Bonus Chapter 1

Working with Colors

In This Chapter

- Specifying colors in your VBA code
- Using VBA conversion functions
- Converting colors to grayscale
- Working with document themes

Back in the pre-Excel 2007 days, a workbook stored a palette of 56 colors. Sure, you could modify any or all of those colors, but there was no way to exceed the 56-color limit for a workbook. But things changed with the introduction of Excel 2007. You now have access to a virtually unlimited number of colors in a workbook — actually, the limit is 16,777,216 colors, but that certainly qualifies as virtually unlimited in my book.

Color My World

In VBA, you can specify a color as a decimal color value, which is a number between 0 and 16,777,215. For example, the VBA statement that follows changes the background color of the active cell to a dark maroon:

```
ActiveCell.Interior.Color = 5911168
```

In addition, VBA has predefined constants for some common colors. For example, vbRed has a value of 255 (the decimal value for pure red), and vbGreen has a value of 65,280.

No one, of course, can keep track of nearly 17 million colors, and the predefined constants are pretty limited. A better way to change a color is to specify the color in terms of its red, green, and blue components — the RGB color system.

The RGB color system

The RGB color system combines various levels of three colors: red, green, and blue. Each of these color values can range from 0 through 255. Therefore, the total number of possible colors is 256 x 256 x 256 = 16,777,216.

- When all three color components are 0, the color is pure black.
- When all three components are 255, the color is pure white.
- When all three are 128 (the half-way point), the color is middle gray.
- The remaining 16,777,213 possible combinations of these three values represent other colors, some of which you may have actually seen.

To specify a color using the RGB system in VBA, use the RGB function. This function accepts three arguments that represent the red, blue, and green components of a color. The function returns a decimal color value, between 0 and 16,777,216.

Here's how it works. The statement that follows uses the RGB function to assign a color that's exactly the same as the one assigned in the preceding section (that dark maroon, with a decimal color number of 5,911,168):

```
ActiveCell.Interior.Color = RGB(128, 50, 90)
```

Table 1-1 shows the RGB values and the decimal color codes of some common colors.

Table 1-1 Color Examples

Name	*Red Component*	*Green Component*	*Blue Component*	*Color Value*
Black	0	0	0	0
White	255	255	255	16777215
Red	255	0	0	255
Green	0	255	0	65280
Blue	0	0	255	16711680
Yellow	255	255	0	65535
Pink	255	0	255	16711935
Turquoise	0	255	255	16776960
Brown	153	51	0	13209
Indigo	51	51	153	10040115
80% Gray	51	51	51	3355443

Converting colors

If you know a color's red, green, and blue component values, converting the color to a decimal color is easy. Just use VBA's RGB function. Assume three variables (r, g, and b), each of which represents a color component value between 0 and 255. To calculate the equivalent decimal color value, use a statement like this:

```
DecimalColor = RGB(r, g, b)
```

To perform this conversion in a worksheet formula, create this simple VBA wrapper function:

```
Function RGB2DECIMAL(R, G, B) As Long
'   Converts from RGB to decimal color
    RGB2DECIMAL = RGB(R, G, B)
End Function
```

The following example worksheet formula assumes the three color values are in A1:C1:

```
=RGB2DECIMAL(A1,B1,C1)
```

Converting a decimal color to its red, green, and blue components is a bit more complicated. Here's a function that returns a three-element array:

```
Function DECIMAL2RGB(ColorVal) As Variant
'   Converts a color value to an RGB triplet
'   Returns a 3-element variant array
    DECIMAL2RGB = Array(ColorVal \ 256 ^ 0 And 255, _
      ColorVal \ 256 ^ 1 And 255, ColorVal \ 256 ^ 2 _
      And 255)
End Function
```

To use the DECIMAL2RGB function in a worksheet formula, the formula must be entered as a three-cell array formula. For example, assume that cell A1 contains a decimal color value. To convert that color value to its RGB components, select a three-cell horizontal range and then enter the following formula. Press Ctrl+Shift+Enter to make it an array formula, and don't enter the braces.

```
{=DECIMAL2RGB(A1)}
```

If the three-cell range is vertical, you need to transpose the array, as follows:

```
{=TRANSPOSE(DECIMAL2RGB(A1))}
```

Figure 1-1 shows the DECIMAL2RGB and RGB2DECMAL functions at work in a worksheet.

