products' in green and so on. Note that I have used a main form - subform arrangement for this because it looks better although it is not necessary for this system to work. When you run the database the main form is displayed showing a list of products and each record is colored depending on the category for that product. You can still use the standard Access facilities to filter or sort the records, for example, right clicking on a **Category** field and selecting 'Sort Ascending' will show the records in category order.

To set up a similar system you should first create the table and a Continuous type form as normal and design a query for the form. If every record on the form is to have a different color then you should add a new field to the table which is an **OLE Object** type field, if, however, multiple records can have the same color (as in the example database), then you should create a separate table that has at least two fields, one field to link back to the main table (usually a **Number** type) and one **OLE Object** field to hold the color information. In the example database the table **tblCategories** already exists so it was only necessary to add the new **OLE Object** field to this table. In the query that the form is bound to, add the new table and add the image field to the query (in the example database the query is named **qryProducts** and the **OLE Object** field is named **ImageColor**).

Next you need to set up the colors for the categories (or whatever the field represents). You should first create a record for each color that you need, in this example there are eight records for the eight categories. To set up the different colors required, open the table as normal (that is - not in Design mode) and right click on the **ImageColor** field (or whatever you called it) for the first record, choose the **Insert Object…** option in the pop up menu, select **Bitmap Image** in the **Object Type** list box in the pop up form and then click **OK** to display the **Windows Image Editor** program (normally **Paint**). Create an image of about 100 pixels wide by 25 pixels high and fill it with the required color, actually the size is not really critical since the field will be resized on the form anyway but it is better to keep the image fairly small as bitmap images can use a fair bit of memory. For most situations the image would normally be all one color but there could be some applications where some sort of pattern image would be more appropriate.

Repeat the above for each record in the table. If you need to change an existing color image, you can right click the field again and choose the **Bitmap Image Object.. -> Edit** option (or the

Open option which seems to be the same as **Edit** for bitmap objects) which will open the same **Paint** program as before and displays the stored image. Alternatively you can just double click the **ImageColor** field to open the **Paint** program in **Edit** mode. Change the image as required and close the form (note that you do not have to specifically 'Save' the image as the table data is amended as soon as you make any changes to the image).

The above procedure applies to Windows 7 (and probably Windows Vista) but for Windows XP the **Paint** program works slightly differently for the edit option. If you use the **Edit** option in the right-click menu, you can only edit the foreground color (it seems) and there are limitations on what you can do. So to edit the image you should select the **Bitmap Image Object.. -> Open** option which works in the same way as the **Insert Object** option described above.

Setting Up the Continuous Form Fields.
That should be all that is necessary on the tables, now open the Continuous form in Design mode and follow the procedure below :-

Add all the 'normal' fields to the form and set them up, as required.

Add the field that holds the color image to the form, if you do this using the **Field List** icon to display a list of fields, the control on the form should automatically be set to a **Bound Object Frame** type.

If you want to add the field manually, as it were, select the **Bound Object Frame** icon on the **ToolBox** and draw the control on the form and delete its associated label. Next set the control's **Control Source** property to the name of the image field in the query.

Set the **Bound Object Frame** control's properties like this :-
Size Mode property = 'Stretch'
OLE Type Allowed property = 'Embedded'
Enabled property = 'No'
Locked property = 'Yes'
Border Style property = 'Transparent'

Position this control behind the controls on the form that you want to show in the different colors and set the **Back Style** property of those controls to 'Transparent' (and for a Continuous form, you may want to ensure that the **Border Style** property is set to 'Solid'). Also make sure the **Bound Object Frame** control is 'sent to the back' (on the **Format** menu).

For the purposes of this example database I have made all the controls transparent and made the image control the whole width of the form but in a real database I think this would be a bit 'overkill' and the form would probably look better with just the relevant field/s showing the colors.

The form should now work, as required.

Using this System in a Split Database.

The database, as it stands, works fine because the tables are all in the same file. However, a working database would normally be split into a Front End / Back End arrangement with the Back End file usually on a Server PC and accessed via a Network. In this situation, this method does produce a problem in that if the table holding the images is stored in the Back End file (i.e. **tblCategories**) the images have to be sent via the Network to the Front End and even though the individual images can be fairly small, the total number of files can be large which slows down the Continuous form's response time considerably.

To get round this problem, when you split the database, it is necessary to store the table that holds the images in the Front End file and any other tables in the Back End file, as normal. If you look at the Relationships window you will see that the two tables are not linked together as they normally would be in this type of set up. This is because it is not possible to link two tables that are in the FE and BE (well, it is possible but it is not possible to enforce **Referential Integrity** so there is not much point).

