TARDIS: Go for Haxe!

A talk by Elliott Stoneham at WWX2014 on 24th May 2014

Haxe as a compilation target for other languages

a new use-case

- 50 years ago: watched very first Dr Who episode
- 40 years ago: started programming (punch cards)
- 20 years ago: became a manager
- 10 years ago: stopped programming
- 1 year ago: discovered Go and Haxe…

Go is like the TARDIS

A small idiosyncratic exterior,
conceals large-scale quality engineering.

Haxe is like a sonic screwdriver

An easy to use tool, built on engineering excellence,
that is useful in a wide range of situations.

overview

- Introduction to Go
- TARDIS Go transpiler
- Go —> Haxe issues
- Go/Haxe interaction
- Hopes & dreams

the birth of Go in 2007

“When the three of us got started, it was pure research. The three of us got together and decided that we hated C++.

...we started off with the idea that all three of us had to be talked into every feature in the language, so there was no extraneous garbage put into the language for any reason.”

—Ken Thompson (who designed and implemented the original Unix operating system)

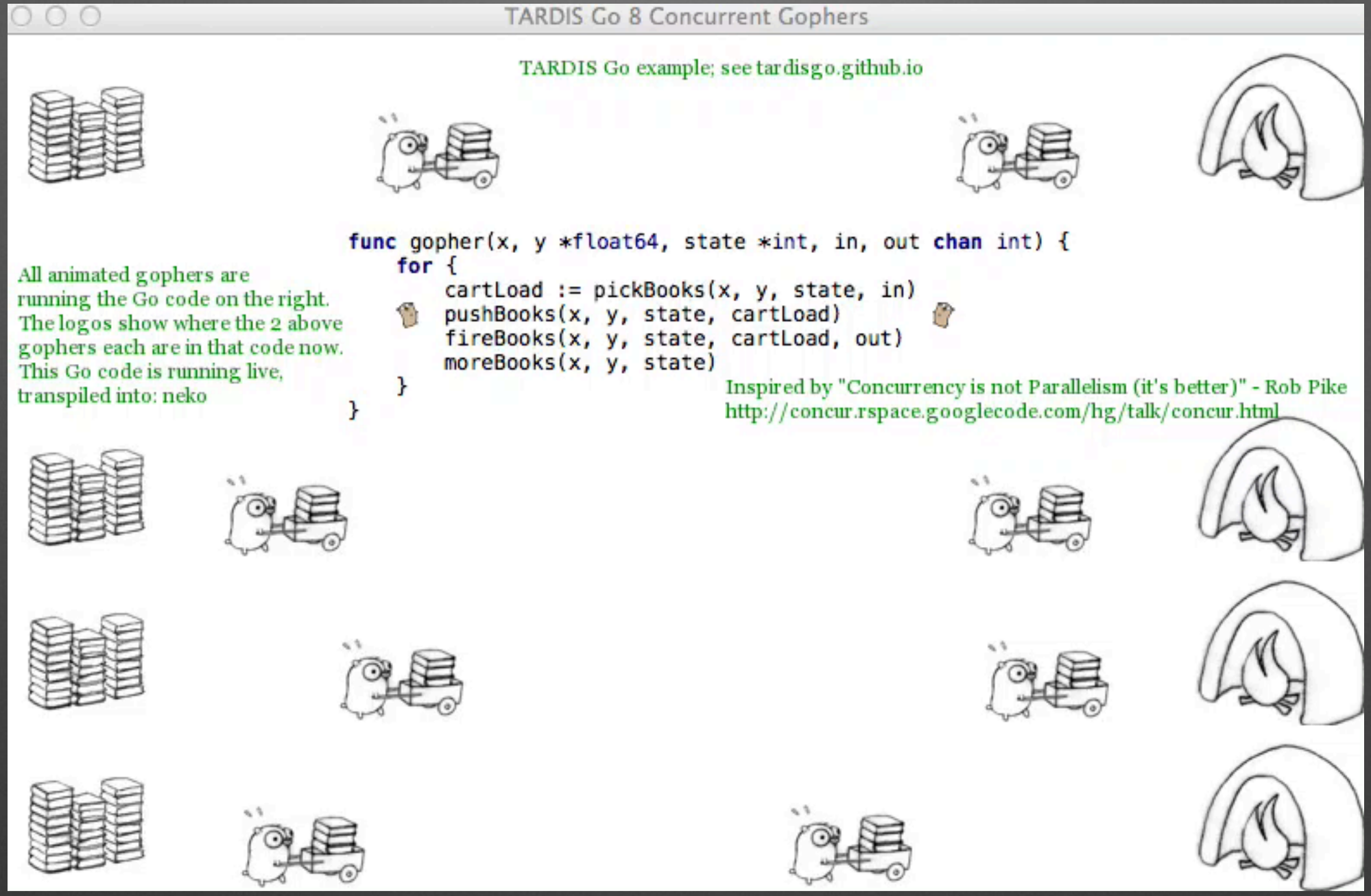
the objectives of Go

- *“Readable, safe, and efficient code.*
- *A build system that scales.*
- *Good concurrency support.*
- *Tools that can operate at Google-scale.*"

— Robert Griesemer (worked on code generation for Google's V8 JavaScript engine)

“Now, Go makes much more sense for the class of problems that C++ was originally intended to solve.”

–Bruce Eckel, author and
founding member of the ANSI/ISO C++ standard committee

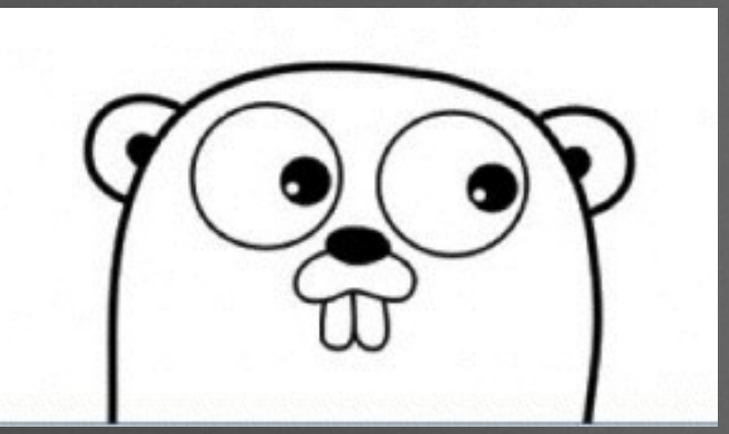


burn those C++ manuals!

Go —> Haxe + Haxe / OpenFL
(running on neko target)