The workbook shown in Figure 1-1 is available at this book's Web site.

	A	B	C	D	E	F	G	H
1	Decimal		Decimal-To-RGB				RGB-To-Decimal	
2	Color Value		R	G	B		Color Value	
3	0		0	0	0		0	
4	167,772		92	143	2		167,772	
5	335,544		184	30	5		335,544	
6	503,316		20	174	7		503,316	
7	671,088		112	61	10		671,088	
8	838,860		204	204	12		838,860	
9	1,006,632		40	92	15		1,006,632	
10	1,174,404		132	235	17		1,174,404	
11	1,342,176		224	122	20		1,342,176	
12	1,509,948		60	10	23		1,509,948	
13	1,677,720		152	153	25		1,677,720	
14	1,845,492		244	40	28		1,845,492	
15	2,013,264		80	184	30		2,013,264	
16	2,181,036		172	71	33		2,181,036	
17	2,348,808		8	215	35		2,348,808	
18	2,516,580		100	102	38		2,516,580	
19	2,684,352		192	245	40		2,684,352	
20	2,852,124		28	133	43		2,852,124	
21	3,019,896		120	20	46		3,019,896	
22	3,187,668		212	163	48		3,187,668	
23	3,355,440		48	51	51		3,355,440	
24	3,523,212		140	194	53		3,523,212	
25	3,690,984		232	81	56		3,690,984	
26	3,858,756		68	225	58		3,858,756	
27	4,026,528		160	112	61		4,026,528	

Figure 1-1: A worksheet that uses the DECIMAL 2RGB and RGB2 DECMAL functions.

More about decimal color values

You may be curious about how the 16,777,216 decimal color values are arranged. Color 0 is black, and color 16,777,216 is white, but what about all those colors in between?

It might help to think of the decimal color values as being generated by nested For-Next loops, as shown in the following code:

```
Sub GenerateColorValues()
    Dim Red As Long, Blue As Long, Green As Long
    Dim AllColors(0 To 16777215) As Long
    Dim ColorNum As Long
    ColorNum = 0
    For Blue = 0 To 255
       For Green = 0 To 255
          For Red = 0 To 255
             AllColors(ColorNum) = RGB(Red, Blue, Green)
             ColorNum = ColorNum + 1
```

```
            Next Red
        Next Green
    Next Blue
End Sub
```

After this procedure runs, the values in the AllColors array correspond exactly to the decimal color values used by Excel.

Understanding Grayscale

When you create worksheets and charts that are intended to be printed, it's important to remember that not everyone has a color printer. And even if your chart is printed on a color printer, it's possible that it may be photocopied, faxed, or viewed by someone who is colorblind (a condition that affects about 8 percent of the male population).

When content is printed on a non-color device, colors are converted to grayscale. Sometimes you'll be lucky, and your colors will display nicely when converted to grayscale. Other times, you won't be so lucky. For example, the columns in a chart may be indistinguishable when the colors are converted.

Every grayscale color has an equal component of red, green, and blue. Pure black is RGB(0, 0, 0). Pure white is RGB(255, 255, 255). Neutral gray is RGB(128, 128, 128). Using this color system produces 256 shades of gray.

To create a 256-color grayscale in a range of cells, execute the procedure that follows. It colors the background of cells in the range A1:A256, starting with black and ending with white. You might want to zoom out on the worksheet to see the entire range.

```
Sub GenerateGrayScale()
    Dim r As Long
    For r = 0 To 255
        Cells(r + 1, 1).Interior.Color = RGB(r, r, r)
    Next r
End Sub
```

Figure 1-2 shows the result. After decreasing the row heights and making column A wider, you've got a nice gradient that goes from pure black to pure white. I don't know what you can do with it, but it looks pretty good.

Figure 1-2: Cells displaying 256 shades of gray.

Converting colors to gray

So what if you want to convert colors to grayscale? One approach is to simply average the Red, Green, and Blue components of a color and use that single value for the Red, Green, and Blue components of its grayscale equivalent. That method, however, doesn't take into account the fact that different colors are perceived as varying levels of brightness. For example, green is perceived to be brighter than red, and red is perceived to be brighter than blue.

