# Spirographs

@am_i_tom

# PureScript Spirographs

# Who am I?

- Tom Harding
- Habito (always hiring!)
- twitter.com/am_i_tom
- github.com/i-am-tom
- tomharding.me

# Spirographs

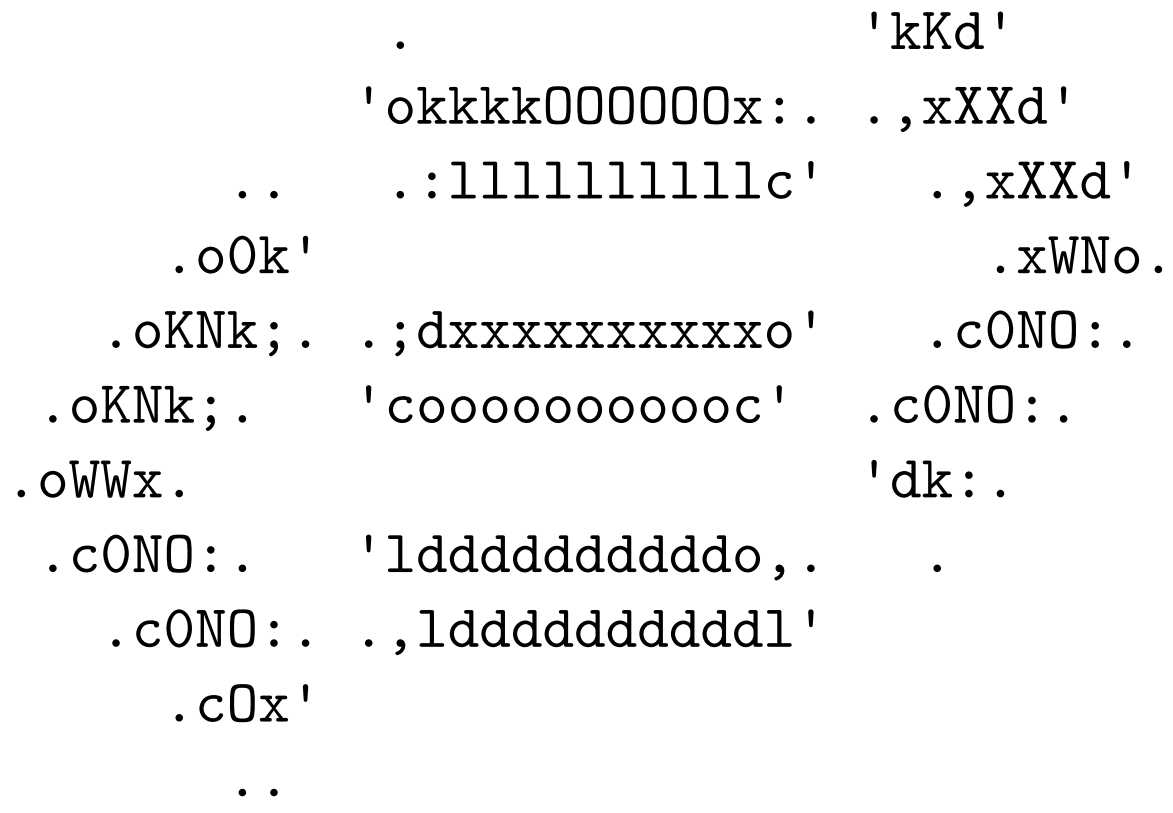
# How does it work?

- Start with a (fixed) circle (as a perimeter).
- Pick a point on a smaller, rolling circle.
- Roll the second circle around the edge of the first.
- Trace the path of the chosen point.
- Repeat until Mum's off the phone.

# Risky Live Moment 1: Spirograph GIF

# PureScript

# In ASCII art?


.
'okkkkOOOOOOx:.
.:lllllllllc'
.oOk'
.oKNk;.
.;dxxxxxxxxxxo'
.oKNk;.
'cooooooooooc'
.oWWx.
.cONO:.
'ldddddddddddo,.
.cONO:.
.,ldddddddddddl'
.cOx'
..
'kKd'
.,xXXd'
.,xXXd'
.xWNo.
.cONO:.
.cONO:.
'dk:.
.

# In code?

```
module Main where

import Prelude
import Effect (Effect)
import Effect.Console (log)

main  Effect Unit
main = do
  log "Hello sailor!"
```

# Maths

# Let's write a type!

```
newtype Coordinate
  = Coordinate
      { x  Number
      , y  Number
      }
```

# What can we do with it?

```
derive instance eqCoordinate
   Eq Coordinate

derive newtype instance semiringCoordinate
   Semiring Coordinate

derive newtype instance ringCoordinate
   Ring Coordinate

derive newtype instance showCoordinate
   Show Coordinate
```

# … How do I write my *own* code?

```
rotate
    Number -> Coordinate
  -> Coordinate

rotate angle (Coordinate { x, y })
  = Coordinate
      { x: cos angle * x - sin angle * y
      , y: sin angle * x + cos angle * y
      }
```

# Where's the rolling circle?

```
rollingCirclePosition
    Number -> Number
  -> Coordinate

rollingCirclePosition sizeRatio time
  = rotate time initial
  where
    initial = Coordinate
      { x: 0.0
      , y: 1.0 - sizeRatio
      }
```