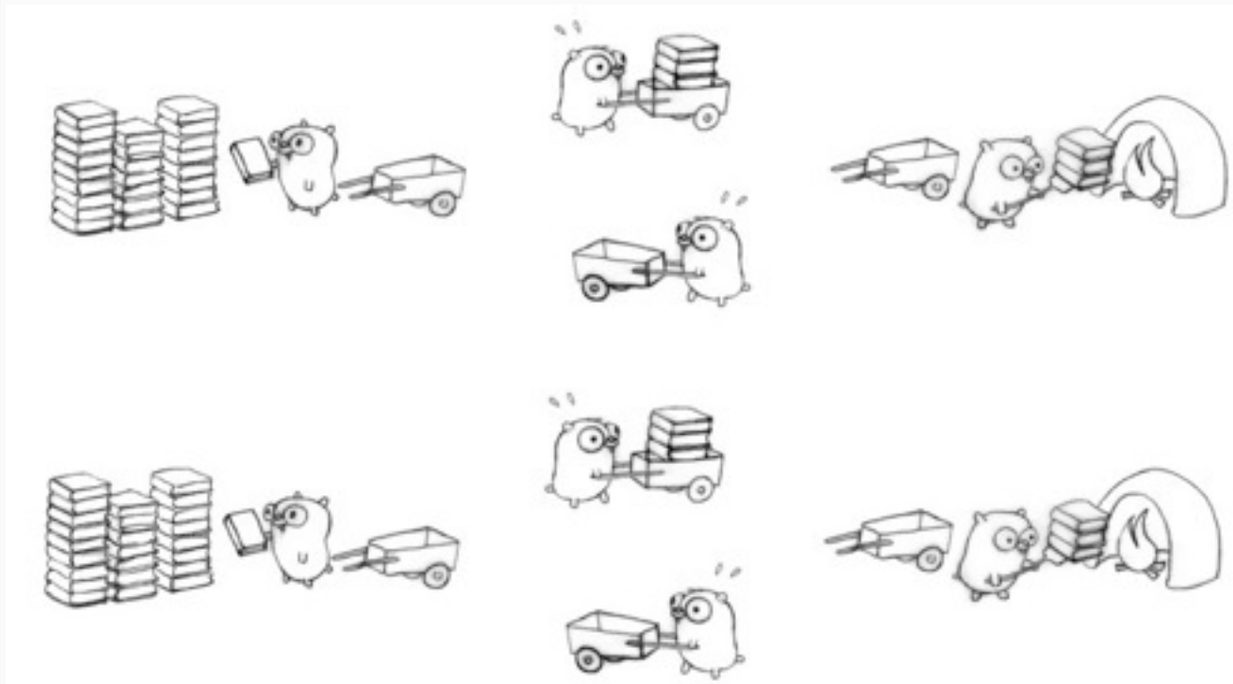
Where is Go now?

- Open-source more than 4 years, 2 official compilers
- Google-scale usage: dl.google.com, youtube.com etc.
- Non-Google: docker.com, nsq.io, juju.ubuntu.com etc.
- 800 attendees at first conference last month in Denver



Go vs Haxe

- Github repos: Go >25,600; Haxe > 1,900
- Twitter: Go >11,100; Haxe >1,700
- Google+: Go+ >13,800; Haxe >800
- TIOBE Index for March 2014: Go #36; Haxe not listed



tardisgo.github.io

TARDIS Go -> Haxe transpiler

TARDIS Go objectives

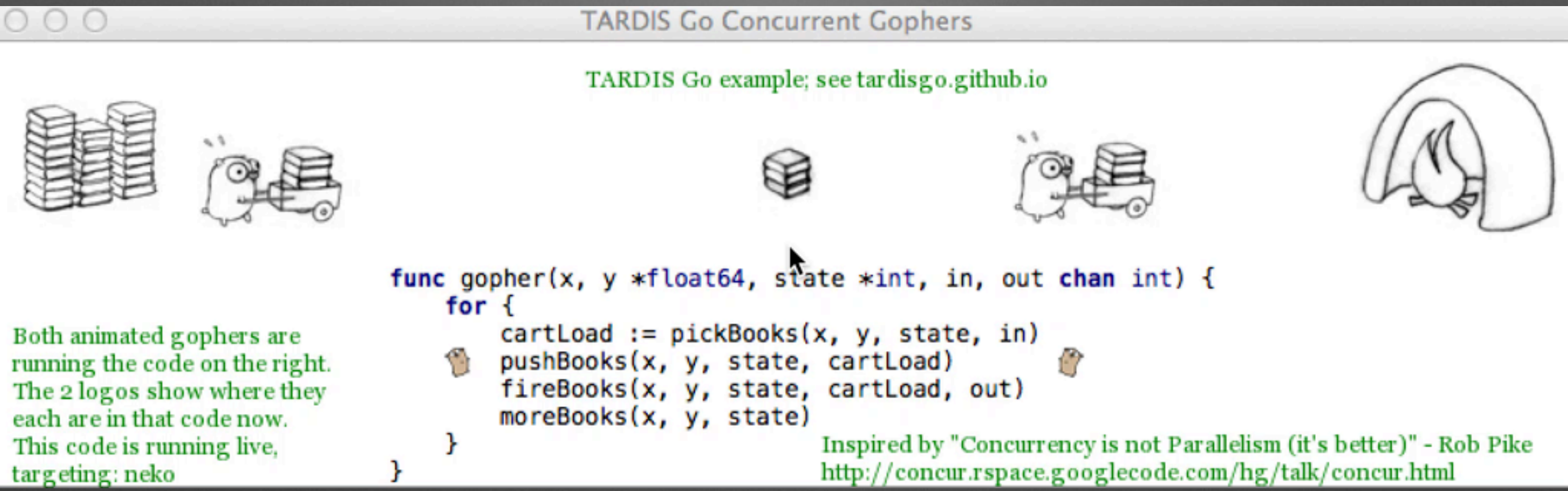
- Go running on Haxe cross-platform targets
- Go using cross-platform UI APIs like OpenFL
- Show that Haxe is viable as an intermediate language
- Create a viable Go/Haxe programming ecosystem
- Influence the future directions of Haxe and Go

