Informe sobre la librería Cairo en Haskell

Introducción

La librería Cairo es una biblioteca de gráficos 2D de propósito general que proporciona una API de alto nivel para renderizar gráficos vectoriales. En Haskell, la librería Cairo está disponible a través del paquete `Cairo` en el ecosistema de Haskell.

Principales Características

- Renderización de gráficos vectoriales: Cairo proporciona herramientas para crear gráficos vectoriales de alta calidad, lo que permite la escalabilidad sin pérdida de calidad.
- Soporte multiplataforma: Cairo es compatible con varias plataformas, lo que lo hace ideal para aplicaciones que necesitan ejecutarse en diferentes sistemas operativos.
- Diversidad de formatos de salida: La librería soporta varios formatos de archivo de salida, incluyendo PNG, PDF, SVG, entre otros.
- API de alto nivel: En Haskell, la librería Cairo ofrece una API de alto nivel que facilita la creación de gráficos complejos con un código limpio y legible.

Uso Básico

```haskell
import Graphics.Rendering.Cairo

import Graphics.Rendering.Cairo.Types

main :: IO ()

main = do

 putStrLn "Hello, Haskell!"

 withImageSurface FormatARGB32 100 100 $ \surface -> do

   renderWith surface $ do

     setSourceRGB 0 0 0
```

```
    moveTo 0 0

    lineTo 100 100

    stroke

  surfaceWriteToPNG surface "out.png"

putStrLn "Goodbye, Haskell!"
```

Explicacion del codigo anterior:

1. `withImageSurface FormatARGB32 100 100 $ \surface ->`:
    - Crea una superficie de imagen en formato ARGB32 de 100x100 píxeles y la utiliza dentro del bloque `do`.
    - `FormatARGB32` indica que la imagen tendrá un canal alfa para la transparencia.
2. `renderWith surface $ do`:
    - Realiza las operaciones de dibujo en la superficie de la imagen creada.
3. `setSourceRGB 0 0 0`:
    - Establece el color de dibujo a negro (RGB 0,0,0).
4. `moveTo 0 0`:
    - Mueve el punto de dibujo a la esquina superior izquierda (0,0) de la superficie.
5. `lineTo 100 100`:
    - Dibuja una línea recta desde el punto actual (0,0) hasta la esquina inferior derecha (100,100).
6. `stroke`:
    - Dibuja el contorno de la línea, haciéndola visible.
7. `surfaceWriteToPNG surface "out.png"`:
    - Guarda la superficie de la imagen como un archivo PNG llamado "out.png".