Perceptual experiments have arrived at the following "recipe" to convert an RGB color value to a grayscale value (go easy on the salt):

- 28.7% of the red component
- 58.9% of the green component
- 11.4% of the blue component

For example, consider color value 16751001, a shade of violet that corresponds to RGB(153, 153, 255). Applying the factors listed previously, the RGB values are

- Red: 28.7% × 153 = 44
- Green: 58.9% × 153 = 90
- Blue: 11.4% × 255 = 29

The sum of these values is 163. Therefore, the corresponding grayscale RGB value for color value 16751001 is RGB(163, 163, 163).

Following is a VBA function that does the math for you. This function accepts a decimal color value as its argument and returns the corresponding grayscale decimal value.

```
Function Grayscale(color) As Long
    Dim r As Long, g As Long, b As Long
    r = (color \ 256 ^ 0 And 255) * 0.287
    g = (color \ 256 ^ 1 And 255) * 0.589
    b = (color \ 256 ^ 2 And 255) * 0.114
    Grayscale = RGB(r + g + b, r + g + b, r + g + b)
End Function
```

Viewing charts as grayscale

Unfortunately, Excel's print preview feature doesn't do grayscale conversion. For example, if you have a black and white laser printer, previewing your print job shows colors — not the grayscale that is actually produced by your printer.

By the way, the Sheet tab of the Page Setup dialog box (displayed by clicking the dialog box launcher in the Page Layout➪Page Setup group) has an option labeled Black And White. When checked, your charts are printed in true black and white — which is not the same as grayscale. Colors are converted to patterns that consist of black and white (no gray). Note that this setting applies only to charts and other graphic objects. When printing in Black And White mode, cells' colors are ignored.

Here's a technique that lets you see how an embedded chart looks when it's converted to grayscale:

1. Select the chart.
2. Press Ctrl+C to copy the chart to the Clipboard.
3. Click a cell and choose Home➪Clipboard➪Paste➪Picture(U).
4. Select the pasted picture and choose Picture Tools➪Format➪Adjust➪Color and then choose the Grayscale color mode from the Recolor section of the drop-down gallery (see Figure 1-3).

These steps are automated in the macro that follows. The ShowChartAsGrayScale procedure copies the active chart as a picture and converts the picture to grayscale. After you've determined whether the colors are satisfactory for grayscale printing, you can delete the picture.

```
Sub ShowChartAsGrayScale()
'   Copies the active chart as a grayscale picture
'   Embedded charts only
    If ActiveChart Is Nothing Then
        MsgBox "Select a chart."
        Exit Sub
    End If
    ActiveChart.Parent.CopyPicture
    ActiveChart.Parent.TopLeftCell.Select
    ActiveSheet.Pictures.Paste
    ActiveSheet.Pictures(ActiveSheet.Pictures.Count). _
      ShapeRange.PictureFormat.ColorType = _
      msoPictureGrayscale
End Sub
```

A workbook with this example is available at this book's Web site.

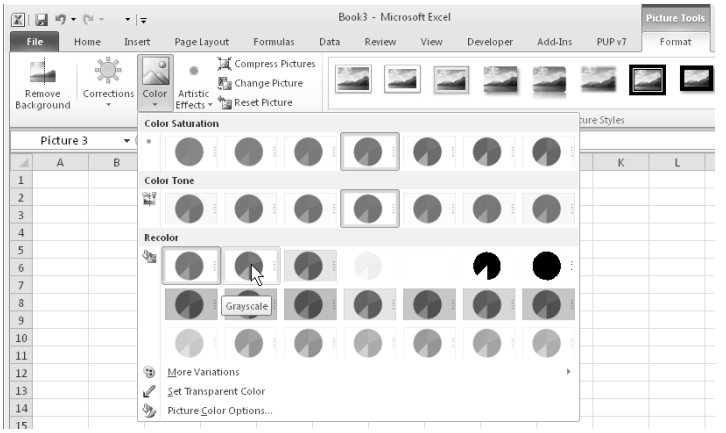

Figure 1-3: Converting a picture of a chart to grayscale.

Don't overlook the built-in grayscale chart styles. The grayscales used in these styles seem to be optimized for showing variations in chart elements.