concurrency

Go provides simple concurrency primitives:

- goroutines
  - lightweight threads: “`go doit(a,b,c)`”
- channels
  - typed thread-safe communication & synchronisation
  - send: “`ch <- 3`” receive: “`x, ok = <-ch`”

tardisgo.github.io



- Each of the two sprites is a Go “goroutine"
- The pile of books in the middle is a Go “channel”

The 3 phases of the TARDIS Go transpiler

`Go program to be compiled =>`

(1) Use Go library packages to create an intermediate form that entirely describes the original Go program

`=> Single Static Assignment (SSA) form =>`

(2) Generate new Haxe code with the same behaviour

`=> Haxe =>`

(3) Use normal Haxe compilation process

The 36 SSA (Single Static Assignment) Instructions

	Value?	Instruction?
*Alloc	✔	✔
*BinOp	✔	✔
*Builtin	✔	✔
*Call	✔	✔
*ChangeInterface	✔	✔
*ChangeType	✔	✔
*Convert	✔	✔
*DebugRef		✔
*Defer		✔
*Extract	✔	✔
*Field	✔	✔
*FieldAddr	✔	✔
*Go		✔
*If		✔
*Index	✔	✔
*IndexAddr	✔	✔
*Jump		✔
*Lookup	✔	✔
*MakeChan	✔	✔
*MakeClosure	✔	✔
*MakeInterface	✔	✔
*MakeMap	✔	✔
*MakeSlice	✔	✔
*MapUpdate		✔
*Next	✔	✔
*Panic		✔
*Phi	✔	✔
*Range	✔	✔
*Return		✔
*RunDefers		✔
*Select	✔	✔
*Send		✔
*Slice	✔	✔
*Store		✔
*TypeAssert	✔	✔
*UnOp	✔	✔

```
func fact(n int) int {
    if n == 0 {
        return 1
    }
    return n * fact(n-1)
}
```

go -> ssa example

```
# Name: main.fact
# Package: main
# Location: glug.go:9:6
func fact(n int) int:
.0.entry:                                                  P:0 S:2
   t0 = n == 0:int                                        bool
   if t0 goto 1.if.then else 2.if.done
.1.if.then:                                                P:1 S:0
   return 1:int
.2.if.done:                                                P:1 S:0
   t1 = n - 1:int                                          int
   t2 = fact(t1)                                           int
   t3 = n * t2                                             int
   return t3
```

