



Programeerproject 1

Fase 2: Verslag "Fire Ant"

Mikhael Swusten



Inhoudstafel

1.	Inleiding	2
1.	ADT's	3
1.1	Positie ADT	3
1.2	doolhof ADT	4
1.3	Voedsel ADT	5
1.4	Power-up ADT	5
1.5	Puzzel ADT	6
1.6	Vuurmier ADT	6
1.7	Schorpioen ADT	7
1.8	Level ADT	8
1.9	Teken ADT	9
1.10	Score ADT	10
1.11	Spel ADT	11
2.	Afhankelijkheidsdiagram	12
3.	Planning	14
4.	Vergelijking van Afhankelijkheidsdiagram en Planning van voorstudie	16
4.1	Afhankelijkheidsdiagram	16
4.2	Planning	16
5.	Uitleg opstarten van het spel	16

1. Inleiding

De inhoud van dit document zal de implementatie omvatten van het spel Fire Ant. Dit zal gerealiseerd worden in de programmeertaal Scheme met behulp van een "gameloop". Hiervoor zullen er een aantal ADT's nodig zijn om het spel Fire Ant te laten werken. Deze worden later verder beschreven wat het exacte nut van elk type ADT bevat.

1.ADT's

1.1 Positie ADT

Het Positie ADT houdt de positie van een bepaald spelobject bij met behulp van een x- coördinaat en een y-coördinaat. Dit vormt de basis waar het vuurmier ADT, schorpioen ADT, voedsel ADT, power-up ADT en puzzel ADT verder op bouwt.

Naam	Signatuur
maak-positie-adt	(number number -> positie)
x	($\emptyset$ -> number)
y	($\emptyset$ -> number)
x!	(number-> $\emptyset$)
y!	(number-> $\emptyset$)
vergelijk?	(positie -> boolean)

- De maak-positie-adt operatie maakt met 2 gegeven nummers een coördinaat aan in de spelwereld.
- De x en y operaties geven respectievelijk de x- en y-coördinaten aan.
- De x! en y! vervangen al bestaande x- en y-coördinaten met nieuwe coördinaten.
- De vergelijk? operatie vergelijkt 2 positie-objecten, kijkt of deze gelijk zijn en geeft een #t of #f terug.

1.2 Doolhof ADT

Het Doolhof ADT geeft matrices weer bij oproep die als doolhof functioneert. De vuurmier, random schorpioen, slot-puzzel en steen-puzzel maakt hier gebruik van.

Naam	Signatuur
maak-doolhof-adt	($\emptyset$ -> doolhof)
level-1-doolhof	vector
level-2-doolhof	vector
level-3-doolhof	vector
einde-doolhof	vector
ij?	(vector number number -> any)
ij!	(vector number number any-> $\emptyset$)

- De maak-doolhof-adt operatie maakt alle doolhoven aan en de operaties die nodig zullen zijn om te kijken op en het aanpassen van een doolhof.
- Level-1-doolhof, level-2-doolhof, level-3-doolhof en einde-doolhof zijn de matrices die de doolhoven voorstellen.
- De ij? procedure gaat kijken aan de hand van een matrix en 2 nummers wat er op die plaats staat.
- De ij! procedure gaat aan de hand van een matrix en 2 nummers op die plaats iets aanpassen.

1.3 Voedsel ADT

Het Voedsel ADT zorgt ervoor dat er voedsel-objecten in de spelwereld komen. Bij aanraking van een vuurmier zal deze verdwijnen en een bepaalde score geven afhankelijk van het soort voedsel.

Naam	Signatuur
maak-voedsel-adt	(positie -> voedsel)
positie	($\emptyset$ -> positie)
opgepakt?	boolean
opgepakt!	(boolean -> $\emptyset$)
vergelijk-voedsel?	(positie -> boolean)

- De maak-voedsel-adt operatie maakt een voedsel object aan.
- De positie operatie geeft de positie van een voedsel weer.
- Opgepakt? is een boolean die dient als status om te zien of een voedsel object opgepakt is of niet.
- Opgepakt! is een operatie die de status aanpast.
- Vergelijk-voedsel? is een operatie die zal kijken of een vuurmier dezelfde positie heeft als het voedsel-object.

