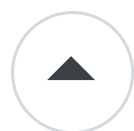


Is there any advantage to using C++ instead of C# when using Direct3D?

Asked 13 years, 8 months ago Modified 13 years, 8 months ago

Viewed 3k times



4



Is there any advantage to using C++ instead of C# when using Direct3D? The tutorials I've found for C++ and DirectX all use Direct3D (which to my knowledge is managed). Similarly, all of the C# tutorials I've found are for Direct3D.



Is Direct3D managed? Is there any difference between using D3D in either of the two languages?



c#

c++

directx

direct3d

Share

Improve this question

Follow

asked Apr 28, 2011 at 22:46



[LunchMarble](#)

5,141 ● 11 ● 67 ● 94


Speed? If there's any crunching besides what Direct3D does for you, it's gonna be the only difference there, really – [sehe](#)
Apr 28, 2011 at 22:47


@sehe are you sure C++ is going to outperform C#? I know it's "common knowledge" that compiled languages are faster than interpreted languages, but I find that to be more of a

"common misconception" myself. – [corsiKa](#) Apr 28, 2011 at 22:49

@sehe, glowcoder This isn't a debate about whether the language C#/C++ is faster/better. It's about how Direct3D is running in either language, will it run any different in C++ than in C#. – [Kyle Uithoven](#) Apr 28, 2011 at 22:54

You might want to look at this thread:

social.msdn.microsoft.com/Forums/en/gametechnologiesdirectx101/... (and yes, it seems like MDX, if that's what you're using, is managed, and at least theoretically almost on par with Native DirectX - that being said, C++ itself could be significantly faster than C# *in some cases*). – [Boaz Yaniv](#) Apr 28, 2011 at 22:56 

- 1 The advantage is speed. C#/Java fans are notorious for generating synthetic benchmarks and claiming parity in cpu computation. If your code is cpu-bound, C++(or Fortran)/inline ASM is the way to go. – [Paul Nathan](#) Apr 28, 2011 at 23:11 

3 Answers

Sorted by:

Highest score (default)



10



DirectX is entirely native. Any impression you may have that it's managed is completely and utterly wrong. There are managed *wrappers* that will allow you to use DirectX from managed code. In addition, DirectX is programmed to be accessed from C++ or C, or similar languages. If you look at the SlimDX project, they encountered numerous issues, especially due to resource collection, because C# doesn't genuinely support non-memory resources being automatically collected and `using` doesn't cut the mustard. In addition, game programming

can be very CPU-intensive, and often, the additional performance lost by using a managed language is untenable, and virtually all existing supporting libraries are for C or C++.

If you want to make a small game, or something like that, there's nothing at all stopping you from using managed code. However, I know of no commercial games that actually take this route.

[Share](#) [Improve this answer](#)

[Follow](#)

answered Apr 28, 2011 at 23:01



Puppy

147k ● 40 ● 266 ● 477

Direct3D is used for many things other than games. I can see plenty of reasons why some developers would want to use Direct3D from a .net application. – [David Heffernan](#) Apr 28, 2011 at 23:03

Just for solid clarification: DirectX *and* Direct3D are native? Or DirectX is native, *but* Direct3D is managed?

– [LunchMarble](#) Apr 28, 2011 at 23:08

@Storm Direct3D is native. I believe all the Direct-whatever libraries are, but not sure about the others. D3D and DInput definitely are. – [ssube](#) Apr 28, 2011 at 23:11

Thank you all for your answers and responses!

– [LunchMarble](#) Apr 28, 2011 at 23:13

@Storm Kierman: There basically is no DirectX anymore except Direct3D- everything from DirectSound, DirectMusic and DirectInput are all gone. Recently, there are new libs like DirectWrite, Direct2D as well, but they're a family and all share the same traits- COM written in native code. – [Puppy](#) Apr 28, 2011 at 23:32



2



The point of Direct3D is to move rendering off the CPU and onto the GPU. If there were to be a significant performance difference it would be for that code that runs on the CPU. Therefore I don't see that there should be any significant performance difference between native and managed code for the part of your code that interfaces with Direct3D.

Direct3D itself is not managed code.

Follow



David Heffernan

612k ● 43 ● 1.1k ● 1.5k

When I reference the Microsoft.DirectX.Direct3D is in located in a directoy called "DirectX for Managed Code". Is there another version built SPECIFICALLY for Unmanaged code? If there is, what are the differences there? – [Kyle Uithoven](#)
Apr 28, 2011 at 23:02

1 @Kyle that's a managed wrapper around DirectX. The version of unmanaged code **is** DirectX. – [David Heffernan](#)
Apr 28, 2011 at 23:04 ✎



0



It depends on what you're doing exactly. As David Heffernan mentioned, one of the objectives of Direct3D is to move as much processing as possible to the GPU. With the advent of vertex shaders, pixel shaders, and much more, we're closer to that reality than ever.



Of course given infinite time and resources, you can usually create more efficient algorithms in C++ than C#. This will affect performance at the CPU level. Today, processing that is not graphics related is still mostly done on the CPU. There are things like CUDA, OpenCL, and even future versions of DirectX which will open up possibilities of moving any parallel-friendly algorithm to the GPU as well. But the adoption rate of those technologies (and the video cards that support it) isn't exactly mainstream just yet.

So what types of CPU-intensive algorithms should you consider C++ for?

- Artificial Intelligence
- Particle engines / n-body simulations
- Fast Fourier transform

Those are just the first things I can think of. At least the first two are very common in games today. AI is often done in a compromised fashion in games to run as quickly as possible, simply because it can be so processor intensive. And then particle engines are everywhere.

Share Improve this answer

edited Apr 28, 2011 at 23:27

Follow

answered Apr 28, 2011 at 23:16



Steve Wortham

22.2k ● 5 ● 70 ● 90
