module Main where

import Crossing
import Control.Monad (replicateM, forM_)
import System.Environment (getArgs)
import System.FilePath (splitExtension)

1. Examples

Here is a simple collar:

diagonals = (\sqrt{2}, \sqrt{2})
(\sqrt{2}, -\sqrt{2})
collarUnit :: M t ()
collarUnit = replicateM 11
  $ do crossing 2 \times (-5, -5) diagonals
       crossing 1 \times (0, 0) diagonals
       crossing 0 \times (5, 5) diagonals
       advance (10, 0)
collarControls :: M Paint ()
collarControls = begin 0 (red Open) \gg collarUnit \gg
  replicateM 3 (begin 0 (paint Open) \gg collarUnit)
collar :: OC \rightarrow [Maybe (Thread Paint)]
collar Open = weave collarControls 4
collar Closed = weave (begin 0 (paint Closed) \gg circular collarUnit) 4

More generally, here is a Turk’s Head. The two parameters are \(k\), the number of strands, and \(n\), the number of crowns in a loop around the collar. If \(k\) and \(n\) are not relatively prime, this code doesn’t currently work.

turksHead :: Int \rightarrow Int \rightarrow OC \rightarrow [Maybe (Thread Paint)]
turksHead k n oc =
  let spacing = 10
      stepping = 5
      unit :: M t ()
      unit = replicateM n
        $ forM_ [k - 2, k - 3 .. 0] (\lambda i.
            let x = spacing/2 - stepping \times fromIntegral i
                y = stepping \times (fromIntegral (k - 2)/2 - fromIntegral i)
                in crossing i (if even i then \notin \ else \in) (x, y) diagonals
              \gg advance (spacing, 0)
controls :: \( M \_ \_ \text{Paint} () \)

\[
\begin{align*}
\text{controls} &= \begin{begin} \text{begin 0 \text{ (red Open) >> unit >>}} \\
& \quad \text{replicateM} \_ \_ (k - 1) \begin{begin} \text{ (begin 0 \text{ (paint Open) >> unit })} \end{begin} \\
\end{begin} \\
\text{in case oc of} \quad & \text{Open \rightarrow weave controls k} \\
& \quad \text{Closed \rightarrow weave \begin{begin} \text{ (begin 0 \text{ (paint Closed) >> circular unit) k} \end{begin} } \\
\end{align*}
\]

The fishbone braid. Quite similar, except for the signs of the crossings. This braid has \(2k + 1\) total strands.

\[
\text{fishbone} :: \text{Int} \rightarrow \text{Int} \rightarrow \text{OC} \rightarrow \text{Maybe (Thread \_ \_ \text{Paint})}
\]

\[
\text{fishbone \ } kk1 \ n \ oc \mid \text{kk1 > 2 \& odd kk1} =
\]

\[
\begin{align*}
\text{let} \quad & \text{k} = \text{kk1}' \text{div} 2 \\
\text{spacing} &= 10 \\
\text{stepping} &= 5 \\
\text{unit} :: & \text{M \_ \_ t()} \\
\text{unit} &= \text{replicateM} \_ \_ n \\
\& \text{forM} \_ \_ [2 \times k - 1, 2 \times k - 2 \ldots 0] \ (\lambda i. \\
\text{let} \quad & x = \text{spacing}/2 - \text{stepping \times fromIntegral i} \\
& y = \text{stepping \times (fromIntegral k - 1/2 - fromIntegral i)} \\
\text{in} \quad & \text{crossing i (if i < k then \( \times \) \text{else} \( \checkmark \)) (x, y) diagonals} \\
& \quad \Rightarrow \text{advance (spacing, 0)} \\
\text{controls ::} & \text{M \_ \_ \text{Paint} ()} \\
\text{controls} &= \begin{begin} \text{begin 0 \text{ (red Open) >> unit >>}} \\
& \quad \text{replicateM} \_ \_ (2 \times k) \begin{begin} \text{ (begin 0 \text{ (paint Open) >> unit })} \end{begin} \\
\end{begin} \\
\text{in case oc of} \quad & \text{Open \rightarrow weave controls kk1} \\
& \quad \text{Closed \rightarrow weave \begin{begin} \text{ (begin 0 \text{ (paint Closed) >> circular unit) kk1} \end{begin} } \\
\end{align*}
\]

Celtic knot 1, another braid, with 4 strands. This one uses two pieces of rope.

\[
\text{celtic1} :: \text{Int} \rightarrow \text{OC} \rightarrow \text{Maybe (Thread \_ \_ \text{Paint})}
\]

\[
\text{celtic1 \ } n \ oc =
\]

\[
\begin{align*}
\text{let unit ::} & \text{M \_ \_ t()} \\
\text{unit} &= \text{replicateM} \_ \_ n \\
\& \text{do} \text{ crossing 2 \( \checkmark \) (0, -5) diagonals} \\
& \quad \text{crossing 1 \( \times \) (5, 0) diagonals} \\
& \quad \text{crossing 0 \( \checkmark \) (10, 5) diagonals} \\
& \quad \text{crossing 1 \( \times \) (15, 0) diagonals} \\
& \quad \text{advance (20, 0)} \\
\text{controls ::} & \text{M \_ \_ \text{Paint} ()} \\
\text{controls} &= \begin{begin} \text{begin 0 \text{ (orange Open) >> unit >>}} \\
& \quad \text{replicateM} \_ \_ 2 \begin{begin} \text{ (begin 0 \text{ (paint Open) >> unit })} \end{begin} \\
& \quad \text{end 0 \text{ (orange Open) >>}} \\
& \quad \text{begin 0 \text{ (paint Open) >> begin 1 \text{ (red Open) >> unit}} \\
\end{begin}
\end{align*}
\]
in case oc of Open  → weave controls 4
Closed → weave (begin 0 (paint Closed) ⇒
begin 1 (red Closed) ⇒ circular unit) 4

