Future proofing a large UI Application - MFC with 2008 Feature pack, or C# and Winforms?

Asked 16 years, 4 months ago Modified 15 years, 10 months ago

Viewed 4k times

12

My company has developed a long standing product using MFC in Visual C++ as the defacto standard for UI development. Our codebase contains ALOT of legacy/archaic code which must be kept operational. Some of this code is older than me (originally written in the late 70s) and some members of our team are still on Visual Studio 6.

However, a conclusion has thankfully been reached internally that our product is looking somewhat antiquated compared to our competitors', and that something needs to be done.

I am currently working on a new area of the UI which is quite separate from the rest of the product. I have therefore been given the chance to try out 'new' technology stacks as a sort of proving ground before the long process of moving over the rest of the UI begins.

I have been using C# with Windows Forms and the .net framework for a while in my spare time and enjoy it, but am somewhat worried about the headaches caused by interop. While this particular branch of the UI won't require much interop with the legacy C++ codebase, I can forsee this becoming an issue in the future.

The alternative is just to continue with MFC, but try and take advantage of the new feature pack that shipped with VS2008. This I guess is the easiest option, but I worry about longevity and not taking advantage of the goodness that is .net...

So, which do I pick? We're a small team so my recommendation will quite probably be accepted as a future direction for our development - I want to get it right.

Is MFC dead? Is C#/Winforms the way forward? Is there anything else I'm totally missing? Help greatly appreciated!

c# c++ winforms mfc user-interface

Share
Improve this question
Follow

edited Aug 17, 2008 at 2:00
Ryan Fox
10.2k ● 5 ● 39 ● 48

asked Aug 14, 2008 at 11:26
Ali Parr
4,787 ● 4 ● 32 ● 35

1 Yes, MFC is dead. WinForms is dying. At this point the way forward is WPF. Switching from MFC to WinForms will cut cost substantially, but moving on from WinForms to WPF will

cut costs dramatically. WinForms is a good option for people who still need to support Windows 2000 machines or Windows Mobile. DirectX is still best for 3D games and CAD. IMHO, everyone else should skip WinForms entirely and move directly to WPF. – Ray Burns Nov 6, 2009 at 5:47

6 Answers

Sorted by: Highest score (default)

9

I'm a developer on an app that has a ton of legacy MFC code, and we have all of your same concerns. A big driver for our strategy was to eliminate as much risk and uncertainty as we could, which meant avoiding The Big Rewrite. As we all know, TBR fails most of the time. So we chose an incremental approach that allows us to preserve modules that won't be changing in the current release, writing new features managed, andporting features that are getting enhancements to managed.

You can do this several ways:

1. Host WPF content on your MFC views (see here)
2. For MFC MDI apps, create a new WinForms framework and host your MFC MDI views (see here)
3. Host WinForms user controls in MFC Dialogs and Views (see here)

The problem with adopting WPF (option 1) is that it will require you to rewrite all of your UI at once, otherwise it'll look pretty schizophrenic.

The second approach looks viable but very complicated.

The third approach is the one we selected and it's been working very well. It allows you to selectively refresh areas of your app while maintaining overall consistency and not touching things that aren't broken.

The Visual C++ 2008 Feature Pack looks interesting, I haven't played with it though. Seems like it might help with your issue of outdated look. If the "ribbon" would be too jarring for your users you could look at third-party MFC and/or WinForms control vendors.

My overall recommendation is that interop + incremental change is definitely preferable to sweeping changes.

After reading your follow-up, I can definitely confirm that the productivity gains of the framework vastly outweigh the investment in learning it. Nobody on our team had used C# at the start of this effort and now we all prefer it.

Share Improve this answer Follow

edited Feb 25, 2009 at 22:41

answered Aug 14, 2008 at 13:32

Aidan Ryan
11.6k ● 13 ● 59 ● 87

You can theme your WPF to look like the older style controls. – Andy Dent Jan 27, 2009 at 5:18

The effort/payoff on that is dismal. Not to mention you would *never* get it looking pixel-by-pixel right. You'll end up with an "uncanny valley" effect. – Aidan Ryan Feb 25, 2009 at 22:39

Getting WPF styles pixel-perfect could take a long time BUT in a matter of hours I was able to get one app's styles so accurate that nobody - not even the QA folks - realized the new sections were slightly different. – Ray Burns Nov 6, 2009 at 5:16

2

Depending on the application and the willingness of your customers to install .NET (not all of them are), I would definitely move to WinForms or WPF. Interop with C++ code is hugely simplified by refactoring non-UI code into class libraries using C++/CLI (as you've noted in your selection of tags).