go -> ssa -> haxe

```
public function run():Pogo_main_fact {
while(true){
switch(_Next) {
case 0: // entry
this.setLatest(9,0);
this.SubFn0();
case 1: // if.then
this.setLatest(9,1);
_res= 1;
this._incomplete=false;
Scheduler.pop(this._goroutine);
return this; // return 1:int *ssa.Return @ glug.go:11:3
case 2: // if.done
this.setLatest(11,2);
this.SubFn1();
_SF1=Pogo_main_fact.call(this._goroutine,[],_t1);
_Next = -1;
return this;
case -1:
this.setLatest(13,-1);
_t2=_SF1.res();
 // _t2 = fact(t1) *ssa.Call @ glug.go:13:15
this.SubFn2();
_res= _t3;
this._incomplete=false;
Scheduler.pop(this._goroutine);
return this; // return t3 *ssa.Return @ glug.go:14:2
default: throw "Next?";}}}
private inline function SubFn0():Void {
var _t0:Bool;
_t0=(p_n==0); // _t0 = n == 0:int *ssa.BinOp @ glug.go:10:7
_Next=_t0 ? 1 : 2; // if t0 goto 1.if.then else 2.if.done *ssa.If near glug.go:10:7
}// end SubFn0
private inline function SubFn1():Void {
_t1=(p_n-1); // _t1 = n - 1:int *ssa.BinOp @ glug.go:13:17
}// end SubFn1
private inline function SubFn2():Void {
_t3=(p_n*_t2); // _t3 = n * t2 *ssa.BinOp @ glug.go:13:9
}// end SubFn2
```

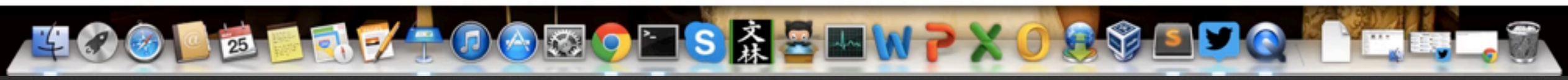
```
func fact(n int) int {
    if n == 0 {
        return 1
    }
    return n * fact(n-1)
}
```

```
# Name: main.fact
# Package: main
# Location: glug.go:9:6
func fact(n int) int:
.0.entry:
   t0 = n == 0:int
   if t0 goto 1.if.then else 2.if.done
.1.if.then:
   return 1:int
.2.if.done:
   t1 = n - 1:int
   t2 = fact(t1)
   t3 = n * t2
   return t3
```

QuickTime Player File Edit View Window Help 100% Sun 08:37 Elliott Stoneham

fact10 — bash — 79×20

```
Elliotts-MacBook-Air:fact10 elliott$
```



```
}
if *allFlag {
    results := make(chan string)
    for _, cmd := range targets {
        go doTarget(cmd, results)
    }
    for _ = range targets {
        fmt.Println(<-results)
    }
}
```

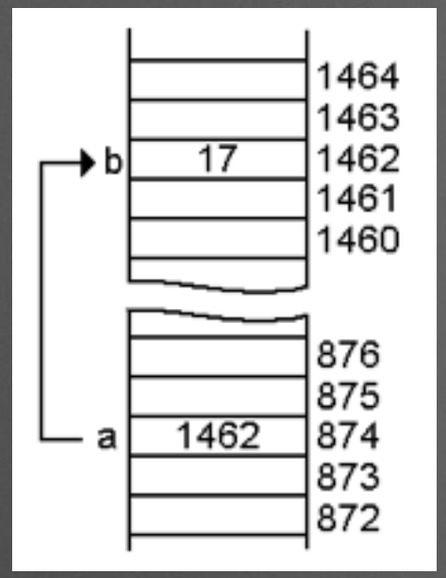
Go to compile and run all the Haxe targets in parallel

