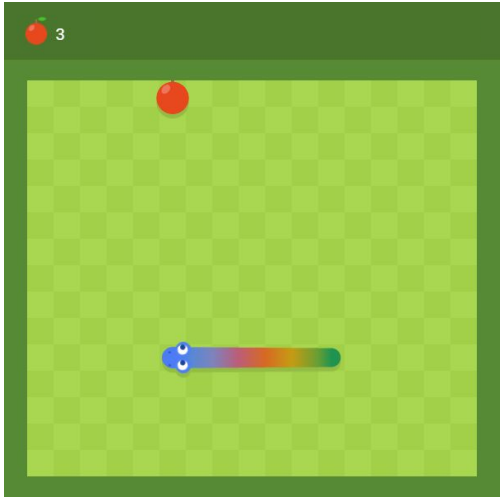
Programmeerproject 1

WPO: Implementatie ADTs + Tutorial Grafische Bibliotheek

Groep 1: Woensdag 1 december (11u-13u)
Groep 2: Donderdag 2 december (14u-16u)
Groep 3: Vrijdag 3 december (9u-11u)

Voorbeeldproject: Snake

- Speler bestuurt een slang
- Slang beweegt in de spelwereld
  Op een grid, horizontaal, of verticaal (niet diagonaal)
- Als de slang een appel opeet, dan wordt de slang langer
  Doel van het spel: zoveel mogelijk appels eten zonder te botsen!
- Scorebord toont de huidige score, en de hoogst behaalde score



Voorbeeldproject: Snake

- Speler bestuurt een slang
- Slang beweegt in de spelwereld
  - Op een grid, horizontaal, of verticaal (niet diagonaal)
- Als de slang een appel opeet, dan wordt de slang langer
  - ~~Doel van het spel: zoveel mogelijk appels eten zonder te botsen!~~
- ~~Scorebord toont de huidige score, en de hoogst behaalde score~~

Download het project

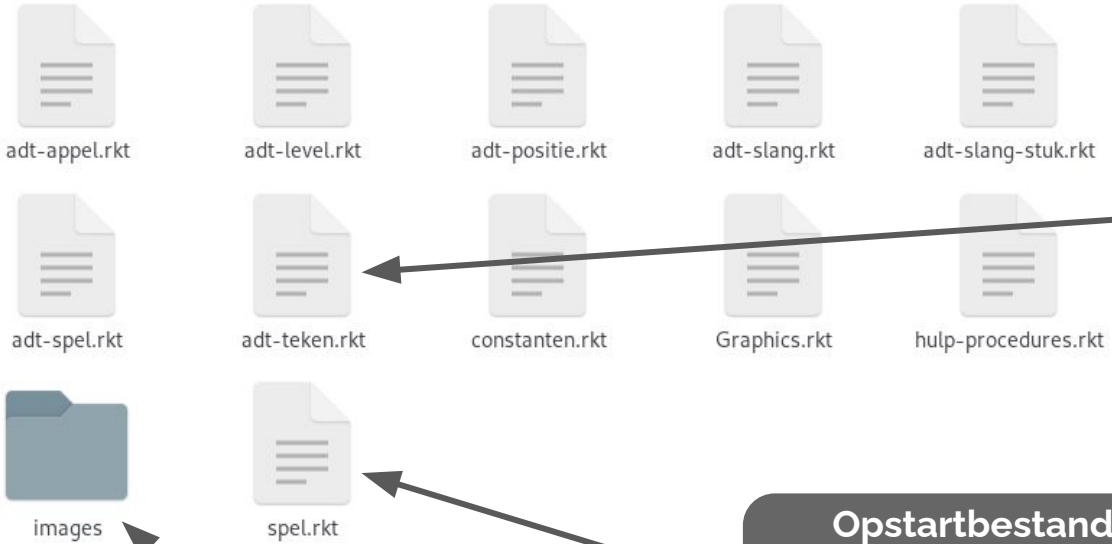
Project beschikbaar op Canvas (cursusruimte: 006328)


```
Documenten
    > WPO Grafische bibliotheek
    > Opgave
    > opgave.zip
```

Zip-archief! **Eerst het bestand uitpakken!**

Alles voor dit WPO is toegevoegd aan dit Zip-archief.

Overzicht project



Scheiding spel- en tekenlogica door middel van een apart Teken ADT

Opstartbestand dat alle andere bestanden inlaad in Scheme en vervolgens het spel opstart.

Aparte map met daarin alle bitmaps (afbeeldingen).