Experimenting with Colors

Figure 1-4 shows a workbook that I created that deals with colors. If you're at all confused about how the RGB color model works, spending some time with this color demo workbook will probably make it all very clear. Or, it may give you a headache.

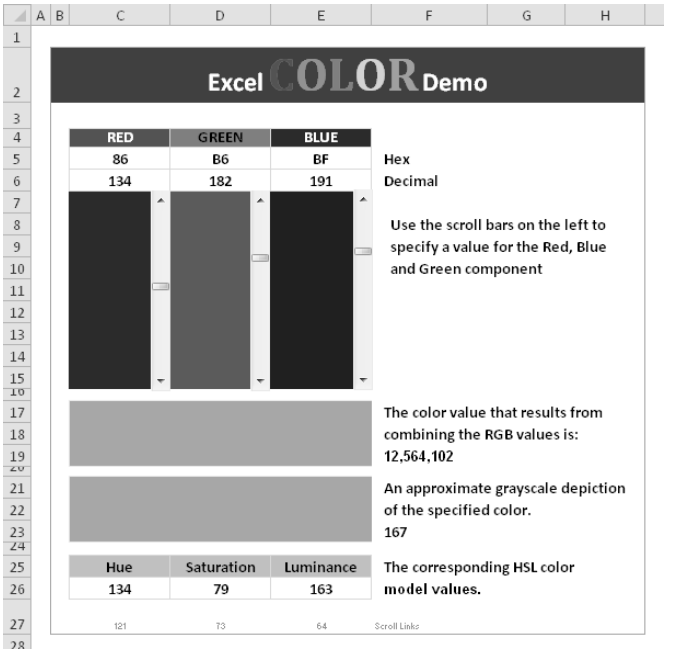

Figure 1-4: This workbook demonstrates how red, green, and blue colors combine.

The workbook shown in Figure 1-4 is available on the companion CD-ROM.

This workbook contains three vertical scrollbars, each of which controls the background color of a range. Use these scrollbars to specify the red, green, and blue components for a color to values between 0 and 255. Moving the scrollbars changes several areas of the worksheet:

- The cells above the scrollbars (in rows 5 and 6) display the color components in hexadecimal (00–FF) and in decimal (0–255). Hexadecimal RGB color values are often used when specifying colors for HTML documents.
- The ranges next to each scrollbar change intensity, corresponding to the scrollbar's position (that is, the value of the color component).
- A range below the scrollbars depicts the combined color, determined by the RGB values you specify.
- A cell displays the decimal color value.
- Another range depicts the color's approximate appearance when it's converted to grayscale.
- Cells in row 26 show the corresponding HSL color values (HSL is another color model used in some software).

Understanding Document Themes

A significant new feature introduced in Excel 2007 was document themes. With a single mouse click, you can change the entire look of a document. A document theme consists of three components: colors, fonts, and effects (for graphic objects). The rationale for using themes is that they may help users produce better-looking and more consistent documents. A theme applies to the entire workbook, not just the active worksheet.

About document themes

Microsoft Office 2010 ships with about 40 document themes, and you can also download or create additional themes. The Ribbon includes several style galleries (for example, the Chart Styles gallery). The styles available in these galleries vary depending on which theme is assigned to the document. If you apply a different theme to the document, the document changes to reflect the new theme's colors, fonts, and effects.

If you haven't explored the wonderful world of document themes, open the workbook named document theme demo.xlsx found on this book's Web site. This workbook contains a range that shows each theme color, two shapes, text (using the headings and body fonts), and a chart. Choose Page Layout➪Themes➪Themes Gallery to see how the worksheet changes with each theme.

Users can also mix and match theme elements. For example, you can use the colors from one theme, the fonts from another theme, and the effects from yet a different theme. In addition, you can create a new color set or a new font set. You can save these customized themes and then apply them to other workbooks.

The concept of document themes is based on the notion that users will apply little, if any, non-theme formatting to the document. If the user applies colors or fonts that aren't part of the current theme, this formatting will not be modified if a new theme is applied to the document. Therefore, it's still very easy to create an ugly document with mismatched colors and too many different fonts.

