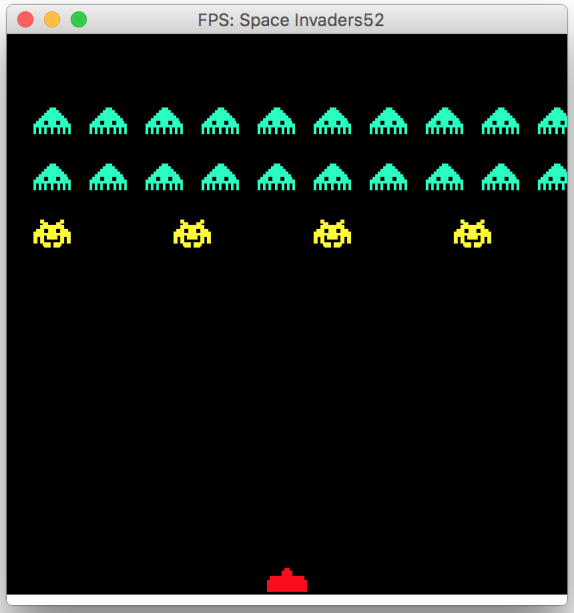
Programmeerproject: Eindverslag Fase 2

Alexandre Kahn
alexkahn@vub.ac.be
0500067
Faculteit Wetenschappen en Bio-ingenieurswetenschappen
Vrije Universiteit Brussel

9 april 2016



Inhoudsopgave

1 Inleiding

Dit is het verslag van de tweede fase. Hiervoor moesten we verder bouwen op wat we in de vorige implementatie gemaakt hebben.
Waar eerst de aliens statisch getekend moesten worden, zouden ze in deze fase moeten bewegen. Het moest ook mogelijk zijn om meerdere kogels af te schieten. Tot slot moest er ook het begin van een menu geïmplementeerd worden. Hierna moet het mogelijk zijn om in een laatste rechte lijn aan de laatste fase te beginnen en het spel af te werken.

2 Abstracte Data Types

Aan het schip-ADT is niets veranderd, de kogel- en alien-ADT's zijn ongeveer hetzelfde gebleven. Beiden hebben vooral een aantal kleine veranderingen in interne functies, maar het grote deel van de veranderingen moest in het Vloot-ADT en het nieuwe Kogels-ADT gebeuren. Deze zorgen dat de beweging en de collision-detection in goede banen verloopt. Tot slot zijn er voor het menu een aantal ADT's bijgekomen, dit om het zo gemakkelijker te maken.

2.1 Alien-ADT

De aliens worden allemaal apart aangemaakt en zijn objecten op hun eigen. Om er meerdere op te roepen is er het `Vloot-ADT`.
Elk alien-object zal zijn eigen positie hebben en bijhouden, alsook de kleur en hoeveel levens deze nog heeft. Het aantal levens is aan de kleur gelinkt.
Door elk hun eigen positie bij te houden kan het vloot-adt gemakkelijk de coördinaten te veranderen.

```
1    (maak-alien-adt <x-positie> <kleur>)
2
```

Message	Parameter	Resultaat
x	/	Geeft de positie van het opgevraagde x-coördinaat terug.
y	/	Geeft de positie van het opgevraagde y-coördinaat terug.
x!	nieuwe-x	Geeft het object een nieuwe x-positie.
y!	nieuwe-y	Geeft het object een nieuwe y-positie.
teken!	teken-adt	Geeft aan het teken-ADT mee dat het, het alien-object moet tekenen.
delete!	teken-adt	Geeft aan het teken-ADT mee dat het alien-object verwijderd moet worden.
geraakt!	kogel-adt teken-adt	Deze message start een loop, die een leven van de alien wegneemt. Indien deze er geen meer heeft wordt de alien gedeletet.

2.2 Vloot-ADT

Bij aanmaak van de vloot moet er meegegeven worden hoeveel aliens men moet hebben. Deze worden dan allemaal in een lijst opgeslagen. Deze begint leeg en

wordt dan aangevuld met alien-objecten. De eersten zijn allemaal groen en die daaronder geel, maar dit kan aangepast worden zonder veel problemen. Momenteel is dit zodat alles zo automatisch mogelijk gebeurt.
Er wordt dan aangevuld tot als het maximum voor een rij bereikt wordt. Hiervoor wordt dan de `afstand-tussen-2` berekend. Voor de volgende rijen gebeurt dan hetzelfde
Verder is er een functie om over de vloot te lopen, `loop-over-vloot`. Deze is zeer belangrijk voor de collision-detection, want er moet voor elke alien gekeken worden of deze geraakt wordt.
De beweging van de aliens wordt hier ook meegegeven. Er wordt altijd gecheckt of er een botsing gebeurt met een zijkant van het scherm en indien dit het geval is wordt de richting omgedraaid. Ook wordt er gekeken of de aliens de onderkant van het scherm raken, in dat geval wordt het menu geladen. Het spel herstart hierna wel niet meer, maar dit zal tegen de volgende implementatie in orde zijn.

```
1  (maak-vloot-adt <aantal> <teken-adt>)
2
```

Message	Parameter	Resultaat
maak-aliens!	Aantal pos-y	Vult de lege lijst aan met nieuwe aliens. En laat het teken-ADT nieuwe alien-tiles aanmaken.
teken!	teken-adt	Zorgt dat elk alien-object zichzelf tekent.
loop-over-vloot	functie	Zorgt dat een bepaalde functie gemapt wordt over alien-objecten.
beweeg!	/	Zorgt dat elk alien-object zich verplaatst.
vloot	/	Geeft lijst met al de aliens terug.

2.3 Schip-ADT

Het schip-adt maakt een schip aan. Deze zal over de onderrand van het scherm moeten bewegen en kogels kunnen afschieten.
De `y-positie` zal altijd 1 zijn, de onderrand van het scherm. In het begin zal de `x-positie` steeds 0.5, of het midden van het venster zijn. De `snelheid` of het aantal stappen waarmee het schip zich zal voortbewegen is 0.3, dit zorgt voor een mooie vloeiende beweging en een gemakkelijk bestuurbaar schip.
De `beweeg!`-functie zal afhankelijk van de meegegeven toetsenbord-input het schip recht of links verplaatsten. Echter indien het zich al in de rand van het venster bevindt zal het niet verder bewegen, dit om te vermijden dat de raket van het spelbord zou verdwijnen.

```
1        (maak-schip-adt)
2
```

Message	Parameter	Resultaat
x!	nieuwe-x	Zorgt ervoor dat de x-positie van het schip aangepast wordt. Moet enkel voor x bestaan, aangezien de y-positie steeds 1 blijft.
x	/	Geeft de x-coördinaat van het schip terug.
beweeg!	richting	Zorgt afhankelijk van de meegegeven richting voor een verplaatsing naar die kant.
teken!	teken-adt	Geeft aan het teken-adt mee om het schip te tekenen.

2.4 Kogel-ADT

Het kogel-ADT wordt niet meer vanaf het begin van het spel aangemaakt, aangezien er nu meerdere van deze ADT's zullen zijn. Ze zullen beheerd worden door een adt dat op het vloot-adt lijkt. Elke kogel zal nog steeds zijn eigen type, staat en snelheid hebben en zal ook zelf zien of het een alien raakt. Ook de beweging zit in dit ADT en zal nagaan of de kogel zich tegen een rand bevindt.

```
1       (maak-kogel-adt posx teken-adt)
2
```

Message	Parameter	Resultaat
x!	nieuwe-x	Zorgt ervoor dat de x-positie van de kogel aangepast wordt.
y!	nieuwe-y	Zorgt ervoor dat de y-positie van de kogel aangepast wordt.
x	/	Geeft de x-coördinaat van de kogel terug.
y	/	Geeft de y-coördinaat van de kogel terug.
beweeg!	teken-adt	Zorgt voor de verplaatsing en kijkt of de kogel tegen de rand komt.
teken!	teken-adt	Geeft aan het teken-adt mee om het schip te tekenen.
staat?	/	Geeft de staat van de kogel terug.
schietlus	teken-adt vloot-adt	Gaat na of de kogel een alien raakt. en indien nodig delete deze de kogel en de alien
type	/	Geeft het type van de kogel terug
geraakt!	alien alien	Delete de alien, zet de kogel op inactief en geeft mee aan het teken-adt om deze te deleten

2.5 Kogels-ADT

Dit ADT zorgt om de meerdere kogels in goede banen te leiden en op een correcte manier aan te spreken. Het doel is om alle kogels in een lijst op te slaan. Dit gebeurt door telkens de nieuwe kogel voor de al bestaande te consen. Het heeft functies om een nieuwe kogel te maken, een functie om functies op alle kogels uit te voeren, zorgt ervoor dat alles getekend wordt en dat alle kogels kunnen bewegen. In de `beweeg!` functie wordt er ook nagegaan of de kogel een alien raakt.

```
(maak-kogels-adt)
```

Message	Parameter	Resultaat
beweeg!	teken-adt vloot-adt	Zorgt dat elke kogel moet bewegen en nagaan of deze botst.
teken!	teken-adt	Geeft aan de kogels mee dat ze zichzelf moeten tekenen.
maak-kogel!	posx, teken-adt	Maakt een nieuw kogel-adt aan en plaatst deze in de lijst. Ook wordt er in het teken-adt een nieuwe kogel aangemaakt.
kogels	/	Geeft de lijst met kogels terug.

