TWEETS 14.1K | PHOTOS/VIDEOS 361 | FOLLOWING 720 | FOLLOWERS 1,051 | FAVORITES 4,086 | More

thattommyhall
@thattommyhall

Well known to those that know him well. Good friend to his good friends. Occasional mountaineer, part time runner. Available in fine bookstores everywhere.

Tweets | Tweets and replies

Pinned Tweet

thattommyhall @thattommyhall · 6 Jun 2013

Borrowed some money to start a painting and decorating company. We have a really nice van and you can use whatever brushes you like #hiring

Escaping DSL hell by having parenthesis all the way down

DSL

Problems with Puppet

Namespaces?

module foo

```
class fooapp {
  exec { 'install':
    command     => '/install/foo',
  }
}
```

module bar

```
class barapp {
  exec { 'install':
    command     => '/install/bar',
  }
}
```

Iteration

```
file { ["/root/one","/root/two"]:
  ensure => present
}


nagios_host { ["first", "second"]:
  use => 'generic-host',
}
```

But you need to know ruby anyway...

```
Puppet::Type.type(:file).provide(:posix) do
    desc "Normal Unix-like POSIX support for file management."

    def create
        File.open(@resource[:name], "w") { |f| f.puts "" } # Create an empty file
    end

    def destroy
        File.unlink(@resource[:name])
    end

    def exists?
        File.exists?(@resource[:name])
    end
end
```

Experimental Features: Lambdas and Iteration

Warning: This document describes an **experimental feature,** which is not officially supported and is not considered ready for production. See here for more information about experimental features in Puppet, especially if you are using Puppet Enterprise.

Status: In Puppet 3.6, the future parser performs much better than it used to. We think it's about on par with the default parser, but haven't extensively speed tested it with real-world manifests. It should still be considered a preview, but can now be used with larger test environments.

The new language features in the future parser are still being designed and considered, and there is ongoing debate over how they should work and whether they should be an official part of Puppet.

Ansible is just YAML...

```yaml
---
- name: Create Instance
  hosts: localhost
  roles:
    - role: ec2_ubuntu_instance
      security_groups: ['all-servers', 'comms', 'commsinternal' ]
      role_name: "commsinternal"
      count: 2
      elastic_ip: True
      zones:
        - a
      subnet_type: private
      instance_type: c1.medium
tasks:
  - ec2_tag: resource="{{ item.1.id }}" region="{{ region }}" state=present
    with_subelements:
      - ec2.results
      - tagged_instances
    args:
      tags:
        Stages: "production"
```

If you give people a 'language' they will expect loops

- maybe lambdas
- probably namespaces

This has been done before…

Chef gets it right

If you can do

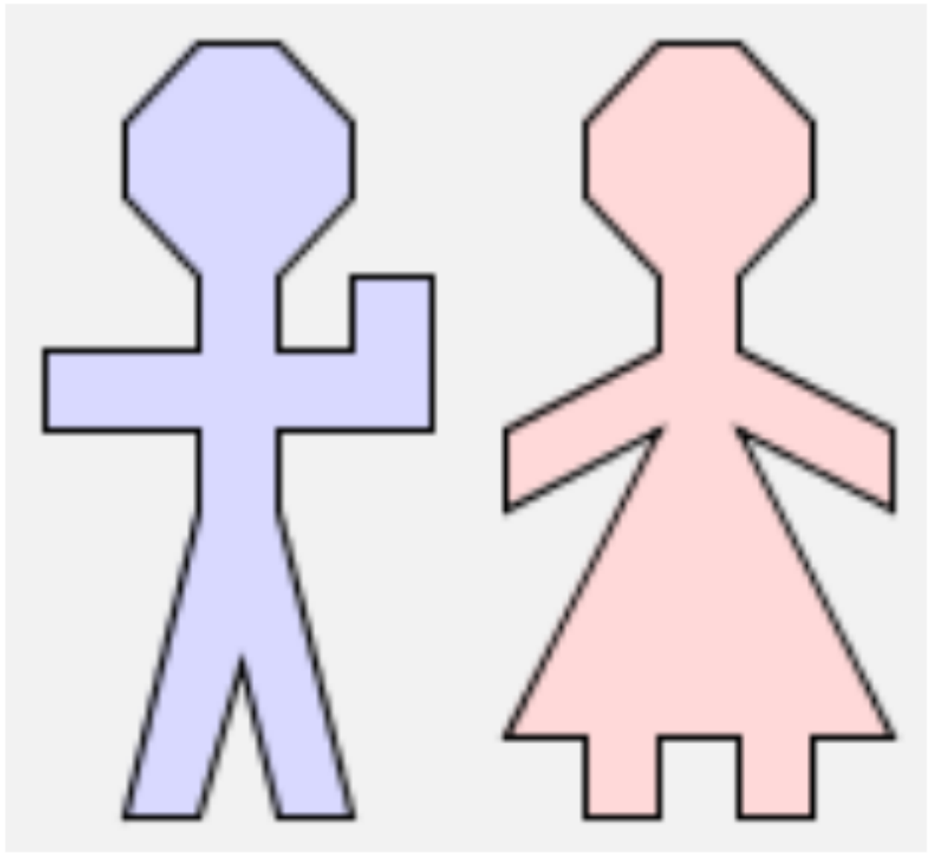
```
file "/path/to/somefile" do
  action :create
end
```

