This TPW unit includes a couple of objects that allow you to use BWCC style bitmapped buttons in your own programs without the BWCC DLL. There are several reasons why you might want to do this. You may have a need to modify a program that previously used BWCC buttons to run without BWCC. This is what happened to me, and is why I wrote this unit. Using the BWCC style buttons, it is easy to implement a toolbar with pushbuttons for commonly used functions, and you may not want to have to distribute the entire BWCC.DLL with your program just for that.

I have not duplicated the full functionality of the BWCC.DLL. I don't believe in reinventing the wheel if I don't have to. This version of the unit does not support the EGA resolution buttons, but since the source code is provided, you shouldn't have any major problems adding it yourself if you need it.

To use this unit, you will need a set of three bitmaps for each button. These bitmaps will be what is actually displayed for each button in its three possible states: Normal, Pressed, and Focused. The bitmap ID's are numbered with the same scheme that Borland uses for BWCC buttons. The normal bitmap is ButtonID+1000, the pressed bitmap is ButtonID+3000, and the focused bitmap is ButtonID+5000. For example, if you have a button with an ID of 500, the bitmaps ID's for Normal, Pressed, and Focused will be 1500, 3500, and 5500, respectively.

Of course, being a programmer like I am, I would suggest finding a way to get the computer to do the work for you. One program that is available in the BPASCAL forum to make this job much easier is BUTTON.ZIP, by N. Waltham. It allows you to easily create and edit the bitmaps as well as the font, color, and style of the button text.

Once you have the bitmaps in your resource file, you will also need a dialog to display the bitmapped buttons in. I'll assume that you're using Borland's Resource Workshop as that's the main Resource Editor that I'm familiar with. First, place a normal button in the dialog where you want your bitmapped button to be. Then use the Align Size menu option to change the size to 32 units wide by 20 units high. It seems that Resource Workshop's dialog units are half what the actual pixel resolution is because the size of the bitmap that will be going into this 32 x 20 button will be 64 pixels by 40 pixels.

When you get all of your buttons placed where you want them to be, then you need to edit each one individually and change it's style from pushbutton or default pushbutton to owner draw. The button will probably not be visible after this if it's not selected, but it is still there. After you're finished editing the dialog, save your work and exit.

To actually use the button in your dialog, you will need to make your dialog object a descendant of TOwnerDialog. Then, in your dialog's Init constructor, you use the NewButton procedure to create an object for each owner draw button. The NewButton procedure has two parameters. The first is the ID of the button, and the second is a Boolean value which specifies if the button is a default button or not. If it's True, then the button will be the first button in the dialog to get the focus when the dialog is created.

There is one last step you will have to do. So far, you have seen how to get your button to be displayed in your dialog. You will also have to create a method in your dialog procedure to respond to this button. This procedure will be of the format:

```
procedure Button1 (var Msg : TMessage); virtual id_First + id_Button;
```

where Button1 is the name of your response procedure for the button, and the id_Button is the actual ID of the button. This is the procedure you would put your code in that responds to the press of the button.

That's the basic procedure to use for using this unit. I have included the sample program so that you can see it in action. This is of course the best way to learn how to do something.

I would welcome any comments, good or bad, on this code. The source code itself is Copyright © 1993 Todd T. Snoddy, but you may use it in your own programs as you see fit. I will accept no liability for any damages which may occur from the use of the unit. If you use it, then you accept full responsibility for it's actions. Although nothing in the source code should cause any kind of damage, I still feel compelled to include the standard basic disclaimer.

Todd T. Snoddy
Compuserve ID 71044,1653