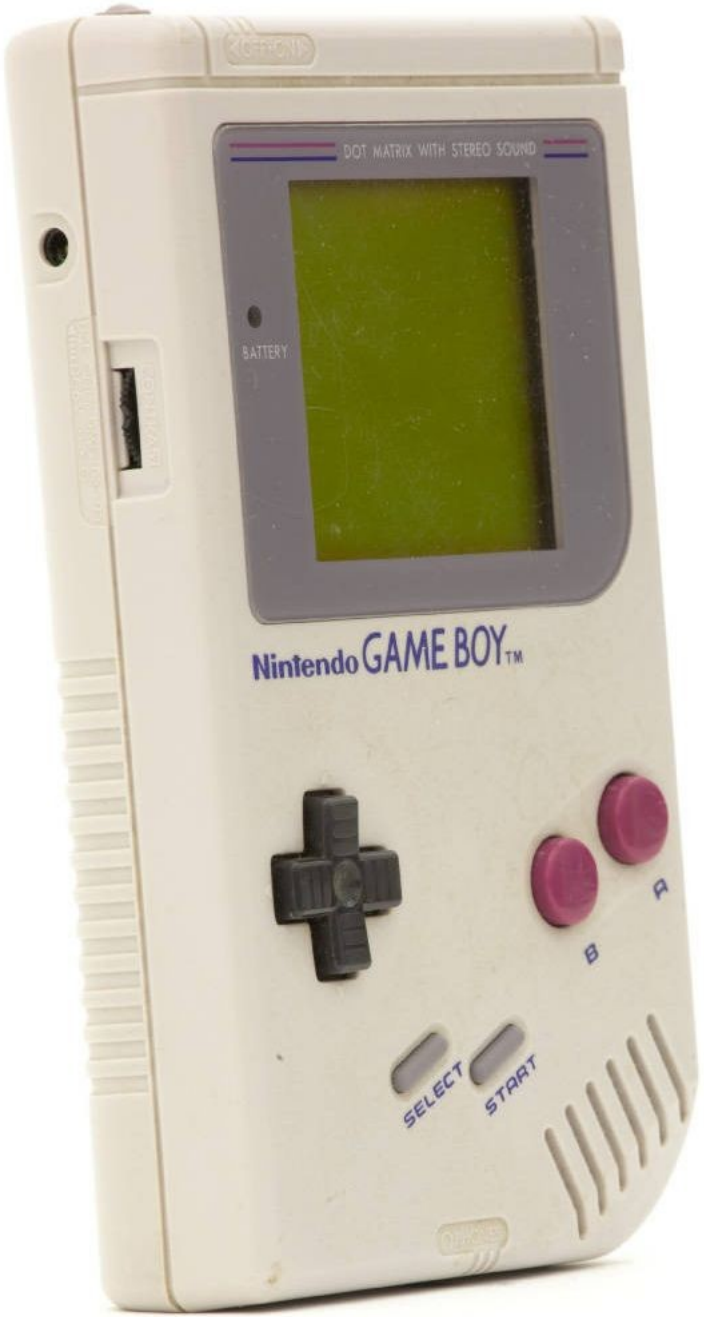
HAXE

Past, Pr[é]sent and Future


DOT MATRIX WITH STEREO SOUND
BATTERY
Nintendo GAME BOY™
B
A
SELECT
START

Past

Past : Origins

- 2001 – Lisp & Ocaml
  - sparkle interest in PL and VM
- 2003 – MotionScript
  - my own VM (inspired by Ocaml – 31bits int)
- 2004 – ASML
  - type checking and type inference of AS2 code
- 2004/2005 – Motion-Types
  - First AS1, then SWF, then MotionScript (Apache)
- 2005 - MTASC

Past : Reasons

- MotionScript VM rewrite (will be NekoVM)
- Motion-Types type system issues
- Unify client and server standard library
- Add JavaScript
- Add Flash9
- No MTASC support for AS3
  - free and standalone adobe compiler