1.4 Power-up ADT

Het Power-up ADT maakt een power-up object aan in het doolhof die door de speler kan worden opgepakt. De power-ups die ik heb gekozen zijn:

- Speciale voeding met een effect op het aantal levens, dit wordt in mijn spel voorgesteld als een gouden-ei.
- Een schild dat de speler tijdelijk beschermt tegen schorpioenen, dit wordt in mijn spel voorgesteld als een chilipeper.

Naam	Signatuur
maak-powerup-adt	(positie -> power-up)
positie	($\emptyset$ -> positie)

- De maak-powerup-adt operatie maakt een powerup object aan.
- De positie operatie geeft de positie van een powerup weer.

1.5 Puzzel ADT

Het Puzzel ADT maakt een puzzel object aan in het doolhof die door de speler kan worden opgepakt. Ik heb ervoor gekozen om de sleutel puzzel en dynamiet puzzel te implementeren.

Naam	Signatuur
maak-powerup-adt	(positie -> power-up
positie	($\emptyset$ -> positie)
slot-gesloten?	boolean
slot-open!	($\emptyset$ -> $\emptyset$)

- De maak-puzzel-adt operatie maakt een puzzel object aan.
- De positie operatie geeft de positie van een puzzel weer.
- De slot-gesloten? is de status van een slot, deze staat in het begin op #f en slot-open! verandert deze op #t.

1.6 Vuurmier ADT

Het Vuurmier ADT zal een vuurmier aanmaken die door de speler bestuurbaar is om door het doolhof te navigeren.

Naam	Signatuur
maak-vuurmier-adt	(positie -> vuurmier)
positie	($\emptyset$ -> positie)

- De maak-vuurmier operatie maakt een vuurmier aan de hand van een (start)positie.
- De positie operatie geeft de huidige positie weer wanneer dit wordt opgeroepen.

1.7 Schorpioen ADT

Het Schorpioen ADT maakt een schorpioen aan die op een al voorgeprogrammeerd traject zal wandelen. Wanneer een vuurmier de schorpioenen aanraakt dan zal de vuurmier terug naar het begin gestuurd worden. Er zijn ook schorpioen die willekeurig bewegen.

Naam	Signatuur
maak-schorpioen-adt	($\emptyset$ -> schorpioen)
positie	($\emptyset$ -> positie)
schorpioen-richting	symbol
schorpioen-kijk-richting	symbol
set-richting!	(symbol -> $\emptyset$)
set-kijk-richting!	(symbol -> $\emptyset$)
start-bewegen	boolean
start-bewegen!	(boolean -> $\emptyset$)
ind	number
ind!	(number -> $\emptyset$)
stap	number
stap!	(number -> $\emptyset$)
schorpioen-tijd	number
schorpioen-tijd!	(number -> $\emptyset$)
soort	symbol
soort!	(symbol -> $\emptyset$)
pad	list
split	(list -> list)
aftrekken-el	(list -> list)
vergelijk-schorpioen?	(positie -> boolean)

- De maak-schorpioen-adt operatie maakt een schorpioen aan.
- De positie operatie geeft de positie van de schorpioen.
- De schorpioen-richting en schorpioen-kijk-richting geeft de richting en kijk-richting van een schorpioen, de set-richting! en set-kijk-richting! past deze aan.
- De start-bewegen, ind en stap zijn waardes die gebruikt worden voor het berekenen van het vast pad van een schorpioen, de respectievelijke mutator versies hiervan passen deze aan.
- Schorpioen-tijd en schorpioen-tijd! geeft en past de schorpioen-tijd aan. Deze worden gebruikt om om de zoveel tijd een stap te zetten.
- De soort en soort! geeft de soort van een schorpioen weer. Deze kan 'normaal' of 'random' zijn.

- Pad is een lijst van lijsten. Deze zijn vaste paden die bestaan uit puntenkoppels.
- De split procedure krijgt een pad en split deze in 2 lijsten, een lijst van alle x-en en een lijst van alle y-en.
- De aftrekken-el is een procedure dat een lijst binnen krijgt en voor elk element in die lijst zichzelf zal aftrekken met het volgende element. Dit wordt gebruikt bij de x-en lijst en y-en lijst. (bv. (8 8 6 6) -> (0 2 0 -2))
- Vergelijk-schorpioen? is een operatie die zal kijken of een vuurmier dezelfde positie heeft als een schorpioen.