2.6 Teken-ADT

Het `teken-ADT` zorgt ervoor dat alle objecten getekend worden. Het maakt hiervoor gebruik `Graphics.rkt`.
Bij het aanmaken van dit ADT, wordt er een venster met zwarte achtergrond getekend. Hierop komen dan verschillende lagen,`'make-layer`, voor de objecten. Deze worden op hun laag getekend als tegels, `'make-tile`. Voor het schip, het menu, de how-to en de rotator is dit simpel, er moet slechts een tegel gemaakt worden. De aliens en kogel zitten daarentegen in lijsten. Voor de beide is dit zodat er meerdere getekend kunnen worden.
Zowel de kogels als de aliens zijn te verwijderen, dit door in de lijst de correcte tegel weg te halen. Dat kan aan de hand van `neem-iets`. De functie zoekt het juiste object en geeft deze dan terug.
Het tekenen van de tiles gebeurt bijna altijd op dezelfde manier, zo wordt indien nodig de juiste tegel gezocht, dan worden de correcte coördinaten berekend, de x-waarde gecentraliseerd en de tegel op deze positie getekend.
Verder zijn er ook de functies om het spel te doen runnen.
Bij het aanmaken moet alles verwijderd worden van het scherm, bij het verwijderen van het menu moet alles terug op de juiste plek komen te staan. Dit gebeurt aan de hand van `verwijder-alles` en `herteken-alles!`.

```
(maak-adt-teken <titel> <hoogte> <breedte>)
```

Message	Parameters	Resultaat
set-toets-functie!	fun	Zorgt dat de toetsenbordinput ingelezen wordt.
set-spel-lus-functie!	fun	Zorgt ervoor dat het speelbord steeds geupdatet wordt.
teken-schip!	schip-adt	Zorgt ervoor dat het schip getekend wordt.
teken-alien!	alien-adt	Zorgt ervoor dat de alien getekend wordt.
teken-kogel!	kogel-adt	Zorgt ervoor dat de kogel getekend wordt.
teken-menu!	menu-adt	Zorgt ervoor dat het menu getekend wordt.
teken-rotator!	rotator-adt	Zorgt ervoor dat de rotator getekend wordt.
teken-how-to!	how-to-adt	Zorgt ervoor dat de how-to getekend wordt.
nieuwe-alien!	alien-adt	Maakt een nieuwe alien-tile en voegt deze toe aan de lijst.
nieuwe-kogel!	kogel-adt	Maakt een nieuwe kogel-tile en voegt deze toe aan de lijst.
herteken-alles	alienlijst	Hertekend alle spelonderdelen op de juiste plaats terug.
maak-rotator	rotator-adt	Voegt de rotator-tile toe aan de laag.
maak-how-to	how-to-adt	Voegt de how-to-tile toe aan de laag.
delete-alien!	alien-adt	Zoekt de correcte alien-tile en verwijderd deze.
delete-kogel!	kogel-adt	Zoekt de correcte kogel-tile en verwijderd deze.
delete-schip!	/	Delete de schip-tile van het scherm.
delete-menu!	/	Delete de menu-tile van het scherm.
delete-rotator!	/	Delete de rotator-tile van het scherm.
delete-how-to!	/	Delete de how-to-tile van het scherm.
verwijder-alles	alienlijst kogellijst	Delete het schip, alle aliens en alle kogels. zo kan het menu op een leeg scherm komen.

2.7 Spel-ADT

Dit ADT bevat de eigenlijke spellus. Er is de functie, `start` om het spel te starten. In de `spel-lus-functie` wordt steeds het verschil in de tijd opgeteld bij de `speltijd`, worden zowel vloot als het schip getekend en wordt de schietlus uitgevoerd indien er een kogel is afgeschoten. Dit is veel efficiënter dan non-stop te kijken of er een botsing voorvalt, aangezien er niet altijd een kogel op scherm is. Ook worden de `spel-lus-functie` en de `toets-naar-spel` functie steeds meegegeven aan het teken-adt.

```
1  (maak-adt-spel)
2
```

Message	Paramater	Resultaat
start	/	Dit is de entry point die ervoor zorgt dat de spellus en hierbij het hele spel start.

2.8 Menu-ADT

Dit ADT dient om het menu weer te geven en is zeer simpel. Het heeft een vaste positie en moet instaat zijn het menu te tekenen, te verwijderen en de staat van

het spel bij te houden.

```
(maak-menu-adt)

```

Message	Paramater	Resultaat
x	/	Geeft de x-coördinaat van het menu terug.
y	/	Geeft de y-coördinaat van het menu terug.
teken!	teken-adt	Laat het teken-adt weten dat het menu getekend moet worden.
staat	/	Geeft de staat van het spel terug.
staat!	nieuwe-staat	Verandert de staat naar de nieuwe staat.
delete!	teken-adt	Verwijdert het menu uit het teken-adt.

2.9 Rotator-ADT

Dit ADT dient voor de selector in het menu. Het moet ook niet veel kunnen, behalve op de correcte posities verschijnen. Hiervoor heeft het ook een staat zodat het spel gemakkelijk kan weten welke keuze er gemaakt wordt.

```
(maak-menu-adt)

```

Message	Paramater	Resultaat
x	/	Geeft de x-coördinaat van de rotator terug.
y	/	Geeft de y-coördinaat van de rotator terug.
teken!	teken-adt	Laat het teken-adt weten dat de rotator getekend moet worden.
staat	/	Geeft de staat van de rotator terug.
up!	/	Doet de rotator stijgen, indien het kan en verandert de staat.
down!	/	Doet de rotator dalen, indien het kan en verandert de staat.