Past : Timeline

- 2005 Oct 22 : Started !
- 2006 Feb 04 : Beta 1
  - only neko and swf8
- 2006 May 17 : 1.0
  - includes neko, swf8 and js
- 2006 Aug 16 : Flash9 (dynamicaly typed)
- 2006 Aug 28 : haxelib
- 2007 July 25 : 1.14 (typed Flash9)

Past : Timeline

- 2008 Jan 13: 1.17 (adds 'inline')
- 2008 Jul 28 : 2.0 (includes PHP)
- 2008 Nov 23 : 2.02 (flash.Memory API)
- 2009 Jul 07 : 2.04 (includes C++, and 'using')
- 2010 : many fixes and improvements

Present

Present - Metadata

- Enrich type/fields with arbitrary data
  - @<ident>(<values>)
- Accessible at runtime :
  - @name("Nicolas")
  - Using haxe.rtti.Meta API
- Compile-time only :
  - @:make({ a : 0, b : false })
  - used by compiler internals (@:final)

Present - Macros

- Initial idea : expr generation

```
class MyMacro {
    @:macro static function compileDate() {
        var p = haxe.macro.Context.currentPos();
        var str = Date.now().toString();
        return { expr : EConst(CString(str)), pos : p };
    }
}
```

Present - Macros

- Can have parameters

```
class MyMacro {
    @:macro static function getFile( e : Expr ) {
        var file = switch( e.expr ) {
            case EConst(c):
                switch( c ) { case CString(s): s; default: null; }
            default: null;
        }
        if( file == null )
            haxe.macro.Context.error("String needed", e.pos);
        var str = neko.io.File.getContent(file);
        return { expr : Econst(CString(str)), pos : e.pos };
    }
}
```

Present - Macros

- Can manipulate parameters

```
class MyMacro {
    @:macro static function makeArray4( e : Expr ) {
        return { expr : EArrayDecl([e,e,e,e]), pos : e.pos };
    }
}
```

