# LESSONS LEARNED MAINTAINING OPEN SOURCE ACTIONSCRIPT PROJECTS

ZEH FERNANDO – FIRSTBORN MULTIMEDIA – OCTOBER 2009

## BACKGROUND

## MC TWEEN

- Tweening “prototypes” for ActionScript 1 (and ActionScript 2)
- First made public in April 2003
- Last version released in April 2006
- Last version had an average 148 downloads/day during the first four months of 2009

## TWEENER

- Static tweening class for ActionScript 2 and 3

- Started in June 2005; first made public in January 2007

- Last version released in June 2009

- Last version had an average 245 downloads/day (latest version, from June to today); around 279,326 downloads in total (stable versions only)

## THE PEOPLE

## THE PEOPLE

If you build it, and it fills a niche, they will come.

Sometimes even if you didn't intend them to.

## THE PEOPLE

Everybody and their mother are building “a framework”.

Focus on stating what it does. How can it help? Why is it different? Why should other people use it?

## THE PEOPLE

Some people will misunderstand what it is about.

They will try to use the code for something it's not meant for and, many times, be really stubborn about it.

## THE PEOPLE

People will request things.

They will insist on asking for some changes, often requesting something that is outside of the original project scope.

They will also come up with some bright ideas from time to time.

## THE PEOPLE

People will report bugs.

They have different scenarios and will reports errors you wouldn't normally catch.

The vast majority of times, however, they will report bugs that never existed.

## THE PEOPLE

Some people will take your project, your project's field or task, or you, WAY too seriously.

## THE PEOPLE

You will be forced to provide support.

Even if you make it clear that you don't want to.

## DOCUMENTATION

## DOCUMENTATION

Documentation takes as much time as writing the code, and it's not as fun.

## DOCUMENTATION

Documentation is vital and must be correct.

```
Tweener.addTween(mymc, {_color:0xffgg33, time:1});
```

## DOCUMENTATION

Suggestion: write documentation as you code. Use ASDoc.

http://livedocs.adobe.com/flex/3/html/help.html?content=asdoc_1.html

```
/**
* This method does something cool.
*
* @param    p1    First parameter description.
* @param    p2    Second parameter description.
*
* @return         Return value description..
*
* @see otherNiceMethod
*/
public function coolMethod(p1:String, p2:Number):Boolean {
   ...
}
```

## API

## API

The approach you take with your API means (almost) everything.

## TATSUO KATO'S DYNTWEEN PROTOTYPE (2000-2002)

```
ball.dynTween({_x:[300, "out", 100], duration: 50});
```

## ROBERT PENNER'S TWEEN CLASS (2002)

```
var twn = new Tween(ball, "_x", Math.easeInOutCirc, ball._x, 300, 50);
```

## MC TWEEN PROTOTYPE (2003)

```
ball.xSlideTo(300, 1, “easeInOutCirc”);
```

## API

API needs change with time.

Communities mature. The language and the platform itself change. You learn more.

## API

Your project's API is pretty much forever.

Changing anything that's public brings pain for everybody.
Think well before publishing; test different approaches.

## API

Much more than I could say:

How to design a good API and why it matters – Joshua Bloch
http://www.youtube.com/watch?v=aAb7hSCtvGw
http://lcsd05.cs.tamu.edu/slides/keynote.pdf

## PROJECT MANAGEMENT

## PROJECT MANAGEMENT

People will want to help, often contributing code of their own even if unrequested.

Be prepared to deal with them – either adding their changes to the project, rejecting it, or bringing them to your team.

## PROJECT MANAGEMENT

Project management with more people is hard.

Prepare to deal with it professionally.

## PROJECT MANAGEMENT

Version control (Subversion, CVS) is an absolute necessity, at least internally, unless it's a small, one-man project.

## PROJECT MANAGEMENT

For larger projects, bug tracking helps dealing with the audience at large.

It also creates a more focused discussion about problems and even feature suggestions.

## PROJECT MANAGEMENT

Having internal documents help, especially when there's more people involved.

Specially useful are the list of project goals, a to-do list, and a development roadmap; for bigger projects, a feature implementation blueprint.

## PROJECT MANAGEMENT

Example of a feature implementation blueprint:
https://wiki.ubuntu.com/SystemCleanUpTool

## PROJECT MANAGEMENT

Suggestion: Google Code offers file and document hosting, SVN repository, and basic bug tracking.

http://code.google.com/projecthosting/

There are alternatives too.

## RELEASE

## RELEASE

Do a lot of testing with each release.

Set test cases and examples, do unit testing if possible.
Release “unstable” (beta; new features) and “stable” (bug correction) versions.

## RELEASE

Pick a distribution license (BSD, GPL, MIT, etc).

Licenses vary a lot and impose limits, obligations and advantages of their own.

http://www.opensource.org/licenses/category

## RELEASE

Not everybody has a version control client on every computer or knows how to operate it.

You can release versions on SVN/CVS more often, but be sure to have a zip version for convenience available somewhere.

## AND FINALLY...

## AND FINALLY...

It can also be really rewarding by making you well-known to the people that matter.

(And you don't have a lot of control over it, so no funny business)

## AND FINALLY...

You will make mistakes.

You will learn, and it's part of the fun. Don't take it personally.

http://code.google.com/events/io/sessions/MythGeniusProgrammer.html

## AND FINALLY...

Suggestion: if you're serious, read this book (it's free!).

Producing Open Source Software – Karl Fogel
http://producingoss.com/