CONTEXT: This is a presentation I recently gave at WWX2015 (WorldWide Haxe conference) in Paris, France.

Briefly stated, *Defender's Quest HD* and *Defender's Quest II* use three core technologies:

- Haxe is the programming language
- OpenFL is our framework
- HaxeFlixel is our game engine

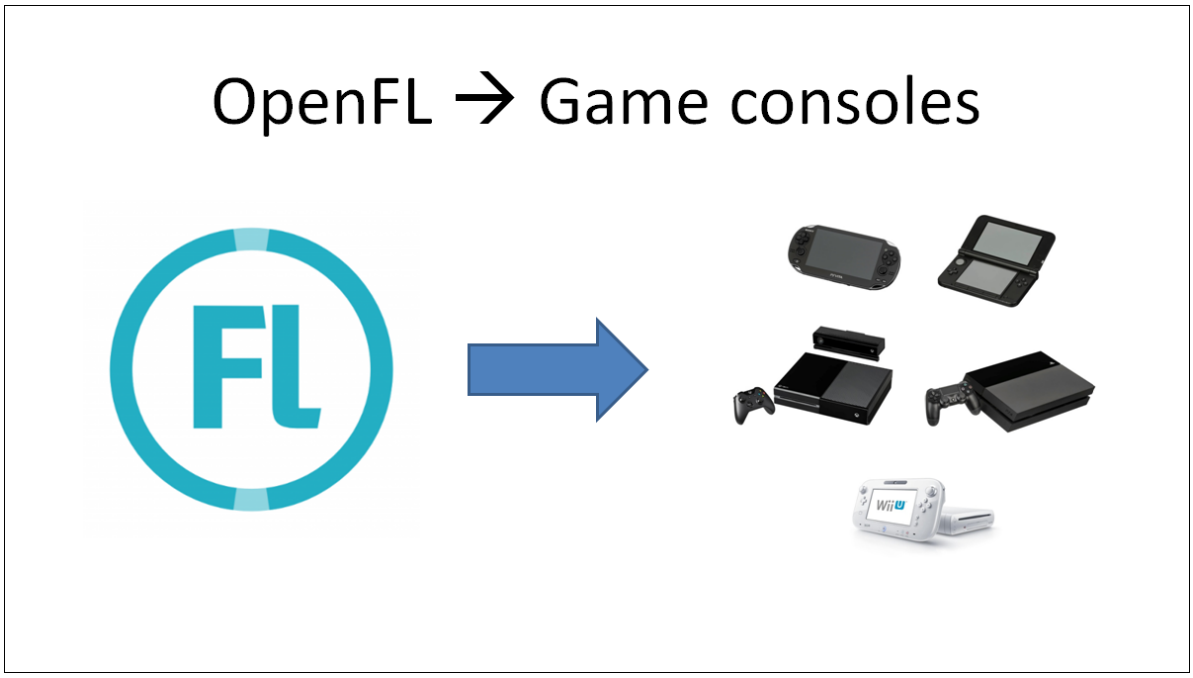
10 second summary: Haxe is a programming language that compiles to other programming languages (everything from C++ to Javascript to Python), it's been

around for about 10 years and is extremely powerful. OpenFL is a hardware-accelerated cross-platform reimplementation of the Flash API, built on top of Haxe (OpenFL does not have the Flash player's performance and security limitations and has nothing to do with Adobe), and HaxeFlixel is a really cool 2D game engine.

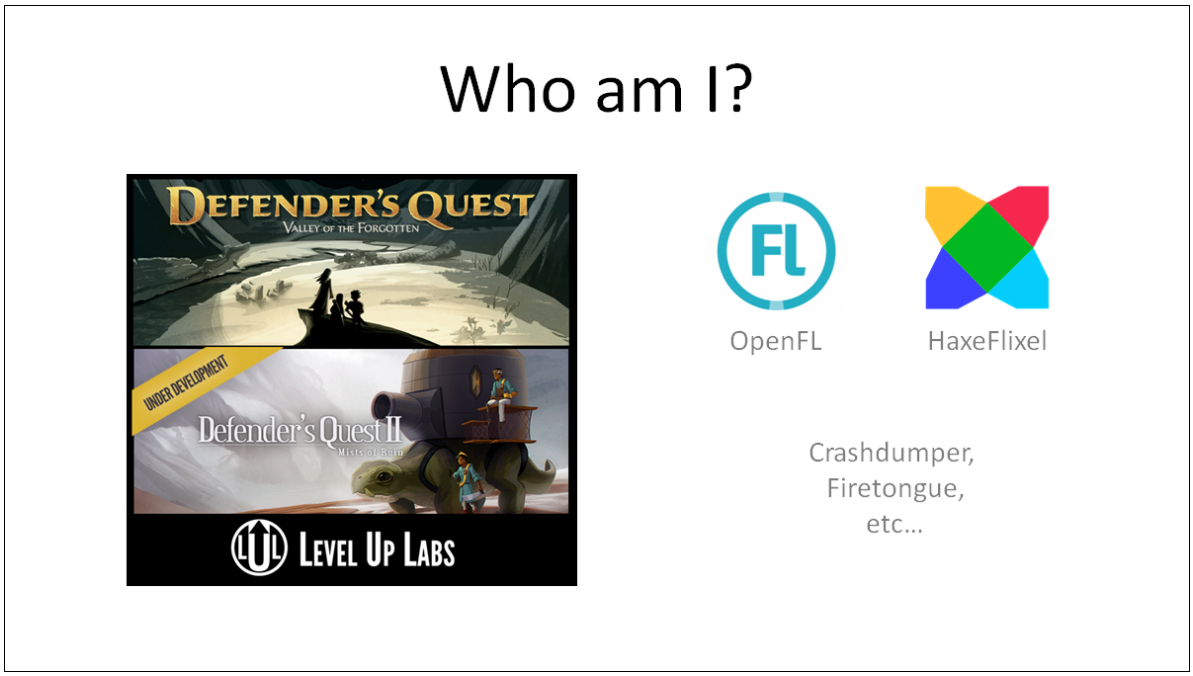
And my talk is about making that entire tech stack work on home game consoles.

Let's go:

(walks up to podium)



Hey everyone, today we're going to talk about how we got Lime and OpenFL to run on home game console hardware, such as WiiU, Xbox One, Playstation 4, etc.

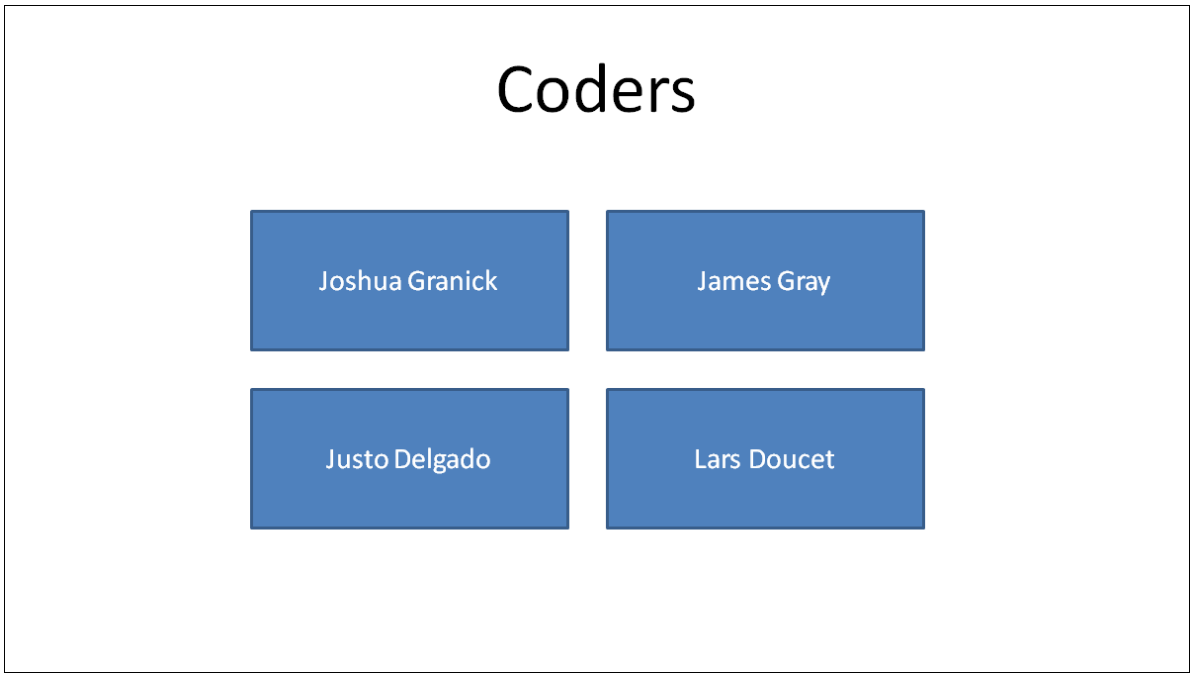


My name's Lars Doucet, I work for Level Up Labs,

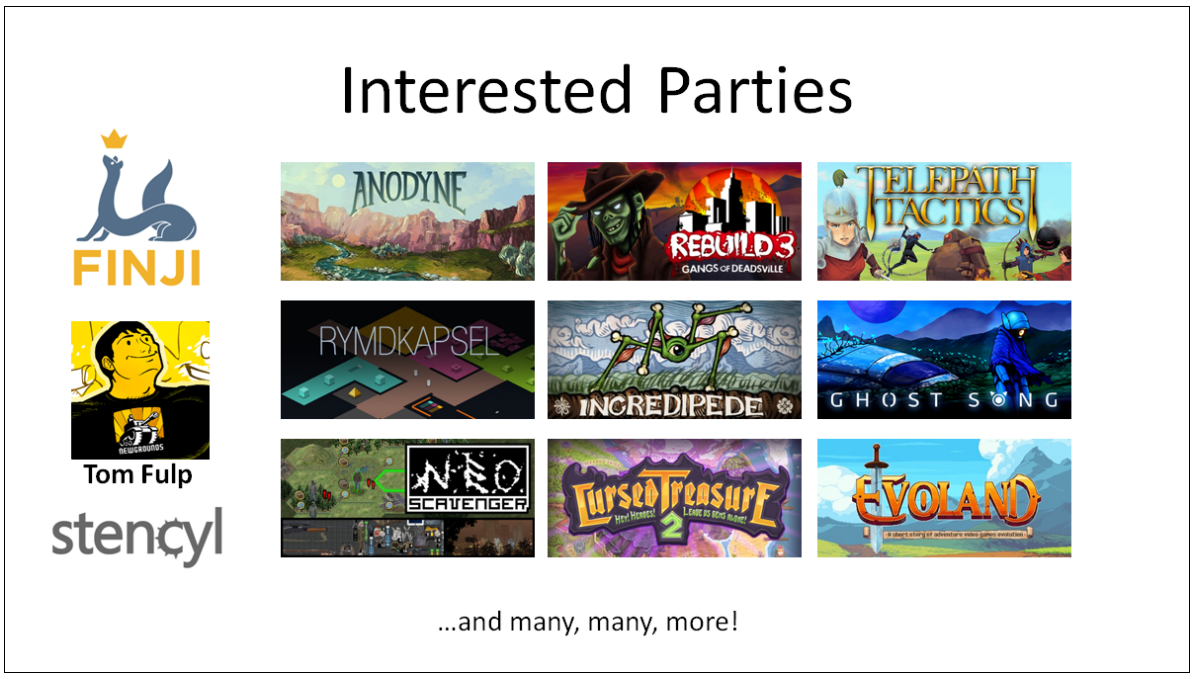
developers of Defender's Quest and Defender's Quest II.