2.10 How-To-ADT

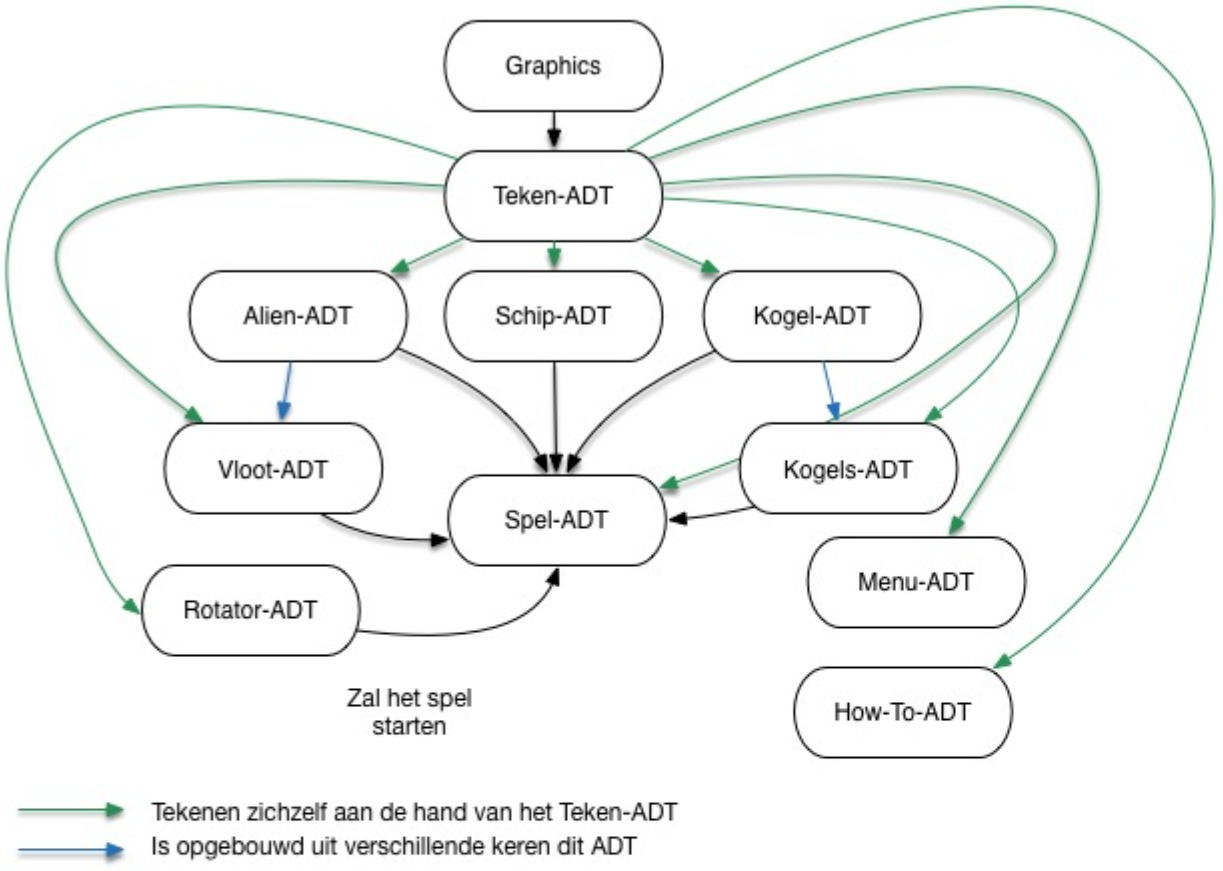
Dit ADT dient om de controls te tonen. Moet ook heel weinig kunnen, buiten getekend worden en van het scherm kunnen verdwijnen.

```
(maak-menu-adt)

```

Message	Paramater	Resultaat
x	/	Geeft de x-coördinaat van de How-To terug.
y	/	Geeft de y-coördinaat van de How-To terug.
teken!	teken-adt	Laat het teken-adt weten dat de How-To getekend moet worden.

3 Afhankelijkheidsdiagram



4 Verloop

Deze fase verliep redelijk vlot, hoewel ik heel lang door een aantal kleine bugs heb vastgezeten. Uiteindelijk was ik nog wel goed voor de deadline klaar en had ik dus nog tijd om mijn code te verduidelijken en te verbeteren. Ook zijn er een aantal zaken waar ik meer heb gedaan als de minimale vereisten. Zo kan het menu altijd opgeroepen worden en zal het spel hierna gewoon kunnen herstarten. Het enige wat niet direct werkte zoals het moest was het heraanmaken van het spel nadat de aliens de onderkant van het scherm geraakt hebben, tegen de volgende implementatie zal dit zeker wel werken.

5 Broncodes

5.1 Abstracties.rkt

```
#lang racket

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;    Alexandre Kahn     ;;;;;;;;;
;;;;;;;;;       500067          ;;;;;;;;;
;;;;;;;;;      1e BA CW         ;;;;;;;;;
;;;;;;;;; alexkahn@vub.ac.be   ;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;Zo kunnen andere files deze bestanden gebruiken.
(provide venster-hoogte
         venster-breedte
         px-venster-hoogte
         px-venster-breedte
         px-alien-hoogte
         px-alien-breedte
         px-alien-reeel
         px-schip-hoogte
         px-schip-breedte
         px-kogel-hoogte
         px-kogel-breedte
         px-start-hoogte
         px-start-breedte
         px-rotator-hoogte
         px-rotator-breedte
         px-how-to-hoogte
         px-how-to-breedte
         helft
         increment
         decrement
         binnen-grenzen?
         check-geraakt?
         centraliseer)

;Speelvenster
(define venster-breedte 1)
(define venster-hoogte 1)

```

```
39 ;Speelvenster in aantal pixels
40 (define px-venster-hoogte 420)
41 (define px-venster-breedte 420)
42
43 ;Grootte van een alien in pixels
44 (define px-alien-hoogte 30)
45 (define px-alien-breedte 30)
46 ;Grootte van een alien op het venster
47 (define px-alien-reeel
48   (/ px-alien-hoogte px-venster-hoogte))
49 ;Grootte van het schip in pixels
50 (define px-schip-hoogte 20)
51 (define px-schip-breedte 30)
52 ;Grootte van de kogel in pixels
53 (define px-kogel-hoogte 8)
54 (define px-kogel-breedte 4)
55 ;Grootte van de start-knop in pixels
56 (define px-start-hoogte 120)
57 (define px-start-breedte 80)
58 ;Grootte van de rotator in pixels
59 (define px-rotator-hoogte 50)
60 (define px-rotator-breedte 50)
61 ;Grootte van de how-to in pixels
62 (define px-how-to-hoogte 300)
63 (define px-how-to-breedte 277)
64
65
66 ;Functie om de helft te berekenen.
67 (define (helft object)
68   (/ object 2))
69
70 ;Functie om een positie aan een bepaalde snelheid te doen stijgen
71 (define (increment positie snelheid)
72   (+ positie snelheid))
73 (define (decrement positie snelheid)
74   (- positie snelheid))
75
76 ;Functie om te zien of het object zich nog binnen de grenzen van
      het spel bevindt?
77 (define (binnen-grenzen? positie omvang)
78   (and (< 0 (- positie (helft omvang)))
79        (> px-venster-hoogte (- positie (helft omvang)))))
80
81 ;Functie om te zien of er een botsing tussen 2 objecten gebeurt.
82 (define (check-geraakt? object1 object2)
83   (and (> (object1 'levens) 0)
84        (and (and (<= (- (object1 'x) (helft px-alien-reeel)) (
       object2 'x))
85                  (>= (+ (object1 'x) (helft px-alien-reeel)) (
       object2 'x)))
86             (and (>= (+ (object1 'y) px-alien-reeel) (object2 'y))
87                  (<= (object1 'y) (object2 'y))))))
88
89
90 ;Functie om objecten te centraliseren. Anders worden ze teveel naar
       rechts getekend.
91 (define (centraliseer positie type)
```

```
92    (cond ((eq? type 'alien) (- positie (helft px-alien-hoogte)))
93          ((eq? type 'kogel) (- positie (helft px-kogel-hoogte)))
94          ((eq? type 'start) (- positie (helft px-start-hoogte)))
95          ((eq? type 'schip) (- positie (helft px-schip-breedte)))
96          ((eq? type 'rotator) (- positie (helft px-rotator-breedte))
      )
97          ((eq? type 'how-to) (- positie (helft px-how-to-breedte)))
98          ))
```

