Controlling designer appearance of double-buffered owner-drawn UserControl in C#/.NET 2.0

Asked 16 years, 1 month ago Modified 14 years, 3 months ago Viewed 1k times

2

I have an owner-drawn UserControl where I've implemented double-buffering. In order to get double-buffering to work without flicker, I have to override the OnPaintBackground event like so:

```
protected override void OnPaintBackground(PaintEventArgs e)
{
    // don't even have to do anything else
}
```

This works great at runtime. The problem is that when I have an instance of the control on a form at design time, it becomes a black hole that shows trails of whatever windows are dragged over it (because the override of the OnPaintBackground event also governs design-time appearance). It's just a cosmetic problem, but it's visually jarring and it always leads new developers on the project to assume something has gone horribly wrong.

Is there any way to have an overridden method like this not be overridden at design time, or is there another solution?

c# .net user-controls .net-2.0

Share Improve this question Follow

asked Nov 16, 2008 at 2:03

MusiGenesis

75.3k ● 41 ● 197 ● 338

Is there some reason you can't use the built-in double buffering? – Greg D Nov 16, 2008 at 2:34

Yes - it doesn't work. If you create a user control with DoubleBuffered = true and then do a bunch of drawing operations on it, you will get flicker. I'm really not sure what that property even does. – MusiGenesis Nov 16, 2008 at 2:45

3 Answers

Sorted by: Highest score (default)

3

Steven Lowe's solution unfortunately cover all scenarios, espcially when user controls come into the picture.

The this.DesignMode flag is very deceptive. Its only scope is to check if the direct parent is within the designer.

For instance, if you have a Form A, and a UserControl B, in the designer:

- A.DesignMode is true when viewed in designer
- B.DesignMode is false when viewing A, but true when looking directly at B in the designer.

The solution to this is a special flag to check (sorry for the ugly C++ code):

```
if(this->DesignMode ||
   LicenseManager::UsageMode == LicenseUsageMode::Designtime)
   return;
```

This variable will always show the proper design boolean.

Share Improve this answer Follow

answered Sep 2, 2009 at 17:26

greggorob64

2,557 ● 2 ● 27 ● 58

Unfortunately, you are right with your solution. I first tried out Steven Lowe's solution and ran into the problems you described. +1 for you. – Marcel Feb 3, 2010 at 20:44

How the hell do I put code in a comment! I've found a much more powerful way of detecting design time detection. I add the following property to pretty much every form\usercontrol I design, and even works properly in constructors\destructors, which are real problem regions for the designer. __ property bool IsDesignMode { bool get () { return (this->DesignMode == true || LicenseManager::UsageMode == LicenseUsageMode::Designtime || AppDomain::CurrentDomain->FriendlyName->Equals(L"DefaultDomain")); } } – greggorob64 Feb 5, 2010 at 14:16

2

```
if (this.DesignMode)
{
    return;    //or call base.OnPaintBackground()
}
```

Share Improve this answer Follow

answered Nov 16, 2008 at 2:25

Steven A. Lowe

61.1k ● 19 ● 135 ● 204

Unfortunately, your soloution does not work in all cases, see greggorob64's post. I first came into trouble using your code. -1, sorry. – Marcel Feb 3, 2010 at 20:49

2

The solution of `greggorob64` in C#:

```
if (DesignMode || LicenseManager.UsageMode == LicenseUsageMode.Designtime)
{
    return;
}
```

Share

Improve this answer

Follow

edited Sep 24, 2010 at 1:31

MusiGenesis

75.3k ● 41 ● 197 ● 338

answered Feb 3, 2010 at 20:47

Marcel

15.7k ● 21 ● 102 ● 161