# Which way up is it?

```
rollingCircleRotation
    Number -> Number
  -> Number

rollingCircleRotation sizeRatio time
  = -time / sizeRatio
```

# Ok, but where's the pen?

```
penOffset
    Number -> Number -> Number
  -> Coordinate

penOffset sizeRatio offsetRatio rotation
  = rotate rotation
  $ Coordinate
      { x: 0.0
      , y: sizeRatio * offsetRatio
      }
```

# Drawing

# How do we draw a point on the canvas?

```
mark
    Configuration -> Seconds
  -> Drawing

mark { sizeRatio, offsetRatio } (Seconds time)
  = filled (fillColor colour)
  $ circle x y 2.0
```

# Where did `x` and `y` come from?

```
  where
    rollingCentre
      = rollingCirclePosition sizeRatio time

    angle
      = rollingCircleRotation sizeRatio time

    Coordinate { x, y }
      = centreForCanvas
      $ rollingCentre
      + penPosition sizeRatio offsetRatio angle
```

# … and colour?

```
colour
  = hsv (time * 180.0 % 360.0) 0.8 0.8
```

# Business logic

# Dealing with the “real world”

```
canvas <- lift (getCanvasElementById "spirograph")
  >>= case _ of
    Just canvas -> pure canvas
    Nothing     -> throwError "No canvas :("

lift $ setCanvasDimensions canvas
  { width: 400.0, height: 400.0 }

context <- lift (getContext2D canvas)
```

# Risky Live Moment 2: The finished product

# Draw me like one of your French curls

```
-- Current time as a stream   vvvvvvv
stopDrawing <- lift $ animate seconds \time -> do
  let config = { sizeRatio, offsetRatio }
  render context (mark config time)
```

# A little more maths?

```
let crossover
      = toNumber
      $ numerator
      $ simplify sizeRatioAsFraction

    completion
      = 2000.0 * pi * crossover

void
  $ lift
  $ setTimeout (ceil completion) stopDrawing
```

# Risky live moment 3: The even finisheder product

Could we have three dimensions

Yes

# Polymorphic accessors

```
getX
   forall wrapper output anythingElse
   . Newtype wrapper { x  output | anythingElse }
  => wrapper
  -> output

getX
  = _.x <<< unwrap
```

# Polymorphic coordinate operations

```
class Coordinate (object  Type) where
  transform
     (Number -> Number)
    -> (object -> object)

  fold
     forall m. Monoid m
    => (Number -> m)
    -> (object -> m)
```

# Type-trickery

```
class GCoordinate
    (row    # Type)
    (list  RowList) where
  transform'
     RLProxy list -> (Number -> Number)
    -> (Record row -> Record row)

  fold'
     forall m. Monoid m
    => RLProxy list -> (Number -> m)
    -> (Record row -> m)
```

# The boring case

```
instance gcoordinateNil
      GCoordinate row Nil where
  transform' _ _ = identity
  fold' _ _ _ = mempty
```

# The interesting case

```
instance gcoordinateCons
      ( GCoordinate row tail, IsSymbol key
        , Row.Cons key Number xyz row )
    => GCoordinate row (Cons key Number tail) where
  transform' _ f record
    = modify (SProxy  SProxy key) f
    $ transform' (RLProxy  RLProxy tail) f record

  fold' _ f record
     = f (get (SProxy  SProxy key) record)
    <> fold' (RLProxy  RLProxy tail) f record
```

# Finally...

```
instance coordinateImpl
      ( RowToList row list
        , GCoordinate row list )
     => Coordinate (Record row) where
  transform = transform' (RLProxy  RLProxy list)
  fold      = fold'      (RLProxy  RLProxy list)
```

# All this for what?

```
offset
   forall row. Coordinate row
  => row -> Number

offset record
  = sqrt total
  where
    folder x = Additive (x `pow` 2.0)
    Additive total = fold folder record
```

# Success

# ... Well, not quite

- Floating point precision!
- `(a * b ) % b === 0`
- `(a * 2 ) % 2 === 0`
- `(a * 3 ) % 3 === 0`
- `(a * pi ) % pi === 0`
- `(a * (pi / 4)) % (pi / 4) === 1.2566`
- *show excuse.png*
- But, with a better number type, *yes*!

“It is left as an exercise to the reader”

# Other exercises to the reader

- Stateful animation with `FRP.Behavior.fixB`.
- *purescript-super-circles*
- Continuous lines (*better laptops*).
- Interactive controls.
- Other types of ellipses to roll.

# Summary

- Present animations on a *newer laptop*.
- Simple canvas drawing with `purescript-drawing`.
- Simple animation with `purescript-behaviors`.
- More examples with `purescript-super-circles`.
- There was life before *the Internet*.

Thank you!

# Questions?

- Tom Harding

# Questions?

- Tom Harding
- Habito (always hiring!)

# Questions?

- Tom Harding
- Habito (always hiring!)
- twitter.com/am_i_tom

# Questions?

- Tom Harding
- Habito (always hiring!)
- twitter.com/am_i_tom
- github.com/i-am-tom

# Questions?

- Tom Harding
- Habito (always hiring!)
- twitter.com/am_i_tom
- github.com/i-am-tom
- tomharding.me

# Questions?

- Tom Harding
- Habito (always hiring!)
- twitter.com/am_i_tom
- github.com/i-am-tom
- tomharding.me
- *github.com/jaspervdj/patat*