Abstracties.rkt

5.2 Alien.rkt

```
1  #lang racket
2
3
4  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
5  ;;;;;;;;;    Alexandre Kahn     ;;;;;;;;;
6  ;;;;;;;;;       500067          ;;;;;;;;;
7  ;;;;;;;;;      1e BA CW         ;;;;;;;;;
8  ;;;;;;;;; alexkahn@vub.ac.be   ;;;;;;;;;
9  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
10
11
12
13 ;;;;;;;;;;;;;;
14 ;;Alien ADT;;
15 ;;;;;;;;;;;;;;
16 ;Het Alien ADT zal gewoon alien's oproepen. Daarentegen zal het
       Vloot ADT ervoor zorgen dat er meerdere samen op het scherm
       verschijnen en deze in een latere fase zullen kunnen bewegen
       door de beweging door te geven.
17 (provide maak-alien-adt)
18 (require "teken-adt.rkt")
19 (require "Abstracties.rkt")
20
21 (define (maak-alien-adt positie-x positie-y kleur)
22   (define levens 0)
23   (cond ((eq? kleur 'geel)
24          (set! levens 1))
25         ((eq? kleur 'groen)
26          (set! levens 2)))
27   ;Hulpfuncties
28   ;Om nieuwe posities mee te geven
29   (define (set-positie-x! nieuwe-x)
30     (set! positie-x nieuwe-x))
31   (define (set-positie-y! nieuwe-y)
32     (set! positie-y nieuwe-y))
33   ;Om de objecten te tekenen of te verwijderen
34   (define (teken! teken-adt)
35     ((teken-adt 'teken-alien!) dispatch-alien))
36   (define (delete! teken-adt)
37     ((teken-adt 'delete-alien!) dispatch-alien))
38   ;Loop van wat er moet gebeuren als een alien geraakt is.
39   ;De kogel moet gedelete worden, zodat deze niet op het scherm
       blijft.
40   ;De alien moet indien hij geen levens meer heeft ook gedeletet
       worden.
```

```
41  (define (geraakt! kogel teken-adt)
42    (teken-adt 'delete-kogel!)
43    (cond ((eq? 'normaal (kogel 'type))
44           (if (< 1 levens)
45               (set! levens (- levens 1))
46               (begin
47                 (delete! teken-adt)
48                 (set! levens 0))))))
49  (define (levens! getal)
50    (set! levens getal))
51  ;Dispatch er met dit object ge nterageerd kan worden.
52  (define (dispatch-alien msg)
53    (cond
54      ((eq? msg 'x!) set-positie-x!)
55      ((eq? msg 'y!) set-positie-y!)
56      ((eq? msg 'x) positie-x)
57      ((eq? msg 'y) positie-y)
58      ((eq? msg 'teken!) teken!)
59      ((eq? msg 'delete!) delete!)
60      ((eq? msg 'geraakt!) geraakt!)
61      ((eq? msg 'levens) levens)
62      ((eq? msg 'levens!) levens!)
63      ((eq? msg 'kleur) kleur)
64      ))
65
66  ;Dispatch oproepen dit is noodzakelijk voor object-georienteerd
      programmeren in Scheme
67  dispatch-alien)
```

Alien.rkt

5.3 Howto.rkt

```
1  #lang racket
2
3
4
5  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
6  ;;;;;;;;;    Alexandre Kahn     ;;;;;;;;;
7  ;;;;;;;;;       500067          ;;;;;;;;;
8  ;;;;;;;;;      1e BA CW         ;;;;;;;;;
9  ;;;;;;;;; alexkahn@vub.ac.be   ;;;;;;;;;
10 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
11
12
13
14
15 (require "teken-adt.rkt")
16 (require "Abstracties.rkt")
17 (provide maak-how-to-adt)
18
19 (define (maak-how-to-adt)
20   (define positie-x 0.3)
21   (define positie-y 0.3)
22   (define (teken! teken-adt)
23     ((teken-adt 'teken-how-to!) dispatch-how-to))
24   (define (dispatch-how-to msg)
```

```
25      (cond
26        ((eq? msg 'x) positie-x)
27        ((eq? msg 'y) positie-y)
28        ((eq? msg 'teken!) teken!)))
29    dispatch-how-to)
```

howto.rkt

5.4 Kogel.rkt

```
1  #lang racket
2  (require "teken-adt.rkt")
3  (require "Abstracties.rkt")
4  (provide maak-kogel-adt)
5
6
7  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
8  ;;;;;;;;;    Alexandre Kahn     ;;;;;;;;;
9  ;;;;;;;;;       500067          ;;;;;;;;;
10 ;;;;;;;;;      1e BA CW         ;;;;;;;;;
11 ;;;;;;;;; alexkahn@vub.ac.be   ;;;;;;;;;
12 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
13
14
15 ;;;;;;;;;;;;;
16 ;;Kogel ADT;;
17 ;;;;;;;;;;;;;
18 ;Dit ADT zal de kogel aanmaken.Het zal nadat het afgeschoten is
       moeten voortbewegen op het scherm.
19
20 (define (maak-kogel-adt positie-x)
21   ;Functie om te schieten is weg. Handiger om elke kogel bij
       aanmaak de juiste x-waarde mee te geven.
22
23   ;Een aantal constantes
24   ;  (define positie-x 0.5)
25   (define positie-y 1)
26   (define snelheid 0.03)
27   (define type 'normaal)
28   (define staat 'actief)
29
30   ;Functies om de posities aan te passen
31   (define (set-positie-x! nieuwe-x)
32     (set! positie-x nieuwe-x))
33   (define (set-positie-y! nieuwe-y)
34     (set! positie-y nieuwe-y))
35
36
37   ;Deze functie zorgt voor de beweging van de kogel. Indien de
       kogel voorbij de rand van het scherm komt wordt hij als het
       ware herladen.
38   (define (beweeg! teken-adt)
39     (cond ((< 0 positie-y)
40            (set-positie-y! (decrement positie-y snelheid)))
41           (else (begin (set! staat 'inactief)
42                        ((teken-adt 'delete-kogel!) dispatch-kogel))
       )))
```

```
43
44  ;Deze functie zorgt voor het verwijderen. teken-adt krijgt de
      boodschap om de kogel te deleten.
45  (define (delete! teken-adt)
46    ((teken-adt 'delete-kogel!) dispatch-kogel
47                                ))
48
49  ;Laat de kogel na een botsing zowel inactief worden als van het
      scherm te verdwijnen. In verdere fase kan dan naar het type
      kogel gekeken worden.
50  (define (geraakt! alien teken-adt)
51    (set! staat 'inactief)
52    (delete! teken-adt)
53    ((alien 'geraakt!) dispatch-kogel teken-adt))
54
55  ;Gaat na of er een botsing is.
56  (define (schietlus teken-adt vloot-adt)
57    (define (schiet-op alien)
58      (if (eq? staat 'actief)
59          (cond ((check-geraakt? alien dispatch-kogel)
60                 (geraakt! alien teken-adt)))
61          'ok))
62    ((vloot-adt 'loop-over-vloot) schiet-op)); Zorgt ervoor dat
      elke alien afgegaan wordt.
63
64  ;Zorgt ervoor dat het teken-adt het object tekent.
65  (define (teken! teken-adt)
66    ((teken-adt 'teken-kogel!) dispatch-kogel))
67
68  (define (dispatch-kogel msg)
69    (cond
70      ((eq? msg 'x!) set-positie-x!)
71      ((eq? msg 'y!) set-positie-y!)
72      ((eq? msg 'x) positie-x)
73      ((eq? msg 'y) positie-y)
74      ((eq? msg 'beweeg!) beweeg!)
75      ((eq? msg 'teken!) teken!)
76      ((eq? msg 'staat?) staat) ;Geeft de staat van de kogel terug
77      ((eq? msg 'geraakt!) geraakt!)
78      ((eq? msg 'type) type)
79      ((eq? msg 'schietlus) schietlus)
80      ))
81  dispatch-kogel)
```

Kogel.rkt

5.5 Kogels.rkt

```
1 #lang racket
2 (provide maak-kogels-adt)
3
4 (require "teken-adt.rkt")
5 (require "Abstracties.rkt")
6
7 (require "Kogel.rkt")
8
9
```

```
10
11 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
12 ;;;;;;;;;    Alexandre Kahn     ;;;;;;;;;
13 ;;;;;;;;;       500067          ;;;;;;;;;
14 ;;;;;;;;;      1e BA CW         ;;;;;;;;;
15 ;;;;;;;;; alexkahn@vub.ac.be   ;;;;;;;;;
16 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
17
18
19 ;;;;;;;;;;;;;;;
20 ;;Kogels ADT;;
21 ;;;;;;;;;;;;;;;
22
23 ;Dit ADT zal als doel hebben om verschillende kogels te maken en te
       zorgen dat deze elk de correcte handelingen doen.
24 (define (maak-kogels-adt)
25   (define kogels '())
26   (define (maak-kogel! posx teken-adt) ;wordt opgeroepen bij
       schieten
27     (let ((nieuwe-kogel (maak-kogel-adt posx)))
28       (set! kogels (cons  nieuwe-kogel kogels)) ;maakt dmv van cons
        cellen een lijst met kogels. Efficienter door van voor te
       consen.
29       ;((nieuwe-kogel 'schiet!) posx teken-adt) ;Zorgt dat de kogel
        afgeschoten wordt
30       ((teken-adt 'nieuwe-kogel!) nieuwe-kogel)
31       ((teken-adt 'teken-kogel!) nieuwe-kogel)
32       ))
33
34   (define (loop-over-kogels functie) ;Zorgt dat functies over de
       kogels gemapt kunnen worden.
35     (map functie kogels))
36
37   (define (teken! teken-adt)
38     (for-each (lambda (kogel-adt)
39                 ((kogel-adt 'teken!) teken-adt))
40               kogels))
41   (define (beweeg! teken-adt vloot-adt)
42     (for-each (lambda (kogel-adt)
43                 ((kogel-adt 'beweeg!) teken-adt)
44                 ((kogel-adt 'schietlus) teken-adt vloot-adt))
45               kogels))
46
47   (define (dispatch msg)
48     (cond ((eq? msg 'maak-kogel!) maak-kogel!)
49           ((eq? msg 'kogels) kogels)
50           ((eq? msg 'beweeg!) beweeg!)
51           ((eq? msg 'teken!) teken!)
52           ))
53   dispatch)
```

