Shipping One Million Lines of Haxe to (Over) One Million Devices

Todd Kulick & Gabriel Dalbec

Overview

- A little bit about TiVo
- TiVo and Haxe
  - Evaluating Haxe for TiVo
  - Converting a large AS3/Flash code base
- Lessons learned
- Future directions

TiVo Company Profile

- Founders Mike Ramsay and Jim Barton
- January 1999, TiVo unveiled it's Personal Television Service at the CES
- Headquarters in Silicon Valley
- Employees: ~600
- 4.5 Million active subscribers

Worldwide Presence

Multiple countries:

United States, Canada, UK, Spain, Sweden, Australia, New Zealand, Mexico, Taiwan, Puerto Rico

Product Line Overview

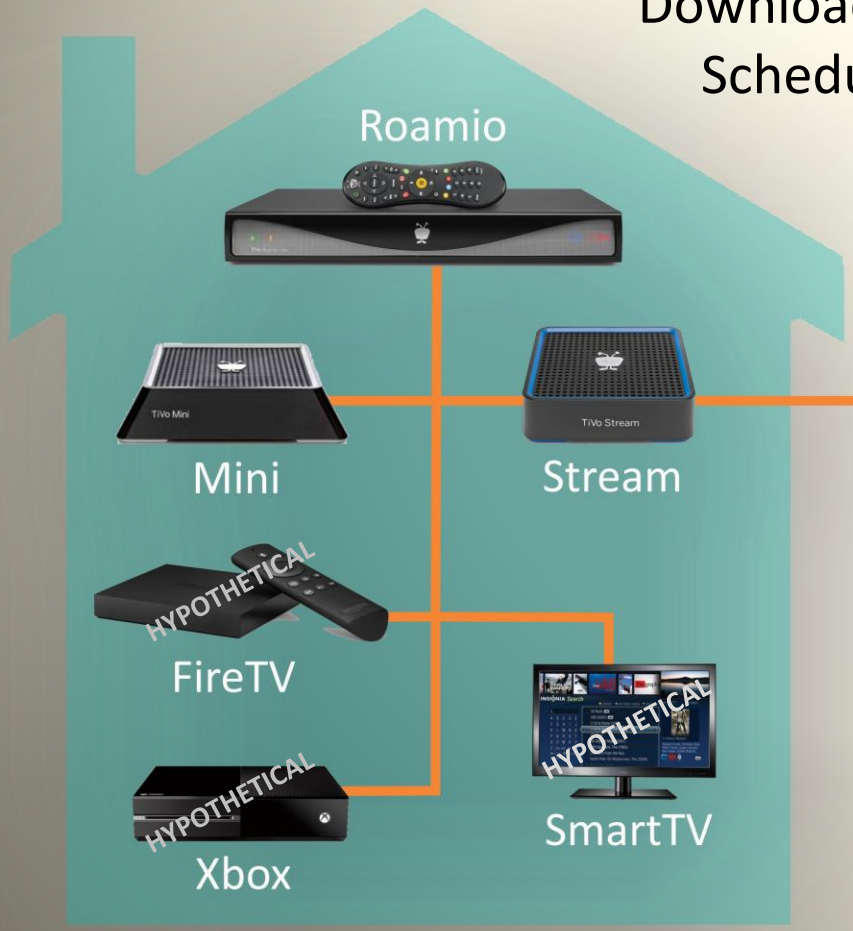
- Digital Video Recorders (Roamio)
  - Powerfully record HD broadcast, cable, IPTV (DVR)
  - Record up to 1000 hours, 6 shows at once
  - MSO/MVPD video-on-demand (VOD) integration
  - Third-party streaming video integration (Netflix, YouTube, Hulu, etc.)
- Second Room Video Devices (Mini)
  - Watch recordings on multiple devices throughout the home
- Mobile Applications
  - Control DVRs (discover content, schedule)
  - Download or stream video
- Online Web Video Portal
  - Stream content to browser

TiVo Anywhere

AT HOME

Stream live or recorded shows
Download recorded shows
Schedule and browse

OUT OF HOME



iPhone - iPad

Andriod
Phone & Tablet

Laptop

TiVo and Haxe

Flash Challenges

TiVo device user interface built on AS3 and Flash.