I'm a regular contributor to Haxe projects, chiefly OpenFL and HaxeFlixel, I also have a few of my own libraries some of you might have used, such as crashdumper, firetongue, etc.

Our funding team is three developers -- Level Up Labs, 3909 (Lucas Pope's company, who did Papers, Please), and Puzzl, who did Yummy Circus.



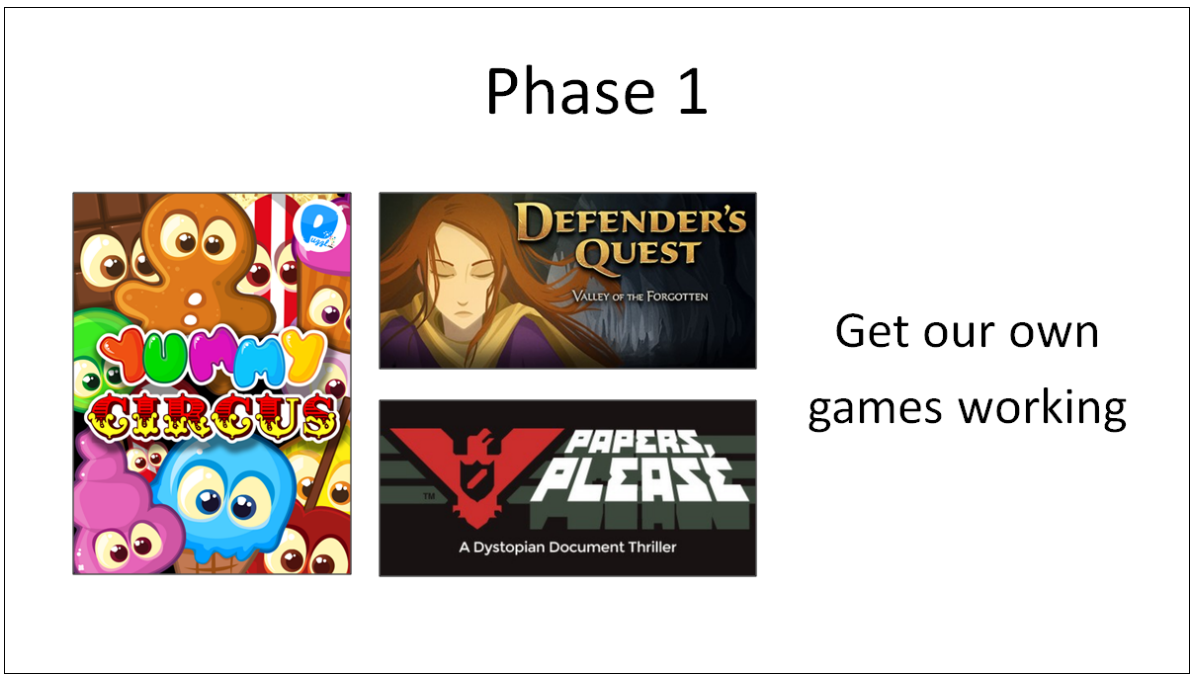
The programmers actually building this are James Gray, Joshua Granick, and Justo Delgado who you might recognize from OpenFL, and myself.



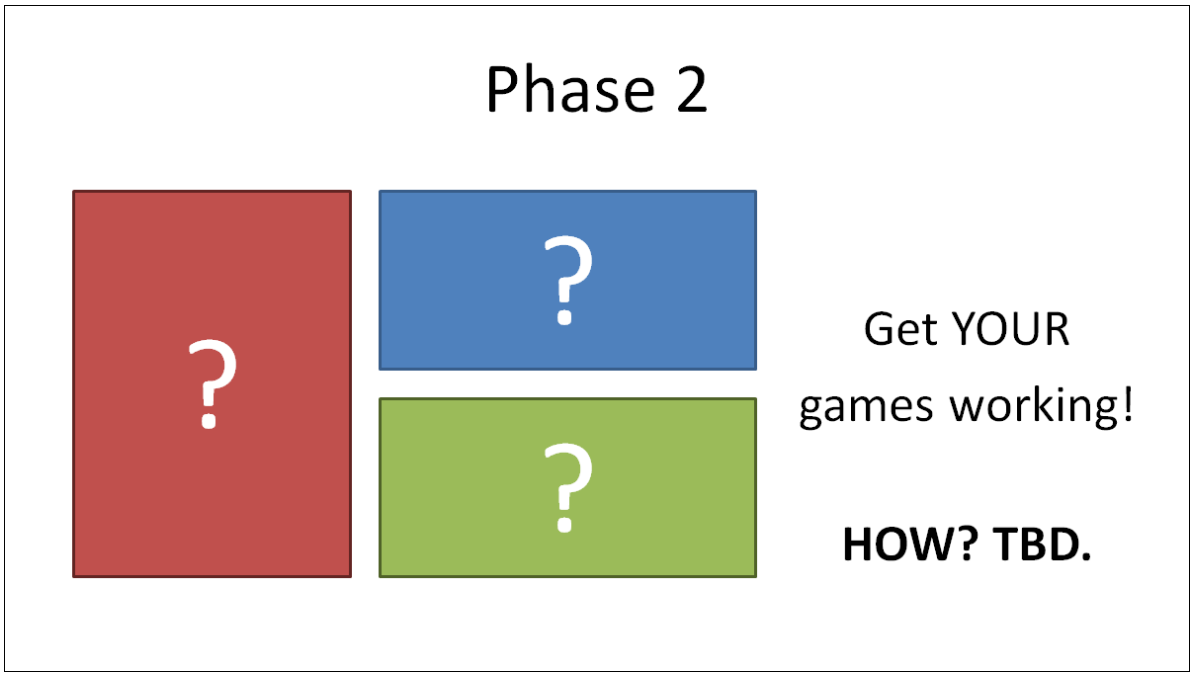
We've gotten a lot of interest in a Haxe/OpenFL console backend.

So here's just a short list of various people who are watching this project -- the developers of games like Anodyne, Rebuild, Rymdkapsel, Ghost Song, NEO Scavenger, Telepath Tactics, Evoland (Haxe's own Nicolas Canasse of course!), as well as the various engines built on OpenFL. There's my own team HaxeFlixel, but also HaxePunk, and tool companies like Stencyl.

Also, I should note Adam Saltsman, developer of the original Flash Flixel, is watching this with a lot of interest, as is Tom Fulp (best known as the founder of Newgrounds and The Behemoth), and legions of current and former flash game developers.

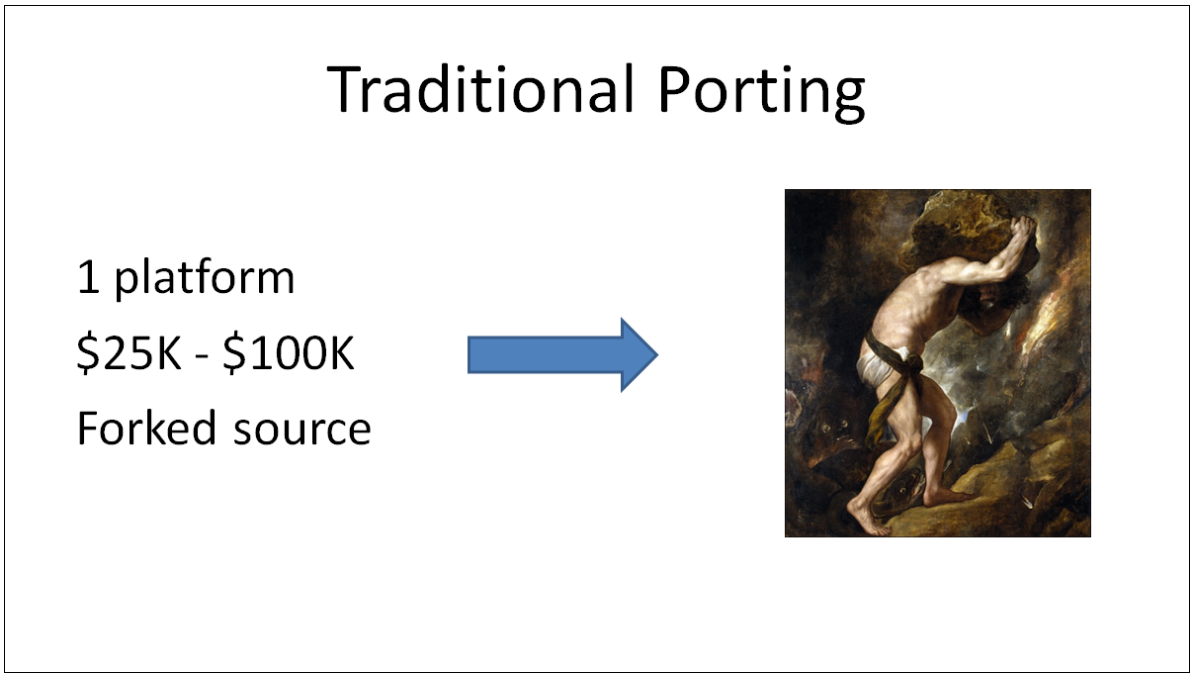


So our first goal is just to get our own games working on consoles at all.
Split the cost, just get it done.



After that, we'll look at how we'll be able to make this available to everyone else. The main reason we can't make it open from the start has to do with technical / legal issues related to console dev, which I'll get into here in a second.