1.8 Level ADT

Het Level ADT omvat alle andere ADT's, beweeg procedures ervan en interacties.

Naam	Signatuur
maak-level-adt	(number number -> level)
level-doolhof	vector
update!	(score teken -> $\emptyset$)
toets!	(any -> $\emptyset$)
vuurmier-adt	vuurmier
schorpioen-adt	list
blad	list
zaad	list
ei	list
sleutel-adt	puzzel
slot-adt	puzzel
dynamiet-adt	puzzel
steen-adt	puzzel
chillie-adt	powerup
goudenei-adt	powerup

- De maak-level-adt operatie maakt een level aan met alle spelobjecten en spel functionaliteiten.
- Level-doolhof geeft een matrix weer dat zal dienen als doolhof.
- De update! procedure zal al de andere procedures in gang laten schieten.
- De toets! procedure zorgt ervoor dat je de vuurmier kan bewegen.

- Vuurmier-adt stelt een vuurmier object voor.
- Schorpioen-adt, blad, zaad en ei zijn objecten die in een lijst zitten.
- Sleutel-adt, slot-adt, dynamiet-adt en steen-adt zijn objecten die worden gemaakt met het puzzel adt.
- Chili-adt en goudenei-adt zijn objecten die worden gemaakt met het powerup adt.

1.9 Teken ADT

Het Teken ADT zorgt ervoor dat al de spel elementen tevoorschijn komen op je scherm.

Naam	Signatuur
maak-teken-adt	(number number -> teken)
... !	($\emptyset$ -> $\emptyset$)
teken-spel!	(score level -> $\emptyset$)
teken-score!	(score -> $\emptyset$)
reset-explosie	($\emptyset$ -> $\emptyset$)
verwijder-level!	($\emptyset$ -> $\emptyset$)
restart!	($\emptyset$ -> $\emptyset$)
herteken!	($\emptyset$ -> $\emptyset$)
set-spel-lus-functie!	((number -> $\emptyset$) -> $\emptyset$)
set-toets-functie!	((symbol any -> $\emptyset$ -> $\emptyset$)

- De maak-teken-adt tekent het venster waarin het spel gebeurt.
- De ... ! dit zijn destructieve operaties die tiles doen verdwijnen. (bv chili!)
- De teken-spel! procedure is om het spel te tekenen.
- De teken-score! procedure tekent de score.
- De reset-explosie procedure reset de explosie tile sequence.
- De verwijder-level! veegt de level lagen schoon.
- De restart! procedure veegt alle lagen schoon.
- De herteken! procedure hertekent het spel.
- De set-spel-lus-functie! zorgt ervoor dat iets kan ververst (heel de tijd opnieuw geëvalueerd) worden.
- De set-toets-functie! procedure zorgt ervoor wanneer een toets wordt ingedrukt, deze gelezen en meegegeven kan worden.

1.10 Score ADT

Het Score ADT maakt een score object aan waarin de levens, score en highscore bewaart zullen worden. Deze zullen visueel worden gemaakt door het teken ADT als een scorebord.

Naam	Signatuur
maak-score-adt	($\emptyset$ -> score)
score	number
verander-score!	(number -> $\emptyset$)
reset-score!	($\emptyset$ -> $\emptyset$)
reset-alle-score!	($\emptyset$ -> $\emptyset$)
highscore	number
update-highscore	($\emptyset$ -> $\emptyset$)
levens	number
reset-levens!	($\emptyset$ -> $\emptyset$)
levens-omhoog!	($\emptyset$ -> $\emptyset$)
levens-omlaag!	($\emptyset$ -> $\emptyset$)
update!	($\emptyset$ -> $\emptyset$)

- De maak-score-adt operatie maakt een score object aan.
- De score en highscore zijn waardes die beginnen op nul.
- De verander-score! Voegt een bepaalde hoeveelheid score toe aan de huidige score. Reset-score! reset de huidige score en reset-alle score! reset zowel de highscore als de normale score.
- De update-highscore vernieuwd de highscore als de normale score hoger wordt dan de highscore.
- De levens is een waarde die begint bij 3.
- De levens-omhoog! en levens-omlaag! procedures verhogen en verlagen van het aantal levens.
- Update! wordt gebruikt om de highscore te updaten.

