LAMBDA LOUNGE

Pixie

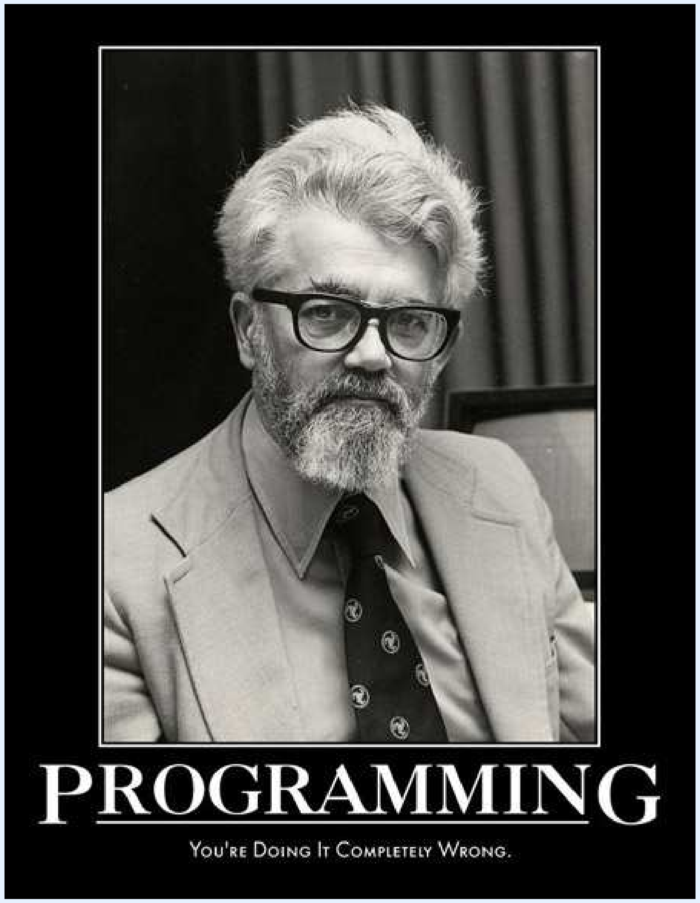
Christopher Mark Gore

`cgore.com`

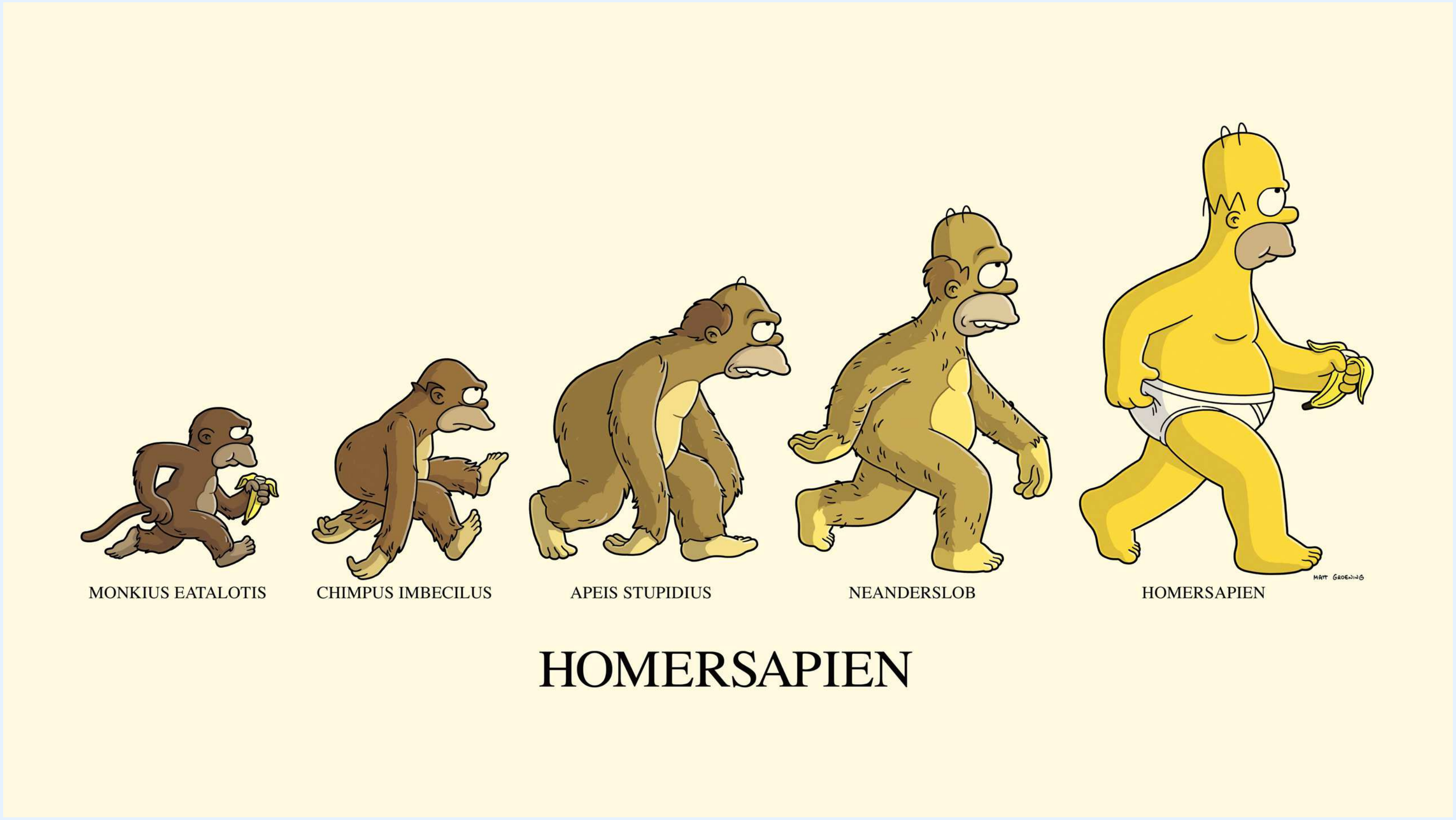
Thursday, December 3, AD 2015

We write Clojure at The Climate Corporation, and we're hiring! Come work with us!

Some people actually program in languages other than Lisp.



I started using Common Lisp in 2004 for evolutionary computation as my M.S. thesis, and quickly learned to love Lisp.



I even think markup languages in web forums should be full-fledged lisps.