A wider version, with 6 strands.

\[
\text{celtic2} :: \text{Int} → \text{OC} → [\text{Maybe (Thread_Paint)}]
\]
\[
\text{celtic2} \ n \ \text{oc} = \\
\begin{align*}
&\text{let} \ \text{unit} :: \text{M}_t () \\
&\ \text{unit} = \text{replicateM}_n \\
&\ \text{do} \ \text{crossing} 4 \bigodot (-10, -10) \ \text{diagonals} \\
&\ \ \text{crossing} 3 \bigodot (-5, -5) \ \text{diagonals} \\
&\ \ \text{crossing} 2 \bigodot (0, 0) \ \text{diagonals} \\
&\ \ \text{crossing} 1 \bigodot (5, 5) \ \text{diagonals} \\
&\ \ \text{crossing} 0 \bigodot (10, 10) \ \text{diagonals} \\
&\ \ \text{crossing} 3 \bigodot (5, -5) \ \text{diagonals} \\
&\ \ \text{crossing} 1 \bigodot (15, 5) \ \text{diagonals} \\
&\ \text{advance} (20, 0) \\
&\ \text{controls} :: \text{M}_t \text{Paint} () \\
&\ \text{controls} = \text{begin} 0 \ (\text{orange Open}) \ ⇒ \ \text{unit} \ ⇒ \\
&\ \ \ \text{replicateM}_3 \ (\text{begin} 0 \ (\text{paint Open}) \ ⇒ \ \text{unit}) \ ⇒ \\
&\ \ \ \text{end} 0 \ (\text{orange Open}) \ ⇒ \\
&\ \ \ \text{begin} 0 \ (\text{paint Open}) \ ⇒ \ \text{begin} 1 \ (\text{red Open}) \ ⇒ \ \text{unit} \ ⇒ \\
&\ \ \ \text{end} 1 \ (\text{red Open}) \ ⇒ \\
&\ \ \ \text{begin} 1 \ (\text{paint Open}) \ ⇒ \ \text{begin} 3 \ (\text{red Open}) \ ⇒ \ \text{unit} \\
\end{align*}
\]

in case oc of Open  → weave controls 6
Closed → weave (begin 0 (paint Closed) ⇒
begin 1 (red Closed) ⇒
begin 3 (red Closed) ⇒ circular unit) 6

\[
\text{celtic3} :: \text{Int} → \text{OC} → [\text{Maybe (Thread_Paint)}]
\]
\[
\text{celtic3} \ n \ \text{oc} = \\
\begin{align*}
&\text{let} \ \text{unit} :: \text{M}_t () \\
&\ \text{unit} = \text{replicateM}_n \\
&\ \text{do} \ \text{crossing} 4 \bigodot (-10, -10) \ \text{diagonals} \\
&\ \ \text{crossing} 3 \bigodot (-5, -5) \ \text{diagonals} \\
&\ \ \text{crossing} 2 \bigodot (0, 0) \ \text{diagonals} \\
&\ \ \text{crossing} 1 \bigodot (5, 5) \ \text{diagonals} \\
&\ \ \text{crossing} 0 \bigodot (10, 10) \ \text{diagonals} \\
&\ \ \text{crossing} 3 \bigodot (5, -5) \ \text{diagonals} \\
\end{align*}
\]
crossing $2 \times (10,0)$ diagonals
crossing $1 \times (15,5)$ diagonals
advance $(20,0)$

controls :: $M_\perp \text{Paint } ()$
controls = begin 0 (orange Open) $\gg$ unit $\gg$
replicate$M_\perp$ 3 (begin 0 (paint Open) $\gg$ unit) $\gg$
end 0 (orange Open) $\gg$
begin 0 (paint Open) $\gg$ begin 1 (red Open) $\gg$ unit $\gg$
begin 1 (paint Open) $\gg$ unit

in case oc of Open $\rightarrow$ weave controls 6
Closed $\rightarrow$ weave (begin 0 (paint Closed) $\gg$
begin 1 (red Closed) $\gg$ circular unit) 6

in case oc of Open $\rightarrow$ weave controls 6
Closed $\rightarrow$ weave (begin 0 (paint Closed) $\gg$
begin 1 (red Closed) $\gg$ circular unit) 6

2. Main program

This program takes any number of file names on the command line, each ending in .svg or .tex. It figures out which SVG or TikZ files to produce by parsing the file name.

main :: IO ()
main = getArgs $\Rightarrow$ map$M_\perp$ ($\lambda$fp.writeFile fp $\$! contents fp)

contents :: FilePath $\rightarrow$ String
contents fp = case splitExtension fp of
  (fp,".svg") $\rightarrow$ svg (contentThreads fp)
  (fp,".tex") $\rightarrow$ tikz (contentThreads fp)
  _ $\rightarrow$ error ("Don’t know how to make " $\oplus$ fp)
data Contents = Collar OC
| Turks Int Int OC
| Fish Int Int OC
| Celtic1 Int OC
| Celtic2 Int OC
| Celtic3 Int OC
| DragonTamer Int OC deriving Read

contentThreads :: String → [Maybe (Thread, Paint)]
contentThreads s = case read (map sp s) of
  Collar oc → collar oc
  Turks k n oc → turksHead k n oc
  Fish kk1 n oc → fishbone kk1 n oc
  Celtic1 n oc → celtic1 n oc
  Celtic2 n oc → celtic2 n oc
  Celtic3 n oc → celtic3 n oc
  DragonTamer n oc → dragonTamer n oc

where sp ' ' = ',
      sp c   = c