- ~500,000 lines of AS3 product code
- ~400,000 lines of AS3 tests

But we had looming problems…

- Support for Flash is waning
  - Especially on embedded chipsets
- User interface performance non-ideal
  - On embedded devices performance was, at best, bearable
- Going forward, fewer and fewer devices would support Flash

Could Haxe solve these problems?

Haxe Evaluation Criteria

- Basic evaluation
  - Will Haxe/NME/OpenFL work for us generally?
  - Is the language good (typed, efficient, easy to use/understand)?
  - Are the right APIs/capabilities present (graphics/network/etc.)?
- Technical evaluation
  - Will it work well on set-top box hardware?
  - Will the computational parts be fast(er)?
  - Will the graphics parts be fast enough?
  - Will it work for an application the size/scale of ours?
- Developer workflow evaluation
  - What is the development workflow like?
  - How fast are incremental compiles?
  - How effective/featureful are the code editing/searching tools?
  - How mature are the debuggers?

Haxe Evaluation

To evaluate Haxe for our purposes, we would create a Haxe prototype of our user interface application by converting some of our AS3 code to Haxe…

- Test the as3tohx conversion tool
  - Roughly converted 85% of the code to prototype quality
- Compare Haxe performance to Flash version
- Compare Haxe memory consumption to Flash version
- Compare Haxe image size to Flash version
- Evaluate Haxe development environment and tools

Haxe Evaluation

Beginning early last year, four developers spent three months building a prototype and evaluating it against our criteria

- AS3 to Haxe conversion process worked reasonably
- Prototype's performance was ~30% better
- Prototype's memory consumption was reasonable
- Prototype's image size was large, but workable
- Haxe development tools and IDEs were not quite as good as using Flash, but were acceptable and could be improved