then you can do

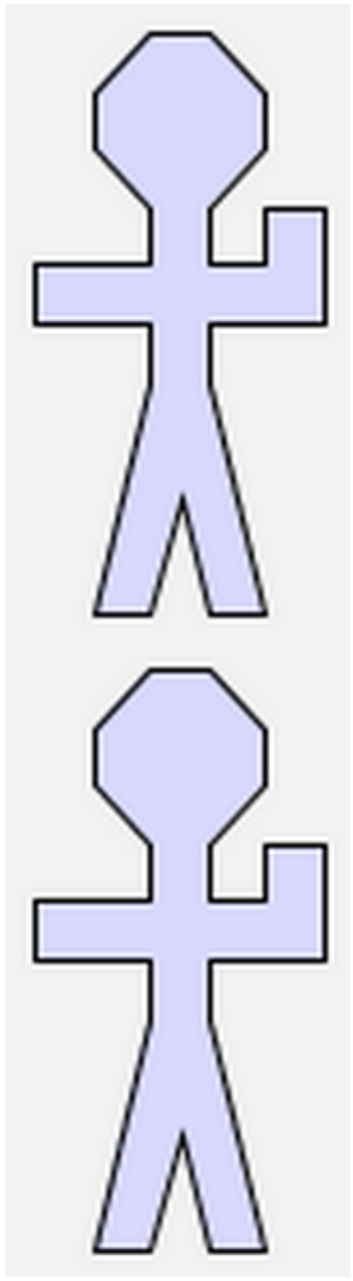
```
filepath = "/some/path/or/other"

["one", "two"].each do |filename|
  file "#{filepath}/#{filename}" do
    action :create
  end
end
```