Understanding document theme colors

When a user applies a color to a cell or object, the color is selected from a Ribbon control like the one shown in Figure 1-5. The control displays the 60 theme colors (10 columns by 6 rows) plus 10 additional standard colors.

Clicking the More Colors option displays the Color dialog box, in which you can specify any of the 16,777,216 available colors.

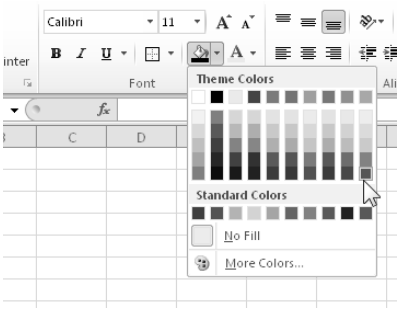

Figure 1-5: A color-selection control.

The 60 theme colors are identified by pop-up ToolTips. For example, the color in the second row of the sixth column is affectionately known as "Accent 2, Lighter 80%."

The first row in each column of colors contains the "pure" color. Below each pure color are six "tint and shade" variations. Figure 1-6 shows the color descriptions for the color picker controls.

Background 1	Text 1	Background 2	Text 2	Accent 1	Accent 2	Accent 3	Accent 4	Accent 5	Accent 6
Darker 5%	Lighter 50%	Darker 10%	Lighter 80%	Lighter 80%	Lighter 80%	Lighter 80%	Lighter 80%	Lighter 80%	Lighter 80%
Darker 15%	Lighter 35%	Darker 25%	Lighter 80%	Lighter 60%	Lighter 60%	Lighter 60%	Lighter 60%	Lighter 60%	Lighter 60%
Darker 25%	Lighter 25%	Darker 50%	Lighter 80%	Lighter 40%	Lighter 40%	Lighter 40%	Lighter 40%	Lighter 40%	Lighter 40%
Darker 35%	Lighter 15%	Darker 75%	Darker 25%	Darker 25%	Darker 25%	Darker 25%	Darker 25%	Darker 25%	Darker 25%
Darker50%	Lighter 5%	Darker 90%	Darker 50%	Darker 50%	Darker 50%	Darker 50%	Darker 50%	Darker 50%	Darker 50%

Figure 1-6: Color descriptions for Excel color pickers.

Keep in mind that these are generic color names; they remain the same even if a different document theme is applied. The document theme colors actually consist of the ten colors displayed in the top row (four text/background colors and six accent colors), and each of these ten colors has five tint/shade variations.

You may find it enlightening to record a macro while you change the fill color and text color of a range. Following is a macro that I recorded when a range was selected. For the fill color, I chose "Accent 2, Darker 25%," and for the text color, I chose "Text 2, Lighter 80%."

```
Sub ChangeColors()
    With Selection.Interior
        .Pattern = xlSolid
        .PatternColorIndex = xlAutomatic
        .ThemeColor = xlThemeColorAccent2
        .TintAndShade = -0.249977111117893
        .PatternTintAndShade = 0
    End With
    With Selection.Font
        .ThemeColor = xlThemeColorLight2
        .TintAndShade = 0.799981688894314
    End With
End Sub
```

First of all, you can safely ignore the three pattern-related properties (Pattern, PatternColorIndex, and PatternTintAndShade). These properties refer to the ugly, old-fashioned (but still supported) cell patterns, which you can specify in the Fill tab of the Format Cells dialog box. These statements are included in the recorded macro to maintain any existing pattern that may exist in the range.

The recorded macro, after I deleted the three pattern-related properties (and added comments), is

```
Sub ChangeColors()
    With Selection.Interior
         '(Accent 2, Darker 25%)
        .ThemeColor = xlThemeColorAccent2
        .TintAndShade = -0.249977111117893
    End With
    With Selection.Font
         '(Text 2, Lighter 80%)
        .ThemeColor = xlThemeColorLight2
        .TintAndShade = 0.799981688894314
    End With
End Sub
```

As you can see, each color is specified in terms of a ThemeColor property and a TintAndShade property. The ThemeColor property is easy enough to decipher. Property values are assigned using built-in constants, and these values correspond to the column number of the 10 x 6 theme color table. For example, xlThemeColorAccent2 has a value of 6. But what about the TintAndShade property?