Go/SSA -> Haxe issues

- Just a proof-of-concept at the moment
- Haxe 3.2: cross-target Haxe unicode, types & bytes
- TODO: cross-target pointers/concurrency/locking
- TODO: Go implementation of Haxe standard library
- Haxe irritation: Auto-mapping of Int “0xFFFFFFFF” into “-1” for js and php, when they are not the same thing

string types

- Go “string” type encoding is utf-8, but Go also has the concept of a “rune” (a 32-bit full Unicode character)
- In Haxe, a string might be implemented as utf-8 or utf-16, depending on the platform
- TARDIS Go uses the native string encoding (for calling consistency) translating on the fly as required, with the runtime distinguishing between the two encodings based on the length of the ‘字’ character



pointer emulation

- Unlike most languages, Go pointers can’t reference outside their type, so memory can be a collection of host objects, with garbage collection by the host runtime

- Currently, emulated pointers are object references + offsets into arrays of dynamic objects (a very simple solution that works for all targets)

- TODO: generate a new Haxe type for each possible Go pointer type so that dynamic values are not required, thus speeding up execution

goroutines

- Implemented using co-routines, emulating multiple stacks
- Each Go function is a Haxe class, with all the local Go variables
- SSA block # used as the Program Counter
- Haxe timer and other events run the co-routines

```
}
#if (!php) private #end class Go_main_fact extends StackFrameBasis implements StackFrame {  // fact10.go:3:6
var p_n:Int;
public function new(gr:Int,_bds:Array<Dynamic>, p_n : Int) {
super(gr,3,"Go_main_fact");
this._bds=_bds;
this.p_n=p_n;
Scheduler.push(gr,this);
}
var _res:Int;
public inline function res():Dynamic {return _res;}
var _t0:Bool=false;
var _t1:Int=0;
var _SF1:StackFrame;
var _t2:Int=0;
var _t3:Int=0;
var _Next:Int=0;
public static inline function callFromHaxe( p_n : Int) : Int {
if(!Go.doneInit) Go.init();
var _sf=new Go_main_fact(0,[], p_n).run();
while(_sf._incomplete) Scheduler.runAll();
return _sf.res();
}
public static inline function callFromRT( _gr, p_n : Int) : Int {
var _sf=new Go_main_fact(_gr,[], p_n).run();
while(_sf._incomplete) Scheduler.run1(_gr);
return _sf.res();
}
public static inline function call( gr:Int,_bds:Array<Dynamic>, p_n : Int) : Go_main_fact
{return new Go_main_fact(gr,_bds, p_n);
}
```

```
public function run():Go_main_fact {
while(true){
switch(_Next) {
case 0: // entry
this.setLatest(3,0);
this.SubFn0();

case 1: // if.then
this.setLatest(3,1);
this.setPH(5);
_res= 1;
this._incomplete=false;
Scheduler.pop(this._goroutine);
return this; // return 1:int *ssa.Return @ fact10.go:5:3

case 2: // if.done
this.setLatest(5,2);
this.SubFn1();
this.setPH(7);
_SF1=Go_main_fact.call(this._goroutine,[],_t1);
_Next = -1;
return this;
case -1:
this.setLatest(7,-1);
_t2=_SF1.res();
 // _t2 = fact(t1) *ssa.Call @ fact10.go:7:17
this.SubFn2();
_res= _t3;
this._incomplete=false;
Scheduler.pop(this._goroutine);
return this; // return t3 *ssa.Return @ fact10.go:7:2

default: Scheduler.bbi();}}}
private inline  function SubFn0():Void {
var _t0:Bool;
this.setPH(4);
_t0=(p_n==0); // _t0 = n == 0:int *ssa.BinOp @ fact10.go:4:7
_Next=_t0 ? 1 : 2; // if t0 goto 1.if.then else 2.if.done *ssa.If near fact10.go:4:7
}// end SubFn0
private inline  function SubFn1():Void {
this.setPH(7);
_t1=Force.toInt32((p_n-1)); // _t1 = n - 1:int *ssa.BinOp @ fact10.go:7:19
}// end SubFn1
private inline  function SubFn2():Void {
_t3=Force.toInt32((p_n*_t2)); // _t3 = n * t2 *ssa.BinOp @ fact10.go:7:11
}// end SubFn2
}
```