Teaching people to program

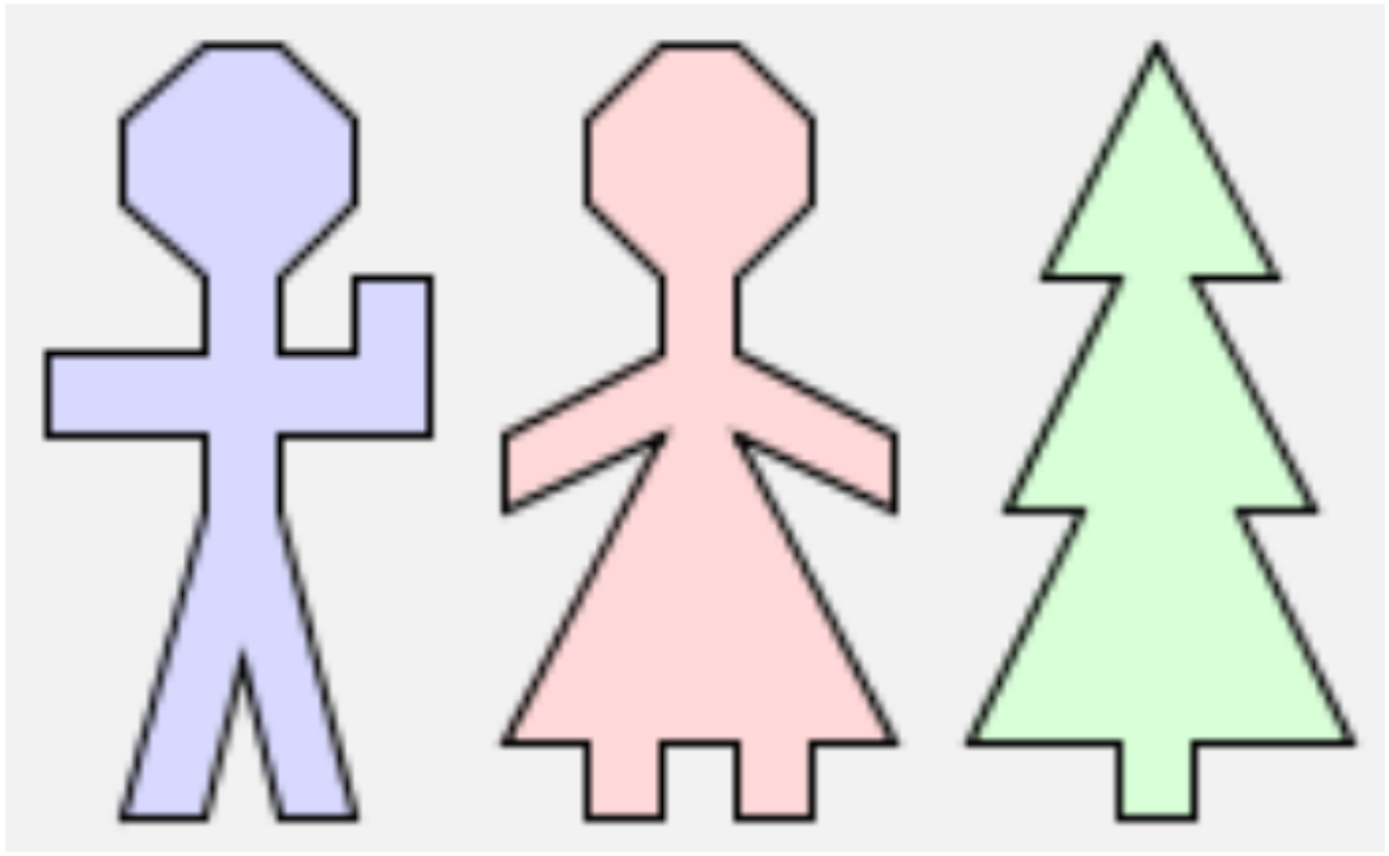
Geomlab



man $ woman

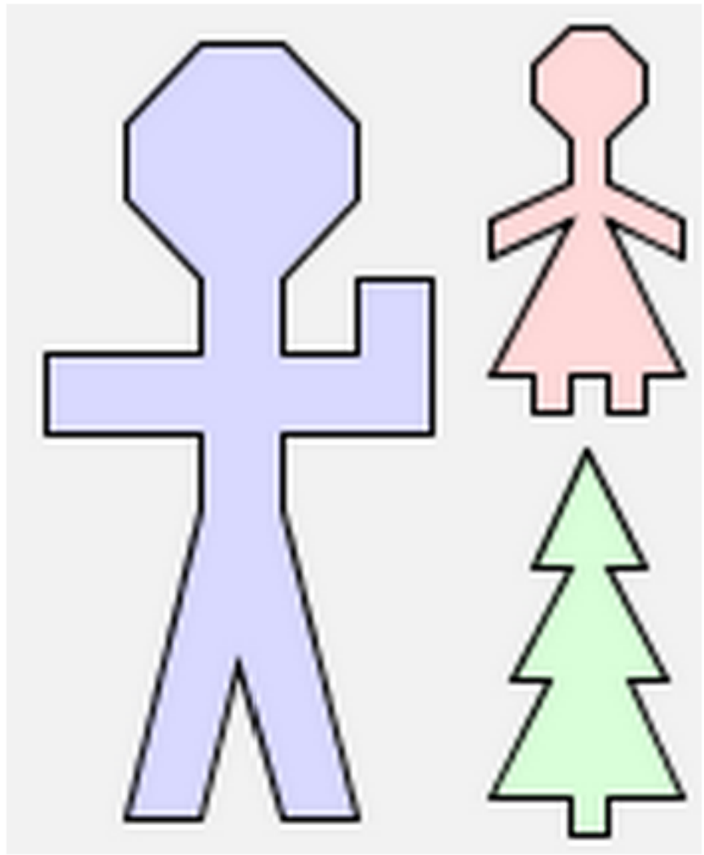


man & man

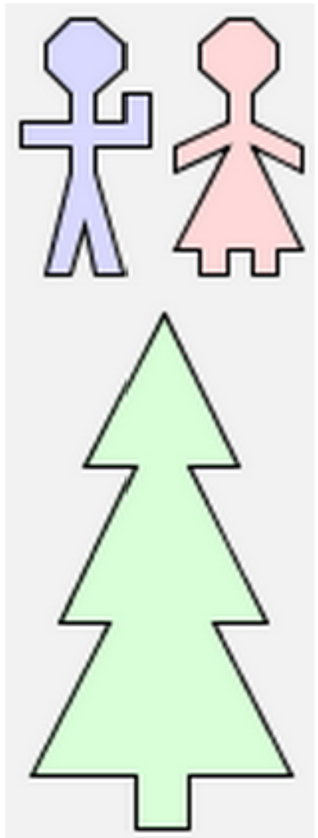


(man $ woman) $ tree

man $ (woman $ tree)