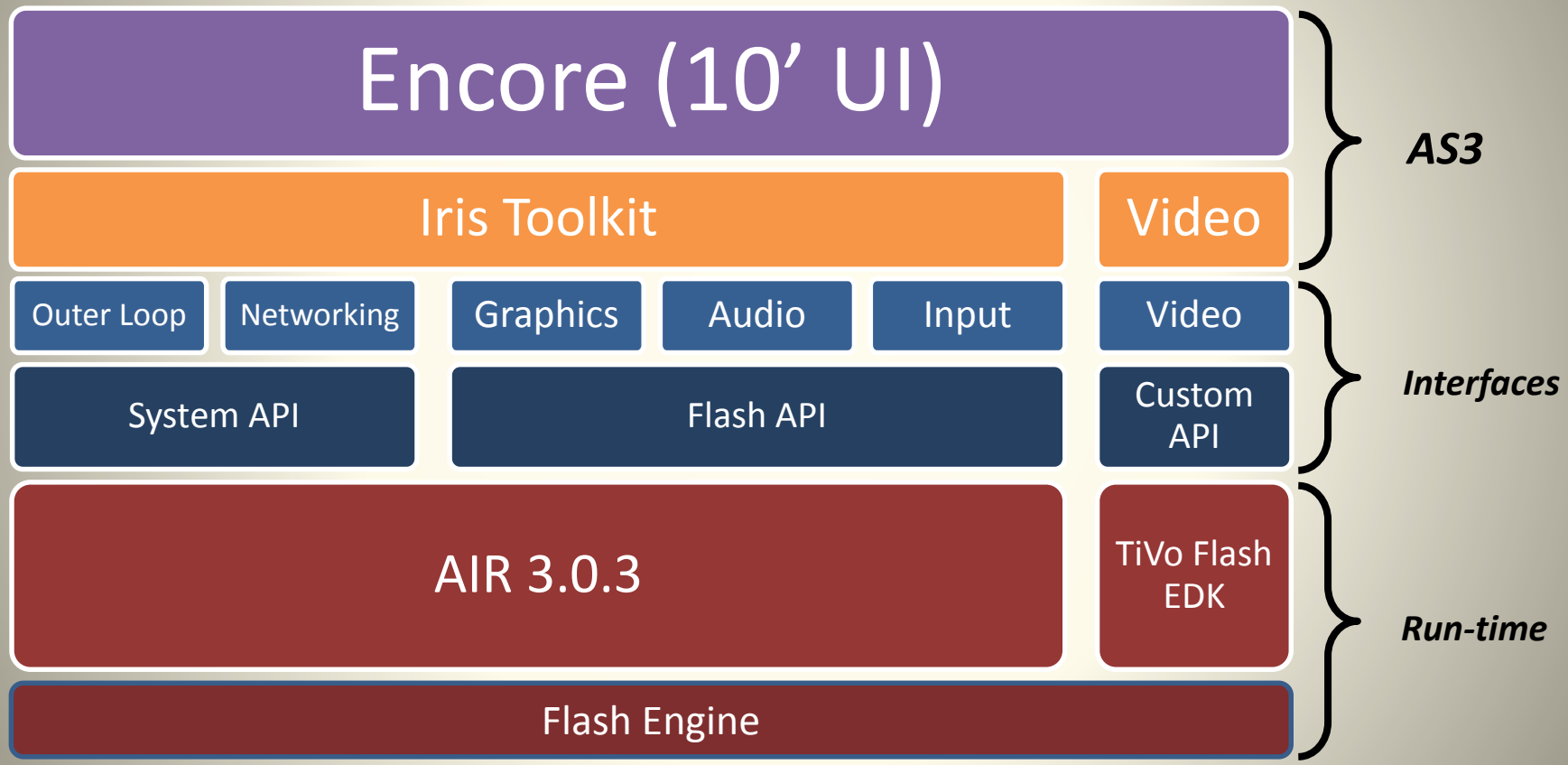
Our evaluation suggested that Haxe *could* meet our strategic, technical and development needs!

So just about one year ago, we began to convert to Haxe!