Objectgebaseerde implementatiestijl

Object-gebaseerde implementatie

Scheme: dispatch-objectjes

```
(define (maak-matrix rijen kolommen)
  ;; We instantieren de vector met nul.
  (define waardes (make-vector (* rijen kolommen) 0))

  ;; Haalt een waarde op positie rij/kolom uit de matrix.
  (define (get rij kolom)
    (let ((index (+ (* rij kolommen) kolom)))
      (vector-ref waardes index)))

  ;; Stelt een nieuwe waarde in op de positie rij/kolom
  (define (set! rij kolom waarde)
    (let ((index (+ (* rij kolommen) kolom)))
      (vector-set! waardes index waarde)))

  ;; ...

  (define (dispatch-matrix msg)
    (cond ((eq? msg 'set!)      set!)
          ((eq? msg 'get)       get)
          ((eq? msg 'kolommen) kolommen)
          ((eq? msg 'rijen)     rijen)
          (else (display "Ongeldig bericht"))))
  dispatch-matrix)
```

- Wat is een object?
- Hoe roept een object een procedure op van een ander object?
- Wat is de relatie met het omgevingsmodel van Scheme?

Objectjes!

```
(define (maak-matrix rijen kolommen)
  ;; We instantieren de vector met nul.
  (define waardes (make-vector (* rijen kolommen) 0))

  ;; Haalt een waarde op positie rij/kolom uit de matrix.
  (define (get rij kolom)
    (let ((index (+ (* rij kolommen) kolom)))
      (vector-ref waardes index)))

  ;; Stelt een nieuwe waarde in op de positie rij/kolom
  (define (set! rij kolom waarde)
    (let ((index (+ (* rij kolommen) kolom)))
      (vector-set! waardes index waarde)))

  ;; ...

  (define (dispatch-matrix msg)
    (cond ((eq? msg 'set!)      set!)
          ((eq? msg 'get)       get)
          ((eq? msg 'kolommen) kolommen)
          ((eq? msg 'rijen)     rijen)
          (else (display "Ongeldig bericht"))))
  dispatch-matrix)
```

Overzicht WPO

Opgaven Positie ADT 1 - 2 - 3

1. Implementeer **2 procedures** om de x- en y-coördinaat aan te passen.
   Hint: Bekijk de implementatie van het dispatch object voor de naamgeving!

2. Implementeer een predicaat (procedure) die twee posities vergelijkt.
   Geeft `#t` terug als beiden dezelfde x- en y-coördinaat hebben, anders `#f`.
   Hint: De procedures die in `(define (maak-adt-positie x y) ...)` zelf gedefinieerd zijn hebben toegang tot de variabelen `x` en `y` van het eigen positie-objectje.

3. Implementeer een operatie die een nieuw positie-object teruggeeft die de originele positie 1 eenheid “opschuift” aan de hand van een symbool.
   Symbolen: `'omhoog`, `'omlaag`, `'links`, en `'rechts`.
   Hint: linksboven is (0, 0), rechtsonder is (w-1, h-1)!

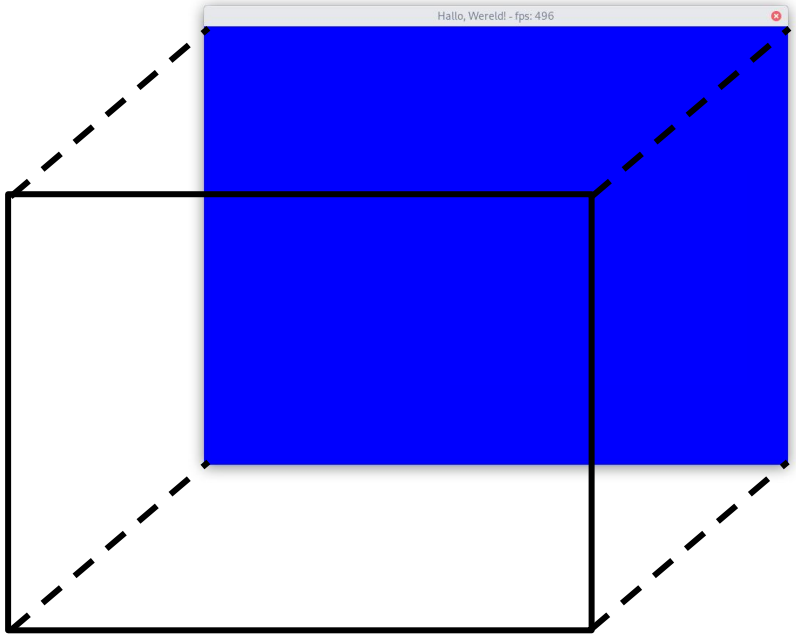
Gebruik de testcode onderaan in `adt-positie.rkt` om je oplossingen te verifiëren in de REPL.