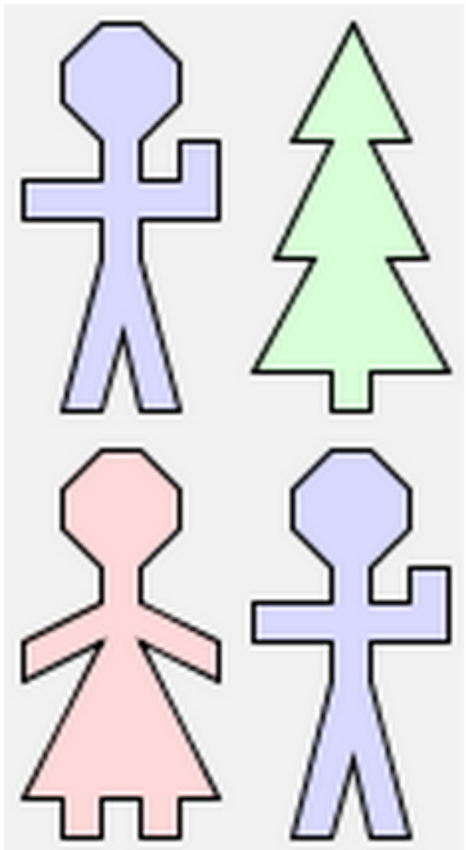


man $ (woman & tree)



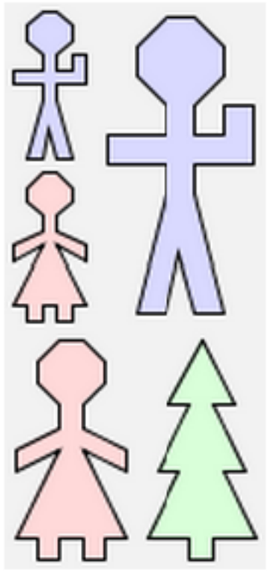
(man $ woman) & tree

man $ woman & tree

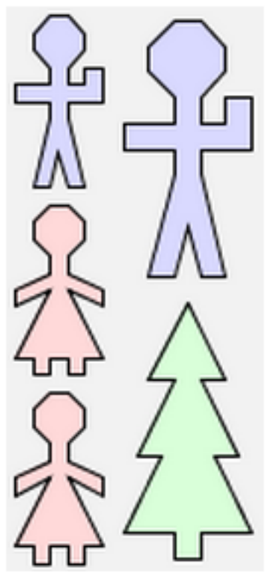


(man $ tree) & (woman $ man)