goroutine TODO

- improve by using threads, for Haxe targets that support them
- Don’t implement goroutines if not used, they add a significant execution overhead and are not always required

calling Haxe libs from Go

- Go packages beginning with an underscore define Haxe functionality; they can be auto-generated from the Haxe code or library documentation with target-specific capital letter prefixes:

```
s := string(_haxeapi.Xhaxe_Http_requestUrl(someURL))
switch tardisgolib.Platform() {
case "cpp": _haxeapi.Pcpp_Lib_println(s)
case "php": _haxeapi.Hphp_Lib_println(s)
default: println(s)
}
```

using Go vars from Haxe

- all Go globals are pointers
- all pointer targets are Dynamic, which needs to change

```
• x = Go.pkgName_varName.load();
• Go.pkgName_varName.store(y);
• aPtr = Go.pkgName_varName.addr(42);
```

calling Go code from Haxe

- `x=Go.Go_pkgNam_funcNam.callFromHaxe(y,z);`
- TODO: there is a need to build a simple "methodFromHaxe" calling interface, as using Go methods from Haxe is currently too complex, for example:
  - `_SF1=Go__star_main_dot_rect_area.call(this._goroutine,[],_t0); // by reference`
  - `_SF2=Go_main_dot_rect_perim.call(this._goroutine,[],Deep.copy(_t5)); // by value`

USE YOUR MOUSE TO PUT THE GOPHER IN THE BOX

awaiting MOUSE_DOWN

```
func handleMouse() {
    for {
        e := <-mouseEvents
        switch e.event {
        case MOUSE_DOWN:
            Status = "MOUSE_DOWN"
            cacheOffsetX = SpriteX - e.x
            cacheOffsetY = SpriteY - e.y
            awaitingMouseUp := true
            for awaitingMouseUp {
                e = <-mouseEvents
                switch e.event {
                case MOUSE_MOVE:
                    Status = "MOUSE_MOVE"
                    SpriteX = e.x + cacheOffsetX
                    SpriteY = e.y + cacheOffsetY
                case MOUSE_UP:
                    Status = "MOUSE_UP"
                    awaitingMouseUp = false
                default:
                    Status = "awaiting MOUSE_UP"
                }
            }
        default:
            Status = "awaiting MOUSE_DOWN"
        }
    }
}
```

final interactive example

adapted from OpenFL handling mouse events example

Haxe calls to generated code

- `Scheduler.timerEventHandler(e); // schedule the Go code`
- `Go_main_MouseDown.callFromHaxe(event.stageX,event.stageY);`
- `Go_main_MouseMove.callFromHaxe(event.stageX,event.stageY);`
- `Go_main_MouseUp.callFromHaxe(event.stageX, event.stageY);`

Haxe -> Go

- https://code.google.com/p/haxego/ by Lee Sylvester seems to be inactive
- Efficient server-side operation & concurrency support
- Shared runtime library with TARDIS Go
- Go style concurrency in Haxe?

? -> Haxe -> ?

- TARDIS Go
- Tarwin's AS3 to Haxe Conversion Script
- Typescript to Haxe Conversion Script
- PHP to Haxe Conversion Utility
- CS2HX - C# to haXe converter
- Write a LLVM or GCC compiler Haxe back-end?

http://tardisgo.github.io

- improve TARDIS Go to be a usable tool
- improve automatic generation of Go definitions for Haxe libraries “gohaxelib” (written in Haxe)
- Work to get Go style concurrency into Haxe?
- Questions?

Image Sources

- Gopher Logo by Renée French
- TARDIS image (c) BBC from http://bbc.co.uk
- http://commons.wikimedia.org/
- http://www.wikipedia.org/
- http://memegenerator.net/Your-Country-Needs-You
- Project related images from relevant project sites
- Other images self-created