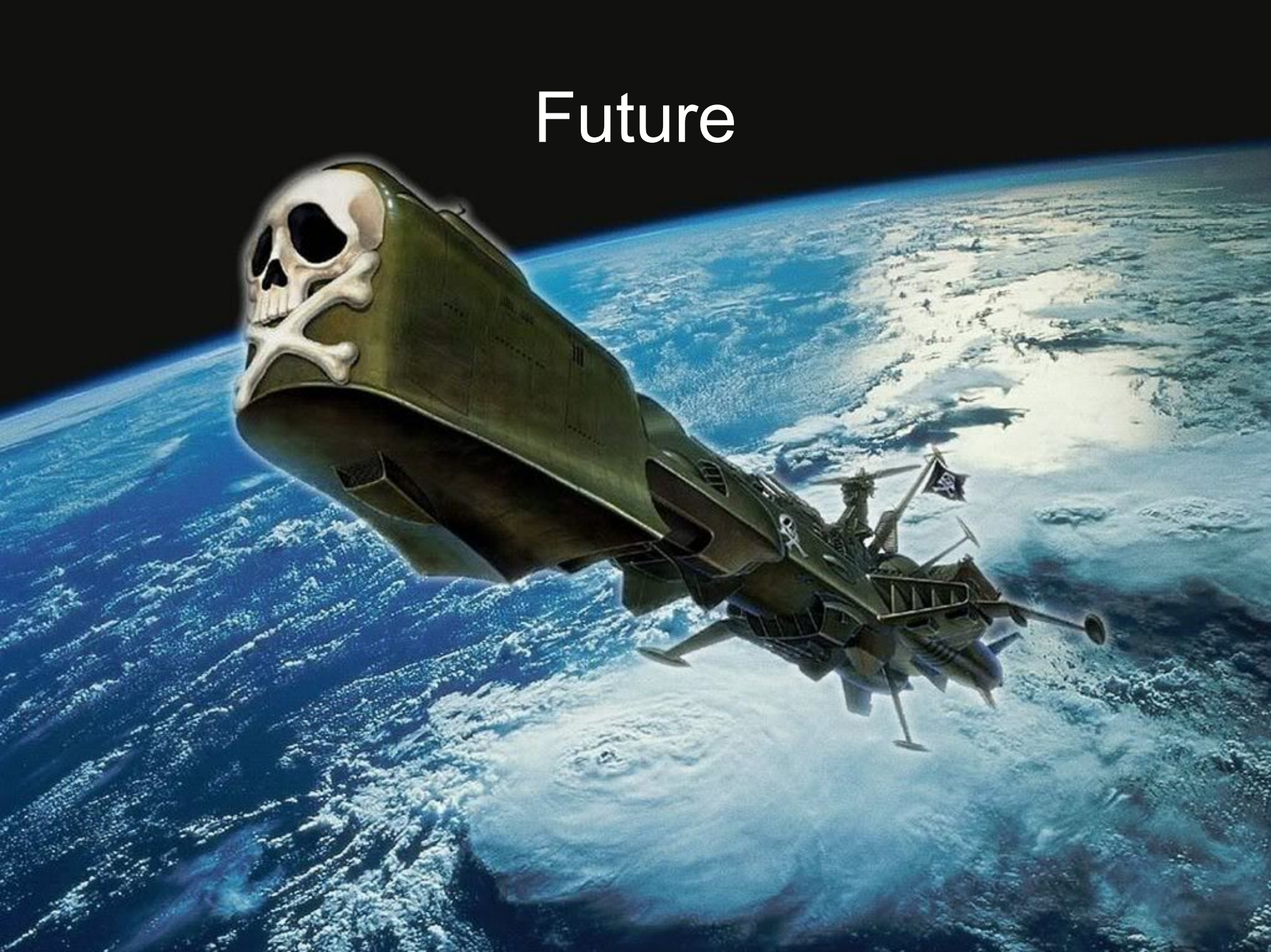
Present - Macros

- Can access types and metadata
- Macros + 'using' on SVN
- SPOD/Macros on SVN
  - a good DSL example
- Compiler configuration :
  - haxe.macro.Compiler API
  - include/exclude lists
  - custom JS output

Present - Flash11

3D API & HxSL Demo

Future



Future - Guidelines

- more syntax, more keywords = bad idea
  - reuse existing syntax with extended meanings
- Everything should by strictly typed
  - unless you use Dynamic
- Keep cross-platformability in mind
  - performances
  - Compile-Time VS Runtime
  - Reduce issues with dynamic systems (inlined JS, scripting, templates, etc)

Future - More Macros !

- better class generation with @:build
- more compiler access
- more features
  - waiting for your requests

Future – More Import !

- import package.*;
- import MyClass.myMethod;
- import MyClass.*;
- import MyClass as M;
- import MyClass.myMethod as F;
- import macro Call(); // ?

Future – haXe 3.0

- New major version = compatibily break
- this in local functions
- More platforms ?
  - C#, Java, etc.
- Packages Overhaul
  - toplevel 'sys' package

Future – haXe 3.0

- Abstract Types

```
abstract Int32 > Int {
    public function toInt( i : Int32 ) : Int {...}
    public function add(a : Int32, b : Int32) : Int32 {...}
    ...
}
```

Future – haXe 3.0

- Switch guards :

```
switch( myEnum ) {
case A(x): if( x < 0 ) foo() else def();
default: def();
}
```

Future – haXe 3.0

- Switch guards :

```
switch( myEnum ) {
case A(x) if( x < 0 ): foo();
default: def();
}
```

Future – haXe 3.0

- Switch guards :

```
switch( myEnum ) {
case A(x) if( x < 0 ): foo();
case A(5): bar();
default: def();
}
```

Future – haXe 3.0

- Switch guards :

```
switch( myEnum ) {
case A(x) if( x < 0 ): foo();
case A(5): bar();
case B(A(0)): somethingElse();
default: def();
}
```

Future – haXe 3.0

- Fix the small quircks
- Waiting for community feedback !

Thank U !