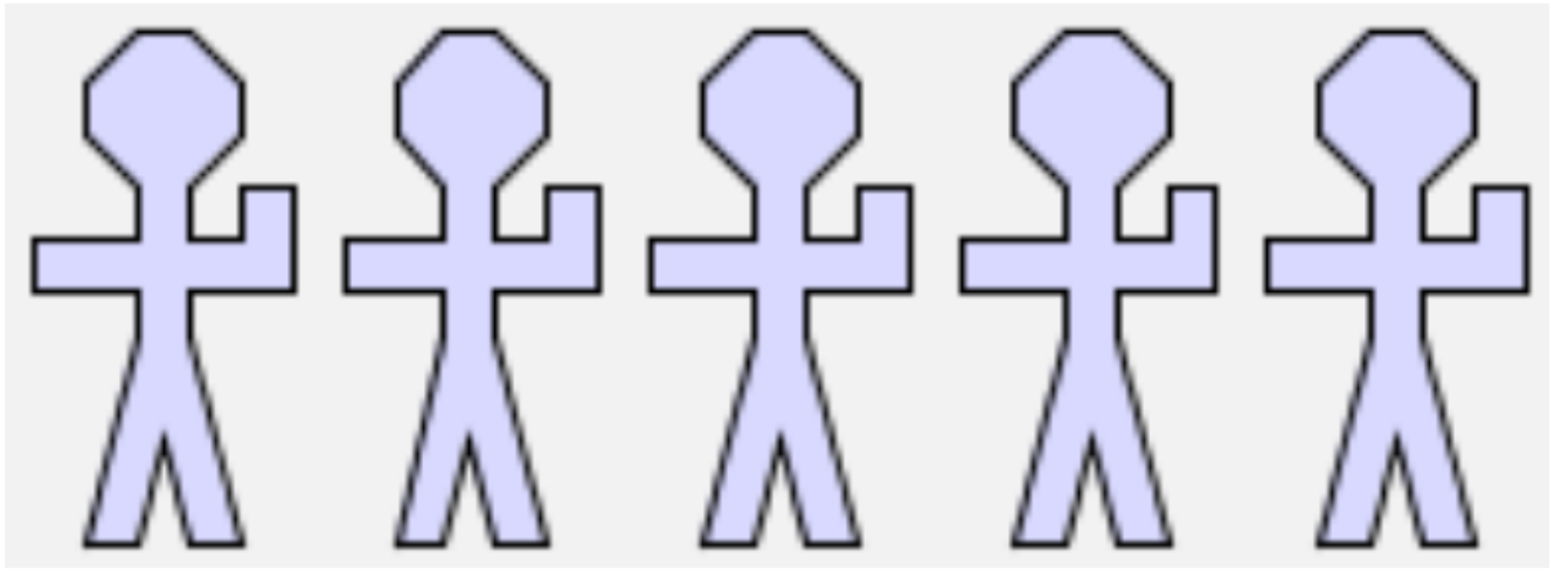
(man & woman) $ (tree & man)