Haxe Conversion Strategy

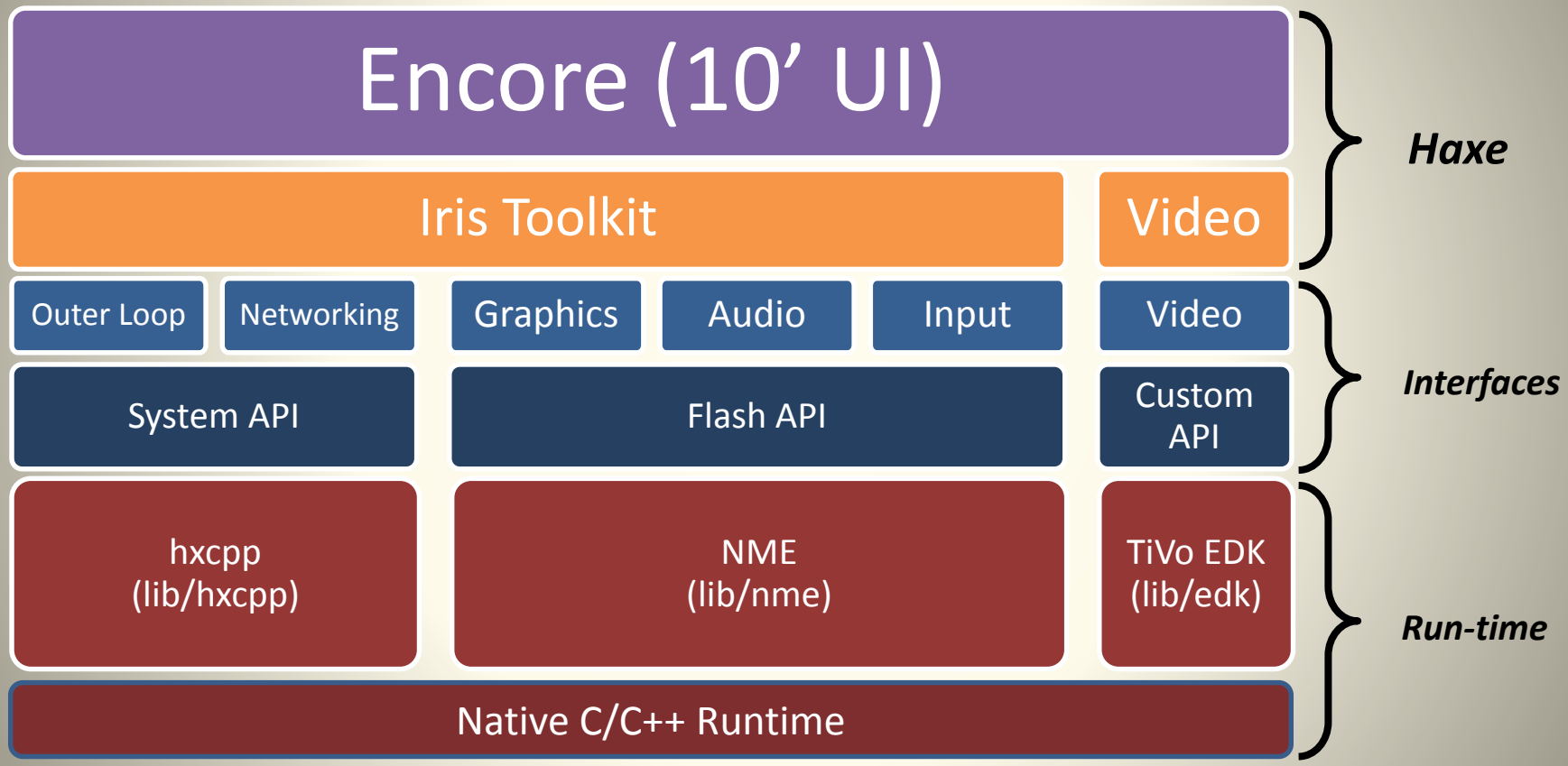
- Strategy
  - Convert AS3 code to Haxe
  - Compile using haxe/hxcpp and TiVo MIPS cross-compilers
  - Use NME/OpenFL to provide Flash API
  - Build new NME back-end on DirectFB API
    - No OpenGL on many embedded set-top boxes
    - Customized for best performance on set-top box devices
- Scope of conversion
  - Total lines of code: >900,000
  - Unit tests: ~17,000