Kogels–adt.rkt

5.6 Menu.rkt

```
1 #lang racket
2 (provide maak-menu-adt)
```

```
3
4  (require "teken-adt.rkt")
5  (require "Abstracties.rkt")
6
7
8  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
9  ;;;;;;;;;    Alexandre Kahn     ;;;;;;;;;
10 ;;;;;;;;;       500067          ;;;;;;;;;
11 ;;;;;;;;;      1e BA CW         ;;;;;;;;;
12 ;;;;;;;;; alexkahn@vub.ac.be   ;;;;;;;;;
13 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
14
15
16 ;;;;;;;;;;;;;;;
17 ;; Menu ADT ;;
18 ;;;;;;;;;;;;;;;
19 (define (maak-menu-adt)
20   ;Een aantal constantes en berekende waarden
21   (define positie-x 0.60)
22   (define positie-y 0.4)
23   (define staat 'pauze)
24
25   ;geeft aan het teken-adt mee dat het menu getekend moet worden
26   (define (teken! teken-adt)
27     ((teken-adt 'teken-menu!) dispatch-menu))
28
29   ;geeft aan het teken-adt mee dat het menu gedeletet mag worden
30   (define (delete! teken-adt)
31     ((teken-adt 'delete-menu!)))
32
33   ;dient om de staat van het spel in op te slaan
34   (define (staat! nieuwe-staat)
35     (set! staat nieuwe-staat))
36
37   ;Dispatch functie voor interactie met het adt
38   (define (dispatch-menu msg)
39     (cond
40       ((eq? msg 'x) positie-x)
41       ((eq? msg 'y) positie-y)
42       ((eq? msg 'teken!) teken!)
43       ((eq? msg 'staat) staat)
44       ((eq? msg 'staat!) staat!)
45       ((eq? msg 'delete!) delete!)
46       (else (display "error, wrong msg " msg))
47       ))
48   dispatch-menu)
```

Menu.rkt

5.7 Rotator.rkt

```
1 #lang racket
2 (provide maak-rotator-adt)
3
4 (require "teken-adt.rkt")
5 (require "Abstracties.rkt")
6
```

```
7
8
9
10 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
11 ;;;;;;;;;    Alexandre Kahn     ;;;;;;;;;
12 ;;;;;;;;;       500067          ;;;;;;;;;
13 ;;;;;;;;;      1e BA CW         ;;;;;;;;;
14 ;;;;;;;;; alexkahn@vub.ac.be   ;;;;;;;;;
15 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
16
17
18
19 ;Maakt het adt aan met de selector.
20 (define (maak-rotator-adt)
21   (define staat 0)
22   (define positie-x 0.45) ;posities zo gekozen dat het mooi uitkwam
23   (define positie-y 0.4)
24
25   (define (up!) ;reageert als je met het pijltje naar boven gaat
26     (if (> staat 0)
27         (begin
28           (set! staat (- staat 1))
29           (set! positie-y (- positie-y 0.18))) ;zorgt dat het mooi
     uitkomt, is omdat het menu 1 foto is
30         'ok))
31
32   (define (down!) ;reageert als je met het pijltje naar boven gaat
33     (if (< staat 2)
34         (begin
35           (set! staat (+ staat 1))
36           (set! positie-y (+ positie-y 0.18))) ;zorgt dat het mooi
     uitkomt, is omdat het menu 1 foto is
37         'ok))
38
39   ;Geeft mee aan het teken-adt om alles te tekenen
40   (define (teken! teken-adt)
41     ((teken-adt 'teken-rotator!) dispatch-rotator))
42   (define (dispatch-rotator msg)
43     (cond
44       ((eq? msg 'x) positie-x)
45       ((eq? msg 'y) positie-y)
46       ((eq? msg 'teken!) teken!)
47       ((eq? msg 'staat) staat)
48       ((eq? msg 'up!) up!)
49       ((eq? msg 'down!) down!)))
50   dispatch-rotator)
```

rotator.rkt

5.8 Schip.rkt

```
1 #lang racket
2 (require "teken-adt.rkt")
3 (require "Abstracties.rkt")
4 (provide maak-schip-adt)
5
6 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;;;;;;;    Alexandre Kahn     ;;;;;;;;;
;;;;;;;;;       500067          ;;;;;;;;;
;;;;;;;;;      1e BA CW         ;;;;;;;;;
;;;;;;;;; alexkahn@vub.ac.be   ;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;



;;;;;;;;;;;;;
;;Schip ADT;;
;;;;;;;;;;;;;
;Het Schip-adt zal het schip aanmaken. Het zal ervoor kunnen zorgen dat het kan voortbewegen.

(define (maak-schip-adt)
  ;Een aantal vaste beginwaarden
  (define positie-x 0.5)
  (define positie-y 1)
  (define snelheid 0.06)

  ;Zorgt dat we een nieuwe positie kunnen geven aan het schip
  ;Geen set-positie-y! nodig aangezien het schip enkel op de x-as beweegt
  (define (set-positie-x! nieuwe-x)
    (set! positie-x nieuwe-x))

  ;Het schip zal moeten bewegen op basis van invoer van het toetsenbord. Ook moet er gezien worden dat het schip niet uit het scherm kan komen
  (define (beweeg! richting)
    (cond
      ((eq? richting 'links)
       (if (< 0 positie-x)
           (set-positie-x! (decrement positie-x snelheid))
           positie-x))
      ((eq? richting 'rechts)
       (if (> 1 positie-x)
           (set-positie-x! (increment positie-x snelheid))
           positie-x))))

  ;Zorgt ervoor dat het teken-adt het object tekent.
  (define (teken! teken-adt)
    ((teken-adt 'teken-schip!) dispatch-schip))

  ;Dispatch er met dit object ge nterageerd kan worden.
  (define (dispatch-schip msg)
    (cond
      ((eq? msg 'x!) set-positie-x!)
      ((eq? msg 'x) positie-x)
      ((eq? msg 'beweeg!) beweeg!)
      ((eq? msg 'teken!) teken!)))

  dispatch-schip)
```

