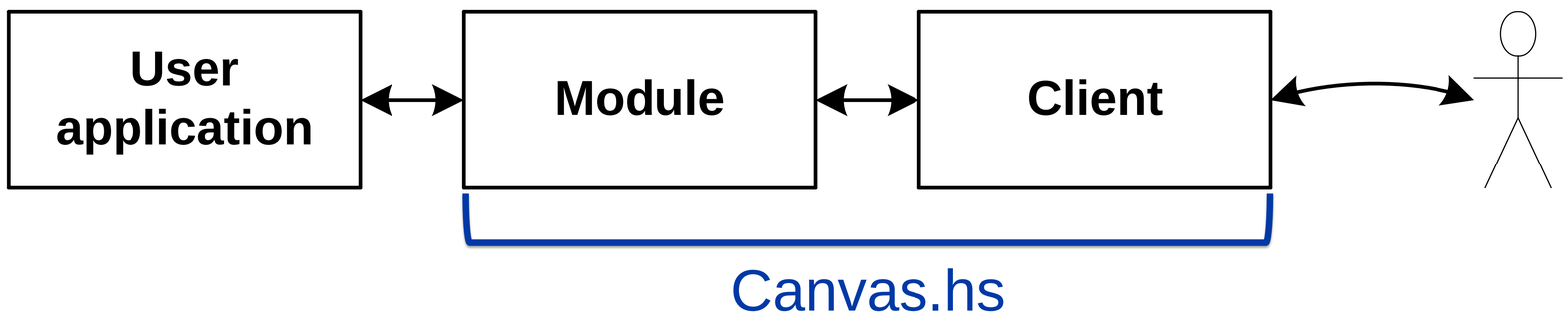
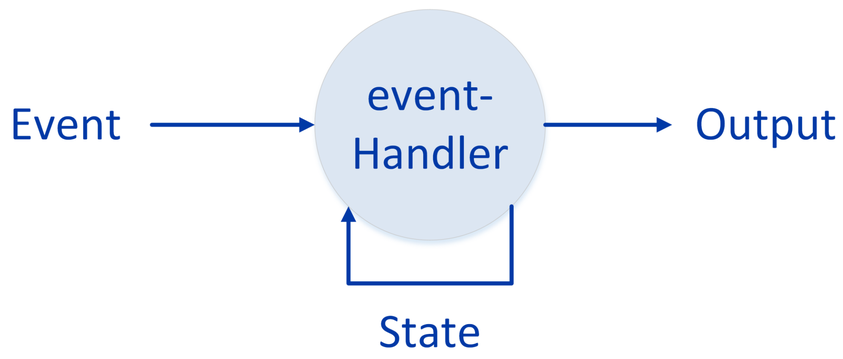
Canvas.hs

Biedt een grafische interface voor Haskell applicaties door middel van event driven I/O met behulp van een browser en HTML5 canvas technologie. Op de achtergrond draait Canvas.hs een lichtgewicht HTTP- en WebSocketserver om de verbinding te onderhouden. Canvas.hs gebruikt KineticJS en jQuery om op het canvas te tekenen en events te onderscheppen.



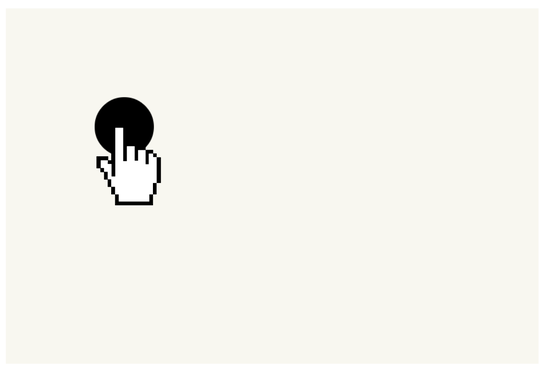
Simpel te gebruiken grafische omgeving voor Haskell zonder gebruik te maken van monadisch programmeren. Dit doet de gebruiker door middel van een eventHandler.

```
eventHandler :: a -> Event -> (a, Output)
```

