Introducción a la programación gráfica con CodeWorld

Dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla

¿Qué es CodeWorld?

Entorno de programación gráfica basado en Haskell

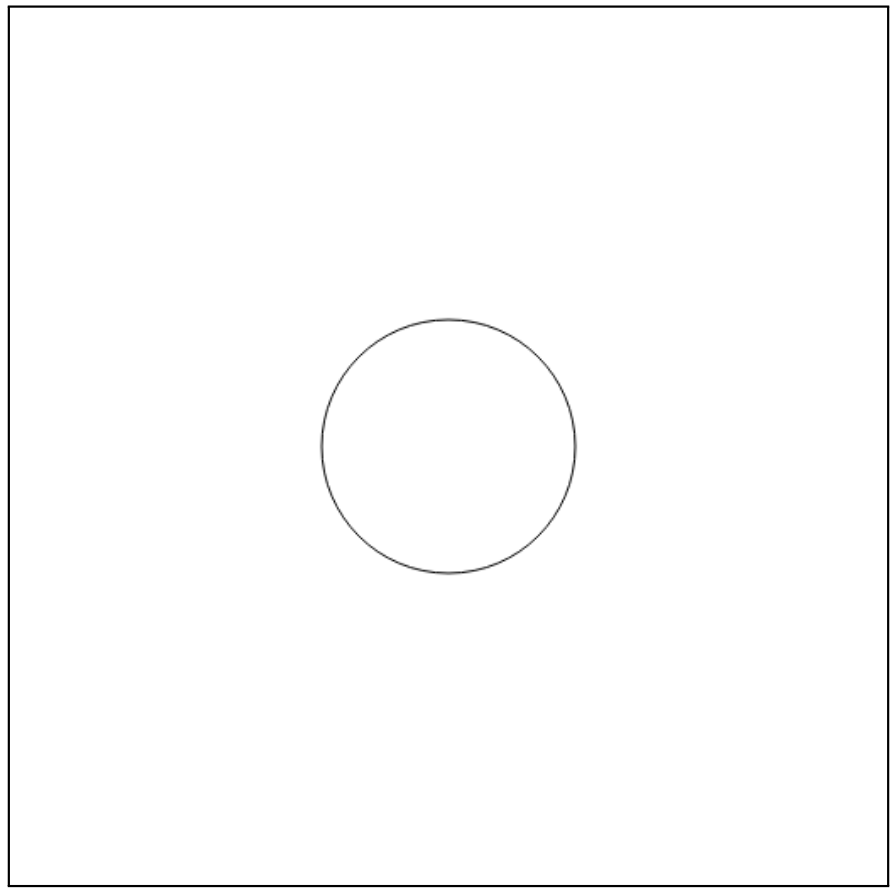
https://code.world/haskell

Prueba el siguiente código:

```
import CodeWorld

main :: IO ()
main = drawingOf miDibujo

miDibujo :: Picture
miDibujo = circle 3
```

```
drawingOf ::  Picture -> IO ()
```

Círculo de radio 3



Tipos y funciones gráficas

- Los dibujos son de tipo `Picture`
- Existe una colección de funciones para producir figuras elementales: rectángulos, polígonos, líneas, arcos, texto...