Schip.rkt

5.9 Spel.rkt

```
#lang racket
(provide maak-adt-spel)
(require "teken-adt.rkt")
(require "Abstracties.rkt")
(require "Alien.rkt")
(require "Vloot.rkt")
(require "Schip.rkt")
(require "Kogel.rkt")
(require "Kogels-adt.rkt")
(require "Menu.rkt")
(require "rotator.rkt")
(require "howto.rkt")


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;    Alexandre Kahn     ;;;;;;;;;
;;;;;;;;;       500067          ;;;;;;;;;
;;;;;;;;;      1e BA CW         ;;;;;;;;;
;;;;;;;;; alexkahn@vub.ac.be   ;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;
;; SPEL ADT ;;
;;;;;;;;;;;;;;;
;Dit adt zal de spellus bevatten. Ook zal het ervoor zorgen dat de
    input van het toetsenbord doorgegeven wordt aan de nodige
    spelelementen. Verder zal de collision detection ook in dit ADT
     gebeuren.

(define (maak-adt-spel)
  ;Maakt alle nodige adt's aan.
  (define teken-adt (maak-adt-teken "Space Invaders"
    px-venster-hoogte px-venster-breedte))
  (define schip-adt (maak-schip-adt))
  (define vloot-adt (maak-vloot-adt 24 teken-adt))
  (define kogels-adt (maak-kogels-adt))
  (define menu-adt (maak-menu-adt))
  (define rotator-adt (maak-rotator-adt))
  (define how-to-adt (maak-how-to-adt))

  (define (pauze!)
    ((menu-adt 'staat!) 'pauze)
    ((teken-adt 'verwijder-alles) (vloot-adt 'vloot)(kogels-adt '
    kogels))
    ((teken-adt 'maak-rotator)))

  (define (restart!)

    (pauze!)
    ;Momenteel geen tijd op een oplossing te vinden om het spel te
    herstarten. Waarschijnlijk gaat dit het beste door de start
    functie op te splitsen zodat enkel de deze opnieuwe opgeroepen
    kan worden.
    ;vloot-adt 'verwijder-vloot!))
    )  ;spellus
```

```
50  ;Leest de toetsenbordinput uit en stuurt dit door naar het spel.
51  ;Afhankelijk van de staat van het spel
52  (define (toets-naar-spel toets)
53    (cond ((eq? (menu-adt 'staat) 'play)
54           (cond
55             ((eq? toets 'right)
56              ((schip-adt 'beweeg!) 'rechts))
57             ((eq? toets 'left)
58              ((schip-adt 'beweeg!) 'links))
59             ((eq? toets #\space)
60              ((kogels-adt 'maak-kogel!) (schip-adt 'x) teken-adt))
61             ((eq? toets #\p)
62              (pauze!))))
63          ((eq? (menu-adt 'staat) 'pauze)
64           (cond
65             ((eq? toets #\return)
66              (cond ((= 0 (rotator-adt 'staat))
67                     (begin
68                       ((menu-adt 'staat!) 'play)
69                       ((menu-adt 'delete!) teken-adt)
70                       ((teken-adt 'herteken-alles!) (vloot-adt '
    vloot))))
71                    ((= 1 (rotator-adt 'staat))
72                     (begin
73                       ((menu-adt 'delete!) teken-adt)
74                       ((menu-adt 'staat!) 'howto)
75                       ((teken-adt 'maak-how-to))
76                       ))
77                    ((= 2 (rotator-adt 'staat))
78                     (exit))))
79             ((eq? toets 'up)
80              ((rotator-adt 'up!)))
81             ((eq? toets 'down)
82              ((rotator-adt 'down!)))
83             ))
84          ((eq? (menu-adt 'staat) 'howto)
85           (cond ((eq? toets #\return)
86                  (begin
87                    ((menu-adt 'staat!) 'pauze)
88                    ((teken-adt 'delete-how-to!))
89                    ((teken-adt 'maak-rotator))
90                    ((teken-adt 'verwijder-alles)(vloot-adt 'vloot)
    (kogels-adt 'kogels))))
91                 ))))
92
93
94  ; Functies voor het starten van het spel
95  (define (start)
96    ((teken-adt 'verwijder-alles) (vloot-adt 'vloot) (kogels-adt '
    kogels))
97    ((teken-adt 'maak-rotator)) ;Is de selector om in het menu te
    kiezen wat we willen openen
98    (define spel-tijd 0)
99    ;;Schietlus, moet enkel opgeroepen worden als er geschoten is
100   ;; Oproepen om het spel te laten lopen
101   (define (spel-lus-functie delta-tijd)
102     (set! spel-tijd (+ spel-tijd delta-tijd))
```

```
103       (cond ((eq? (menu-adt 'staat) 'play) ;De toetsenbordinput om
      het spel mee te spelen
104              (begin
105                ((schip-adt 'teken!) teken-adt)
106                ((vloot-adt 'teken!) teken-adt)
107                ((vloot-adt 'beweeg!) dispatch-spel)
108                ;Indien de kogel afgeschoten is wordt de collision
      detection gedaan. Is performanter op deze manier, anders moet
      er elke beurt voor elke alien gecheckt worden.
109                ((kogels-adt 'beweeg!) teken-adt vloot-adt)
110                ((kogels-adt 'teken!) teken-adt)
111                ))
112              ((eq? (menu-adt 'staat) 'pauze) ; Input voor in het
      menu
113               (begin
114                 ((menu-adt 'teken!) teken-adt)
115                 ((rotator-adt 'teken!) teken-adt)
116                 ))
117              ((eq? (menu-adt 'staat) 'howto) ;Input voor een extra
      menu
118               ((how-to-adt 'teken!) teken-adt))
119              ))
120
121
122     ;(nu worden de adt's zoals het moet in een lus hertekend)
123     ((teken-adt 'set-spel-lus-functie!) spel-lus-functie)
124     ((teken-adt 'set-toets-functie!) toets-naar-spel))
125
126   ;; Dispatch functie
127   (define (dispatch-spel msg)
128     (cond
129       ((eq? msg 'start) start)
130       ((eq? msg 'pauze!) pauze!)
131       ((eq? msg 'restart!) restart!)
132       ))
133   dispatch-spel)
134
135 ;Maakt het ADT van het spel, moet wel nog opgeroepen worden.
136 (define spel (maak-adt-spel))
137
138 ;Entry point om alles te laden
139 ((spel 'start))
```