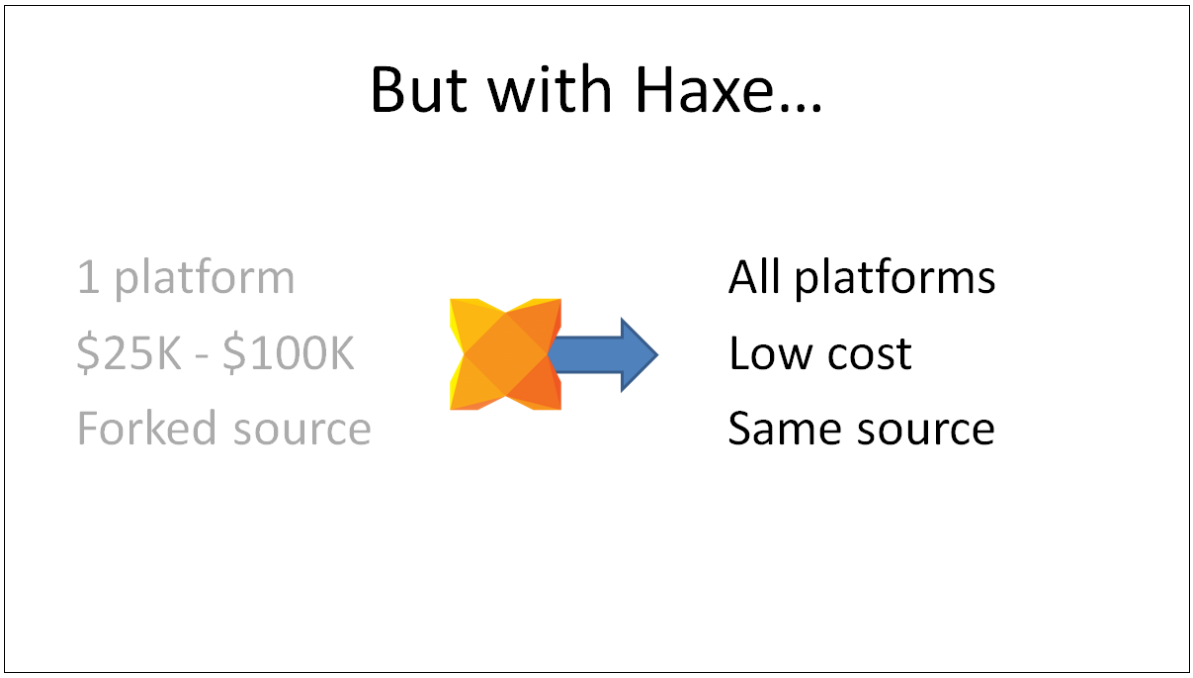
But first let's talk about what "traditional" console porting looks like.



So traditional porting is frankly a pain in the ass. Let's take a decent-sized indie game about the same size as Defender's Quest. I've seen porting companies charge anywhere from 25 to 100 thousand US dollars.

That price actually isn't even unreasonable, because it's a LOT of work to port a game "the hard way."

Usually this gets you ONE platform, coded mostly from scratch. Assuming everything even goes well, you wind up with a forked codebase so you have to patch bugs in parallel now.



So we all know Haxe is great.

Instead of porting one platform at a time, you can access them all at once from the same code base, at a low incremental cost per platform, and keep one unified codebase so any updates or bugfixes will instantly propagate to all of your supported targets.

The Problem

Now, the problem is consoles are different. You can do "lime build linux" or "lime build ios" but one does not simply "lime build xbox."

Here's why.

Console Challenges

Closed SDK

Dev kits

NDA

Consoles are HARD to build for, especially for an open source project. Unlike android or iOS, the SDKs are closed. You have to be an approved developer to even look at them. Even if you have the SDK, you also need hardware to test on, which means a devkit. You're sworn to secrecy and you usually need to fork out some money. And it takes forever to do the paperwork.

Even worse, console development cuts off the standard lifelines we open source developers are used to using. I can't reveal the specifics of the various console SDKs, but I think it's safe to say that they generally use proprietary APIs for most things, especially graphics. You can't just use SDL or OpenGL right out of the box and expect it to work.

"No problem, I'll just google it," you say.

No you can't! The SDK is under NDA, so there's no information on the public web about it. No info on Stack Overflow either, and you definitely can't just dump your code that touches the console SDKs on github and ask for help that way.

There is some help -- the console owners have

documentation and their own internal sites with forums and mailing lists. It's just harder than what we're normally used to.

Certification is Scary

Risky

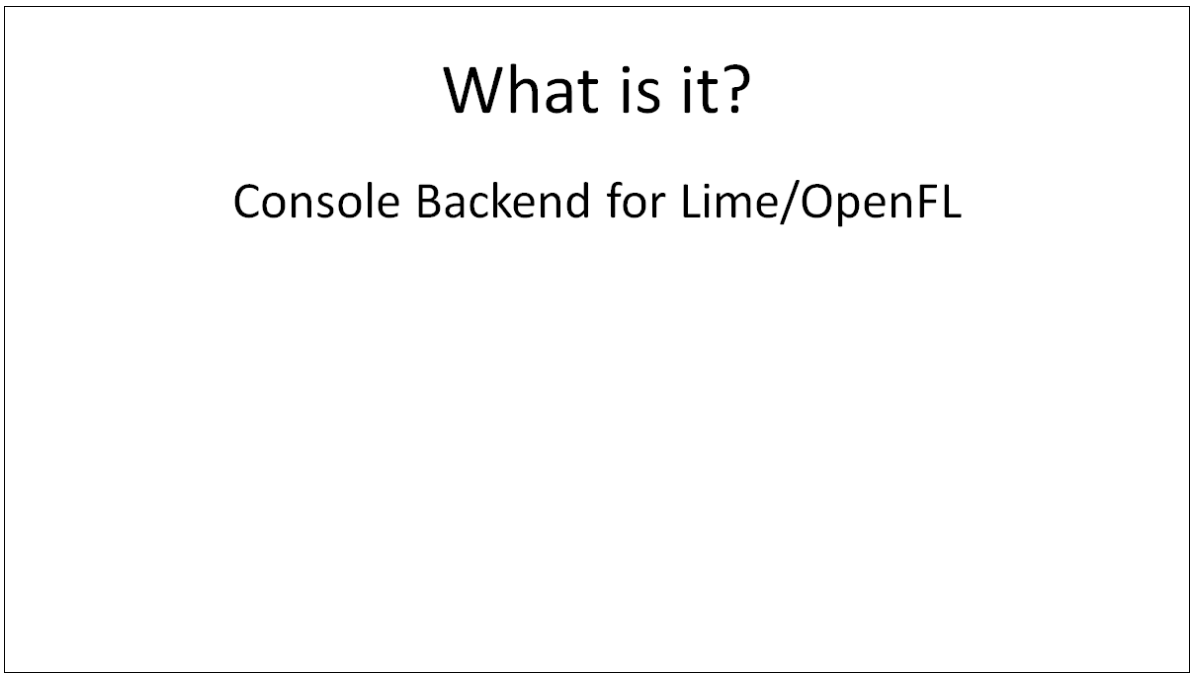
Expensive

Time-consuming

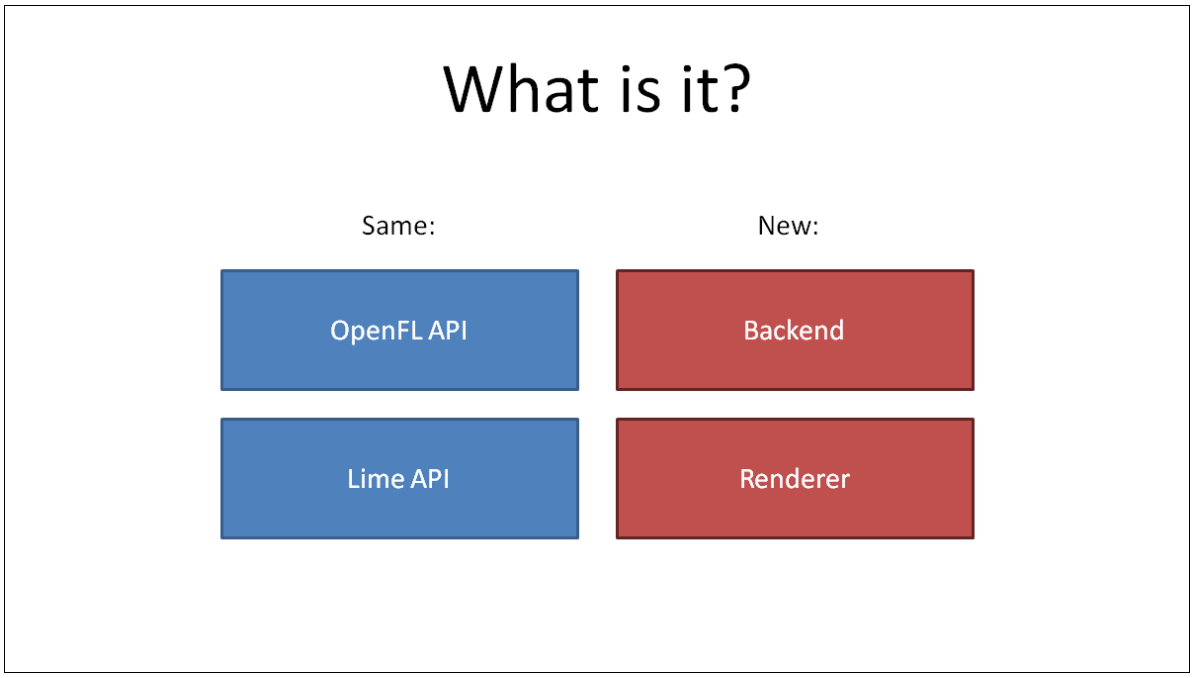
There's lots of horror stories in console development. You go through a gauntlet before release, and you can't put out a patch without it getting checked out.

They're trying to make it better but it will never be like Steam where you can put out an unsupervised patch in five minutes. So it's kind of scary to push something new like Haxe/OpenFL to consoles for the first time.

So, we've come up with a solution to all of these problems. It doesn't have a name yet.



It's a new cross-console backend for Lime/OpenFL.

