



[Desktop Development](#) » [Button Controls](#) » [Owner-draw buttons](#)

## CxShadeButton

By [Davide Pizzolato](#)

An owner-drawn button class that gives a professional look to your buttons.

VC6Win2K, MFC, Dev

Posted: **18 May 2001**

Updated: **5 Nov 2001**

Views: **373,218**

Bookmarked: **122 times**

122 votes for this article.

Popularity: 9.68 Rating: 4.64 out of 5

- [Download source files - 16 Kb](#)
- [Download demo project - 45 Kb](#)



### Introduction

With this class you can easily give a professional look to your buttons in few steps. **No bitmap resources are needed** - all bitmaps are generated at runtime. Parts of the code come from the [CxSkinButton](#) article. The goal is to replace the standard buttons, check boxes and radio buttons with minimal modifications in the application code.

1. Add "xShadeButton.cpp" and "xShadeButton.h" to the project.
2. Include "xShadeButton.h" in the header file where the controls are defined
3. Create (or edit) a member variable for each button you want to customize as `CxShadeButton`. If the Class Wizard doesn't show the `CxShadeButton` type, select `CButton` and then edit the code manually.
4. In the window initialization add the `CxShadeButton` methods:

```

BOOL CxShadeButtonDemoDlg::OnInitDialog()
{
    // ...
    m_btn1.SetTextColor( RGB(255,0,0) );
    m_btn1.SetToolTipText( "Button1" );
    m_btn1.SetShade( SHS_DIAGSHADE, 8,10,5, RGB(55,255,55) );
    // ...
}

```

### CxShadeButton Class Members & Operations

`CxShadeButton` is derived from `CButton`. The `BS_OWNERDRAW` style is added automatically, you don't need to set the "Owner draw" property in the resource editor. You can change some styles (flat, push-like, text alignment, group,...) using the resource editor, however not all the styles are currently supported. If you change the aspect of the button at runtime, to avoid flicker first call the functions that don't cause invalidation (like `SetShade`, `SetIcon` or `SetFont`) and than invalidate the button, for example with `SetWindowText`, or directly with `Invalidate`.

```
void SetShade(UINT shadeID=0, BYTE granularity=8,
             BYTE highlight=10, BYTE coloring=0, COLORREF color=0);
```

Generates the button bitmaps.

**Important**

- **shadeID** : can be one of these effects:

```
SHS_NOISE = 0
SHS_DIAGSHADE = 1
SHS_HSHADE = 2
SHS_VSHADE = 3
SHS_HBUMP = 4
SHS_VBUMP = 5
SHS_SOFTBUMP = 6
SHS_HARDBUMP = 7
SHS_METAL = 8
```

- **granularity** : this parameter add an uniform noise to the button bitmaps. A good value is from 5 to 20; 0 to disable the effect. The noise has a positive effect because it hides the palette steps.
- **highlight** : sets the highlight level when the mouse is over the button. A good value is from 5 to 20; 0 to disable the effect.
- **coloring** : sets the percentage of `color` to blend in the button palette. The value can range from 0 to 100; 0 to disable the effect.
- **color** : if `coloring` is greater than zero, `color` is mixed with the standard button colors.

**Remarks** : the `coloring` and `color` parameter should be used carefully to guarantee a good aspect in all the situations.

```
void SetToolTipText(CString s, CString sDown="");
```

Sets or changes the tool tip text.

**nice**

- **s**: String displayed in normal state.
- **sDown**: (optional) Specifies a second text to be displayed when a check box or radio button is checked.

```
COLORREF SetTextColor(COLORREF new_color);
```

Sets or changes the button text color. Returns the previous button text color.

**nice**

```
void SetIcon(UINT nIcon, UINT nIconAlign=BS_CENTER, UINT nIconDown=0,
            UINT nIconHighLight=0);
```

Similar to the `BS_ICON` style.

**nice**

- **nIcon** : ID number of the icon resource
- **nIconAlign** : icon alignment, can be one of the following values:  

```
BS_CENTER
BS_LEFT
BS_RIGHT
```
- **nIconDown** : (optional) ID number of the icon resource displayed when the button is checked.
- **nIconHighLight** : (optional) ID number of the icon resource displayed when the mouse pointer is over the button.