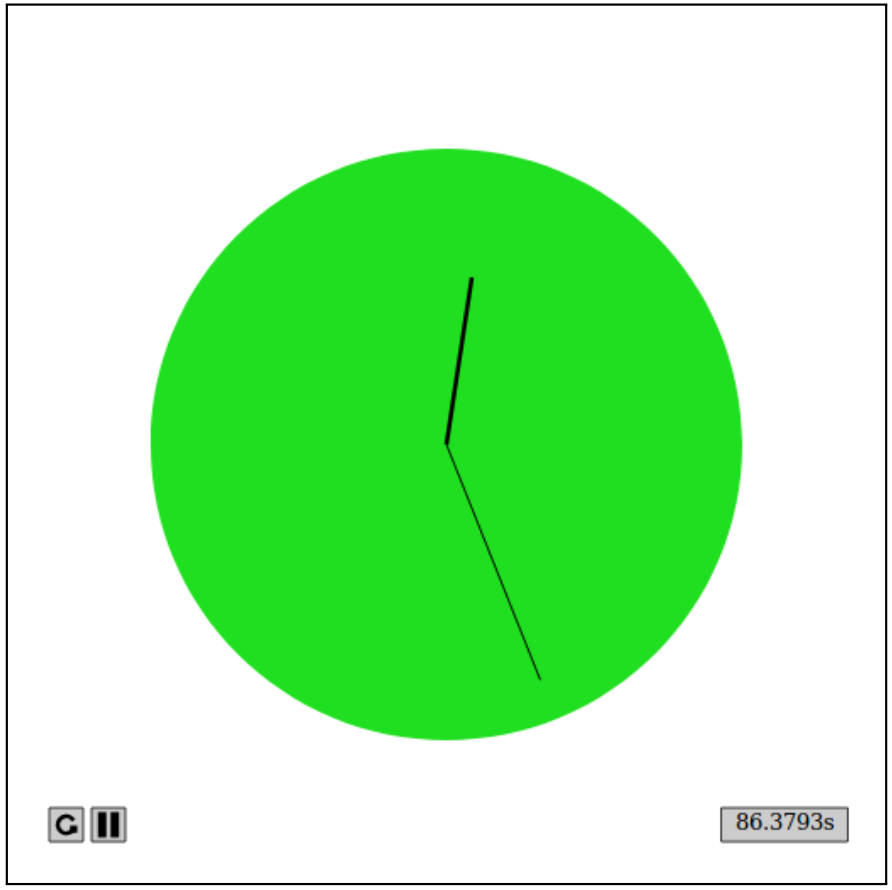
```
circle :: Double -> Picture
```

- Existen funciones que modifican los dibujos, por ejemplo:
  - Color: `colored :: Color -> Picture -> Picture`
  - Posición: `translated :: Double -> Double -> Picture -> Picture`
- Los dibujos se pueden combinar para dar lugar a dibujo más elaborados.
  - Combinar dos dibujos: `(&) :: Picture -> Picture -> Picture`
  - Dibujo vacío: `Blank`
  - Concatenación de lista de dibujos: `pictures :: [Picture] -> Picture`

Un poco más elaborado

```
import CodeWorld
import Data.Text (pack)

main :: IO ()
main = animationOf cronometro  -- una animación (no interactivo)

cronometro :: Double -> Picture
cronometro t = minutero t & segundero t & fondo & tiempo t

fondo :: Picture
fondo = colored green (solidCircle 7)

segundero :: Double -> Picture
segundero t = rotated (-(2*pi/3600)*t) (thickPolyline 0.1 [(0,0), (0,4)])

minutero :: Double -> Picture
minutero t = rotated (-(2*pi/60)*t) (polyline [(0,0), (0,6)])

tiempo :: Double -> Picture
tiempo t = translated (4.5) (-7) (scaled 0.7 0.7 (texto (t)))

texto t = lettering (pack (take 7 ("t="++(show t))))
```

Cronómetro



Ejemplos en 1º del Grado de Matemáticas-Física

- Tema introductorio
- Ejemplos desarrollado por alumnos

Y ahora interactivo...

```
{-# LANGUAGE OverloadedStrings #-} -- pragma necesario para KeyPress
import CodeWorld

main :: IO()
main = activityOf inicio manejaEvento pintaEstado

type Estado = (Double, Double)

inicio :: Estado
inicio = (0, 0)

manejaEvento :: Event -> Estado -> Estado
manejaEvento (KeyPress "Right") (x, y) = (x+1, y)
manejaEvento (KeyPress "Left") (x, y)  = (x-1, y)
manejaEvento (KeyPress "Up") (x, y)    = (x, y+1)
manejaEvento (KeyPress "Down") (x, y)  = (x, y-1)
manejaEvento _ (x, y)                  = (x, y)

pintaEstado :: Estado -> Picture
pintaEstado (x, y) = translated x y (colored red (solidRectangle 1 1))
```

```
activityOf ::  a -> (Event -> a -> a) -> (a -> Picture) -> IO()
```

Usando el ratón y el paso del tiempo

```
import CodeWorld

data Ficha = Blanca | Negra | Roja

dibujaFicha :: Ficha -> Picture
dibujaFicha Blanca = circle 5 & coordinatePlane
dibujaFicha Negra = solidCircle 5 & coordinatePlane
dibujaFicha Roja = colored red (solidCircle 5) & coordinatePlane

contraria :: Ficha -> Ficha
contraria Blanca = Negra
contraria Negra = Blanca
contraria _ = undefined

modificaFicha :: Event -> Ficha -> Ficha
modificaFicha _ Roja = Roja
modificaFicha (PointerPress (x, y)) ficha    -- Detecta pulsación del ratón
  | x**2 + y**2 <= 25 = (contraria ficha)
modificaFicha (TimePassing dt) f             -- Detecta paso del timpo (útil para animación)
  | dt > 0.07 = Roja
  | otherwise = f
modificaFicha _ f = f

main :: IO()
main = activityOf Blanca modificaFicha dibujaFicha
```