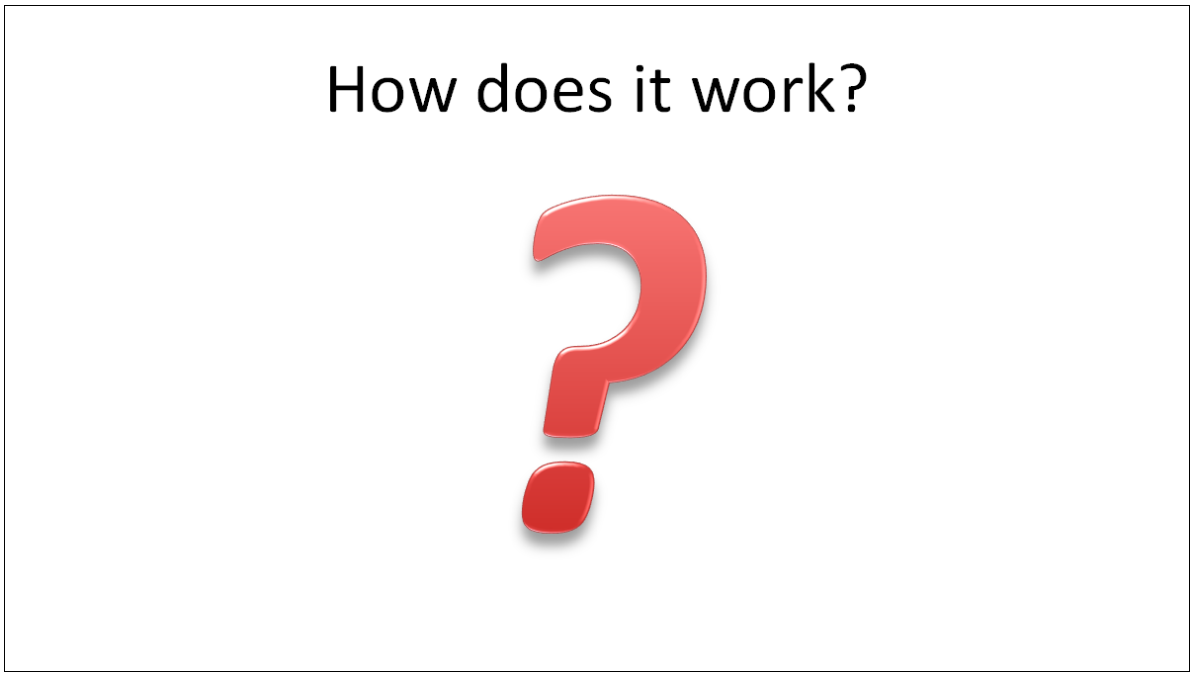


You can use the exact same OpenFL and Lime APIs you've been using so far, although it will have a new "console" renderer that is different from the existing GL, DOM, and CANVAS renderers Lime has right now.

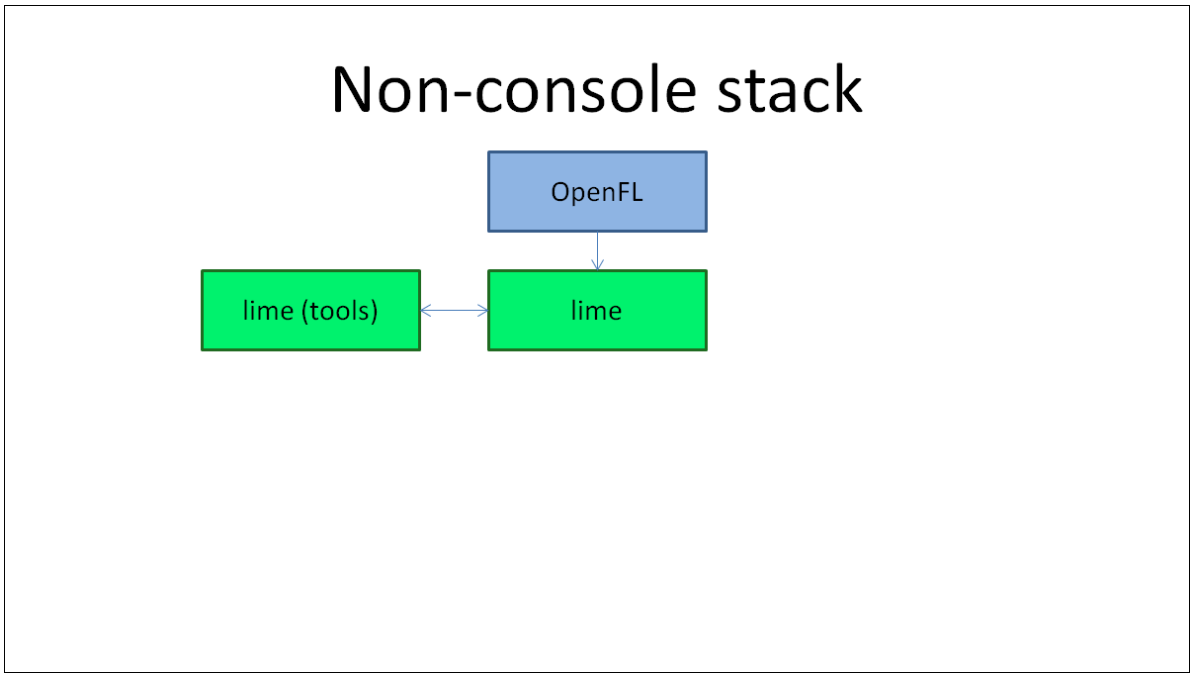
So if you're using OpenFL you won't have to write any new code, but if you're using Lime and you're directly using, say, the GL renderer, you might need to write some new graphics code for your console targets. But once you do that you'll be able to hit ALL of the consoles we support.

We plan on supporting all of these consoles eventually: WiiU and 3DS, Xbox One, Playstation 4, Playstation Vita, and Playstation 3. As of right now we have dev kits for all of these systems except the 3DS.

As of today (May 30th at WWX), we currently have samples compiling and running on WiiU. More on that later.



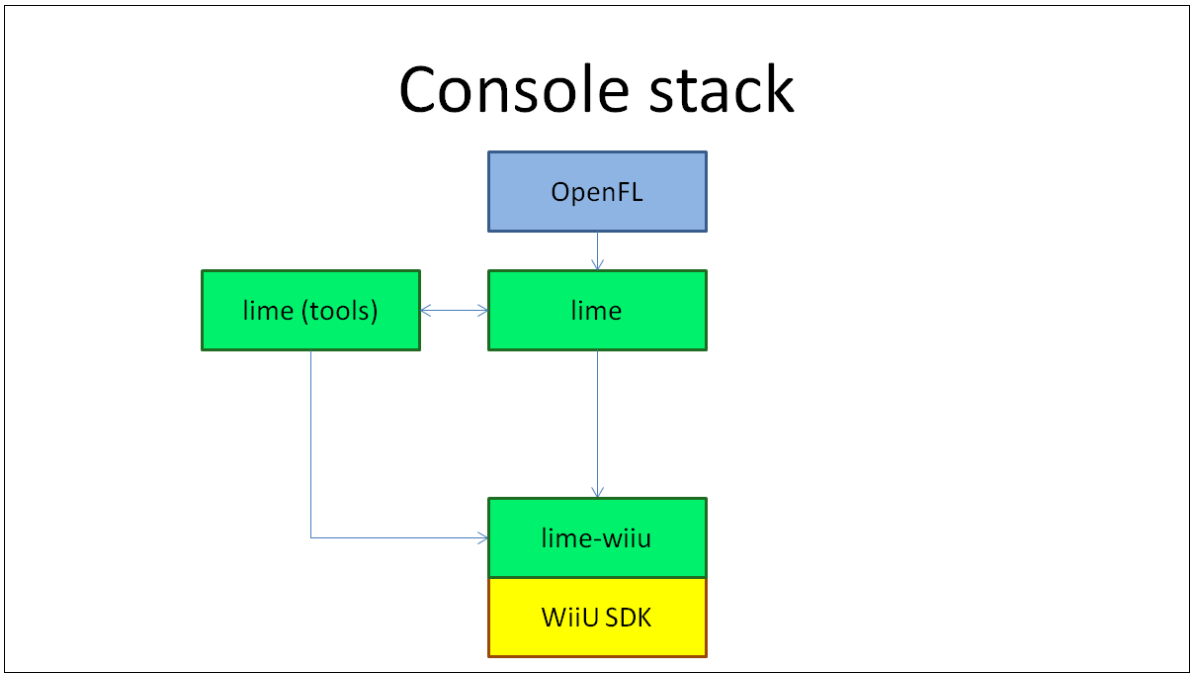
Okay, so how does it work? Let's start by looking at the new OpenFL architecture.



The new OpenFL architecture is a lot simpler now. There's only two libraries: OpenFL, and Lime, with tools included in Lime.

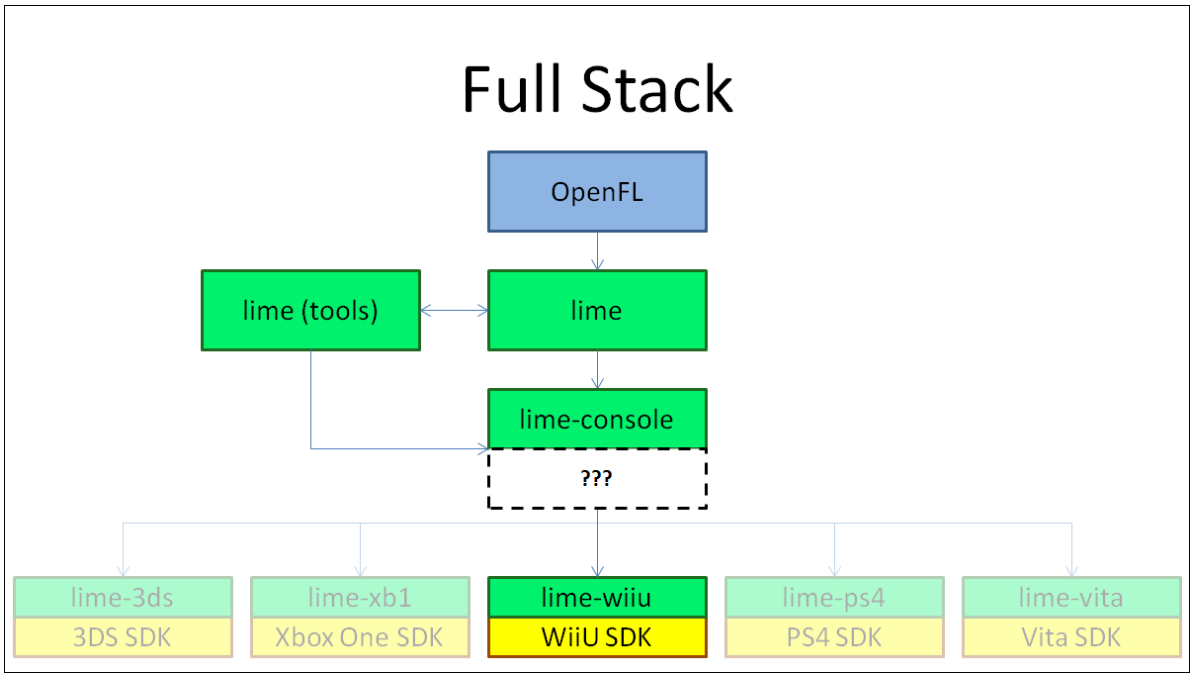
OpenFL provides high level stuff like the flash API and the various renderers, whereas Lime provides the low level platform support, rendering context, and tools. There's no longer any openfl-native, or openfl-html5, it's all encapsulated in these two libraries.

Now let's add a console target, like WiiU.