**Remarks** : the button text is automatically placed so that the icon and the text don't overlap.

```
bool SetFont(CString sFontName, long lSize=0, long
            lWeight=400, BYTE bItalic=0, BYTE bUnderline=0);
bool SetFont(LOGFONT* pNewStyle); / LOGFONT* GetFont();
```

Changes the text font.

nice

- **sFontName** : specifies the typeface name of the font.
- **lSize** : (optional) text height
- **lWeight** : (optional) text weight can range from 0 to 1000; 100=thin, 300=light, 400=normal, 700=bold
- **bItalic** : (optional) italic style
- **bUnderline** : (optional) underline style

**Remarks** : use **GetFont/SetFont** with a LOGFONT structure to get/set the complete attributes of the font. **GetFont** returns NULL if the button is using the default system font.

```
void SetTextAlign(UINT nTextAlign=BS_CENTER);
```

- **nTextAlign** : button text alignment, can be one of the following values:

optional

```
BS_CENTER
BS_LEFT
BS_RIGHT
```

```
void SetFlat(bool bFlag);
```

- **bFlag** : sets the border style:  
**FALSE** = standard 3D border.  
**TRUE** = flat border.

optional

## Release History

### v1.00 - 12/05/2001

- basic implementation and interface.

### v1.10 - 23/05/2001

- added text shift on button down.  
- fixed many CxDib bugs.  
- fixed SHS\_HARDBUMP bug.  
- added icon support.  
- added text alignment.  
- added flat style.

### v1.20 - 23/06/2001

- fixed keyboard shortcut bug.  
- check box & radio button add on.  
- 2nd icon & 2nd tooltip add on.  
- memory DC for painting operations.

### v1.30 - 03/08/2001

- fixed SetIcon bug.  
- added Font support.

### v1.40 - 23/09/2001

- fixed memory leakage bug in DrawItem() and SetIcon().  
- fixed second tooltip initialization bug.  
- fixed OnLButtonUp() problem with drag & drop.  
- added multiline tooltip support.

### v1.41 - 4/11/2001

- fixed memory leakage bug in SetIcon() and in the destructor.  
- added 3rd icon for highlighted state.

## Compatibility

Win95, WinNT = Yes, requires IE3.0 or higher

Win98, WinME, W2K, WinXP = Yes

For any questions, e-mail to: [ing.davide.pizzolato@libero.it](mailto:ing.davide.pizzolato@libero.it)

Thanks to all the Code Project developers!

Special thanks to:

[Milan Gardian](#) for mouse and keyboard tracking code.

[Davide Calabro](#) for [CButtonST](#) code snippets.

[Rainer Mangold](#) for radio-button and check-box code.

[Jeremy Davis](#), [Andre Brogli](#), [Richard Cunday](#), [Shanker Chandrabose](#), [Luis](#),

[Gilad](#), [Rui Lopes](#), [Tom Archer](#), [Tommy Svensson](#), [David Scambler](#),

[Orioli Alessandro](#), [João Filipe de Castro Ferreira](#), [Jesper Kinnås](#), [Derek Lakin](#) for suggestions, debugging & support.


## License

This article has no explicit license attached to it but may contain usage terms in the article text or the download files themselves. If in doubt please contact the author via the discussion board below.

A list of licenses authors might use can be found [here](#)

## About the Author


### Davide Pizzolato

Location:  Italy



Member

## Discussions and Feedback

 **78 messages** have been posted for this article. Visit

<http://www.codeproject.com/KB/buttons/cxshadebutton.aspx> to post and view comments on this article, or click [here](#) to get a print view with messages.

[PermaLink](#) | [Privacy](#) | [Terms of Use](#)

Last Updated: 5 Nov 2001

Editor: [Chris Maunder](#)

Copyright 2001 by Davide Pizzolato  
Everything else Copyright © [CodeProject](#), 1999-2009  
Web16 | [Advertise on the Code Project](#)