The only issue with WPF is that it may be hard to maintain the current look-and-feel. Moving to WinForms can be done while maintaining the current look of your GUI. WPF uses such a different model that to attempt to keep the current layout would probably be futile and would definitely not be in the spirit of WPF. WPF also apparently has poor performance on pre-Vista machines when more than one WPF process is running.

My suggestion is to find out what your clients are using. If most have moved to Vista and your team is prepared to put in a lot of GUI work, I would say skip WinForms and move to WPF. Otherwise, definitely look seriously at WinForms. In either case, a class library in C++/CLI is the answer to your interop concerns.

Share Improve this answer Follow

answered Aug 14, 2008 at 11:40

Zooba
11.4k ● 3 ● 38 ● 40

Please provide references for the poor WPF performance with multiple instances - this could be a key issue for us! – Andy Dent Jan 27, 2009 at 5:19

Windows Vista Display Driver Model (msdn.microsoft.com/en-us/library/aa480220.aspx), from MSDN – Zooba Jan 29, 2009 at 8:39

One of my devel machines is a dual-monitor Windows XP and I frequently have several WPF applications visible onscreen. I have not noticed any performance issues. Of course none of the applications I use for development are real heavy-duty graphics applications with lots of moving objects. If that were the case, the GPU sharing issue might become noticeable. – Ray Burns Nov 6, 2009 at 5:24

2

You don't give a lot of detail on what your legacy code does or how it's structured. If you have certain performance criteria you might want to maintain some of your codebase in C++. You'll have an easier time doing interop with your old code if it is exposed in the right way - can you call into the existing codebase from C# today? Might be worth thinking about a project to get this structure right.

On the point of WPF, you could argue that WinForms may be more appropriate. Moving to WinForms is a big step for you and your team. Perhaps they may be more comfortable with the move to WinForms? It's better documented, more experience in the market, and useful if you still need to support windows 2000 clients.

You might be interested in Extending MFC Applications with the .NET Framework

Something else to consider is C++/CLI, but I don't have experience with it.

Share Improve this answer Follow

answered Aug 14, 2008 at 12:04

Brian Lyttle
14.6k ● 15 ● 69 ● 106

2

Thank you all kindly for your responses, it's reassuring to see that generally the consensus follows my line of thinking. I am in the fortunate situation that our software also runs on our own custom hardware (for the broadcast industry) - so the choice of OS is really ours and is thrust upon our customers. Currently we're running XP/2000, but I can see a desire to move up to Vista soon.

However, we also need to maintain very fine control over GPU performance, which I guess automatically rules out WPF and hardware acceleration? I should have made that point in my original post - sorry. Perhaps it's possible to use two GPUs... but that's another question altogether...

The team doesn't have any significant C# experience and I'm no expert myself, but I think the overall long term benefits of a managed environment probably outweigh the time it'll take to get up to speed.

Looks like Winforms and C# have it for now.

Share Improve this answer Follow

answered Aug 14, 2008 at 13:30

Ali Parr
4,787 ● 4 ● 32 ● 35

1 GPU also seems to have a huge influence on WPF memory usage. – Aidan Ryan Feb 25, 2009 at 22:42

1 I don't know if this affects your situation or not, but there is at least one situation in which the GPU is not used at all when running WPF: Remote Desktop (RDP / Terminal Services). To be more precise, the remote machine's GPU is not utilized at all when running WPF applications there. Under RDP 6.0 the primitives are passed back to the client for rendering. – Ray Burns Nov 6, 2009 at 5:38

1

Were you to look at moving to C# and therefore .NET, I would consider Windows Presentation Foundation rather than WinForms. WPF is the future of smart clients in .NET, and the skills you pick up you'll be able to reuse if you want to make browser-hosted Silverlight applications.

Share Improve this answer Follow

answered Aug 14, 2008 at 11:28

Matt Hamilton
204k ● 61 ● 392 ● 321

0

I concur with the WPF sentiment. Tag/XML based UI would seem to be a bit more portable than WinForms.

I guess too you have to consider your team, if there is not a lot of current C# skills, then that is a factor, but going forward the market for MFC developers is diminishing and C# is growing.

Maybe some kind of piecemeal approach would be possible? I have been involved with recoding legacy applications to C# quite a bit, and it always takes a lot longer than you would estimate, especially if you are keeping some legacy code, or your team isn't that conversant with C#.

Share Improve this answer Follow

answered Aug 14, 2008 at 11:36

JamesSugrue

15k ● 10 ● 62 ● 93