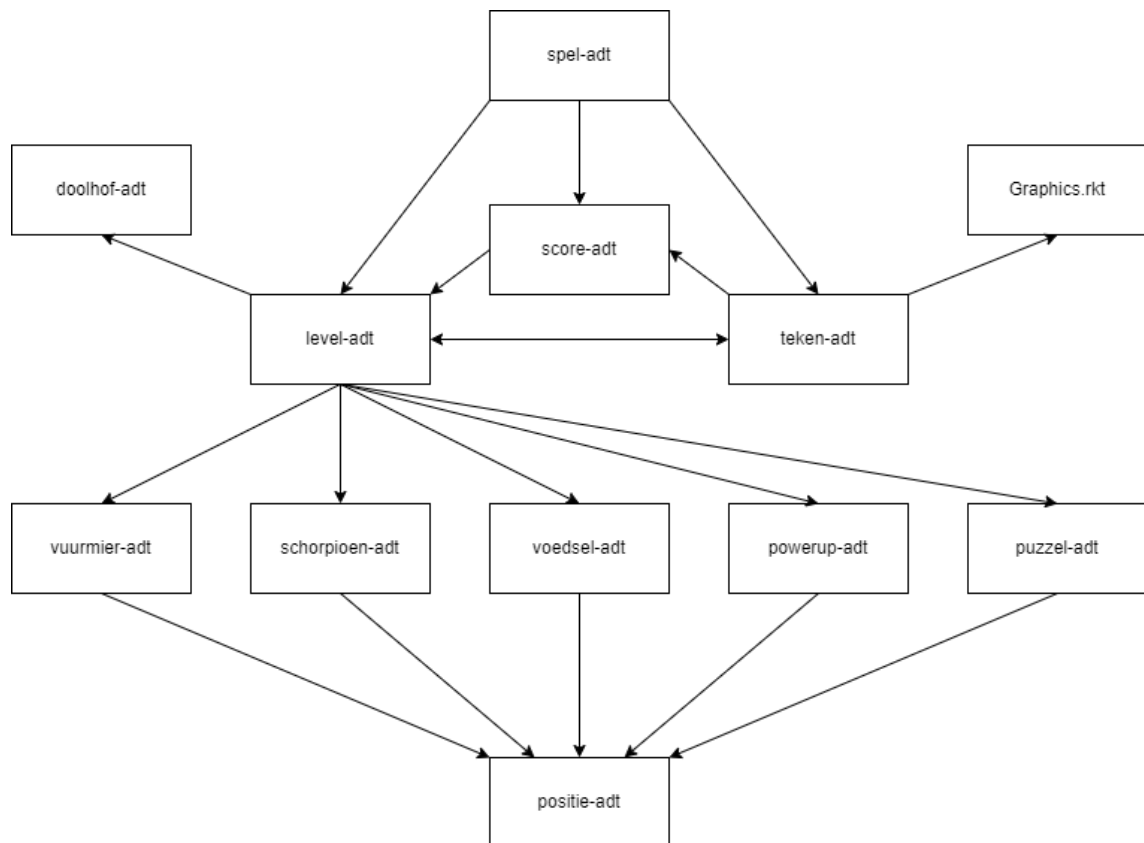
1.11 Spel ADT

Het Spel ADT omvat alle functionaliteiten die nodig zijn om het spel te laten werken. Dit ADT laat de spellogica en tekenlogica samenwerken.

Naam	Signatuur
maak-spel-adt	$(\emptyset \rightarrow \text{spel-adt})$
start	$(\emptyset \rightarrow \emptyset)$

- De maak-spel-adt operatie maakt een spel aan door maak-level-adt en maak-teken-adt op te roepen.
- De start operatie zorgt ervoor dat je effectief het spel kan starten.

2. Afhankelijkheidsdiagram



Het vuurmier ADT, schorpioen ADT, voedsel ADT, powerup ADT en Puzzel ADT zijn afhankelijk van het positie ADT want elk object heeft een positie nodig.

Het level ADT is afhankelijk van het doolhof ADT, vuurmier ADT, schorpioen ADT, voedsel ADT, powerup ADT en Puzzel ADT want dit zijn de spelobjecten die met elkaar zullen interageren.