De Grafische Bibliotheek

```
1.   (#%require "Graphics.rkt")
2.
3.   (define venster (make-window 800 600 "Hallo, Wereld!"))
4.   ((venster 'set-background!) "blue")
```

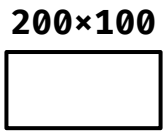
De Grafische Bibliotheek

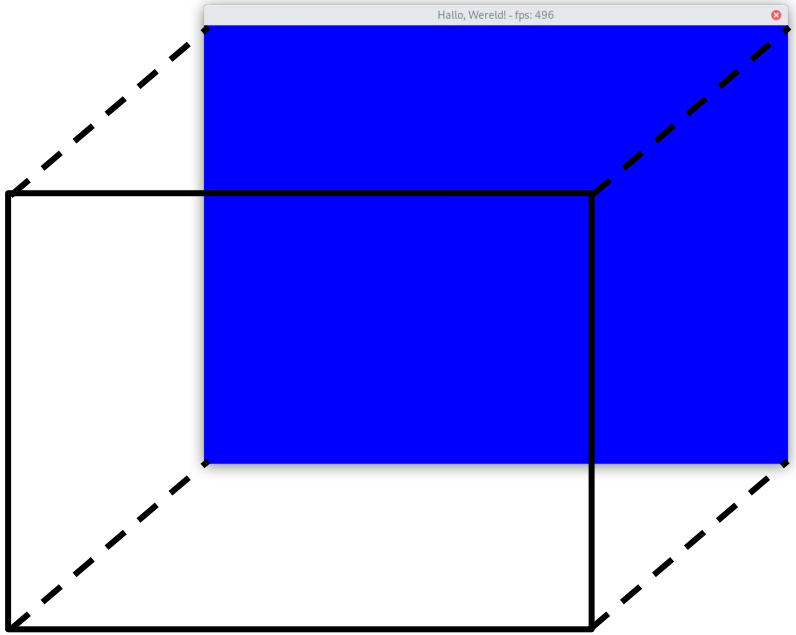
```
1.    (#%require "Graphics.rkt")
2.
3.    (define venster (make-window 800 600 "Hallo, Wereld!"))
4.    ((venster 'set-background!) "blue")
5.
6.    (define mijn-eerste-laag (venster 'make-layer))
```



De Grafische Bibliotheek

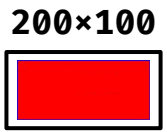
```
1.   (#%require "Graphics.rkt")
2.
3.   (define venster (make-window 800 600 "Hallo, Wereld!"))
4.   ((venster 'set-background!) "blue")
5.
6.   (define mijn-eerste-laag (venster 'make-layer))
7.
8.   (define mijn-tile (make-tile 200 100))
```

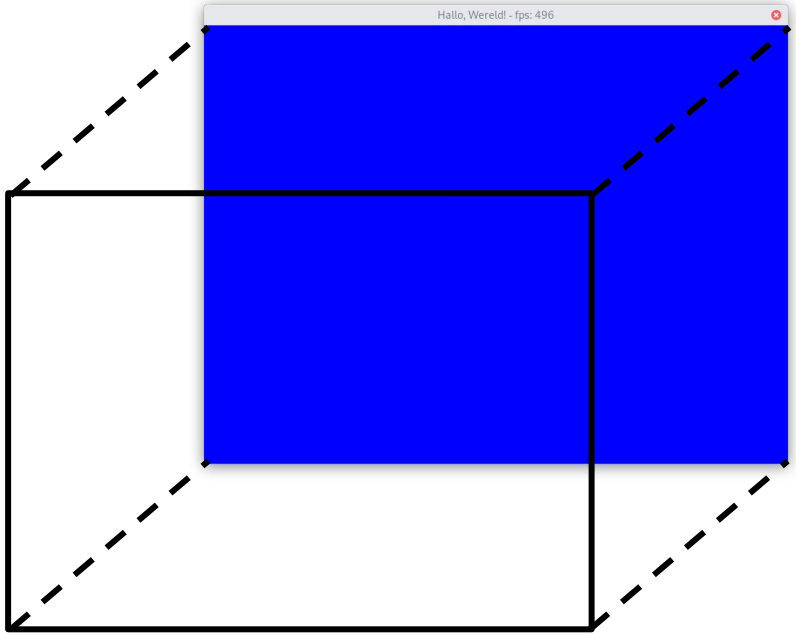




De Grafische Bibliotheek

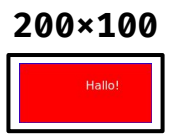
```
1.   (#%require "Graphics.rkt")
2.
3.   (define venster (make-window 800 600 "Hallo, Wereld!"))
4.   ((venster 'set-background!) "blue")
5.
6.   (define mijn-eerste-laag (venster 'make-layer))
7.
8.   (define mijn-tile (make-tile 200 100))
9.
10.  ((mijn-tile 'draw-rectangle) 10 10 180 80 "red")
```