Flash AIR Deployment



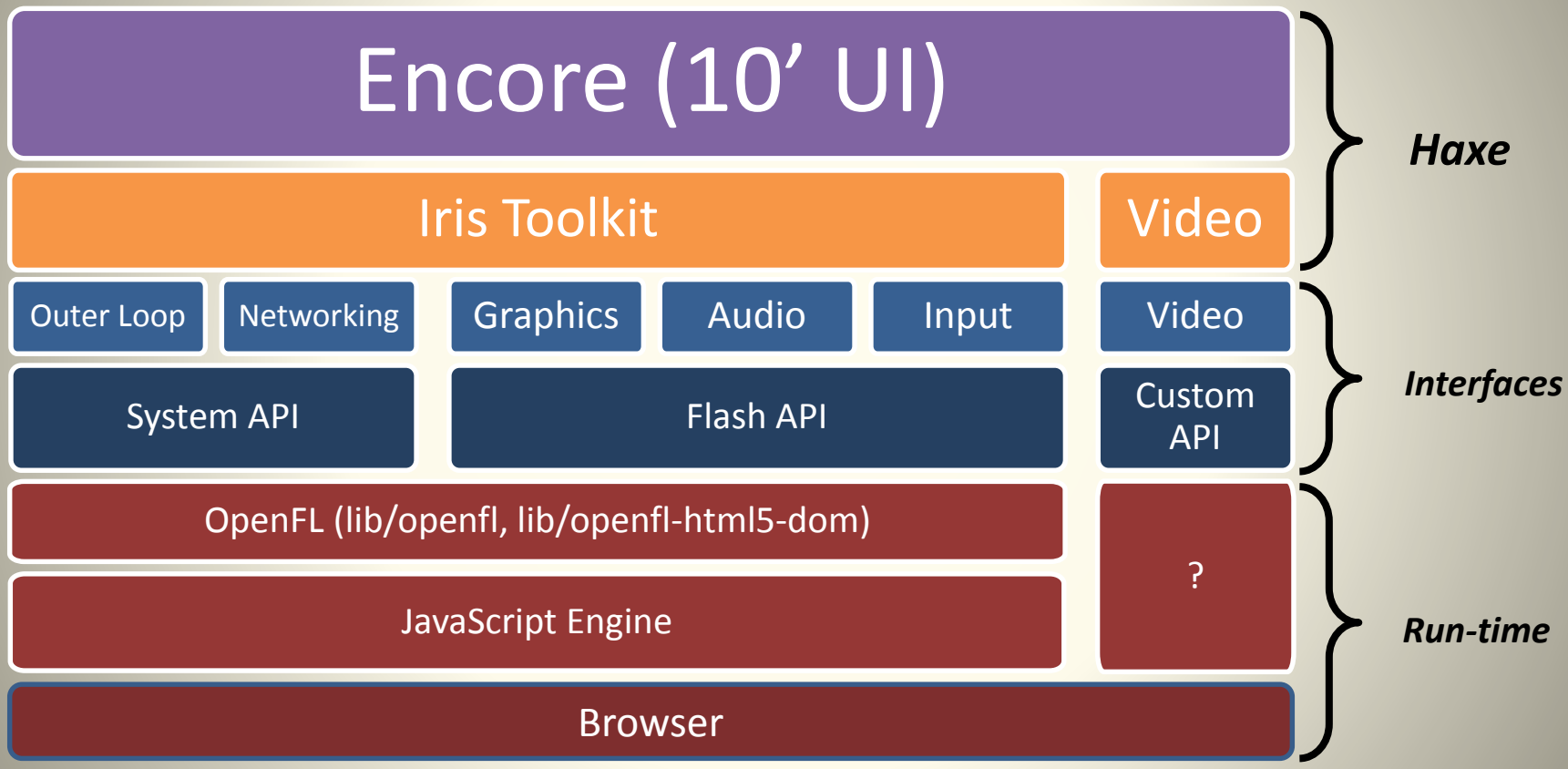
Devices: TiVo set-top boxes

NME C++ Deployment



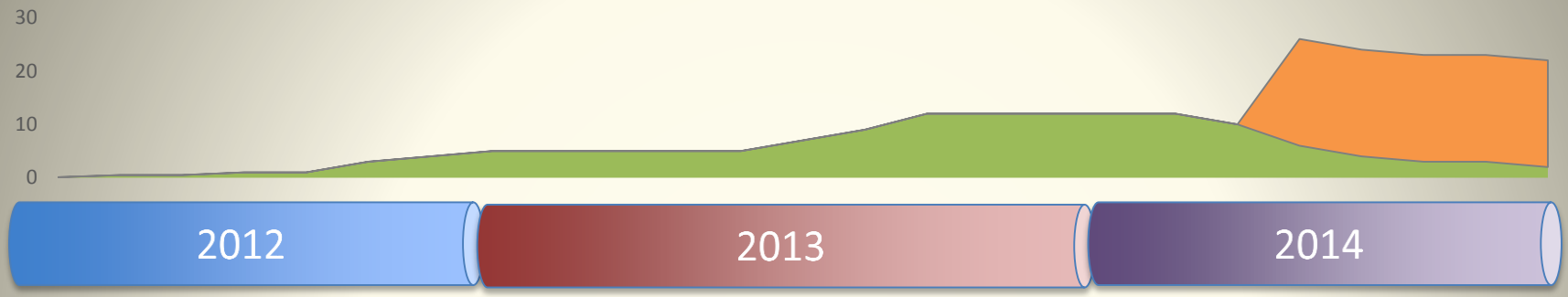
Devices: TiVo set-top boxes, Android TV devices

OpenFL HTML/JS Deployment



Devices: Browsers, Connected TVs & more…

Haxe Conversion Timeline



Spring: Looking for a strategy for “after Flash”

Summer: Began learning about Haxe

Fall: Began evaluating Haxe in earnest

Spring: Decision to switch to Haxe; WWX 2013

Summer: Conversion from AS3 to Haxe begins

Winter: Conversion code complete

Spring: WWX 2014 – TiVo Haxe demo!

