Code (Wednesday Week 2)

Using the same `ShapeGraphics` library found in this week's exercise.

Haskell

```haskell
module FractalTrees where

-- needed to display the picture in the playground
import Codec.Picture
-- our graphics programming interface
import ShapeGraphics

fracTree :: Picture
fracTree
  = fTree (Point  (400 - width / 2) 800) (Vector 0 (-height))
                  (Vector width 0) red depth
  where
    depth = 10
    height = 100
    width = 15
    angle = pi/8

    -- Nudge a colour to be less red and more blue.
    toBlue :: Colour -> Colour
    toBlue (Colour r g b o) =
      Colour (max 0 (r - 15)) g (min 255 (b + 15)) o

    fTree :: Point -> Vector -> Vector -> Colour -> Int -> Picture
    fTree pos vec1 vec2 col n
      | n == 0 = []
      | otherwise =
          [Polygon [pos, movePoint vec1 pos,
                      movePoint vec2 $ movePoint vec1 pos,
                      movePoint vec2 pos]
                col
                Solid
                SolidFill] ++
          fTree (movePoint vec1 pos)
                (scaleVector 0.8 $ rotateVector (0.5 * angle) vec1)
                (scaleVector 0.8 $ rotateVector (0.5 * angle) vec2)
                (toBlue col) (n - 1) ++
          fTree (movePoint vec1 pos)
                (scaleVector 0.8 $ rotateVector (-angle) vec1)
                (scaleVector 0.8 $ rotateVector (-angle) vec2)
                (toBlue col) (n - 1)
```

```
scaleVector :: Float -> Vector -> Vector
scaleVector fac (Vector x y)
  = Vector (fac * x) (fac * y)

rotateVector :: Float -> Vector -> Vector
rotateVector alpha (Vector vx vy)
  = Vector (vx * cos alpha - vy * sin alpha )
           (vx * sin alpha  + vy * cos alpha)

movePoint :: Vector -> Point -> Point
movePoint (Vector xv yv) (Point xp yp)
  = Point (xv + xp) (yv + yp)

writeToFile pic
  = writePng "output.png" (drawPicture 3 pic)
```