While keeping the table with the images in the Front End works OK there are some issues that you should be aware of. As it is not possible to enforce Referential Integrity it is not possible to automatically update the linking field (**CategoryID** in this demo) if the value in this field in **tblCategories** is changed. So if the value in this field is changed you would also need to change the corresponding value in any tables that are dependent on the same field with some VBA code. Also, if a color is changed in the **tblCategories** table by the users it will only affect their own Front End file and not any other user Front Ends that may be connected to the same Back End and the next time an updated Front End file is supplied by the database developer, the table images will be changed back to the original colors.

So, for these reasons, I think this system would work best where the colors of the images are more or less fixed at design time rather than allowing the users to set up their own colors (although it would probably be possible to store that information in a table in the Back End and then copy it to each Front End file at start up, or whenever).

Editing the Color codes.

If you should ever need to change any of the colors for the various categories, you can open the table that holds the color image field and edit the image, as described above. However, if you want the end users to be able to do that it is not a good idea to allow them access to the tables directly.

A better option is to give them a form in which they can see the colors and change them manually. In the demo database, I have also provided a form (**frmEditColors** and **frmEditColorsSub**) which shows one method for doing this. On the main form in the demo database click the **Edit Category Colors** button to show the **Category Colors List** form. This simple form just shows all records in the table, **tblCategories**, with the relevant fields in the table. To change a color, just double click on the color box to be changed and the **Paint** program will display the image. Edit the image as required and exit the **Paint** program, the main form colors will be updated automatically when you close the form. Again, as mentioned above, if you are using Windows XP you need to right click the color field and choose **Bitmap Image Object.. -> Open** to edit the color image (and for Windows XP you should probably disable the double-click facility, see below).

Of course, any inexperienced user could still mess things up by deleting the image or choosing invalid colors or whatever so this facility should be used with caution, perhaps only being made available to database administrators.

Also, the **ImageColor** control has some other properties which can be used like this :-
The **Auto Activate** property can be changed from 'Double-Click' to 'Manual' which would prevent users opening the **Paint** program by double clicking the control. However, the users can still right click on the control and open the **Paint** program with the **Edit** or **Open** option to allow them to edit the images.

If the control's **Locked** property is set to 'Yes', the users cannot use the **Insert Object** option when they right click the control although they can still use the **Edit** (and **Open**) options.

To disable the right click pop-up menu completely you can uncheck the **Allow Default Shortcut Menus** check box on the Access **Tools -> Startup** form but this will affect the whole database (including the tables) which means that the only way to add new images would be with VBA code. Note that if you do change this setting, it will only take effect the next time you run the database.

There are a few other facilities and properties associated with **OLE Object** controls but these are not relevant for this type of application. For more information on these - Google is your friend!

Programming OLE Object Controls with VBA Code.

There may be occasions when you need to fetch an image from or insert an image into an **OLE Object** field in a table. This is possible but there are certain limitations as detailed below.

To fetch an image from a table you can just use the standard **DLookup** function like this :-

```
Dim vImage As Variant

vImage = DLookup("ImageColor", "tblCategories", "CategoryID = 8")
```

The variable must be dimensioned as a **Variant** type and the image just copied into it.

Copying an image into a field in a table is a bit more complicated. It is not possible to use the standard SQL functions (i.e. INSERT INTO or UPDATE) because bitmap objects could be far larger than the SQL functions can handle. The method to use is to write the data directly to the table like this :-

```
Dim rst As Recordset

Set rst = CurrentDb.OpenRecordset("SELECT ImageColor FROM tblCategories WHERE
CategoryID = 1")
rst.Edit
rst!ImageColor = vImage
rst.Update
rst.Close
Set rst = Nothing
```

This code fragment assumes that the image to save has been copied into variable **vImage** (as above). First create a one record recordset for the table record that is to be used, open the record for edit, copy the image to the required field and update the table.

On the main form in the attached database, in order to demonstrate this code, I have provided a button called **Swap Colors 1 & 8** which copies the color image in record 1 into record 8 (as defined by the **CategoryID** field) and then copies the image from record 8 into record 1. Record 1 is for category ‘Beverages’ and record 8 is for category ‘Seafood’. Make a note of the colors used for those two categories and then click the button to test it. The VBA code used is very similar to that described above.

If you need some VBA code that is more comprehensive than this you should Google the Internet for ‘Access BLOB’. This Web site would be a good place to start :-
http://www.ammara.com/articles/imagesaccess.html

History.

Version 1.0 June 2011.

--ooOOoo--