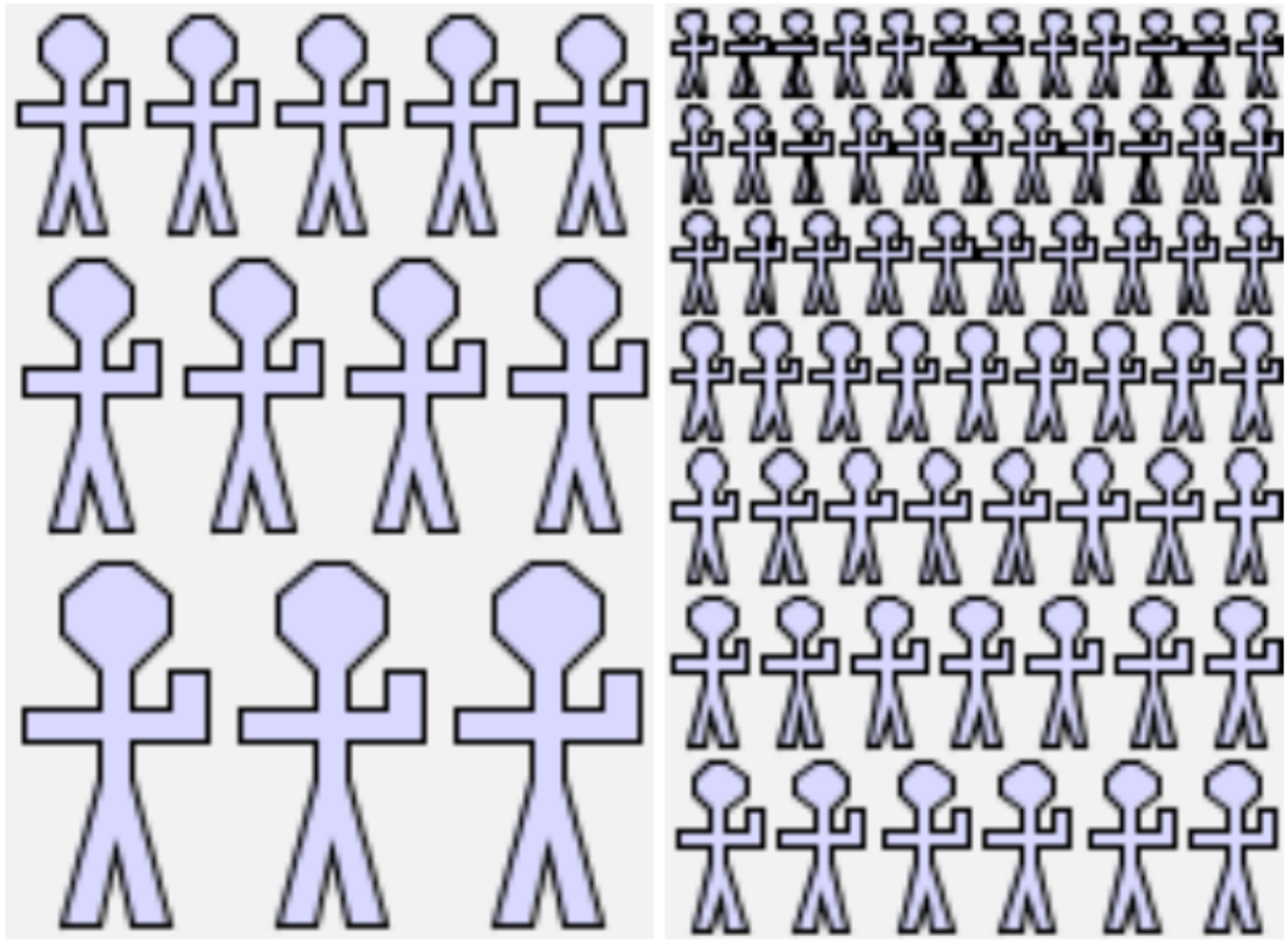
((man & woman) $ man) & (woman $ tree)