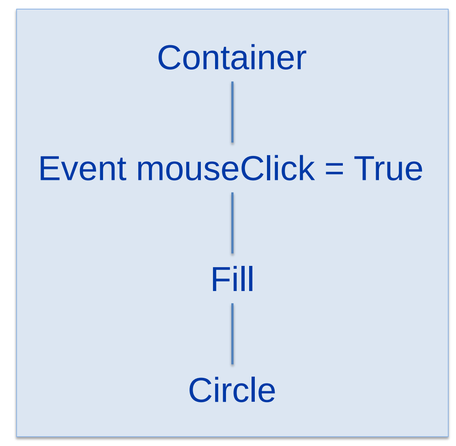


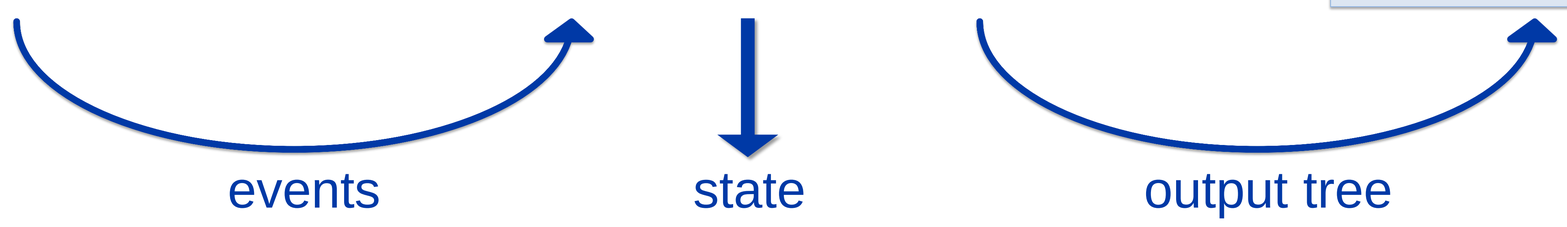
Event driven I/O

De programmeur hoeft alleen een eventHandler te implementeren. Events worden door Canvas.hs naar de eventHandler gestuurd. De eventHandler hoeft alleen nieuwe output terug te geven. Met behulp van de state variabele kan de programmeur de staat van het programma bijhouden.



```
import CanvasHs
import CanvasHs.Data

type State = Int -- Onze state is een getal, dit is in te stellen

initState = 0 -- De initiele state

-- registreer jezelf bij Canvas.hs met de functie handler en de initiele state 0
main = installEventHandler handler 0

-- de functie die vanuit Canvas.hs aangeroepen wordt
handler :: State -> Event -> (State, Output)
handler state StartEvent = (state, shape $ Container 900 600 [Event defaults{eventId="circle", mouseClick=True}
                                                                $ Circle (200, 200) 50])

-- deze patternmatch wordt uitgevoerd als er een mouseclick met eventid circle is.
handler state (MouseClick (x,y) "circle") = (state, shape $ Container 900 600 [ Event defaults{eventId="circle",
                                                    mouseClick=True} $ Fill (255,0,0,1.0) $ Circle (200, 200) 50 ])
```





Events

Er zijn verschillende soorten events die Canvas.hs ondersteund, waaronder klik, toets en scroll events.

Actions

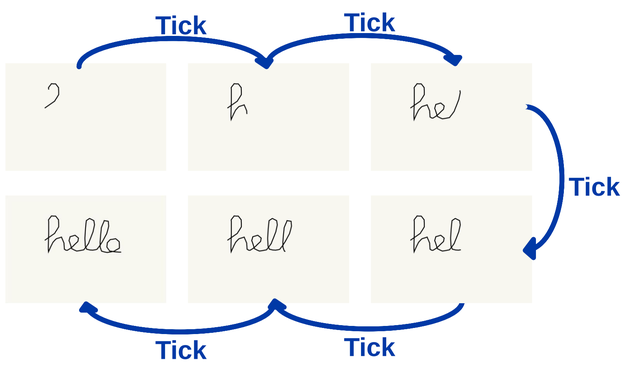
De eventHandler kan naast het opleveren van een nieuwe grafische boom ook nog een aantal uit te voeren acties opleveren; hierbij kan gedacht worden aan bijvoorbeeld het lezen of schrijven van bestanden.

Output
Block $ LoadFileString “save.txt”

Event
(**FileLoadedString** “save.txt" contents)

Timers

Timers zijn acties die op een gezet interval een event voor het programma genereren. Bij het afgaan van de timer genereert het programma een tick-event.