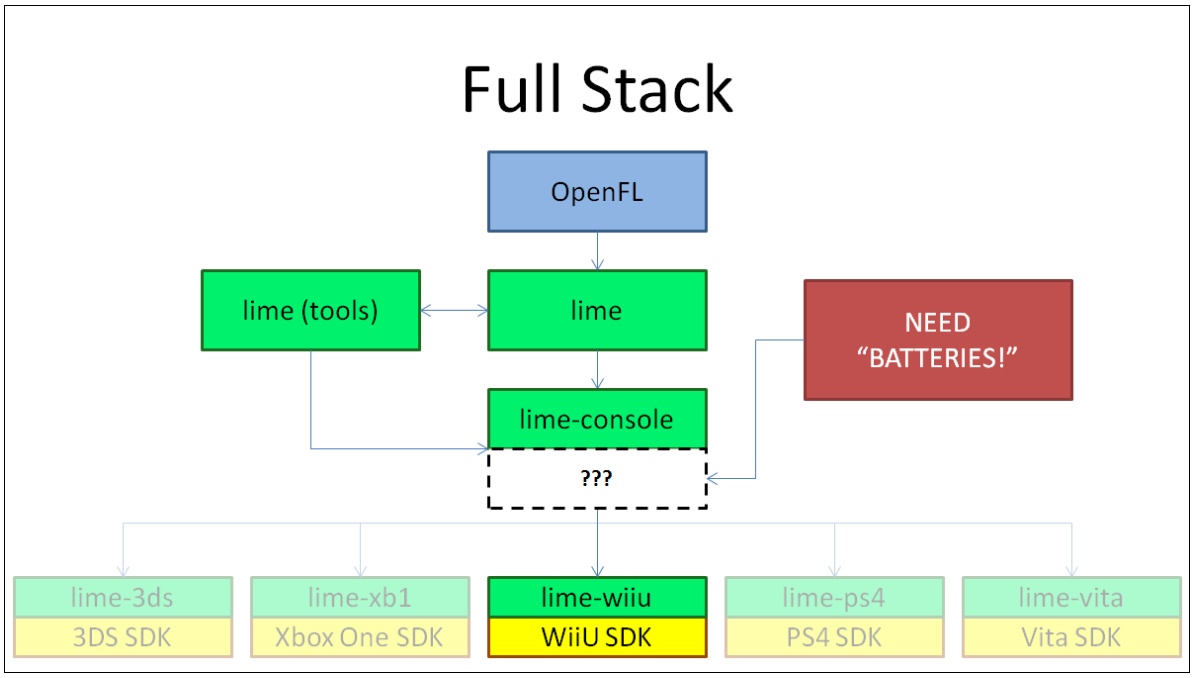


So you add the "lime-wiiu" library. This comes with pre-built binaries for the WiiU, and obviously you need the WiiU SDK and a devkit to use it. But now you can write an OpenFL or Lime app, and type "lime build wiiu" to make an executable that will run on your devkit!

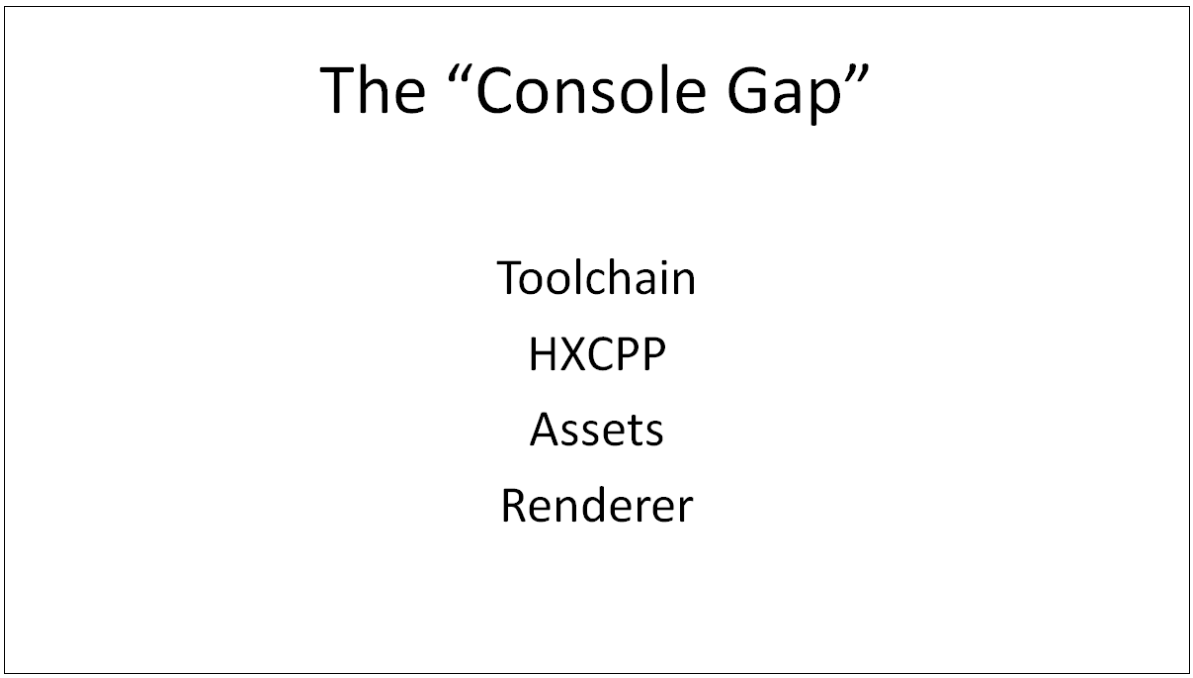
There will be a lime-{whatever} library for each console target.



So here's the full developer stack that we use internally to compile the entire thing from scratch. There's an intermediate library called "lime-console" that serves as a bridge between Lime itself and each native implementation. A lot of this is work we can do entirely on our own, but you notice there's a missing piece...



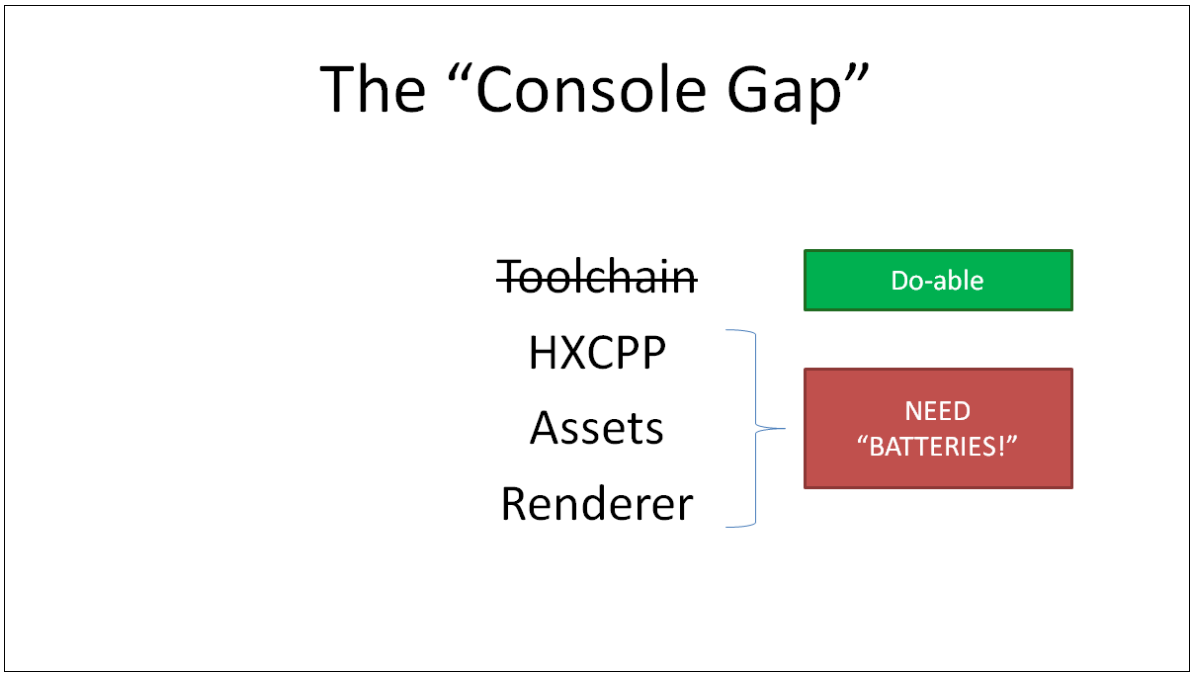
Basically, we need some "batteries" that we can slide in here to finish out the last mile of the console compatibility. So let's look at what this "console gap" looks like.



There are four major parts to the console gap. First, there's figuring out the toolchain, which is basically a big ugly XML file that Lime defines for each platform target, that gets Haxe/Lime to compile and run on that platform at all.

This is a big bundle of compiler flags, defines, etc, that we hand tweak.
This has to be done individually for every new platform.

But it's manageable.

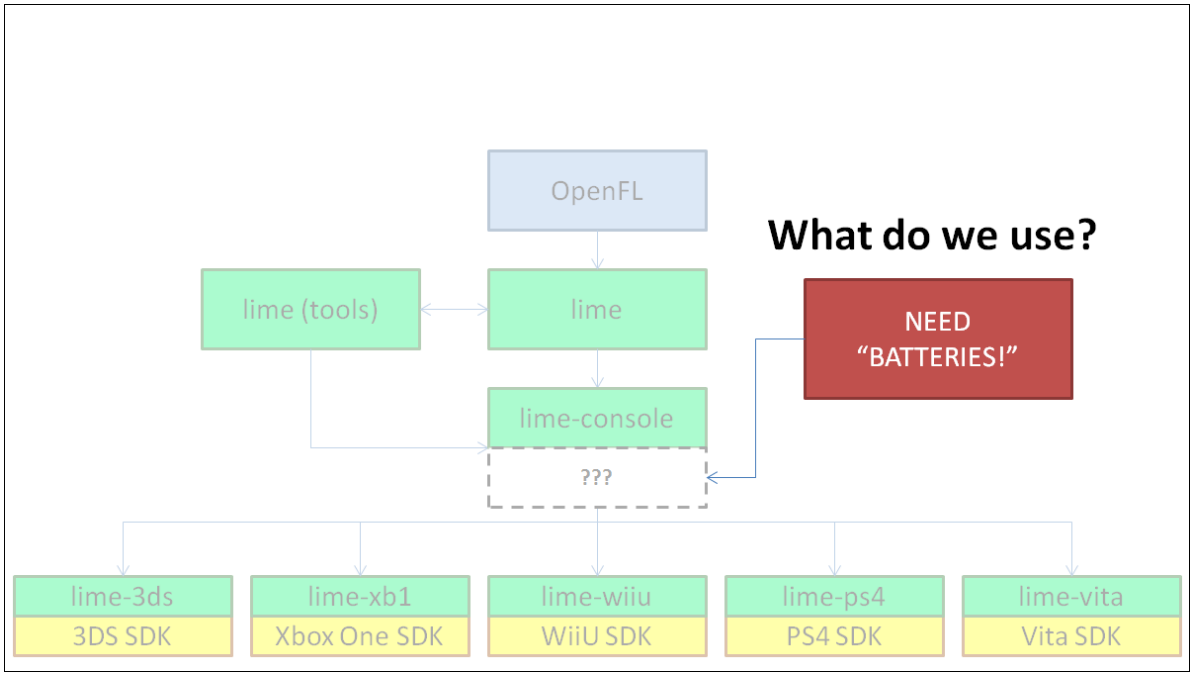


The other three are harder problems.