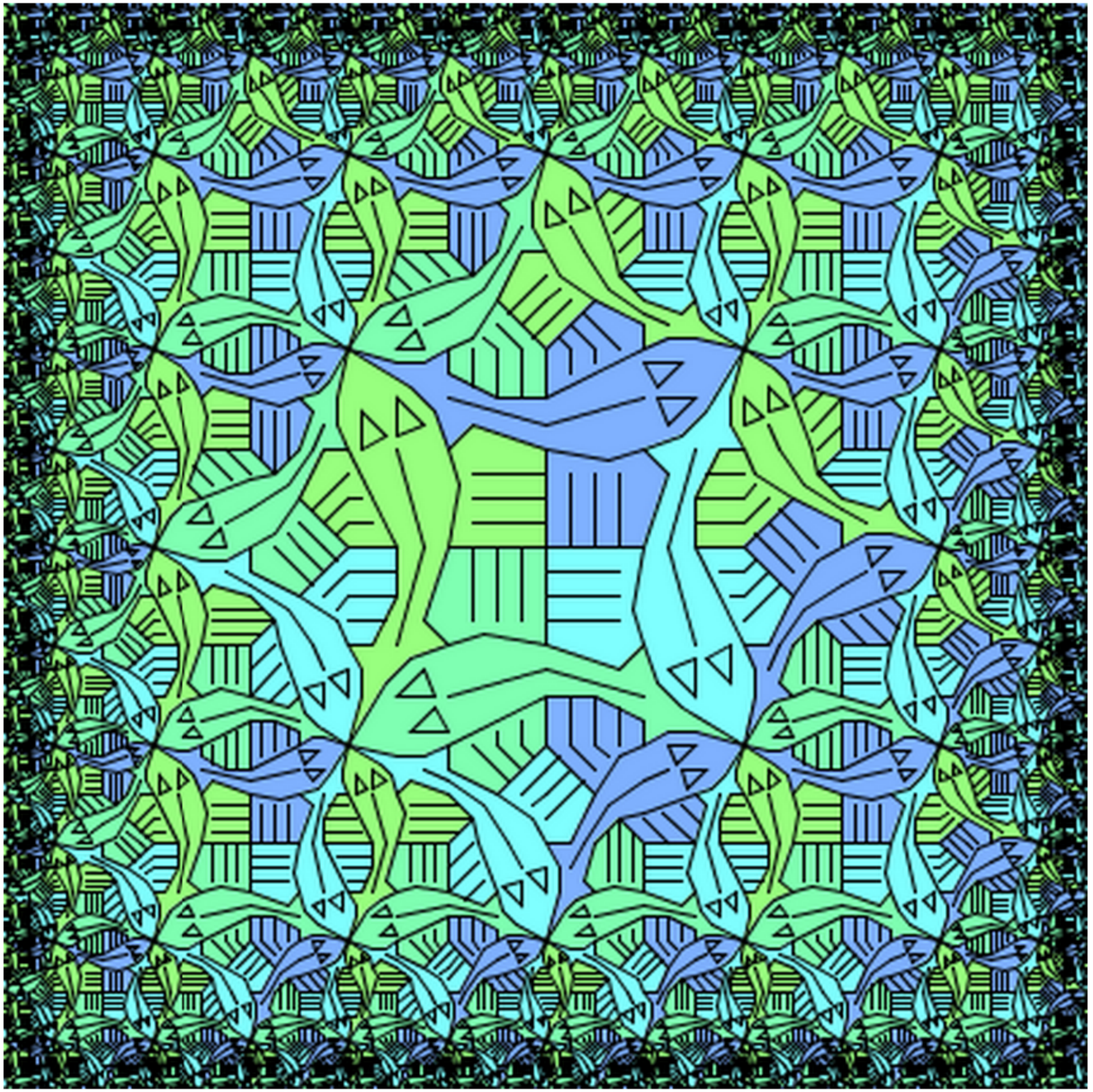


((man & woman) & woman) $ (man & tree)



```
define manrow(n) = manrow(n-1) $ man when n > 1
                 | manrow(1) = man
```





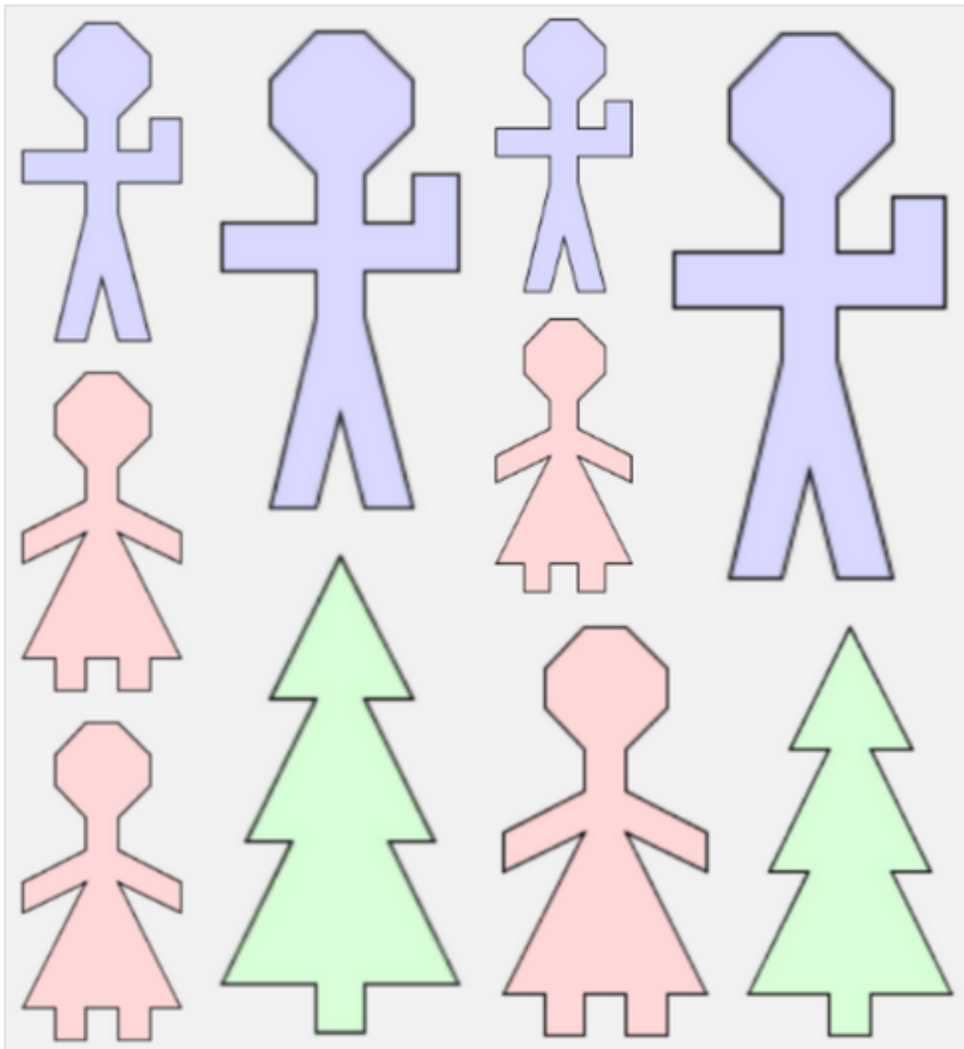
Where do we go now?

