Programmeerproject 1: FireAnt

Fase 1

Erneste Richard Rwema Rugema
(erneste.richard.rwema.rugema@vub.be)
rolnummer: 0580198

Academiejaar 2021-2022

Inhoudsopgave

1 Intorductie

Dit document beschrijft het implementatie van het spelletjes `Fire Ant` in scheme **r5rs**. Het maakt deel uit van het opleidingsonderdeel "Programmeerproject 1".

1.1 Spelbeschrijving

In dit spel is het de bedoeling dat een speler een *vuurmier* bestuurt, die tracht zijn mierenkoningin te redden van schorpioenenleger. Om dit te voltooien moet de speler een serie van levels, die toenemen in moeilijkheidsgraad, oplossen zodat hij tot bij de koningin kan geraken. Zo een level is een doolhof dat gevuld is met (verschillende soorten) puzzels en schorpioenen die de doorgangen bewaken. Enkel door deze puzzels op te lossen en de schorpioenen te vermijden, kan de speler het level oplossen. Het spel eindigt dan tot dat alle levels opgelost zijn of tot dat de speler al zijn levens kwijt is geraakt.

1.2 Implementatie

Voor het spel te implementeren zal er gewerkt worden in twee fases. In deze document zal enkel de implementatie van het eerste fase besporken worden. Het doel van deze fase zal het implementeren van de elementaire componenten en handelingen zijn.

Om het spel te implementeren zal er gebruikt gemaakt worden van een spellus. Deze spellus moet de invoer van de speler behandelen, de spellogica en de elementen in het spel updaten bij elke iteratie/frame.
Omdat dit enkel het eerste fase is zullen er enkel de volgende elementaire functionaliteiten geïmplementeerd worden:

- Een spelwereld dat een verzameling zal moeten bevatten van doolhoven (levels).
- Een doolhof dat de plaatsing van de muren, in- en uitgang bepaalt.

- Een vuurmier dat kan bestuurt worden op basis van de gebruikersinvoer.
- Een schorpioen die een vaste pad bezit en kan zien of hij de vuurmier (speler) aanraakt.
- Eitjes die een vast positie hebben in het doolhof en kunnen check of ze door de vuurmier (speler) zijn opgeraapt.

Het tekenlogica voor elk van deze elementen zal respectievelijk zelf door het element bijgehouden en geupdate worden. Met andere woorden, elke element dat op het spel weergegeven moet worden zal, naast het spellogica, ook nog een andere element bijhouden dat het grafische visalisatie zal tekenen.

2 ADTs

In deze sectie zullen alle ADTs beschreven dat men verwacht te gebruiken, om de elementaire functionaliteiten in fase 1 van `Fire Ant` te implementeren.

2.1 Game ADT

Het `Game ADT` is het centrale ADT det het spel op te start. Deze is verantwoordelijk om het initialisatie en onthouden van het juiste level en deze te updaten of te herstarten wanneer nodig. De `Game ADT` is ook verantwoordelijk om naar de invoer te luisteren en deze door te geven aan de `Level ADT`. Het luisteren zal gebeuren aan de hand van een spellus.

Naam	Signatuur
newGame	$(list \rightarrow Game)$
start!	$(\varnothing \rightarrow \varnothing)$
¡??¿	¡??¿

Tabel 2.1: Operaties van het `Game ADT`

Het `Game ADT` bevat de volgende operaties:

- **newGame**: Dit is de aanmaak operatie van het `Game ADT`. Voor objecten van `Game ADT` te maken, moet er een lijst meegegeven worden van strings. Deze strings zijn de namen van tekst bestanden die zullen gebruikt worden door het `Level ADT`.

- **start!**: Deze operatie start het spel op. Wanneer deze operatie uitgevoerd wordt zal er een venster aangemaakt worden. Ook de operatie om een nieuw ant en het corresponderende level worden aangeroepen in het spel lus van het spel.
- **¡??¿**: ¡??¿

2.2 Level ADT

Het `Level ADT` is het ADT verantwoordelijk om het level te vormen aan de hand van een tekst bestand en alles op zijn juiste plaats te zetten. Deze is verantwoordelijk om het level, characters en items te initialiseren. ¡??¿

Naam	Signatuur
newLevel	(string Ant → Level)
update!	(Ant → ∅)
isFinished?	(∅ → boolean)
¡??¿	(¡??¿ → ¡??¿)

Tabel 2.2: Operaties van het `Level ADT`

Het `Level ADT` bevat de volgende operaties:

- **newLevel**: Dit is het aanmaak operatie van het `Level ADT`. Voor een object van het `Level ADT` aan te maken moet er een string en een object van het `Ant ADT` meegegeven worden. Het string geeft het naam van het bestand weer dat door het `Maze ADT`, die in het `Level ADT` zit, zal gebruikt worden. Het object van het `Ant ADT` is belangrijk voor het positioneren van de speler in het maze.¡??¿
- **update!**: Deze operatie is verantwoordelijk om alle objecten van het `Character ADT` en het `Item ADT`, die respectievelijk opgeslagen zijn in een lijst, te updaten. Het updaten van elk object gebeurt door hun respectievelijke ***update!*** operatie aan te roepen.
- **isFinished**: Deze operatie checkt of het level gedaan is of niet.
- **¡??¿**: ¡??¿

2.3 Positie ADT

De `Positie ADT` dit is een simpel ADT dat de positie in een level voorstelt. isFinished?

