Fase 3: Verslag

Programmeerproject 1: Frogger

Simon Vervisch
1BA Computerwetenschappen - VUB
0557757
simon.vervisch@vub.be
20 mei 2019

Inhoudstafel

1 Introductie

Dit is de laatste fase van het Programmeerproject 1. De bedoeling van dit project was Frogger na te maken in Scheme, de programmeertaal die wordt gegeven aan de Bachelor Computerwetenschappen van de VUB. Het dialect van Scheme dat gebruikt werd is Racket, de modernste variant.

Dit is mijn eerste groot project - langer dan 100 lijntjes code -. Dit is het 'moeilijkste vak' voor mij van het eerste jaar. Dit betekent dat dit ook een van de leerrijkste vakken was voor mij. Het is het omzetten van de geleerde kennis naar de praktijk (in een nogal theoretische richting).

A) Hoe werkt het spel?

Het doel is de (veilig) de overkant halen. Hij moet daarvoor auto's ontwijken en alle muntjes en een pil opeten. Ook zijn er struiken in de weg staan en een gevaarlijke rivier (waar hij sterft). Elk level versnellen ofwel de auto's of worden er voorwerpen toegevoegd. In het zesde level gebeurt er iets speciaals.

B) Feedback op vorige fase/voorstudie

Bij de vorige fase werd er constructieve kritiek gegeven op de code. In deze fase werd er wat aan gedaan.

Zo werd het gevraagd om direct een Positie-ADt mee te geven met de objecten, in plaats van x- en y-positie. Enkel bij de auto's heb ik dit niet gedaan, daar was het handiger om de oorspronkelijk meegegeven positie (de startpositie), te behouden.

Ook heb ik meer geabstraheerd in het Teken-ADT. Het tekenen en verwijderen van tiles is in één grote procedure gestoken.

C) Extra's

Enkele kleine features die toegevoegd zijn aan het spel: in het laatste level zijn de pijltjestoetsen omgekeerd. Dit geeft een extra uitdaging.

Daarnaast is er ook functionaliteit toegevoegd die het spel aangenamer maken om te spelen. Zo is er een startscherm (met uitleg). De scores, het aantal levens, het aantal seconden dat de kikker nog ondschendbaar is en het level zijn duidelijk zichtbaar. Daarnaast is er ook een mogelijkheid om de highscore te resetten of om het spel te pauzeren.

Ook is de grafische bibliotheek lichtjes aangepast door mij. Om het verwijderen van drawables op de laag te versnellen heb ik een 'clear commando aangemaakt. Dit maakt van de drawables lijst een lege lijst.

D) Start het spel

De instructies zijn als volgt: open **"spel.rkt"** en druk op Run. Druk op s om het spel te starten. Met de pijltjes toetsen kan de kikker in de correcte richting bewegen. Een extra commando om de highscore te resetten is 'h. Let op! Dit is niet-omkeerbaar! Daarnaast is er zoals eerder gezegd een mogelijkheid tot pauzeren. Ook kan het spel stopgezet worden door op 'escape te drukken.

2 Abstracte Data Types

De ADT's stellen objecten voor. Aan deze objecten kunnen commando's meegegeven worden. Deze commando's geven een bepaalde interactie met het object (bijvoorbeeld *beweeg!*). Niet alle procedures worden besproken, sommige zijn gewoon abstracties, die niet veel info bijbrengen aan de reeds getoonde commando's.

A) Oude ADT's

Er werd veel gesleuteld aan oude ADT's. De code van Kikker-ADT is verbeterd en ook de spellogica is sterk aangepast (in het Spel-ADT).

I) Positie

Alle objecten die gedurende het spel van plaats veranderen, bezitten een positie. Het score-ADT en het Teken-ADT bezitten dit dus niet. De getters (get-x en get-y staan hier niet aangeduid). Het positie ADT staat niet aangeduid bij de ADT's. Indien het niet duidelijk is als een object positie-adt gebruikt, kijk naar het afhankelijkheidsdiagram en kijkt of het het positie-adt rechtstreeks gebruikt.