First, on consoles Haxe has holes in HXCPP, the runtime support for the Haxe standard library on C++ targets. And

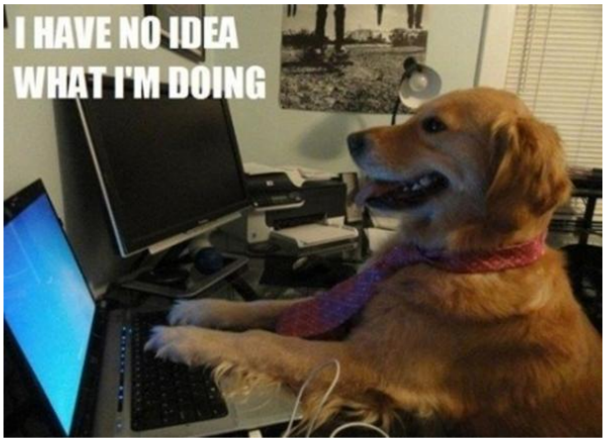
the holes are in different places on each platform. We could patch it in a custom way for each platform, but that's messy and inconsistent.

Second, we need a cross-platform solution that wraps asset processing for each console, and Third, we need a cross-platform console renderer so users don't have to support half a dozen different proprietary graphics APIs themselves.



So how do we fill this gap? What do we use for our "batteries?"

Choice 1: Do it Yourself



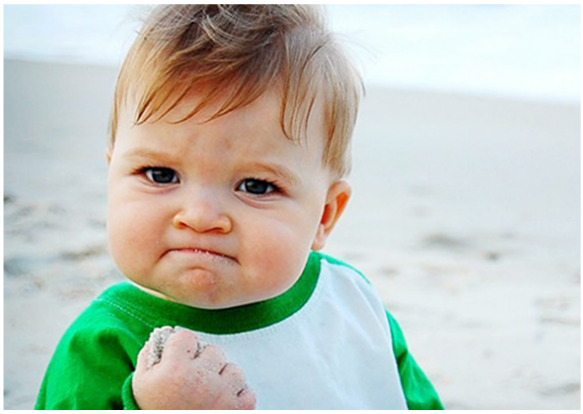
Well, you can try to do it yourself and bridge the console gap with your own C++ wrappers.

Okay, so now you have to hook up six different consoles running on drastically different hardware, running different software with different architectures, all with different rendering and asset pipelines, all with fresh, untested code that will somehow manage to pass certification and produce consistent results.

Easy!

If you're a hero, you're welcome to try this, but I'm a mere mortal so we decided to...

Choice 2: License some Tech



It just makes sense to partner with someone who knows what they're doing in the console space. So we're proud to announce that we're working with...

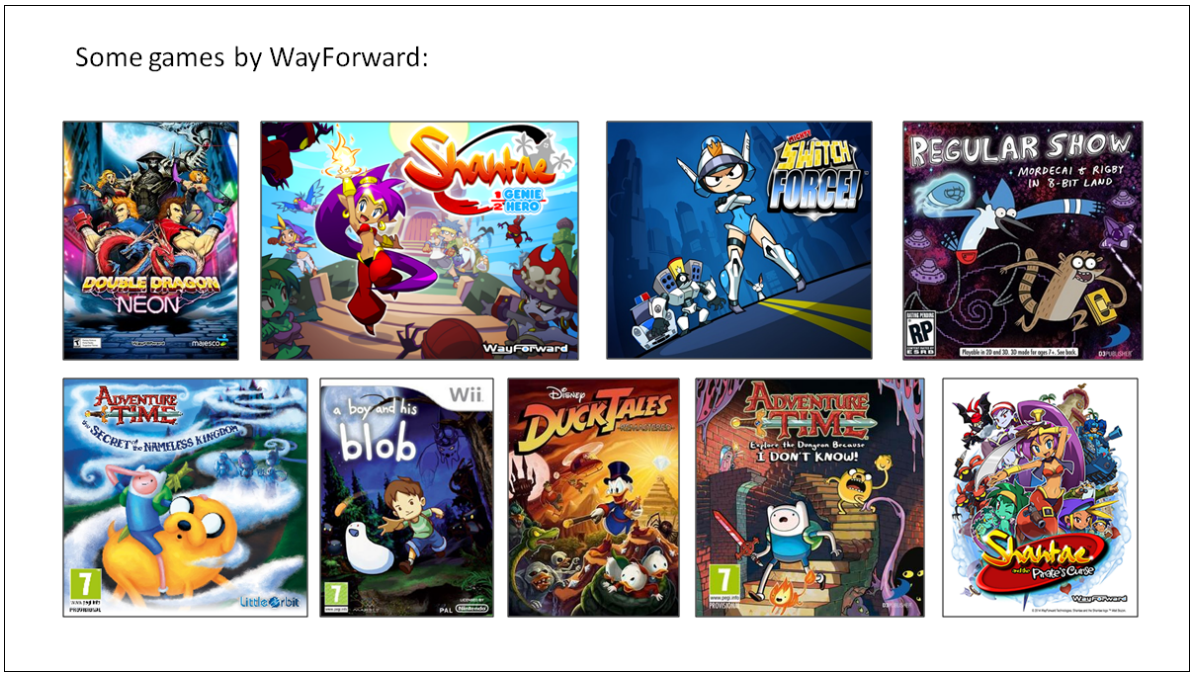
WayForward

Est. 1990

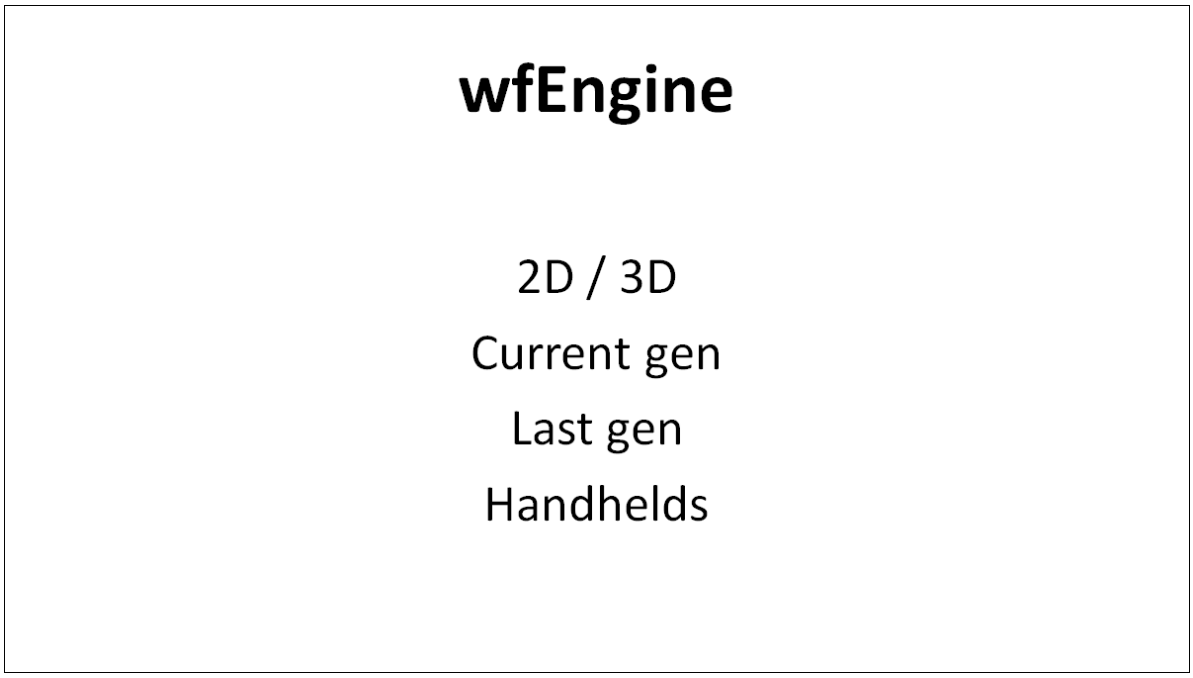
100+ games

25 platforms

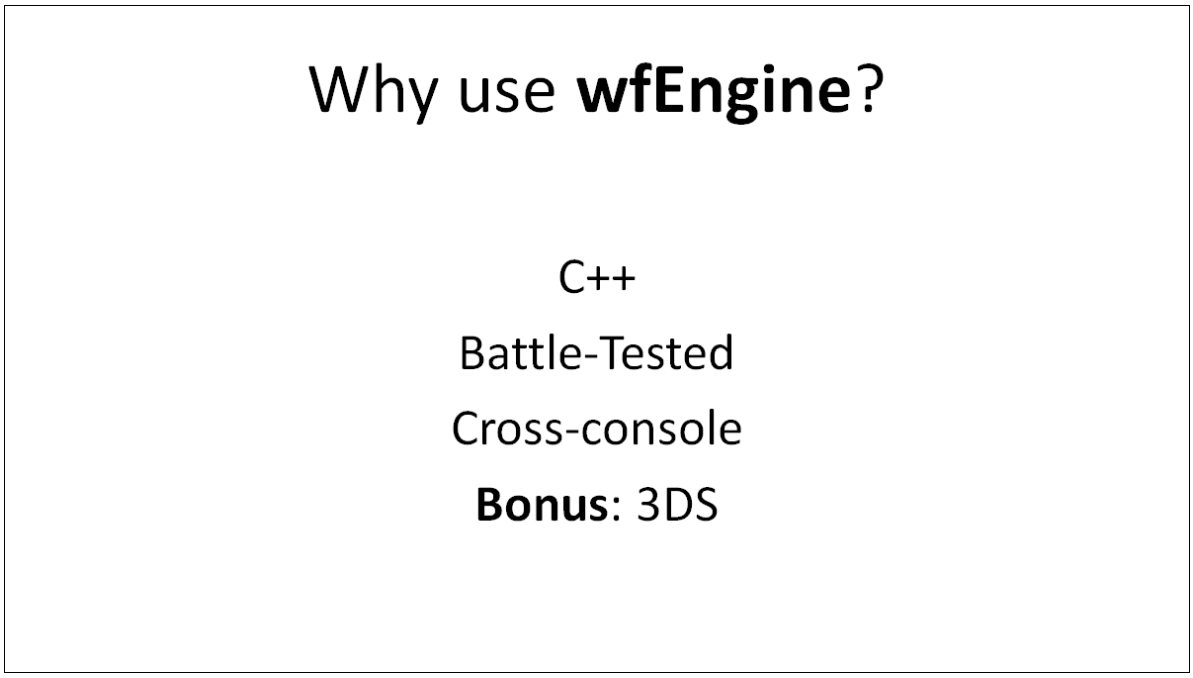
WayForward is a well-known console game developer. They've been in business for 25 years, they've published well over a hundred games on more than two dozen platforms. If there's anyone who knows how to do this, it's them.