```
Welcome to the future of crapflooding!

\defun{\crapflood [\n]
  \dotimes{\n
    \b{Netcraft \blink{confirms} it;}
    the JVM is naked and petrified!
    \br
  }
}
\crapflood{1000}

\it{Wasn't that fun?}
```

And then I got a real job doing embedded C for an avionics firm up in Milwaukee.

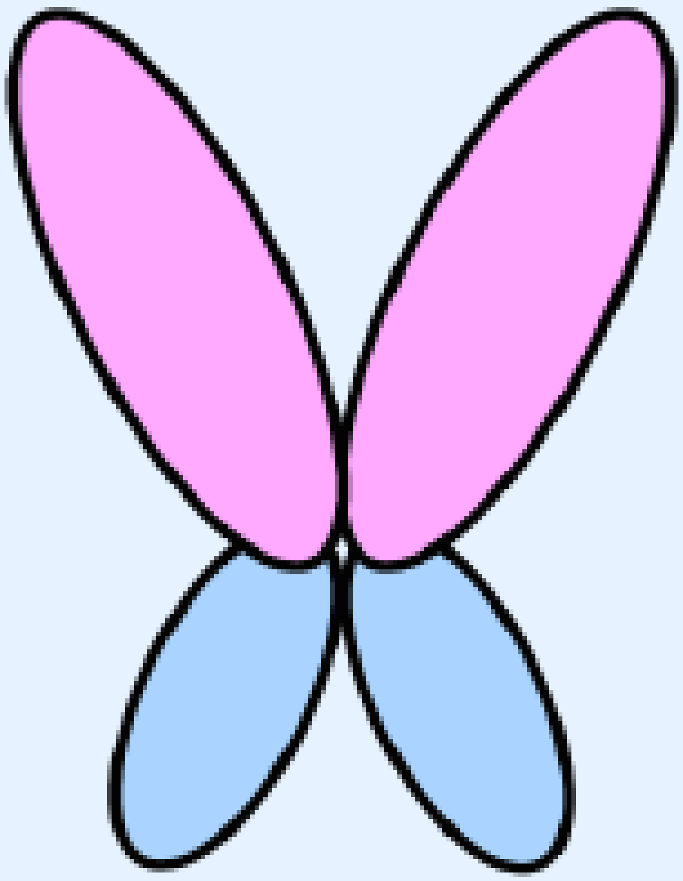
Around 2009 I started messing around with Ruby a lot, and it's actually pretty nice for a not-quite-Lisp.

