VRIJE UNIVERSITEIT BRUSSEL

VOORSTUDIE FASE I

PROGRAMMEERPROJECT I

SAMUEL VAN DE VEN
SAMUEL.ROELAND.D.VAN.DE.VEN@VUB.BE
2021-2022

VAK: PROGRAMMEERPROJECT I
FACULTEIT WETENSCHAPPEN & BIO-INGENIEURSWETENSCHAPPEN

Inhoudstafel

1 Inleiding

In deze inleiding van mijn voorstudie van het programmeerproject bespreek ik kort wat de taak inhoudt en hoe het verslag is ingedeeld. De gegeven opdracht: Fire Ant programmeren in de Scheme programmeertaal. In sectie 2: Functionele veristen, geef ik een beschrijving van de functionele veriesten van het spel. Vervolgens zal ik in sectie 3 de verschillende Abstracte Data Types die ik zal gebruiken uitleggen. In sectie 4 geef ik een uitleg hoe de verschillende ADT's met elkaar interageren. Ten slotte geef ik in sectie 5 mijn planning.

2 Functionele Vereisten

Het spel Fire Ant speelt zich af in een doolhof. Het doel van het spel is de mierenkoniging te bevrijden. Elk level verschilt in moeilijkheidsgraad en wanneer alle levels zijn voltooid is het spel ten einde. In elk level zullen er schorpioenen aanwezig zijn die het leven van de mier moeilijk maken. Er bevinden zich ook stukken eten en powerups die de mier helpen zijn einddoel te bereiken.

3 Abstracte Data Types

3.1 ADT position

Het abstract data type positie zal gebruikt worden door andere adt's om een positie bij te houden. De constructor make-adt-position zal een nieuwe positie instantie maken. De accessor x zal de x-waarde van het object terug geven. De accessor y zal de y-waarde van het object terug geven. De accessor compare? zal twee posities vergelijken en een boolean als resultaat teruggeven. De mutator x! zal de x-waarde van een object aanpassen. De mutator y! zal de y-waarde van een object aanpassen.

Constructor:	
Naam	**Signatuur**
make-adt-position	(number, number → *position*)

Accessor:	
Naam	**Signatuur**
x	(/ → *number*)
y	(/ → *number*)
compare?	(position → *boolean*)

Mutator:	
Naam	**Signatuur**
x!	(number → /)
y!	(number → /)

3.2 ADT game

De constructor make-adt-game zal een instantie van het game adt aanmaken. De mutator start! zorgt ervoor dat het spel wordt gestart. Elk adt zal een voor een worden opgeroepen.

Constructor:	
Naam	**Signatuur**
make-adt-game	$(/ \rightarrow \mathit{Game})$

Mutator:	
Naam	**Signatuur**
start!	$(/ \rightarrow /)$

3.3 ADT ant

De constructor make-adt-ant maakt een nieuwe instantie van het ant adt. De accessor get-position? zal de positie van de ant teruggeven. De mutator move-ant! zal een oproep doen aan het positie adt en de ant van plaats veranderen. De mutator position! zal de huidige positie van de ant veranderen.

Constructor:	
Naam	**Signatuur**
make-adt-ant	$(\text{position} \rightarrow \mathit{ant})$

Accessor:	
Naam	**Signatuur**
get-position?	$(/ \rightarrow \mathit{position})$

Mutator:	
Naam	**Signatuur**
move-ant!	$(/ \rightarrow \mathit{position})$
position!	$(\text{position} \rightarrow /)$

3.4 ADT level

De constructor make-adt-level zal de hoogte en de breedte van het level meekrijgen als argumenten. Elk level zal een nieuw speelveld creeren. update! zal het speelvenster updaten, met een verplaatsing van de ant of een van de scorpions of food dat op het speelvenster wordt getekend.

Constructor:	
Naam	**Signatuur**
make-adt-level	(number, number → *level*)

Accessor:	
Naam	**Signatuur**

Mutator:	
Naam	**Signatuur**
update!	(number → "*updateSpeelveld*")

3.5 ADT draw

De constructor make-adt-draw zal als parameters de pixel hoogte en de pixel breedte van het speelveld meekrijgen. De mutator draw-object! zal indien opgeroepen vanuit een andere functie het gevraagde object met tile op het speelvenster tekenen. **De mutator set-game-loop-function! zal ervoor zorgen dat het spelvenster steeds wordt ververst na elke update**.

Constructor:	
Naam	**Signatuur**
make-adt-draw	(number, number → *draw*)

Accessor:	
Naam	**Signatuur**

Mutator:	
Naam	**Signatuur**
draw-object!	(obj, tile → *ant*)
set-game-loop-function!	(fun → /)

3.6 ADT scorpion

De constructor make-adt-scorpion zal een instantie maken van het scorpion adt. De accessor get-position? zal de huidige positie van de scorpion terug geven. De mutator set-position! zal de huidige positie van de scorpion aanpassen. De mutator move-scorpion! zal de scorpion van plaats doen veranderen. De mutator set-init-position! zal de start positie van de scorpion initialiseren. De mutator draw-scorpion! zal een oproep naar het draw adt uitvoeren om de scorpion op het speelvenster te tekenen.