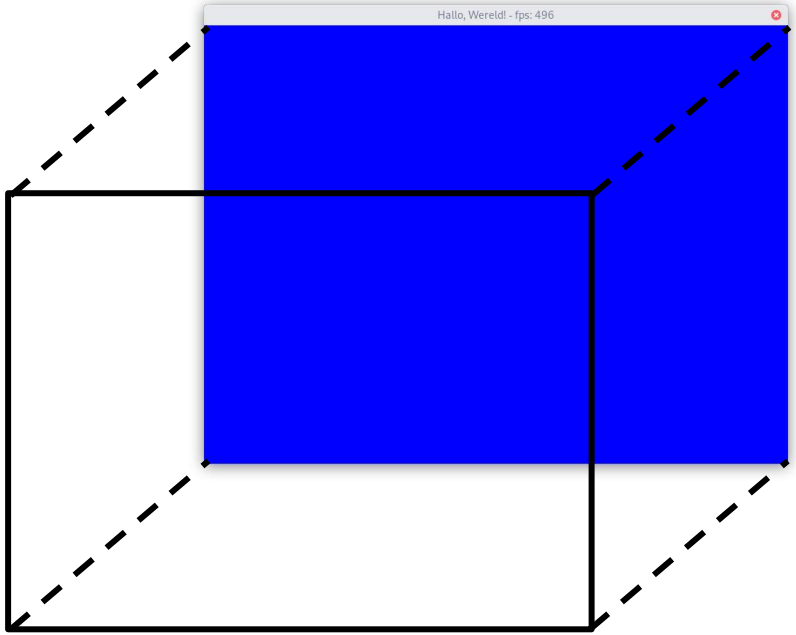




De Grafische Bibliotheek

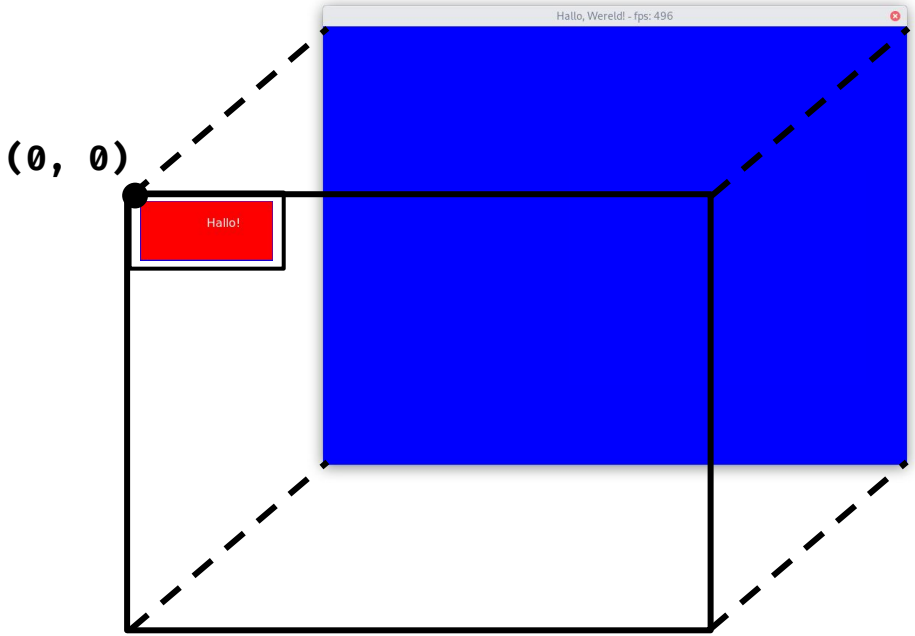
```
1.   (#%require "Graphics.rkt")
2.
3.   (define venster (make-window 800 600 "Hallo, Wereld!"))
4.   ((venster 'set-background!) "blue")
5.
6.   (define mijn-eerste-laag (venster 'make-layer))
7.
8.   (define mijn-tile (make-tile 200 100))
9.
10.  ((mijn-tile 'draw-rectangle) 10 10 180 80 "red")
11.  ((mijn-tile 'draw-text) "Hallo!" 12 100 30 "white")
```





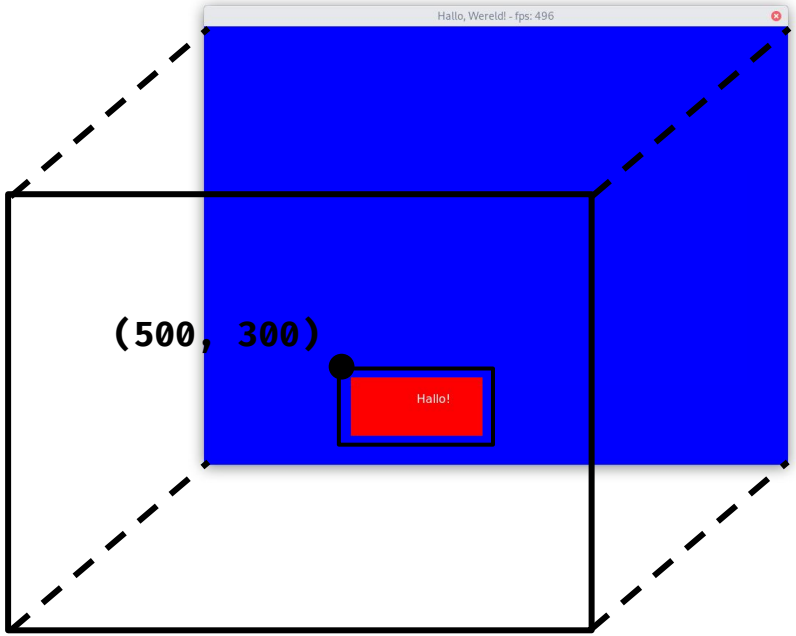
De Grafische Bibliotheek

```
1.   (#%require "Graphics.rkt")
2.
3.   (define venster (make-window 800 600 "Hallo, Wereld!"))
4.   ((venster 'set-background!) "blue")
5.
6.   (define mijn-eerste-laag (venster 'make-layer))
7.
8.   (define mijn-tile (make-tile 200 100))
9.
10.  ((mijn-tile 'draw-rectangle) 10 10 180 80 "red")
11.  ((mijn-tile 'draw-text) "Hallo!" 12 100 30 "white")
12.
13.  ((mijn-eerste-laag 'add-drawable) mijn-tile)
```