You'll recognize some of these games. They're most famous for their original Shantae series of games, but they've also done a lot of popular licensed games, such as Double Dragon Neon, Adventure Time, DuckTales Remastered, Regular Show, etc.



So we're licensing their in-house game tech, "wfEngine." We can't go into too much detail about it today, but the broad strokes are that it supports both 2D and 3D, and it works on both current and last gen consoles. Most importantly, it also supports handhelds!



So, why this engine?

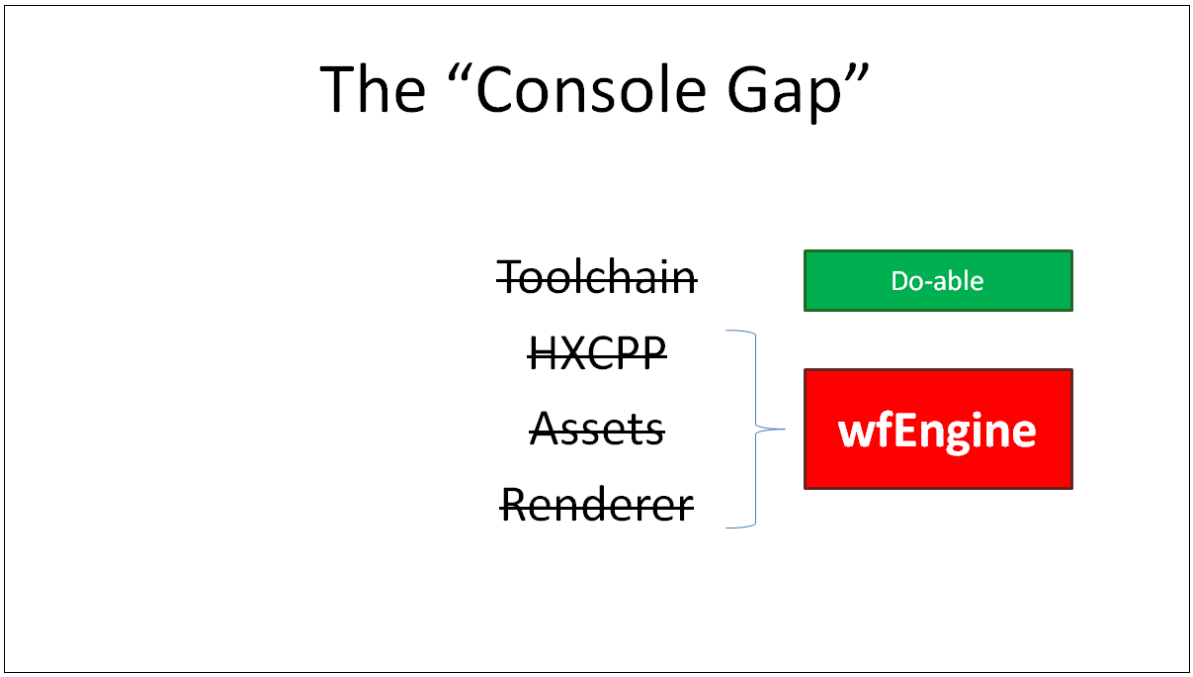
First, it's C++ based, so we can use the hxcpp target. This is way faster than using a virtual machine target like C# and connecting to Unity or Mono to power the backend.

Second, it's battle-tested. It's been used in tons of games that have all passed cert and already shipped.

Third, it's cross-console; it supports every platform we want to reach.

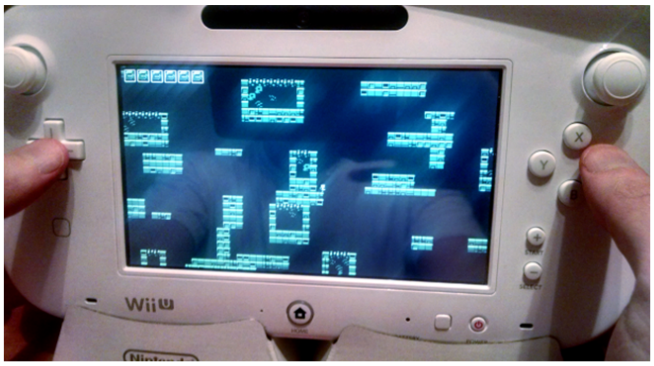
And as a special bonus, one of those platforms is the 3DS, which is actually the best-selling console of the current generation. GameMaker, Unity, etc, can't run on the 3DS -- (although Unity has just announced some partial support for the 3DS, likely games will only be able to run on the

"NEW 3DS" model).



And so going back to our console gap, this plugs our remaining holes. We can use files in the lime-console library to patch HXCPP with wfEngine's implementations of anything that's missing. We can also use its pipeline and processing code to build out the console version of Lime's asset packaging. And of course, the new OpenFL "console" renderer connects to wfEngine via Lime.

Pics or it didn’t happen

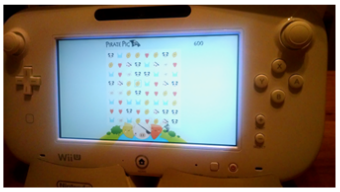


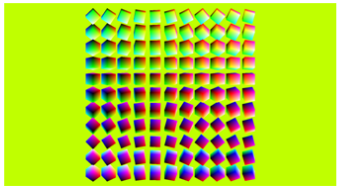
Here's HaxeFlixel's stock MODE demo running on a WiiU. That's not HTML5 either, that's real C++ running natively!

Current Progress

Wii U

MODE (HaxeFlixel)
PiratePig
3D Cubes
BunnyMark
etc





So as of today we have some demos compiling and running using this tech stack on WiiU. We've got HaxeFlixel's MODE, OpenFL's PiratePig, a 3D rotating cubes demo, a gamepad input demo, bunnymark, and audio all confirmed as working.

We also have access to these devkits -- WiiU, PS4, PS3, PSVita, Xbox One. The only thing we don't have yet is the 3DS but we're working on that.

The great news is that once we get one target up and running (as we have done with the WiiU), the cross-platform architecture makes it easier to get each next target.

Step 1

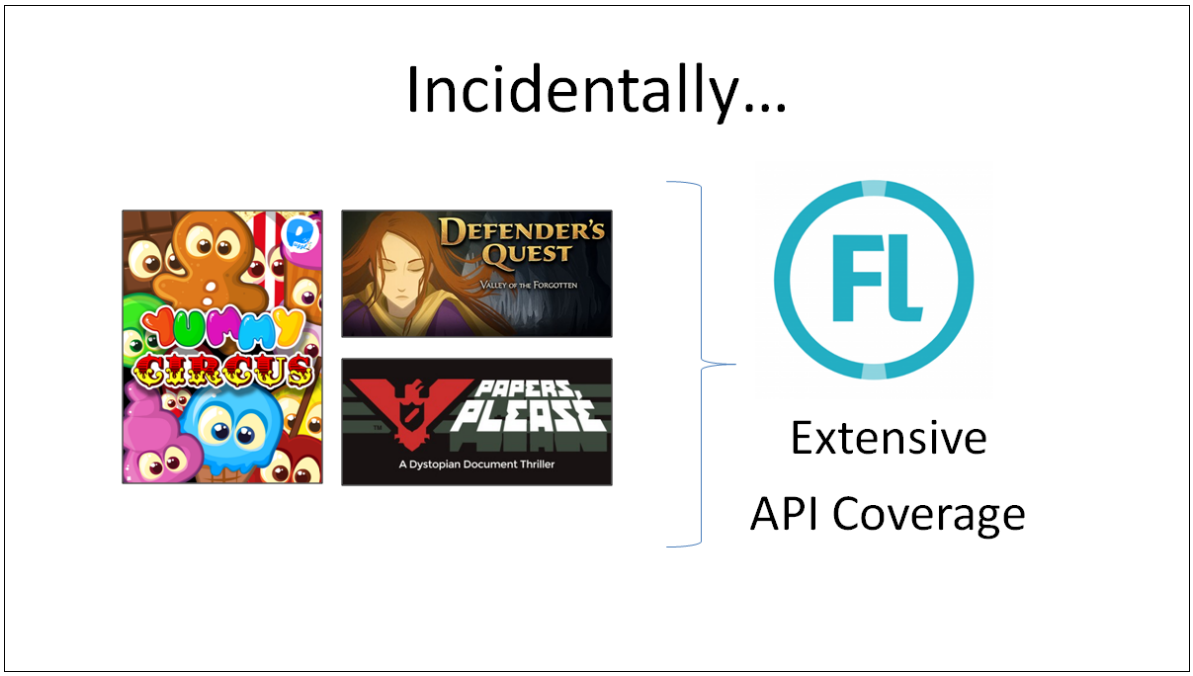
Launch
"Phase 1" games

So what's next? Well, step one is to proceed with our "phase 1" launch games:
Yummy Circus, Papers Please, Defender's Quest HD.

Step 2

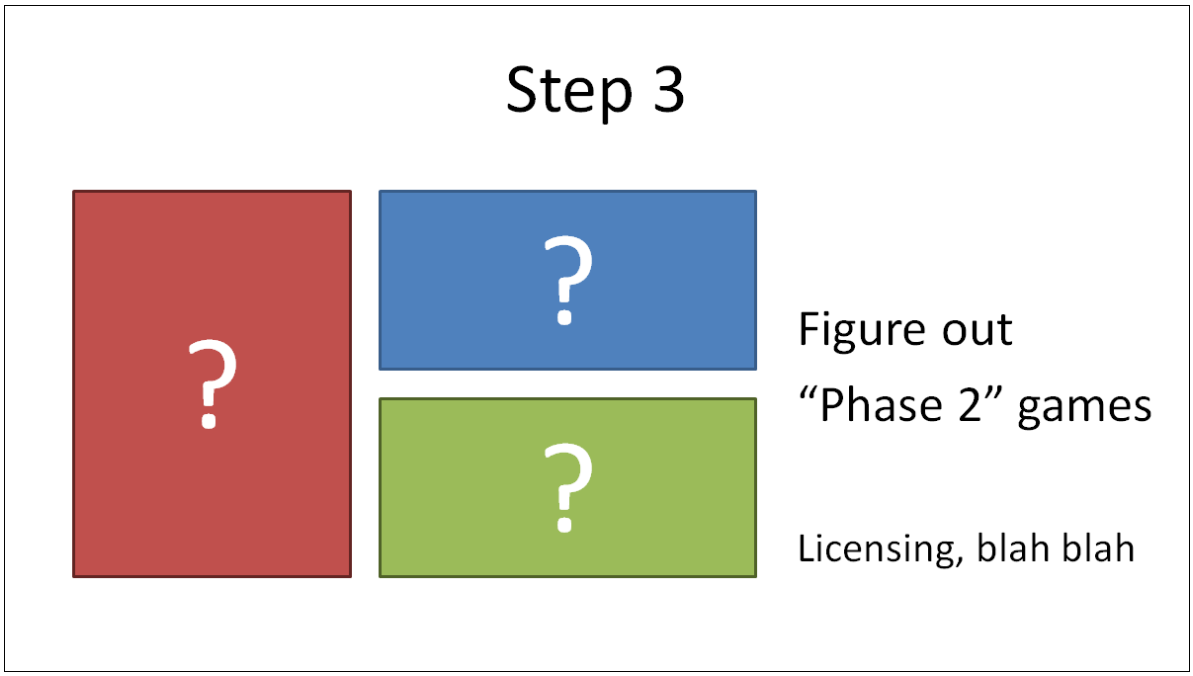
Feedback/
Work out kinks

Step 2 is to see how that went and get feedback, work out the bugs, etc.
Basically confirm that yes, this works.