Spel.rkt

5.10 Teken-adt.rkt

```
1  #lang racket
2
3  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
4  ;;;;;;;;;    Alexandre Kahn     ;;;;;;;;;
5  ;;;;;;;;;       500067          ;;;;;;;;;
6  ;;;;;;;;;      1e BA CW         ;;;;;;;;;
7  ;;;;;;;;; alexkahn@vub.ac.be   ;;;;;;;;;
8  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
9
10 (require "Graphics.rkt")
```

```
11 (require "helpers.rkt")
12 (require "Abstracties.rkt")
13 ;Debug functies komen uit de helpers.rkt file die met het
       snakeproject meegegeven zijn. Deze werden tijdens het schrijven
        van de code veel gebruikt om zaken mee te testen.
14 ;zodat ander files hieraan kunnen
15
16 (provide maak-adt-teken)
17 ;Zorgt evoor dat het teken-adt vanuit de spelfile aangesproken kan
       worden.
18
19
20 ;;;;;;;;;;;;;;;;;
21 ;;Het teken-adt;;
22 ;;;;;;;;;;;;;;;;;
23
24 ;Maakt het teken-adt met een venster waarop getekend kan worden
25 (define (maak-adt-teken titel h-pixels v-pixels)
26   (define venster (make-window v-pixels h-pixels titel))
27   ((venster 'set-background!) "black")
28
29   ;;;;;;;;;;
30   ;;CONFIG;;
31   ;;;;;;;;;;
32
33
34   ;;;;;;;;;;;;;;;;;;;;;;;;
35   ;; Config voor alien ;;
36   ;;;;;;;;;;;;;;;;;;;;;;;;
37
38   ;;; Maakt een alien-laag aan. Hierop zullen de alien(-tiles)
       getekend worden.
39   (define alien-laag (venster 'make-layer))
40
41   ;;; Beginnen met een lege lijst aan tiles, zodat we er in de
       globale omgeving aankunnen
42   (define vloot-tiles '())
43
44   ;;; Functie om alien-tiles aan de vloot toe te voegen.
45   (define (voeg-alien-toe! alien-adt)
46     (define kleurfoto '())
47     (cond ((eq? 'geel (alien-adt 'kleur))
48            (set! kleurfoto (make-bitmap-tile "monster-geel.jpg")))
       ;moet met set! anders geen actie gedaan
49           ((eq? 'groen (alien-adt 'kleur))
50            (set! kleurfoto (make-bitmap-tile "monster-groen.jpg")))
       )
51     (set! vloot-tiles (cons (cons alien-adt kleurfoto) vloot-tiles)
       )
52     ((alien-laag 'add-drawable) kleurfoto))
53
54   ;;; Functie om de correcte alien of kogel uit de lijst te halen.
55   (define (neem-iets iets-adt uit-tiles)
56     (cdr (assoc iets-adt uit-tiles)))
57
58   ;;; Functie om een alien te verwijderen. Zowel uit de lijst van
       de vloot-tiles als effectief van het scherm (remove-drawable)
```

```
59  (define (verwijder-alien! alien-adt)
60
61    ;(debug "delete-alien")
62    (let* ((alien-tile (neem-iets alien-adt vloot-tiles)))
63      ((alien-laag 'remove-drawable) alien-tile))
64    (set! vloot-tiles (remove alien-adt vloot-tiles (lambda (r e) (
      eq? (car e) e)))))
65  (define (reset-vloot!)
66    (set! vloot-tiles '()))
67
68
69  ;;;;;;;;;;;;;;;;;;;;;
70  ;;Config voor schip;;
71  ;;;;;;;;;;;;;;;;;;;;;
72
73  ;;;De laag waarop het schip getekend zal worden
74  (define schip-laag (venster 'make-layer))
75
76  ;;; In dit geval is er slechts een tile dus die kan hardcoded
      staan
77  (define schip-tile (make-bitmap-tile "schip.jpg" "schip-mask.jpg"
      ))
78
79  ;;; Zorgt ervoor dat het schip getekend kan worden. Moet niet te
      deleten zijn.
80  ((schip-laag 'add-drawable) schip-tile)
81
82  ;verwijderen van het schip
83  (define (verwijder-schip!)
84    ((schip-laag 'remove-drawable) schip-tile))
85
86
87  ;;;;;;;;;;;;;;;;;;;;;
88  ;;Config voor kogel;;
89  ;;;;;;;;;;;;;;;;;;;;;
90  ; Laag waarop de kogel getekend zal worden
91  (define kogel-laag (venster 'make-layer))
92
93  ; Een lege lijst voor de kogel, zo zal deze pas op het scherm
      komen nadat men hem nodig heeft.
94  (define kogel-tiles '())
95
96  ; Functie die de kogel aanmaakt en in assoc lijst steekt
97  (define (maak-kogel! kogel-adt)
98    (let ((kogel-obj (make-bitmap-tile "kogel.jpg")))
99      (set! kogel-tiles (cons (cons kogel-adt kogel-obj)
      kogel-tiles))
100
101     (map (lambda (kogel-tile) ((kogel-laag 'add-drawable) (cdr
      kogel-tile))) kogel-tiles)
102     ))
103
104 ; Functie om een getekende kogel te verwijderen
105 (define (verwijder-kogel! kogel-adt)
106   (let* ((kogel-obj (neem-iets kogel-adt kogel-tiles)))
107     ((kogel-laag 'remove-drawable) kogel-obj))
108   (set! kogel-tiles (remove kogel-adt kogel-tiles (lambda (r e) (
```

```
      eq? (car e) e)))))
109
110
111   ;;;;;;;;;;;;;;;;;;;;
112   ;;Config voor menu;;
113   ;;;;;;;;;;;;;;;;;;;;
114   ; Laag waarop het menu getekend zal worden
115   (define menu-laag (venster 'make-layer))
116   (define menu-tile (make-bitmap-tile "Menu.png"))
117
118   ;Functie om het menu te verwijderen
119   (define (verwijder-menu!)
120     ((rotator-laag 'remove-drawable) rotator-tile)
121     ((menu-laag 'remove-drawable) menu-tile))
122
123   ;Functie om alles wat weggehaald wordt bij het aanmaken van een
        menu te hertekenen.
124   (define (herteken-alles! alienlijst)
125     ((schip-laag 'add-drawable) schip-tile)
126     (define (redraw-aliens alienlijst)
127       (if (not (null? alienlijst))
128           (begin
129             (cond ((>= ((car alienlijst) 'levens) 1) ;Zorgt dat
        dode aliens niet hertekend worden
130                    (voeg-alien-toe! (car alienlijst))))
131             (redraw-aliens (cdr alienlijst))) ;zorgt voor de
        iteratie
132           'ok))
133     (redraw-aliens alienlijst))
134
135   ;Functie om alles weg te halen bij het aanmaken van het menu.
        Zorgt ervoor dat dit ook al spelende kan gebeuren.
136   (define (verwijder-alles! alienlijst kogellijst)
137     (define (verwijder-alle-kogels kogellijst) ;delete alle kogels
138       (if (not (null? kogellijst))
139           (begin
140             (verwijder-kogel! (car kogellijst))
141             (verwijder-alle-kogels (cdr kogellijst)))
142           'ok))
143     (define (verwijder-alle-aliens alienlijst) ;delete alle aliens
144       (if (not (null? alienlijst))
145           (begin
146             (verwijder-alien! (car alienlijst))
147             (verwijder-alle-aliens (cdr alienlijst)))
148           'ok))
149     (verwijder-alle-aliens alienlijst)
150     (verwijder-alle-kogels kogellijst)
151     (verwijder-schip!)
152     ((menu-laag 'add-drawable) menu-tile)) ;tekent het menu
153
154   ;;;;;;;;;;;;;;;;;;;;;;
155   ;;Config voor rotator;;
156   ;;;;;;;;;;;;;;;;;;;;;;
157
158   ; Laag waarop de rotator getekend zal worden
159   (define rotator-laag (venster 'make-layer))
160   (define rotator-tile (make-bitmap-tile "rotator.jpg"))
```

```
161  (define (maak-rotator)
162    ((rotator-laag 'add-drawable) rotator-tile))
163  (define (verwijder-rotator!)
164    ((rotator-laag 'remove-drawable) rotator-tile))
165
166  ;;;;;;;;;;;;;;;;;;;;
167  ;;Config voor howto;;
168  ;;;;;;;;;;;;;;;;;;;;
169
170  ; Laag waarop het how-to scherm getekend zal worden
171  (define how-to-laag (venster 'make-layer))
172  (define how-to-tile '())
173
174  (define (maak-how-to)
175    (let ((tile (make-bitmap-tile "HowTo.jpg")))
176      (set! how-to-tile tile)
177      ((how-to-laag 'add-drawable) tile)))
178  (define (verwijder-how-to!)
179    ((how-to-laag 'remove-drawable) how-to-tile)
180    (set! how-to-tile '()))
181
182  ;;;;;;;;;;;;;;;;;;
183  ;;TEKEN FUNCTIES;;
184  ;;;;;;;;;;;;;;;;;;
185
186  ;;;;;;;;;
187  ;;Schip;;
188  ;;;;;;;;;
189
190  (define (teken-schip! schip-adt)
191    (let* ((schip-x (* h-pixels (schip-adt 'x)))
192           (schip-y (- v-pixels px-schip-hoogte))
193           (absolute-x (centraliseer schip-x 'schip)))
194      ((schip-tile 'set-x!) absolute-x)
195      ((schip-tile 'set-y!) schip-y)
196      ;(debug "schip tekenen" schip-x " " schip-y)
197      ))
198
199  ;;;;;;;;;
200  ;;Alien;;
201  ;;;;;;;;;
202
203  (define (teken-alien! alien-adt)
204    (let* ((alien-x (* h-pixels (alien-adt 'x)))
205           (alien-y (* v-pixels   (alien-adt 'y)))
206           (alien-tile (neem-iets alien-adt vloot-tiles))
207           (absolute-x (centraliseer alien-x 'alien)))
208      ((alien-tile 'set-x!) absolute-x)
209      ((alien-tile 'set-y!) alien-y)
210      ;(debug "Alien tekenen" alien-x " " alien-y)
211      ))
212
213  ;;;;;;;;;
214  ;;Kogel;;
215  ;;;;;;;;;
216
217  (define (teken-kogel! kogel-adt)
```

```
218     (let* ((kogel-x (* h-pixels (kogel-adt 'x)))
219            (kogel-y (* v-pixels (kogel-adt 'y)))
220            (kogel-tile (neem-iets kogel-adt kogel-tiles))
221            (absolute-x (centraliseer kogel-x 'kogel)))
222       ((kogel-tile 'set-x!) absolute-x)
223       ((kogel-tile 'set-y!) kogel-y)
224       ;(debug "kogel tekenen" kogel-x " " kogel-y)
225       ))
226
227   ;;;;;;;;
228   ;;Menu;;
229   ;;;;;;;;
230
231   (define (teken-menu! menu-adt)
232     (let* ((start-x (* h-pixels (menu-adt 'x)))
233            (start-y (* v-pixels (menu-adt 'y)))
234            (absolute-x (centraliseer start-x 'start)))
235       ((menu-tile 'set-x!) absolute-x)
236       ((menu-tile 'set-y!)  start-y)
237       ))
238
239   ;;;;;;;;;;;
240   ;;Rotator;;
241   ;;;;;;;;;;;
242
243   (define (teken-rotator! rotator-adt)
244     (let* ((rotator-x (* h-pixels (rotator-adt 'x)))
245            (rotator-y (* v-pixels (rotator-adt 'y)))
246            (absolute-x (centraliseer rotator-x 'rotator)))
247       ((rotator-tile 'set-x!) absolute-x)
248       ((rotator-tile 'set-y!)  rotator-y)
249       ))
250
251   ;;;;;;;;;;
252   ;;How-To;;
253   ;;;;;;;;;;
254   (define (teken-how-to! how-to-adt)
255     (if (not (null? how-to-tile))
256         (begin
257           (let* ((how-to-x (* h-pixels (how-to-adt 'x)))
258                  (how-to-y (* v-pixels (how-to-adt 'y)))
259                  (absolute-x (centraliseer how-to-x 'how-to)))
260             ((how-to-tile 'set-x!) how-to-x)
261             ((how-to-tile 'set-y!)  how-to-y)
262             )
263           )
264         'ok))
265
266   ;;;;;;;;;;;;;;;;;;;;;
267   ;; Spellus functies;;
268   ;;;;;;;;;;;;;;;;;;;;;
269
270   (define (set-spel-lus-functie! fun)
271     ((venster 'set-update-callback!) fun))
272
273   (define (set-toets-functie! fun)
274     ((venster 'set-key-callback!) fun))
```

```
275
276
277   ;;;;;;;;;;;;;;
278   ;; Dispatch ;;
279   ;;;;;;;;;;;;;;
280
281   (define (dispatch-teken-adt msg)
282     (cond ((eq? msg 'set-toets-functie!) set-toets-functie!)
283           ((eq? msg 'set-spel-lus-functie!) set-spel-lus-functie!)
284           ;; Teken functies.
285           ((eq? msg 'teken-schip!) teken-schip!)
286           ((eq? msg 'teken-alien!) teken-alien!)
287           ((eq? msg 'teken-kogel!) teken-kogel!)
288           ((eq? msg 'teken-menu!) teken-menu!)
289           ((eq? msg 'teken-rotator!) teken-rotator!)
290           ((eq? msg 'teken-how-to!) teken-how-to!)
291           ;; Delete en maak functies
292           ((eq? msg 'herteken-alles!) herteken-alles!)
293           ((eq? msg 'nieuwe-alien!) voeg-alien-toe!)
294           ((eq? msg 'nieuwe-kogel!) maak-kogel!)
295           ((eq? msg 'maak-rotator) maak-rotator)
296           ((eq? msg 'maak-how-to) maak-how-to)
297           ((eq? msg 'delete-alien!) verwijder-alien!)
298           ((eq? msg 'delete-kogel!) verwijder-kogel!)
299           ((eq? msg 'delete-schip!) verwijder-schip!)
300           ((eq? msg 'delete-menu!) verwijder-menu!)
301           ((eq? msg 'delete-rotator!) verwijder-rotator!)
302           ((eq? msg 'delete-how-to!) verwijder-how-to!)
303           ((eq? msg 'verwijder-alles) verwijder-alles!)
304           (else (display "error, message ") (display msg) (display
      "not understood"))))
305   dispatch-teken-adt)
```