But for the last two years, I’ve been doing Clojure as my main gig, and that’s been pretty awesome.

There’s just one problem I really have with Clojure …

Java™

Pixie is very early in development, inspired by Clojure (but not a port/fork/clone), and doesn't run on top of the JVM.

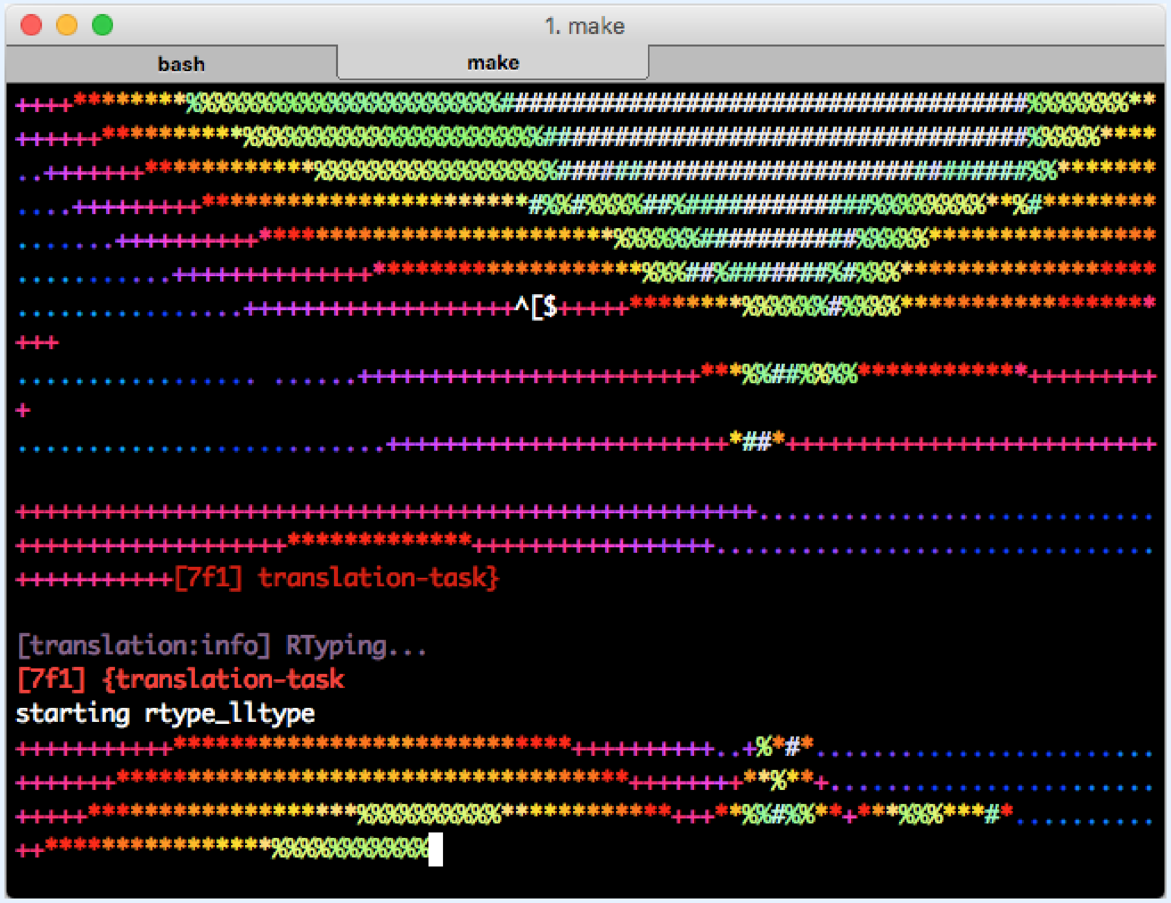


Pixie sits on top of RPython, a subset of Python originally created for PyPy.

Let’s make a Pixie!

```
$ git clone git@github.com:pixie-lang/pixie.git
$ cd pixie
$ make build_with_jit
$ ./pixie-vm # REPL = goodness
```

Building Pixie takes a while, but at least it’s pretty to watch it go.



```
++++********%%%%%%%%%%%%%%%%%%%%%%%####################################%%%%%%%%**
++++++**********%%%%%%%%%%%%%%%%%%%%%##################################%%%%%****
..+++++++************%%%%%%%%%%%%%%%%%#################################%%*******
....+++++++++***********************#%%#%%%%##%#############%%%%%%%%**%#********
.......++++++++++*************************%%%%%%###########%%%%%****************
...........++++++++++++++*******************%%%##%########%#%%******************
................+++++++++++++++++++^[$+++++********%%%%%%#%%%%******************
+++
.................  ......++++++++++++++++++++++++***%%##%%%%************++++++++
+
..........................++++++++++++++++++++++++*##*++++++++++++++++++++++++++

++++++++++++++++++++++++++++++++++++++++++++++++++++............................
+++++++++++++++++++*************+++++++++++++++++...............................
+++++++++++[7f1] translation-task}

[translation:info] RTyping...