Summer: **TiVo Haxe shipping!**

Haxe Conversion Goals

- Adobe/Flash independence
  - …while *leveraging our current user interface codebase*
  - Run-time smaller than embedded Adobe Flash engine, easier to improve, more opportunities
- Performance benefits
  - Better user experience
  - Makes new, additional look & feel features possible
  - No JIT “hiccups”, no JIT bugs(!)
- Multiplatform development efficiencies
  - Build great user experiences that can be targeted at many devices
  - Improvements made in the user experience would potentially have a longer life span and broader applicability

Challenges

Workflow / development environment issues

- IDE support/quality in Linux/OSX
  - No integrated debugging
  - No integrate unit test execution

Application requirements issues

- Application must share graphics/audio resources
- NME designed for constant drawing/games

Haxe architectural issues

- UTF-8 / multilingual support rough
- String vs. Bytes confusion
- Map implementation handling of interfaces as keys

Challenges

Run-time issues

- Endianness issues in hxcpp
- Object churn and memory footprint
- Resulting executable size
  - Generated reflection code large in hxcpp
  - Object/function boxing in hxcpp
  - Template size in hxcpp (not using "void*" design pattern)

Small compatibility issues

- No Date object support of UTC/localtime
- Weak reference bugs in hxcpp

Demo(s)


POPULAR ON NOW
SUGGESTION
Pitch Perfect (2012)
SUGGESTION
Argo (2012)
NEW ON DEMAND
8:02pm
TiVo Central
My Shows
Find TV, Movies, & Videos
Manage Recordings & Downloads
Music & Photos
Showcases & Apps
Settings & Messages
Search
Browse TV & Movies
Browse Web Videos
WishList Saved Searches
Hulu Plus
Amazon Instant Video
Netflix
YouTube
AOL On

TiVo Haxe Contributions

- Improvements in as3tohx convertor (*Dominguez*)
- New hxcpp debugger
- New hxcpp compiled object caching
- IntelliJ IDEA Haxe plugin OSS'd & improved
- Numerous smaller Flash compatibility improvements in NME/OpenFL (*Granick*)
- New NME back-end for TiVo STBs (DirectFB, custom-video, etc.)
  - Smarter redraw logic (only when necessary)
- New OpenFL HTML5 back-end (*Granick*)
  - Support for DOM rendering, optimized for low-end browsers

TiVo Future Directions

- Deployments through other Haxe back-ends/languages/devices
  - HTML/JavaScript
  - Android/iOS
- Pull requests for some additional TiVo improvements
- Continuing IDE/development environment improvements
  - Better IntelliJ environment (building, debugging, etc.)
  - Better multi-threaded support for GC/debugger in hxcpp
- Continuing hxcpp improvements
  - Image size, run-time memory footprint, performance, GC, unreflective
- Continuing Haxe/NME/OpenFL churn improvements
- Continuing performance study/tuning

Haxelibs We Use/Love

TiVo Haxe Wishlist

- Better development environment
  - Better IDE support on Linux
  - Integrate debugger into IDE
  - Integrate munit into IDE
  - Continuing optimization of compiler and language back-ends
- Partial compilation for Haxe (libraries)
- Executable image size reduction when using hxcpp
- Foreign function integration API for “call in”
  - CFFI only defines Haxe “call out”
- Short lambdas(?!)

Haxe has helped TiVo to successfully create a faster, better user experience for our customers.

Questions?