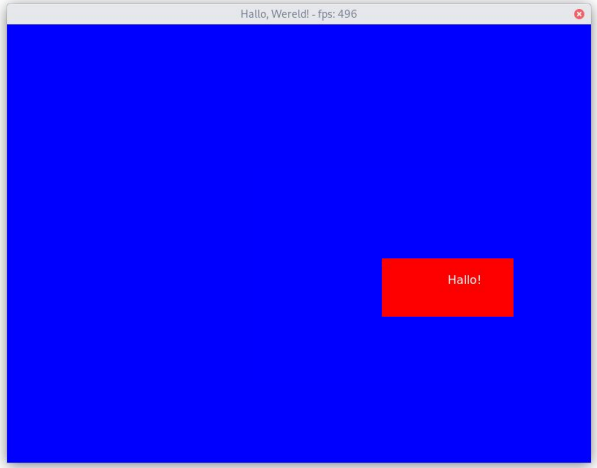
De Grafische Bibliotheek

```
1.   (#%require "Graphics.rkt")
2.
3.   (define venster (make-window 800 600 "Hallo, Wereld!"))
4.   ((venster 'set-background!) "blue")
5.
6.   (define mijn-eerste-laag (venster 'make-layer))
7.
8.   (define mijn-tile (make-tile 200 100))
9.
10.  ((mijn-tile 'draw-rectangle) 10 10 180 80 "red")
11.  ((mijn-tile 'draw-text) "Hallo!" 12 100 30 "white")
12.
13.  ((mijn-eerste-laag 'add-drawable) mijn-tile)
14.
15.  ((mijn-tile 'set-x!) 500)
16.  ((mijn-tile 'set-y!) 300)
```



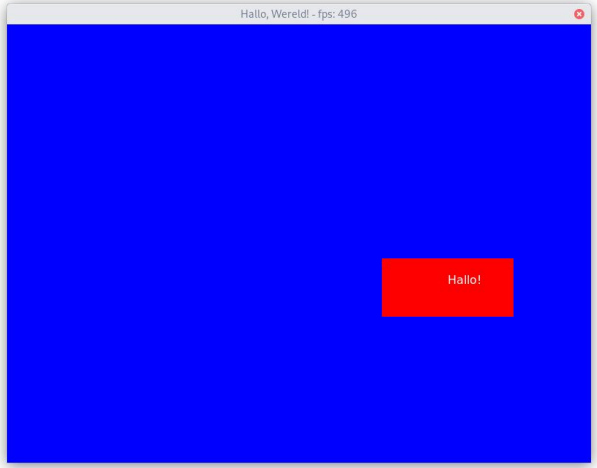
De Grafische Bibliotheek

```
1.   (#%require "Graphics.rkt")
2.
3.   (define venster (make-window 800 600 "Hallo, Wereld!"))
4.   ((venster 'set-background!) "blue")
5.
6.   (define mijn-eerste-laag (venster 'make-layer))
7.
8.   (define mijn-tile (make-tile 200 100))
9.
10.  ((mijn-tile 'draw-rectangle) 10 10 180 80 "red")
11.  ((mijn-tile 'draw-text) "Hallo!" 12 100 30 "white")
12.
13.  ((mijn-eerste-laag 'add-drawable) mijn-tile)
14.
15.  ((mijn-tile 'set-x!) 500)
16.  ((mijn-tile 'set-y!) 300)
```