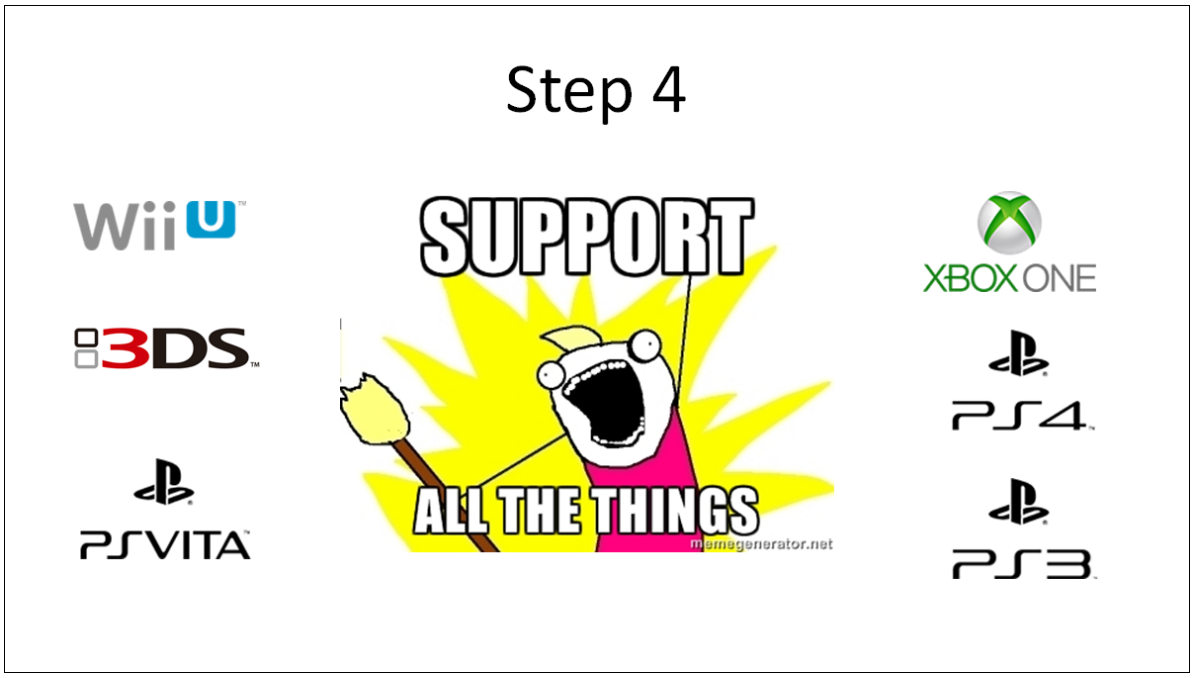
I should mention these 3 particular games give us a lot of API coverage. Yummy Circus is a networked party game with controller input, Papers Please will focus strongly on the Vita target and mobile/touch input, and Defender's Quest uses HaxeFlixel, probably the most popular Haxe-based game engine, as well as its user interface library, Flixel-UI.

So when we get these three games up and running, we'll be very confident that we can get lots of different kinds of games running.



After that is when we figure out how we're going to make this available to others outside of Phase 1, working out terms with the console platform holders and WayForward.

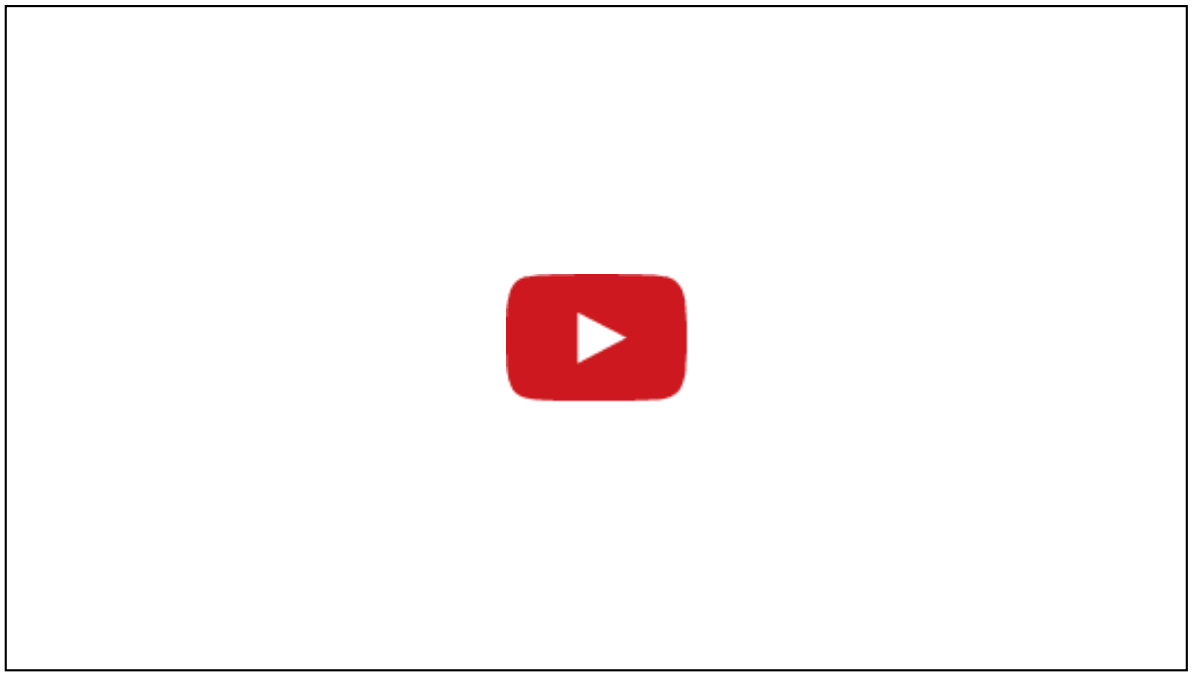
One of the reasons I came to this conference is to find other Haxe developers interested in this solution; If you're among them then you should get in touch with us now so that we can make sure your interests are represented in the final version of this solution.



And after that, we'll focus on getting each and every one of these six consoles fully supported as quickly as we can.

Now, let's watch some videos :)

This first one is a slightly customized demo of OpenFL's stock PiratePig demo. The only thing I've done is add controller input. As you can see, the input is driven by the actual WiiU gamepad controller. Apologies for the lighting, I suck at videos.



This is HaxeFlixel's MODE demo running on the WiiU. Please note that any slowdown you see is caused by two things: first, it's running in debug mode spewing out tons of debug symbols with no optimizations, and second, whenever you blow up one of the ship spawners there's apparently an intentional slow-mo effect programmed into the demo that has nothing to do with performance constraints.

Now for questions. (I'm going to do my best to remember these).

Q: What about 3D? Will this support 3D?

A: Yes. Obviously our first three lead games are 2D, but this technology is 3D-capable. Lime exposes a simple 3D context that you can build something more sophisticated on top of. You can look to tools like Away3D that build on top of Lime/OpenFL to provide more sophisticated 3D engine structure, we're just providing the basics.

Q: What about shaders? Will they be unified across all the platforms or not?

A: Yes, that's the goal. I can't go into too many specifics just yet, but we should be able to provide a unified shader

interface across the console space. We're still working out whether we'll also be able to unify that with the existing desktop/mobile/html5 shaders which are GL based. But the goal is to make it so you don't have to write your shaders 6 times.

Q: What should I do if I want to use this?

A: You should get in touch with us now! Email leveluplabs at gmail dot com for now.

Q: What's the timeline?

A: We hope to have our phase 1 games ready before the end of the year.

Q: Is there a way to test my games on the console stack before I get access to everything?

A: Yes, there will be. There is also a "lime-console-pc" target in addition to "lime-wiiu", "lime-ps4, "lime-xbox1" etc. Basically, wfEngine has a PC target. This is normally redundant to us because we already have Mac/Windows/Linux well supported in OpenFL. However, it provides a way to make a target that represents what the console stack will behave like. So even if you don't have an agreement in place with Nintendo or Microsoft or Sony, you could at least try the console-pc target and

get a reasonable idea of what the console stack will do with your game, and test it on your PC. Also, we're going to stick as close as we can to the OpenFL and Lime APIs, so if your game is already using those you will not be far off from having something that will run on a console.

Q: How will I use this when it's available?

A: The idea is we'll provide you with the "lime-XYZ" library where XYZ is a console platform you are a certified developer for. So sign up with Nintendo for the WiiU, then you can work with us to legally get "lime-wiiu", etc. We're in early talks with the various console maker's middleware departments to make things more convenient, but nothing has been finalized yet. The first stage is just to prove the technology (Phase 1). After that we will work out all the details for Phase 2.

Q: How much will it cost / what are the licensing terms?

A: We're still working that out and we can't announce any fixed numbers yet, but suffice it to say we know people have options and we have no plans to get into the red-ocean, for-profit commercial game engine business. We see this as a cost-sharing method for our community and we will provide this at a cost that allows us to support and maintain it. When someone like Unity offers individual developers console licenses "for free," Unity almost

certainly has a subsidy agreement worked out with partners. We're looking into that sort of thing but I want to emphasize that nothing has been agreed to just yet.

Q: Can this be used for stuff other than games?

A: You betcha! At the conference, TiVo expressed a particular interest, presumably for the sake of getting their apps running on consoles natively.

That's all folks!

If you're interested in this, please get in touch. And if you're a Haxe / OpenFL developer, please help spread the word!

UPDATE June 5 2015:

I'm drowning a bit in email and asking for the same information over and over, so here's a form as a starting point for those who are interested: https://docs.google.com/forms/d/1nOWzWk1ABMbvnXVXe5XZJT7sFfuhVVdZ46tQ9t0OJ3o/viewform?usp=send_form

---

Lars Doucet

Norwegian+Texan Eastern Orthodox Christian game designer with Tourette's Syndrome and Narcolepsy.

Texas • *http://www.leveluplabs.com*