The TintAndShade property can have a value between –1 and +1. A value of –1 results in black, and a value of +1 results in white. A TintAndShade property value of 0 gives the pure color. In other words, as the TintAndShade

value goes negative, the color gets increasingly darker until it's pure black. As the TintAndShade value goes positive, the color gets increasingly lighter until it's pure white. The TintAndShade value corresponds to the color name displayed in the color selection controls.

If the color variation is expressed as "Darker," the TintAndShade property value is negative. If the color variation is expressed as "Lighter," the TintAndShade property value is positive.

By the way, I don't know why the TintAndShade values have such a high level of precision in recorded macros. It's certainly not necessary. For example, a TintAndShade property value of –0.249977111117893 produces the same visual result as a TintAndShade property value of –0.25. It's just one of those VBA mysteries.

Displaying all theme colors

I wrote a macro that displays all 60 theme color variations in a range of cells. These 60 colors are the colors that appear in the color selection controls.

```
Sub ShowThemeColors()
  Dim r As Long, c As Long
  For r = 1 To 6
    For c = 1 To 10
        With Cells(r, c).Interior
        .ThemeColor = c
        Select Case c
            Case 1 'Text/Background 1
            Select Case r
                Case 1: .TintAndShade = 0
                Case 2: .TintAndShade = -0.05
                Case 3: .TintAndShade = -0.15
                Case 4: .TintAndShade = -0.25
                Case 5: .TintAndShade = -0.35
                Case 6: .TintAndShade = -0.5
            End Select
        Case 2 'Text/Background 2
            Select Case r
                Case 1: .TintAndShade = 0
                Case 2: .TintAndShade = 0.5
                Case 3: .TintAndShade = 0.35
                Case 4: .TintAndShade = 0.25
                Case 5: .TintAndShade = 0.15
                Case 6: .TintAndShade = 0.05
            End Select
```

```
        Case 3 'Text/Background 3
            Select Case r
                Case 1: .TintAndShade = 0
                Case 2: .TintAndShade = -0.1
                Case 3: .TintAndShade = -0.25
                Case 4: .TintAndShade = -0.5
                Case 5: .TintAndShade = -0.75
                Case 6: .TintAndShade = -0.9
            End Select
        Case Else  'Text/Background 4, and Accent 1-6
            Select Case r
                Case 1: .TintAndShade = 0
                Case 2: .TintAndShade = 0.8
                Case 3: .TintAndShade = 0.6
                Case 4: .TintAndShade = 0.4
                Case 5: .TintAndShade = -0.25
                Case 6: .TintAndShade = -0.5
            End Select
        End Select
       Cells(r, c) = .TintAndShade
        End With
    Next c
  Next r
End Sub
```

Figure 1-7 shows the result of executing the ShowThemeColors procedure (it's more impressive in color). If you switch to a different document theme, the colors will be updated to reflect those in the new theme.

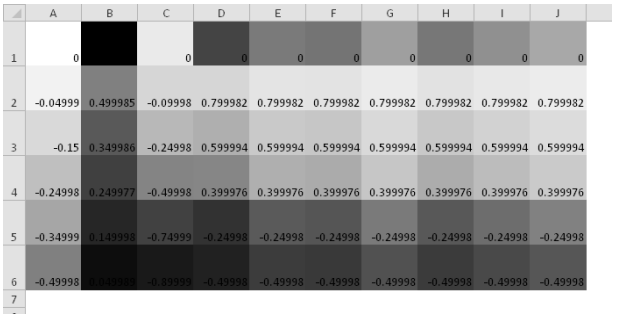

Figure 1-7: A VBA macro generated these theme colors.

This example is available at this book's Web site.

Earlier in this chapter, I described how to change the fill color of a range by setting the Color property of the Interior object. As I noted, using the VBA RGB function makes this easier. These two statements demonstrate how to change the fill color of a range (they both have the same result):

```
Range("A1:F24").Interior.Color = 5913728
Range("A1:F24").Interior.Color = RGB(128, 60, 90)
```

It's important to understand that assigning a color in this way doesn't make it a theme color. In other words, if the user switches to a new document theme, the range A1:F24 won't change colors. To change cell colors in a way that is consistent with themes, you *must* use the ThemeColor and (optionally) the TintAndShade property.