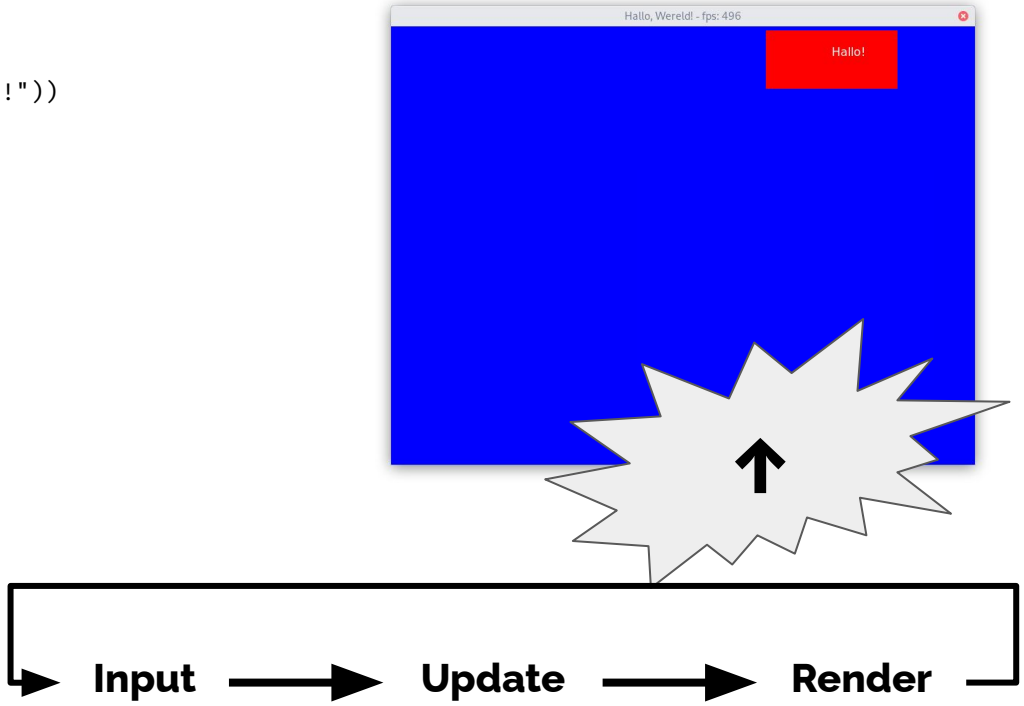
De Grafische Bibliotheek

```
 1.   (#%require "Graphics.rkt")
 2.
 3.   (define venster (make-window 800 600 "Hallo, Wereld!"))
 4.   ((venster 'set-background!) "blue")
 5.
 6.   (define mijn-eerste-laag (venster 'make-layer))
 7.
 8.   (define mijn-tile (make-tile 200 100))
 9.
10.   ((mijn-tile 'draw-rectangle) 10 10 180 80 "red")
11.   ((mijn-tile 'draw-text) "Hallo!" 12 100 30 "white")
12.
13.   ((mijn-eerste-laag 'add-drawable) mijn-tile)
14.
15.   ((mijn-tile 'set-x!) 500)
16.   ((mijn-tile 'set-y!) 300)
17.
18.   ((venster 'set-key-callback!)
19.    (lambda (type toets)
20.      (if (and (eq? type 'pressed) (eq? toets 'up))
21.          ((mijn-tile 'set-y!) 0))))
```



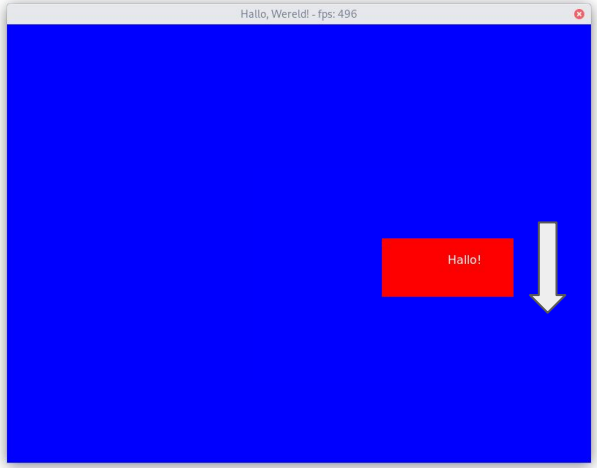
De Grafische Bibliotheek

```
 1.   (#%require "Graphics.rkt")
 2.
 3.   (define venster (make-window 800 600 "Hallo, Wereld!"))
 4.   ((venster 'set-background!) "blue")
 5.
 6.   (define mijn-eerste-laag (venster 'make-layer))
 7.
 8.   (define mijn-tile (make-tile 200 100))
 9.
10.   ((mijn-tile 'draw-rectangle) 10 10 180 80 "red")
11.   ((mijn-tile 'draw-text) "Hallo!" 12 100 30 "white")
12.
13.   ((mijn-eerste-laag 'add-drawable) mijn-tile)
14.
15.   ((mijn-tile 'set-x!) 500)
16.   ((mijn-tile 'set-y!) 300)
17.
18.   ((venster 'set-key-callback!)
19.    (lambda (type toets)
20.      (if (and (eq? type 'pressed) (eq? toets 'up))
21.          ((mijn-tile 'set-y!) 0))))
```