Naam	Signatuur
newPositie	(number number Maze $\rightarrow$ *Positie* $\cup \{\#f\}$)
move!	(number number $\rightarrow \varnothing \cup \{\#f\}$)
eqPosistie?	(Positie $\rightarrow$ boolean)
¡??¿	(¡??¿ $\rightarrow$ ¡??¿)

Tabel 2.3: Operaties van het `Positie ADT`

Het `Positie ADT` bevat de volgende operaties:

- **newPositie**: Deze operatie maakt een nieuw object van het `Positie ADT` aan. Die twee numbers opneemt dat respectievelijk de x- en y-coördinaat zijn in het meegegeven `Maze ADT`.
- **move!**: Verandert het positie van het `Positie ADT` als dat een geldige positie is in het `Maze ADT`.
- **eqPosistie?**: Controleert dat het meegegeven Positie een gelijkwaardige positie heeft. Dat wil zeggen dat het x- en y-coördinaat een aan een gelijk zijn van elkaar.
- **¡??¿**: ¡??¿

2.4 Item ADT

Het `Item ADT` stelt elk voorwerp voor dat kan opgeraapt worden door een `Ant` (speler). ¡??¿

Het `Item ADT` bevat de volgende operaties:

- **newItem**: Maakt een nieuw object van het `Item ADT`. Het eerste argument dat de operatie neemt is een `Positie ADT` dat een positie in het maze (doolhof) aangeeft. Vervolgens het tweede argument, een functie dat wordt toegepast bij het oprapen van het `Item ADT`.
- **isCollected?**: Check of het `Item ADT` is opgeraapt of niet.

Naam	Signatuur
newItem	(Positie (any *rightarrow* any) → Item)
isCollected?	(∅ → boolean)
update!	(Ant → ∅)
¡??¿	(¡??¿ → ¡??¿)

Tabel 2.4: Operaties van het `Item ADT`

- **update!**: Deze operatie checkt of het Ant object op dezelfde positie heeft als het item. Als dat het geval is wordt het te toepassen effect toegepast door het functie dat werd meegegeven aan **newItem**. Anders doet het operatie niks.

2.5 Character ADT

Het `Character ADT` moet een elke beweegbare character voorstellen in het spel. Deze is verantwoordelijk voor alle algemeen functies die een object van deze `ADT` bezit.

Naam	Signatuur
newCharacter	(Positie → Character)

Tabel 2.5: Operaties van het `Character ADT`

Het `Character ADT` bevat de volgende operaties:

- **newCharacter**: Deze operatie maakt een object van het `Character ADT` aan.
- **¡??¿**: ¡??¿

2.6 Maze ADT

Het `Maze ADT` is verantwoordelijk om het spellogica van het doolhof te construeren.

Het `Maze ADT` bevat de volgende operaties:

- **¡++¿**: ¡++¿

Naam	Signatuur
¡++¿	(¡++¿ → ¡++¿)

Tabel 2.6: Operaties van het `Maze ADT`

- ¡++¿: ¡++¿

2.7 View_Maze ADT

¡++¿

Naam	Signatuur
¡++¿	(¡++¿ → ¡++¿)

Tabel 2.7: Operaties van het `View_Maze ADT`

Het `View_Maze ADT` bevat de volgende operaties:

- ++: ¡++¿
- ¡++¿: ¡++¿

2.8 Scorpion ADT

¡++¿

Naam	Signatuur
¡++¿	(¡++¿ → ¡++¿)

Tabel 2.8: Operaties van het `Scorpion ADT`

Het `Scorpion ADT` bevat de volgende operaties:

- ¡++¿: ¡++¿
- ¡++¿: ¡++¿

Naam	Signatuur
¡++¿	(¡++¿ → ¡++¿)

Tabel 2.9: Operaties van het `View_Scorpion ADT`

2.9 View_Scorpion ADT

¡++¿

Het `View_Scorpion ADT` bevat de volgende operaties:

- ¡++¿: ¡++¿
- ¡++¿: ¡++¿

2.10 Ant ADT

¡++¿

Naam	Signatuur
¡++¿	(¡++¿ → ¡++¿)

Tabel 2.10: Operaties van het `Ant ADT`

Het `Ant ADT` bevat de volgende operaties:

- ¡++¿: ¡++¿
- ¡++¿: ¡++¿

2.11 View_Ant ADT

¡++¿

Naam	Signatuur
¡++¿	(¡++¿ → ¡++¿)

Tabel 2.11: Operaties van het `View_Ant ADT`

Het `View_Ant ADT` bevat de volgende operaties:

- ¡++¿: ¡++¿
- ¡++¿: ¡++¿

2.12 Eggs ADT

¡++¿

Naam	Signatuur
¡++¿	(¡++¿ → ¡++¿)

Tabel 2.12: Operaties van het `Eggs ADT`

Het `Eggs ADT` bevat de volgende operaties:

- ¡++¿: ¡++¿
- ¡++¿: ¡++¿

2.13 View_Eggs ADT

¡++¿

Naam	Signatuur
¡++¿	(¡++¿ → ¡++¿)

Tabel 2.13: Operaties van het `View_Eggs ADT`

Het `View_Eggs ADT` bevat de volgende operaties:

- ¡++¿: ¡++¿
- ¡++¿: ¡++¿

3 Afhankelijkheidsdiagram

...

4 Planning

...