[7f1] {translation-task
starting rtype_lltype
+++++++++++****************************++++++++++..+%*#*........................
+++++++************************************++++++++**%**+.......................
+++++*******************%%%%%%%%%%************+++**%%#%%**+***%%%***#*..........
++****************%%%%%%%%%%%
```

There's already a lot of cool stuff there.

```
$ ./pixie-vm
user => "Hello,␣Pixie!"
"Hello,␣Pixie!"
user => (println "Hello,␣Pixie!")
Hello, Pixie!
nil
user => (+ 1 2 3)
6
user => (defn foo [x] (+ x 4077))
<inst pixie.stdlib.Var>
user => (foo 12)
4089
```

Namespaces work in manner just like in Clojure.

```
user => (ns foo (:require [pixie.math :as math]))
nil
foo => (math/sin 1.2)
0.932039
foo => (math/sin 0.0)
0.000000
foo => (math/sin 3.14159)
0.000003
foo => (math/sin (/ 3.14159 2))
1.000000
```

Lots of basic stuff isn't quite there yet though.

```
foo => math/PI
ERROR:
 in pixie function repl_fn

in pixie/repl.pxi at 27:24
                    (let [x (eval form)]
                              ^
in internal function eval

in <unknown> at 5:1
math/PI
^
RuntimeException: :pixie.stdlib/AssertionException
Var PI is undefined
```

Numerics work as expected.

```
user => (+ 1 2)
3
user => (+ 1 2.0)
3.000000
user => (/ 1 2)
1/2
user => (/ 1 2.0)
0.500000
user => (/ 12)
1/12
```

Strings work as expected.

```
user => "foo"
"foo"
user => (ns foo (:require [pixie.string :as s]))
nil
foo => (str "foo" "bar")
"foobar"
foo => (count "foo")
3
foo => (s/upper-case "why␣should␣we␣shout?")
"WHY␣SHOULD␣WE␣SHOUT?"
```

Conclusion

Questions?