How can I use WS_CLIPCHILDREN and still be able to draw a control with a transparent background?

devblogs.microsoft.com/oldnewthing/20181005-00

October 5, 2018

Raymond Chen

A customer was using an MFC `CHtmlDialog` as a child dialog and found that they needed to add the `WS_ CLIPCHILDREN` style to ensure that the contents appeared on the screen; otherwise, the parent dialog would paint its background over the child dialog, and the child dialog would consequently appear to vanish.

On the other hand, they also had some transparent image controls on the parent dialog, and adding the `WS_ CLIPCHILDREN` style prevented those controls from drawing transparently, meaning that instead of having the parent dialog's background show through, it just used a black background.

How can they get the best of both worlds, with `WS_ CLIPCHILDREN` style applying to some child controls but not others?

The controls that need the background to show through can ask the parent dialog to draw its background. They can do this the same way standard controls do it, by sending a `WM_ CTL-COLORxxx` message to the parent to request the background brush and text colors. Of course, since it's an image control, the text colors aren't interesting.

```
case WM_PAINT:
{
 PAINTSTRUCT ps;
 HDC hdc = BeginPaint(hwnd, &ps);

 HBRUSH hbrBackground = (HBRUSH)
       SendMessage(GetParent(hwnd), WM_CTLCOLORSTATIC,
             (WPARAM)hdc, (LPARAM)hwnd);

 FillRect(hdc, &ps.rcPaint, hbrBackground);

 DrawTheImage(...);

 EndPaint(hwnd, &ps);
 return 0;
}
```

This technique assumes that the parent dialog responds to `WM_ CTLCOLORxxx` messages. If the parent dialog has standard colors, then the `DefWindowProc` function will return standard colors, and everything will work out. But if the parent dialog uses custom colors, it will have to add additional message handlers for these messages.

Alternatively, the image controls could render its background by forwarding the `WM_ ERASEBKGND` message to its parent. Since the child and parent have different coordinates, you'll have to do some coordinate manipulation to get the parent to receive a device context whose origin is what it expects.

```
case WM_ERASEBKGND:
{
 HDC hdc = (HDC)wParam;

 HWND hwndParent = GetParent(hwnd);

 POINT ptOffset{};
 MapWindowPoints(hwnd, hwndParent, &ptOffset, 1);
 OffsetWindowOrgEx(hdc, ptOffset.x, ptOffset.y, &ptOrig);
 LRESULT lres = SendMessage(hwndParent, WM_ERASEBKGND, wParam, lParam);
 SetWindowOrgEx(hdc, ptOrig.x, ptOrig.y, nullptr);
 return lres;
}
```

Note that neither of these techniques actually draw the control transparently. Rather, they attempt to simulate the effect by asking the parent to draw its background into the control. It also means that if the parent draws something interesting in its `WM_ PAINT` handler, it won't show up in the "background" of the control.

Nevertheless, this usually works well enough for most purposes.

Raymond Chen

Follow