teken–adt.rkt

5.11 Vloot.rkt

```
1  #lang racket
2  (provide maak-vloot-adt)
3  (require "teken-adt.rkt")
4  (require "Abstracties.rkt")
5  (require "Alien.rkt")
6
7
8  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
9  ;;;;;;;;;    Alexandre Kahn     ;;;;;;;;;
10 ;;;;;;;;;       500067          ;;;;;;;;;
11 ;;;;;;;;;      1e BA CW         ;;;;;;;;;
12 ;;;;;;;;; alexkahn@vub.ac.be   ;;;;;;;;;
13 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
14
15
16
17 ;;;;;;;;;;;;;;
18 ;;Vloot ADT;;
19 ;;;;;;;;;;;;;;
20 ;Het vloot-adt zal in een lijst alle aliens bijhouden. Ook zal dit
```

```
    ervoor zorgen dat bepaalde functies op alle aliens toegepast
    kunnen worden.
21
22 (define (maak-vloot-adt aantal teken-adt)
23   ;De (+ aantal 1) is om te zorgen dat ze mooi gecentreerd staan
24   ;(define afstand-tussen-2 0)
25   ;Dit maakt een lege vloot aan in de vorm van een lijst.
26   (define vloot '())
27   (define richting-vloot 'links)
28   (define horizon-snelheid 0.0005)
29   (define vert-snelheid 0.0005)
30   (define afstand-tussen-2 0)
31
32   ;maak-aliens! maakt aliens aan, deze zullen dan een positie en
       dergelijke hebben, met maximaal 10 per rij. en worden in een
       lijst opgeslagen.
33   ;het teken-adt wordt ook aangesproken om daar de nieuwe
       alien-tiles aan te maken
34   (define (maak-aliens! aantal pos-y)
35     (define (maak-rij-aliens! teller pos-x pos-y aantal-op-rij
       kleur)
36       (if (< teller aantal-op-rij)
37           (begin  (let* ((nieuwe-alien (maak-alien-adt pos-x pos-y
       kleur)))
38                     (set! vloot (cons nieuwe-alien vloot))
39                     ((teken-adt 'nieuwe-alien!) nieuwe-alien))
40                   (maak-rij-aliens! (+ teller 1) (- pos-x
       afstand-tussen-2) pos-y aantal-op-rij kleur))
41           'ok))
42     (if (> aantal 10)
43         (begin
44           (set! afstand-tussen-2 (/ venster-breedte 10))
45           (maak-rij-aliens! 0 (- venster-breedte afstand-tussen-2)
       pos-y 10 'groen)
46           (maak-aliens! (- aantal 10) (+ pos-y 0.1)))
47         (begin
48           (set! afstand-tussen-2 (/ venster-breedte aantal))
49           (maak-rij-aliens! 0 (- venster-breedte afstand-tussen-2)
       pos-y aantal 'geel))))
50   ;Zorgt ervoor dat voor elke alien oproept om zichzelf te tekenen
51   (define (teken! teken-adt)
52     (for-each (lambda (alien-adt)
53                 ((alien-adt 'teken!) teken-adt))
54               vloot))
55
56   ;Zorgt ervoor dat men een functie kan uitvoeren op elke alien (
       nodig voor de collision detection).
57   (define (loop-over-vloot functie)
58     (map functie vloot))
59
60   (define (verwijder-alle-aliens)
61     (for-each (lambda (alien-adt)
62                 ((alien-adt 'delete!) teken-adt))
63               vloot))
64
65   (define (verwijder-vloot!)
66     (for-each (lambda (alien-adt)
```

```
67                ((alien-adt 'levens!) 0))
68              vloot)
69    (verwijder-alle-aliens)
70    (maak-aliens! aantal 0))
71
72  ;Zorgt voor de beweging van de aliens
73  ;Gaat na of de aliens met de rand van het scherm botsen en laat
      ze dan terug draaien.
74  (define (beweeg! spel)
75    (for-each (lambda (alien-adt)
76                (let ((y (alien-adt 'y))
77                      (x (alien-adt 'x)))
78                  (if (< y 1)
79                      (begin
80                        ((alien-adt 'y!) (increment y vert-snelheid
      ))
81                        (cond
82                          ((eq? richting-vloot 'links) (if  (< 0 x)
83                                                            ((
      alien-adt 'x!) (decrement x horizon-snelheid))
84                                                            (begin
85                                                              (set!
       richting-vloot 'rechts)
86                                                              ((
      alien-adt 'x!) (increment x horizon-snelheid)))
87                                                            ))
88                          ((eq? richting-vloot 'rechts) (if (> 1 x)
89                                                            ((
      alien-adt 'x!) (increment x horizon-snelheid))
90                                                            (begin
91                                                              (set!
       richting-vloot 'links)
92                                                              ((
      alien-adt 'x!) (increment x horizon-snelheid)))))))
93                        ((spel 'restart!)))))
94              vloot))
95  (maak-aliens! aantal 0)
96
97  ;Dispatch er met dit object ge nterageerd kan worden.
98  (define (dispatch msg)
99    (cond ((eq? msg 'maak-aliens!) (maak-aliens!))
100         ((eq? msg 'teken!) teken!)
101         ((eq? msg 'loop-over-vloot) loop-over-vloot)
102         ((eq? msg 'beweeg!) beweeg!)
103         ((eq? msg 'verwijder-vloot!) verwijder-vloot!)
104         ((eq? msg 'vloot) vloot)
105         ))
106   dispatch)
```

vloot.rkt

6 Bibliografie

Sussman G.J. Abelson, A. *Structure and Interpretation of Computer Programs*, The MIT Press, 1996.