What if we did it in Clojure(script)?

```
(below man
       woman)

(beside man (below tree
                   star))
```

cljs html css

```clojure
(ns thattommyhall.geomlab.demo
  (:require [thattommyhall.geomlab.core
             :refer [man woman tree below beside draw]]))

(defn left [] (below (beside (below man
                                    woman)
                             man)
                     (beside woman tree)))

(defn right [] (beside (below (below man woman)
                              woman)
                       (below man tree)))

(defn manstack [n]
  (if (= n 0)
    man
    (below man (manstack (- n 1)))))

(defn manrow [n]
  (if (= n 0)
    man
    (beside man (manrow (- n 1)))))

(defn init []
  (draw (beside (right) (left))))
```



```clojure
(ns thattommyhall.geomlab.demo
  (:require [thattommyhall.geomlab.core
             :refer [man woman tree below beside draw]]))

(defn left [] (below (beside (below man
                                    woman)
                             man)
                     (beside woman tree)))

(defn right [] (beside (below (below man woman)
                              woman)
                       (below man tree)))

(defn init []
  (draw (beside (right) (left))))
```

What's easier to learn?

```
define manrow(n) = manrow(n-1) $ man when n > 1
  | manrow(1) = man
```

OR

```
(defn manrow [n]
  (if (= n 0)
    man
    (beside man (manrow (- n 1)))))
```

Extension?

cljs html css

```clojure
(ns thattommyhall.geomlab.bruce
  (:require [thattommyhall.geomlab.core
             :refer [below beside draw hflip]]))

(defn image-from-url [url]
  (let [image (js/Image.)]
    (set! (.-src image) url)
    image))

(def bruce1 (image-from-url "http://www.thattommyhall.com/bruce.jpg"))
(def bruce2 (image-from-url "http://cdn.oreillystatic.com/en/assets/1/ev

(defn img-stack [img n]
  (if (= n 0)
    img
    (below img (img-stack img (- n 1)))))

(defn imgfinity [img depth init]
  (if (= depth 1)
      (img-stack img (dec init))
      (beside (img-stack img (dec init))
              (imgfinity img (dec depth)
                         (* 2 init)))))

(defn brucefinity []
  (beside (imgfinity bruce1 5 3)
          (hflip (imgfinity bruce2 5 3))))
```

STAND BACK

I'M GOING TO TRY

SCIENCE

SANTA FE INSTITUTE

```
to wiggle
  rt random 40
  lt random 40
  if not can-move? 1 [ rt 180 ]
end

to go
  ask turtles
  [ wiggle
    fd 1 ]
  tick
end
```

```
(defn move [t]
  (fd t 1)
  (rt t (rand 60))
  (lt t (rand 60)))

(go
 (loop []
   (<! (timeout 10))
   (doseq [t turtles]
     (move t))
   (recur)))
```

Conclusion

- You probably dont need to make a new language
- If you do it will probably be rubbish
- - At least for a while
- Think about its power and reach

You should embed *deeply* into clojure

- http://www.twitter.com/otfrom
- http://cljsfiddle.net/fiddle/thattommyhall.geomlab.core
- http://cljsfiddle.net/fiddle/thattommyhall.geomlab.demo

- http://cljsfiddle.net/fiddle/thattommyhall.geomlab.bruce

- http://www.complexityexplorer.org/

- http://cljsfiddle.net/fiddle/thattommyhall.ants.core
- http://ccl.northwestern.edu/tortoise/2013-10-25/Ants.html