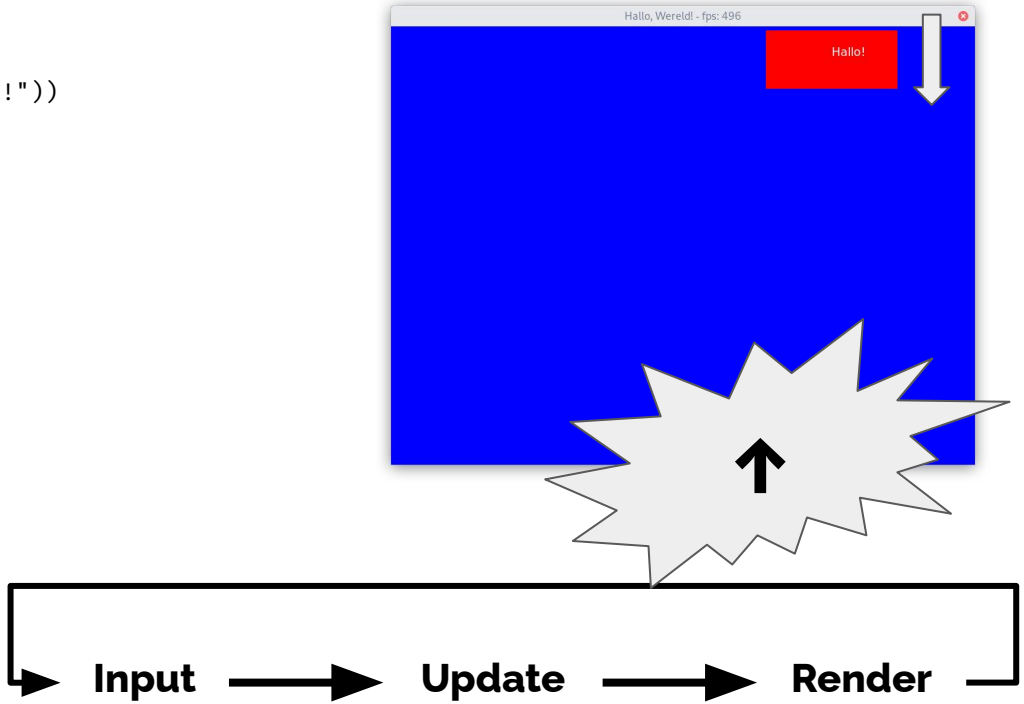
De Grafische Bibliotheek

```
 1.   (#%require "Graphics.rkt")
 2.
 3.   (define venster (make-window 800 600 "Hallo, Wereld!"))
 4.   ((venster 'set-background!) "blue")
 5.
 6.   (define mijn-eerste-laag (venster 'make-layer))
 7.
 8.   (define mijn-tile (make-tile 200 100))
 9.
10.   ((mijn-tile 'draw-rectangle) 10 10 180 80 "red")
11.   ((mijn-tile 'draw-text) "Hallo!" 12 100 30 "white")
12.
13.   ((mijn-eerste-laag 'add-drawable) mijn-tile)
14.
15.   ((mijn-tile 'set-x!) 500)
16.   ((mijn-tile 'set-y!) 300)
17.
18.   ((venster 'set-key-callback!)
19.    (lambda (type toets)
20.      (if (and (eq? type 'pressed) (eq? toets 'up))
21.          ((mijn-tile 'set-y!) 0))))
22.
23.   ((venster 'set-update-callback!)
24.    (lambda (ms)
25.      (let* ((huidige-y (mijn-tile 'get-y))
26.             (nieuwe-y (+ huidige-y (* 0.10 ms))))
27.        ((mijn-tile 'set-y!) nieuwe-y))))
```



De Grafische Bibliotheek

```
 1.   (#%require "Graphics.rkt")
 2.
 3.   (define venster (make-window 800 600 "Hallo, Wereld!"))
 4.   ((venster 'set-background!) "blue")
 5.
 6.   (define mijn-eerste-laag (venster 'make-layer))
 7.
 8.   (define mijn-tile (make-tile 200 100))
 9.
10.   ((mijn-tile 'draw-rectangle) 10 10 180 80 "red")
11.   ((mijn-tile 'draw-text) "Hallo!" 12 100 30 "white")
12.
13.   ((mijn-eerste-laag 'add-drawable) mijn-tile)
14.
15.   ((mijn-tile 'set-x!) 500)
16.   ((mijn-tile 'set-y!) 300)
17.
18.   ((venster 'set-key-callback!)
19.    (lambda (type toets)
20.      (if (and (eq? type 'pressed) (eq? toets 'up))
21.          ((mijn-tile 'set-y!) 0))))
22.
23.   ((venster 'set-update-callback!)
24.    (lambda (ms)
25.      (let* ((huidige-y (mijn-tile 'get-y))
26.             (nieuwe-y (+ huidige-y (* 0.10 ms))))
27.        ((mijn-tile 'set-y!) nieuwe-y))))
```



Spellogica en Tekenlogica

***** het effectieve “tekenen op het scherm” wordt volledig door de grafische bibliotheek afgehandeld. Het is de **`Update`** die, na de spellogica uit te voeren, de tiles op de juiste plaatsen moet zetten zodanig dat de **`Render`** weet wat er moet getekend worden.