Constructor:	
Naam	**Signatuur**
make-adt-scorpion	(/ → *scorpion*)

Accessor:	
Naam	**Signatuur**
get-position?	(/ → *position*)

Mutator:	
Naam	**Signatuur**
move-scorpion!	(/ → *position*)
set-init-position!	(position → /)
draw-scorpion!	(position → *scorpion*)

3.7 ADT wall

Het wall adt zal beschikken over een lijst met hierin paren van x y coordinaten. Deze paren zullen de posities van een wall voorstellen en hier kan een ant of scorpion niet over wandelen. De constructor make-adt-wall zal een instantie van een wall maken. De accessor get-position? zal de x,y-coordinaten van een wall teruggeven. Hierdoor zal er kunnen worden afgeleid of scorpion of ant hier kan wandelen.

Constructor:	
Naam	**Signatuur**
make-adt-wall	(/ → *scorpion*)

Accessor:	
Naam	**Signatuur**
get-position?	(/ → *position*)

3.8 ADT food

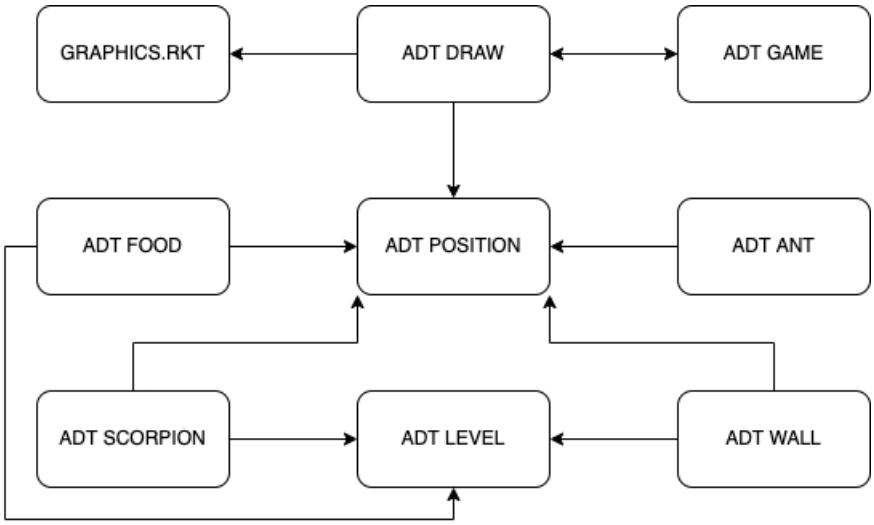
De constructor make-adt-food zal een instantie maken van het food adt. De accessor get-position? zal indien er een eitje in het veld ligt deze positie teruggeven. De accessor get-time? zal de tijd teruggeven van hoelang een eitje al gespawnt is.

Constructor:	
Naam	**Signatuur**
make-adt-food	$(/ \rightarrow scorpion)$

Accessor:	
Naam	**Signatuur**
get-position?	$(/ \rightarrow position)$
get-time?	$(/ \rightarrow number)$

Mutator:	
Naam	**Signatuur**

4 Afhankelijkheidsdiagram



Verklaring afhankelijkheidsdiagram: Het ADT Game zal afhankelijk zijn van het ADT Draw omdat het ADT Draw ervoor zal zorgen dat er een speelvenster verschijnt. Het ADT Draw is afhankelijk van het ADT Game, het ADT Game zal zorgen voor de spellus en dus calls doen om het spelvenster te herladen en dus opnieuw te tekenen. Het ADT Draw is afhankelijk van de graphics.rkt bibliotheek. het adt zal deze nodig hebben voor interne functionaliteit zoals tekenen, tile aanmaken, spelvenster tevoorschijn laten komen. Het ADT Draw is ook afhankelijk van het ADT Position. Indien er iets nieuw op het scherm getekend zal worden zullen de x,y-coordinaten via het position adt verkregen worden. Het ADT Food is afhankelijk van het position adt, indien er een nieuw stuk food wordt aangemaakt zal deze en positie nodig hebben en dus een call doen naar position adt. Het ADT Food is ook afhankelijk van het huidig level waarin de speler zich bevindt. Elk level zal verschillende stukken eten of powerups op het veld zetten. Het ADT Scorpion is afhankelijk van het position adt, het zal een positie nodig hebben. Het ADT Scorpion zal ook afhankelijk zijn van het level adt. Indien de speler zich in een hoger level bevindt zal/zullen de scorpions de moeilijkheidsgraad verhogen. Het ADT Ant zal afhankelijk zijn van het position adt. De ant zal een positie moeten bijhouden. Het ADT Wall zal afhankelijk zijn van het level adt. Elk level zal zijn eigen uniek doolhof hebben. Het ADT Wall zal ook afhankelijk zijn van het position adt. Elke muur heeft een unieke positie.

5 Planning

Planning:				
Task	Deadline	My actions	My deadline	Done?
Indienen voorstudie fase I	**26/11**			X
		Implementeren: adt draw, adt game, adt position,adt level	2/11	
		Implementeren: adt ant, adt scorpion	4/12	
		Implementeren: adt wall	15/12	
		Code nakijken	17/12	
Tussentijds indienmoment	**20/12**			
		Implementeren: adt food	25/12	
		Implementeren: adt score	1/02	
Tussentijds indienmoment	**14/02**			
Indienen code en verslag	**21/02**			
Projectverdediging	**week 23**			
FASE II				
Indienen voorstudie Fase 2	**week 26 (14/03**			
Tussentijds indienmoment	**week 28 (1/04)**			
Tussentijds indienmoment	**week 34 (9/05)**			
Indienen code en verslag	**week 37 (30/05)**			
Projectverdediging	**week 37**			