# Why I’m Using C

Casper Beyer Jan 1, 2020

# Reliability

Languages, especially dynamic ones and their runtime environments come and go like fashion. When your target platform suddenly gets deprecated from the runtime you’re relying on, … you’re … out of luck.

I need something [that] will have a reliable toolchain available on all the platforms I could ever possibly want in perpetuity without having to worry about compiler bugs….

The ecosystem around C provides all of this. Even for the most obscure platform you can imagine there will be solid stable toolchain with a good set of native libraries that provide access to system and device interfaces for said platform.

# Performance

Avoiding cache misses is key to [performance], so having any kind of dynamic language with an interpreter is hopeless. Even in the best case scenario that the platform of choice provides you with one of the magical JIT compilers available today, they’re still magical black boxes which makes it difficult to reason about performance characteristics object boxing/unboxing and cache locality.

Same goes for stop the world garbage collection, again there are some fairly impressive implementations out there; Go’s garbage collector comes to mind but it’s still a garbage collector and it will still lead to frame drops. If you have to use object pools to tip-toe around garbage collector killing the frame-rate then what’s the point of having one in the first place.

But [it’s] not just about running fast, battery life matters [as well.] Fewer cycles means less battery consumption which is a major win if you ask me.

Lastly, when it comes to the web and mobile the executable size also has to be considered. While it’s not a deal breaker it’s a nice bonus that if you work on it a little bit C executables built with WebAssembly essentially have zero bytes of overhead by default. It might seem like a minor thing but it’s the difference between loading instantly and taking 3–5 minutes to load on a connection.

# Simplicity

While it’s not a critical requirement, a nice thing about C is that it is an extremely simple language which is a nice welcome break from working in monolithic languages that get jam packed with new features every time someone on the design team or committee learns about a new feature in another language causing a paradigm shift until the next iteration which causes another paradigm shift leaving the whole ecosystem in a bit of a tangled mess.

When I’m saying C [is] simple I don’t necessarily mean easy. If you don’t know what you’re doing C will absolutely blow up in your face and make you spend the day trying to figure out what you did wrong. But again it’s a simple language so it’s really not too hard learning how to write well-behaved programs….

Now there’s definitively features that would be nice to have in C; I wouldn’t hate having quality of life things like arrays that don’t decay to naked pointers, modules or deferred statements for example but not having them aren’t total deal breakers.

At the end of the day I prefer having a limited surface area in the language than having a language that gets in the way bringing absurd amounts of complexity and bad abstractions to the problem.

Retrieved from <https://medium.com/bytegames/why-im-using-c-2f3c64ffd234>

on June 30, 2020.