Spellogica = Toetsenbordinvoer uitlezen (input)
+ Nieuwe spelsituatie berekenen (update)

Tekenlogica = (Nieuwe) spelsituatie tekenen (render)

Implementeer bij je spellogica *wat* er getekend moet worden…

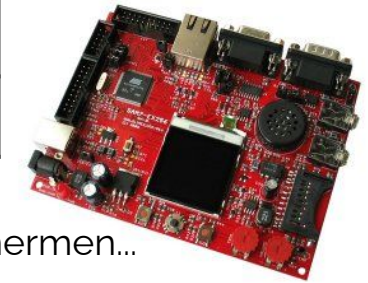
Maak gebruik van een (of meerdere) ADT(s) die bepalen *hoe* er op het scherm getekend moet worden (bijvoorbeeld: Teken ADT in het Snake-spel)

Scheiding spel- en tekenlogica: Coördinatensysteem

w cellen breed

h cellen hoog

(0, 0)	(1, 0)	(2, 0)	...	(w-1, 0)
(0, 1)	(1, 1)	(2, 1)	...	(w-1, 1)
(0, 2)	(1, 2)	(2, 2)	...	(w-1, 2)
...	...	...	...	...
(0, h-1)	(1, h-1)	(2, h-1)	...	(w-1, h-1)



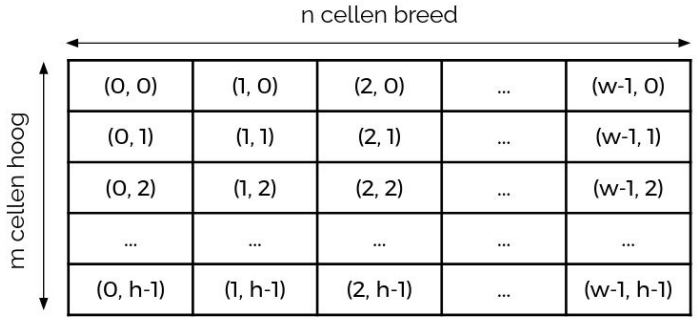
Spellogica moet onafhankelijk werken van de specifieke hardware:

Verschillende resolutie, Matrix van LED-lichtjes, arcadekast met meerdere schermen...

⇒ Schrijf je spellogica **NIET** in pixels!

Opgave Teken ADT 4 - 5

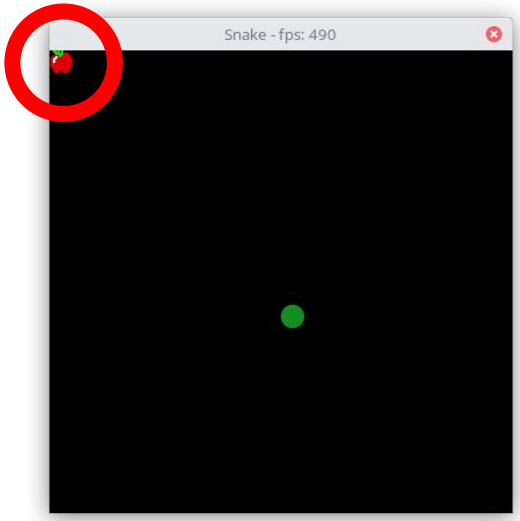
4. Implementeer in het Teken ADT de procedure om de tile van een spelobject op een positie op het scherm te tekenen (verplaatsen).



5. Implementeer de logica om een slang te tekenen. Merk op dat een slang bestaat uit meerdere stukken. Maak gebruik van de operaties van het Slang ADT voor iets te doen met elk slangstuk.

Test het spel!

Als alles correct geïmplementeerd is zou het spel nu speelbaar moeten zijn!



Maar geen interactie met de appel!

Appel ADT is nog niet geïmplementeerd.

Waarom crasht het spel niet?

Bestaande code controleert of er een appel aangemaakt is.

Waarom wordt de appel al wel getekend?

Het Teken ADT maakt al wel een tile aan voor de appel.

Opgaven Appel ADT en Level ADT 6 - 7

6. Implementeer het Appel ADT.

`ADT appel`

`maak-adt-appel`	`(Positie → Appel)`
`positie`	`(/ → Positie)`
`positie!`	`(Positie → /)`

7. Maak een instantie (object) van het Appel ADT aan in het Level ADT.

Test daarna of het volledige spel werkt!

Algemene tips

- Meng geen tekenlogica en spellogica.
  De `beweeg!` operatie in Snake ADT verandert niet de locatie van de tiles!

- Maak je implementatie robuust voor een dynamisch aantal spelelementen.
  Bv. voeg ondersteuning toe voor 0..n appels (zoals met de slangstukken)

- Maak gebruik van de vastgelegde Scheme conventies voor de benamingen van procedures.
  Predikaatprocedures eindigen op `?`, en destructieve procedures op `!`... (zie 1.3.5 in r5rs)

- Wees consistent! En, ofwel alles in het Nederlands, *or everything in English*!

- Vermijd code-duplicatie en 🪄 magische constanten.

- Maak geen gebruik van globale variabelen (m.u.v. constanten!).

- Verkies symbolen boven strings. Die laatste zijn vaak minder performant!

Contact

programmeerproject1@soft.vub.ac.be