|  |
| --- |
| 避免内存泄漏 |

当在托管的 XAML 应用程序中使用 Win2D 控件，护理必须采取避免可以防止过被垃圾回收器回收这些控件的引用计数周期。

http://microsoft.github.io/Win2D/icons/SectionExpanded.png你有一个问题，如果...

您正在使用 Win2D 从一种.NET 语言如 VB (不是 c + +)

您使用 Win2D XAML 控件之一:

* [CanvasControl](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fmicrosoft.github.io%2FWin2D%2Fhtml%2FT_Microsoft_Graphics_Canvas_UI_Xaml_CanvasControl.htm)
* [CanvasVirtualControl](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fmicrosoft.github.io%2FWin2D%2Fhtml%2FT_Microsoft_Graphics_Canvas_UI_Xaml_CanvasVirtualControl.htm)
* [CanvasAnimatedControl](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fmicrosoft.github.io%2FWin2D%2Fhtml%2FT_Microsoft_Graphics_Canvas_UI_Xaml_CanvasAnimatedControl.htm)
* [CanvasSwapChainPanel](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fmicrosoft.github.io%2FWin2D%2Fhtml%2FT_Microsoft_Graphics_Canvas_UI_Xaml_CanvasSwapChainPanel.htm)
* 你订阅 Win2D 控件的事件的 (例如 绘制，CreateResources，SizeChanged...)
* 您的应用程序多个 XAML 页之间来回移动

如果满足所有这些条件，引用计数周期将阻止 Win2D 控制过被垃圾收集。新的 Win2D 资源分配每次应用程序移动到一个不同的页，但旧的永远不会被释放，所以内存泄漏。要避免此问题，必须添加代码来显式地打破这种循环。

http://microsoft.github.io/Win2D/icons/SectionExpanded.png如何修复它

打破的引用计数循环，让你的页面进行垃圾回收:

挂钩 XAML 页面，其中包含 Win2D 控制空载的事件

在空载处理程序，调用 RemoveFromVisualTree Win2D 控制

在卸载处理程序中，释放 (通过设置为 Nothing) 对 Win2D 控制的任何显式引用

示例代码:

**VB**

[副本](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fmicrosoft.github.io%2FWin2D%2Fhtml%2FRefCycles.htm%23)

Private Sub page\_Unloaded(sender As Object, e As RoutedEventArgs)

Me.canvas.RemoveFromVisualTree

Me.canvas = Nothing

End Sub

工作示例，请参见任何[示例库](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fgithub.com%2FMicrosoft%2FWin2D-Samples%2Ftree%2Fmaster%2FExampleGallery%2FShared)演示页面。

http://microsoft.github.io/Win2D/icons/SectionExpanded.png如何测试周期是否有泄漏

若要测试是否您的应用程序正确打破使循环，将添加到包含 Win2D 控件的任何 XAML 页的终结器方法:

**VB**

[副本](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fmicrosoft.github.io%2FWin2D%2Fhtml%2FRefCycles.htm%23)

Protected Overrides Sub Finalize()

Debug.WriteLine("回收画布")

MyBase.Finalize()

End Sub

在您的应用程序的构造函数，建立了一个计时器，它将使确定垃圾收集发生在固定的时间间隔:

**VB**

[副本](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fmicrosoft.github.io%2FWin2D%2Fhtml%2FRefCycles.htm%23)

Dim gcTimer As New DispatcherTimer

AddHandler gcTimer.Tick, Sub() GC.Collect

gcTimer.Interval = TimeSpan.FromSeconds(1)

gcTimer.Start

导航到页面，然后从它到其他页面上。如果所有周期都碎都了，你会看到在 Visual Studio 中的 Debug.WriteLine 输出输出窗格在一两秒内。

请注意，调用 GC。收集具有破坏性和伤害性能，所以您应该删除此测试代码，只要你完成测试泄漏!

残酷的细节

对象 A 具有参考 B，同时 B 也有提及 A.时发生一个周期或者当 B 和 B 的引用引用 C，而 C 引用 A 等。

当订阅事件的 XAML 控制，这种周期是几乎不可避免:

* XAML 页保留对它所包含的所有控件的引用
* 控件保持对已订阅它们的事件处理程序委托的引用
* 每个委托保存到其目标实例的引用
* 事件处理程序通常是实例方法的 XAML 页类，所以他们目标实例引用点返回到 XAML 页面，创建一个循环

如果在.NET 中实现所有涉及的对象，这种循环不是问题因为.NET 垃圾回收，垃圾回收算法是能够识别并回收的对象组，即使它们链接在一个周期。

与.NET 人，不同的是 c + + 管理内存的引用计数，是无法检测和回收循环对象。尽管这种限制，使用 Win2D 的 c + + 应用程序没有任何问题，因为 c + + 事件处理程序默认为控股弱而不是他们的目标实例的强引用。因此页面引用该控件，而控件引用的事件处理程序委托，此委托未引用返回到页面，所以没有任何周期。

问题在于当.NET 应用程序使用 c + + WinRT 组件如 Win2D:

* XAML 页是应用程序的一部分，所以使用垃圾回收
* Win2D 控制在 c + + 中实现，因此，使用引用计数
* 事件处理程序委托是应用程序的一部分，所以使用垃圾回收，认为对其目标实例的强引用

一个周期是存在的但参加这个周期的 Win2D 对象不使用.NET 垃圾回收。这意味着垃圾收集器是无法看到整个链，因此它不能检测或回收的对象。当这发生时，应用程序必须通过显式打破循环帮忙。这可以通过释放所有引用从页面到控件 (如上文所建议) 或通过都释放从控制到可能指向页面 (使用页卸载事件取消订阅所有事件处理程序) 的事件处理程序委托的所有引用。