Het level ADT en teken ADT zijn van elkaar afhankelijk want er worden oproepen vanuit beiden gedaan die nodig zijn om het spel te laten werken. Zoals het teken ADT is afhankelijk van het level ADT om de spelobjecten te tekenen. Het level ADT is afhankelijk van teken ADT voor het verwijderen van afbeeldingen van objecten.

Het score ADT is afhankelijk van het level ADT want de score wordt bepaald bij de interacties in het level ADT.

Het teken ADT is afhankelijk van de Graphics bibliotheek voor het tekenen van de spelobjecten. Het teken ADT is ook afhankelijk van het score ADT omdat deze in een scorebord getekend en telkens vernieuwd zal worden.

Het spel ADT is afhankelijk van het score ADT, level ADT en teken ADT want deze worden alle drie aangemaakt en bij het starten van het spel worden deze opgeroepen.

3. Planning

Planning van Voorstudie 2	
Week	Actie
Week 26	Voorstudie indienen
Week 27	WPO massaprogrammeren
Week 27	Aanpassen van bestaande code en implementatie van meerdere voedsel objecten en een beetje beginnen aan scorebord
Week 28	Tussentijds indienmoment
Week 29	Lentevakantie
Week 29	Implementatie power-up ADT
Week 30	Implementatie puzzel ADT
Week 31	Implementatie van scorebord
Week 32	Verder werken aan wat nog niet klaar is.
Week 33	Spel uitproberen voor bugs
Week 34	Tussentijds indienmoment en WPO massaprogrammeren
Week 35	Verder werken aan wat nog niet klaar is.
Week 37	Indienen code en verslag

Planning van Fase2	
Week	Actie
Week 26	Voorstudie indienen
Week 27	WPO massaprogrammeren
Week 27	/
Week 28	/
Week 29	/
Week 29	/
Week 30	Code aangepast en andere ADT's aangemaakt en de bijhorende interacties.
Week 31	/
Week 32	Score ADT geïmplementeerd en verder bugs gefixt.
Week 33	Level overgang geïmplementeerd
Week 34	Code aanpassen die niet werkten door level overgang
Week 35	Verslag maken en commentaar bij code voegen
Week 37	Indienen code en verslag

4. Vergelijking van Afhankelijkheidsdiagram en Planning van voorstudie

4.1 Afhankelijkheidsdiagram

Dit afhankelijkheidsdiagram verschilt veel met het afhankelijkheidsdiagram van de voorstudie. In dit afhankelijkheidsdiagram en het afhankelijkheidsdiagram van de voorstudie zijn de vuurmier ADT, schorpioen ADT, voedsel ADT, power-up ADT en puzzel ADT afhankelijk van het positie ADT. In het afhankelijkheidsdiagram van dit verslag is het level ADT afhankelijk van het vuurmier ADT, schorpioen ADT, voedsel ADT, power-up ADT, puzzel ADT en doolhof ADT wat niet zo was bij het afhankelijkheidsdiagram van de voorstudie. Ook is het level ADT en teken ADT afhankelijk van elkaar en zit het score ADT er ook nog tussen. Ten slotte is het spel ADT dan nog afhankelijk van het score ADT, level ADT en teken ADT omdat ze daar alle drie worden aangemaakt en worden opgeroepen.

4.2 Planning

Het volgen van mijn planning van de voorstudie was niet gelukt. Van week 26 tot week 29 heb ik niet gewerkt aan het programeer project. Pas week 30 ben ik effectief begonnen. Ik had vooral al bestaande code aangepast zodat die wat efficiënter werkten en minder plaats innamen. Ik heb ook verschillende ADT's geïmplementeerd zoals power-up en puzzel. Dit ging vele vlotter dan verwacht, zeker ten opzichte van de vorige fase. Ook heb ik het ei ADT verwacht met voedsel ADT sinds er niet veel nut was om aparte ADT's te maken. Week 32 heb ik het score ADT geïmplementeerd en bugs gefixt. Op week 33 heb ik het level overgang geïmplementeerd wat heel wat problemen met zich meebracht. In week 34 heb ik dit dan moeten oplossen. Ik ben een beetje laat begonnen maar alles is min of meer gelukt op tijd.

5. Uitleg opstarten van het spel

Het spel is makkelijk op te starten. Je opent het "spel.rkt" bestand en drukt op "run". Het spel zal automatisch opstarten en zo kun je beginnen met spelen.