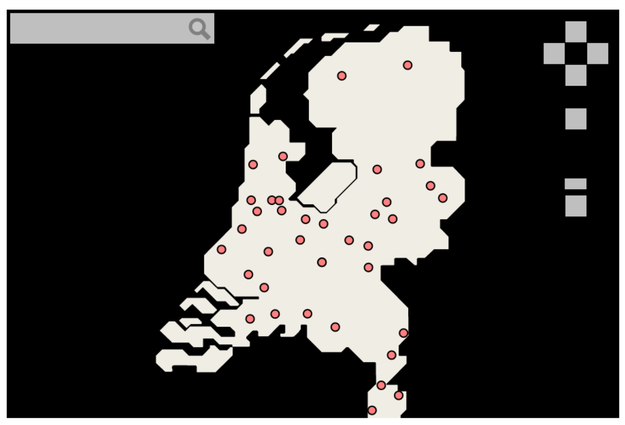


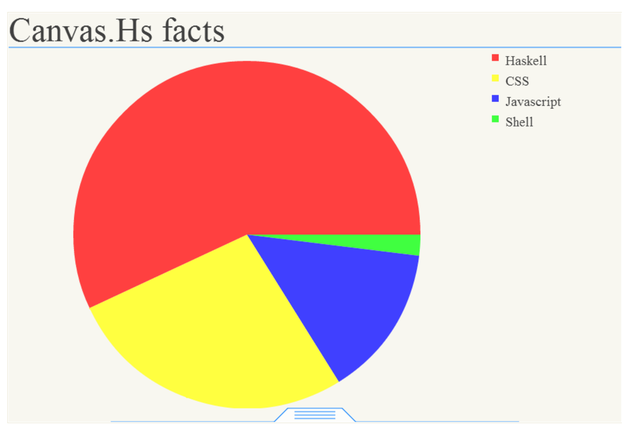
State

De programmeur bepaalt zelf in welke structuur hij de staat van het programma wilt bijhouden. De programmeur hoeft alleen zijn eventHandler het correcte type te geven. Hieronder heeft de state het type Int.

handler :: Int -> Event -> (Int, Output)

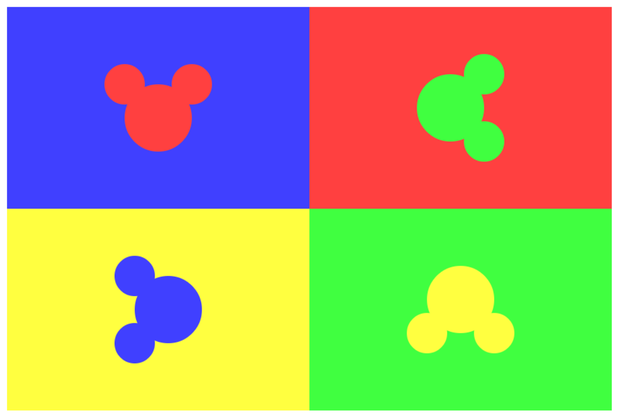
Toepassingen



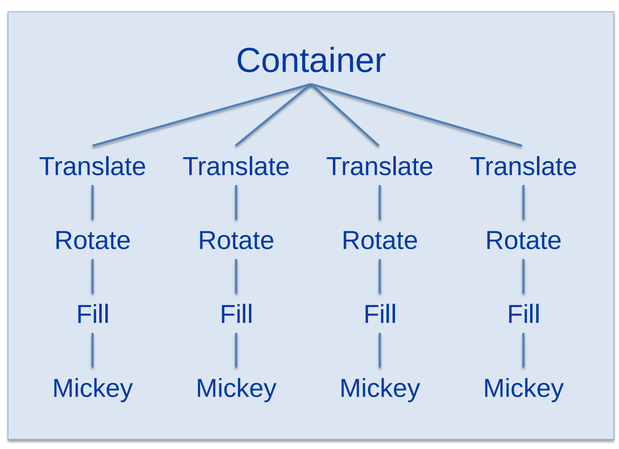


Output tree

Output wordt gedefinieerd in de vorm van een grafische boom. Er worden verschillende grafische primitieven ondersteund, waaronder vierkanten, lijnen, polygonen, cirkels en tekst. Een container-element kan gebruikt worden om meerdere primitieven te bevatten. Met behulp van translaties worden de rotatie, positie, vulkleur en lijnkleur aangepast van het element.



Deze gedeeltelijke grafische boom demonstreert het gebruik van translaties en rotaties op Mickey.



UNIVERSITY OF TWENTE.

GitHub https://github.com/CanvasHS/Canvas.hs