Procedure-naam	Procedure-type	Uitleg
maak-adt-positie	(Number, Number ->Positie-ADT)	Maakt nieuwe positie aan
x!	(Number ->/)	Past x-positie aan
y!	(Number ->/)	Past y-positie aan

II) Kikker

De kikker eet muntjes, insecten en de pil op. Als hij de pil opeet, wordt hij enerzijds blauw en wandelt trager en is onschendbaar voor de auto's. Door het eten van de pil en alle muntjes en de overkant te bereiken, kan de kikker het volgende level behalen. Raakt hij een van de auto's zonder onschendbaar te zijn of valt hij in de rivier, dan verliest hij een leven. Hij start met drie levens.

procedure-naam	*procedure-type*	uitleg
maak-adt-kikker	(Number, Number -> Kikker-ADT)	maakt kikker aan
volgende-positie	(Number, Number ->Pair)	Berekent volgende positie
set-beweging!	(Symbol ->/)	Verandert tag beweging
teken!	(Teken-ADT ->/)	Zorgt dat kikker getekend word
beweeg!	(/ ->/)	Beweegt de kikker
normaal!	(Teken-ADT ->/)	Maakt kikker normaal
onschendbaar!	(Teken-ADT ->/)	Maakt kikker onschendbaar
check-onschendbaarheid	(Number, Teken-ADT ->/)	Controleert of kikker nog onschendbaar moet zijn
dood!	(Teken-ADT ->/)	Vermindert levens
levens-reset!	(Teken-ADT ->/)	Zet levens op maximum
reset!	(Teken-ADT ->/)	Maakt kikker normaal en zet op oorspronkelijke positie

III) Teken

Het ADT handelt interactie met de grafische bibliotheek af. Het tekent de tekst en objecten op het scherm. De data is sterk ge-encapsuleerd. Als de grafische bibliotheek vervangen wordt, zullen er enkel aanpassingen in het Teken-ADT moeten gebeuren.

Procedure-naam	Procedure-type	Uitleg
maak-adt-teken	(String, Number, Number ->Teken-ADT)	Maakt Teken-ADT aan
teken-scherm!	(/ ->/)	Tekent achtergrond
teken-score!	(Score-ADT ->/)	Tekent score
teken-kikker!	(Kikker-ADT, Symbol ->/)	Tekent kikker
teken-adt!	(Auto-ADT, Munt-ADT, Insect-ADT, Pil-ADT ->/)	Tekent auto, munt, pil of insect
verwijder-eetbaar-adt!	(Munt-ADT, Pil-ADT, Insect-ADT ->/)	Verwijdert Pil-/Munt-/Insect-ADT
teken-welkom!	(Symbol, Score-ADT ->/)	Tekent welkomstscherm
game-over	(Symbol ->/)	Tekent game-overscherm
update-seconden!	(Kikker-ADT ->/)	Tekent correct aantal seconden op scherm
teken-score!	(Score-ADT ->/)	Tekent Score op scherm
teken-levens!	(Kikker-ADT ->/)	Tekent levens op scherm
teken-level!	(Level-ADT ->/)	Tekent Level op scherm
verander-kikker-tile!	(Kikker-ADT ->/)	Verander kleur kikker op scherm
set-spel-lus-functie!	(procedure ->/)	Zet gameloop
set-toets-functie!	(procedure ->/)	Zet toetseninput

IV) Munt

Munten verhogen simpelweg de score. Het eten van alle muntjes is nodig om de overkant te bereiken.

Procedure-naam	Procedure-type	Uitleg
maak-adt-munt	(Positie-ADT ->Munt-ADT)	Maakt munt-ADT aan
verwijder!	(Teken-ADT, Score-ADT ->/)	Verbergt muntje
teken!	(Teken-ADT ->/)	Tekent muntje
reset!	(Teken-ADT ->/)	Berekent nieuwe positie en hertekent muntje voor nieuw level

V) Spel

In het Spel-ADT is het meeste veranderd. Het spel heeft nu een level-structuur en start in level 1. Dit gaat door tot level 6 bereikt is of kikker alle levens verloren heeft. Het spel bevat ook alle collision detection. Daarnaast zijn er ook 2 gameloops (één voor de auto's en één voor de kikker) en een klein loopje dat de onschendbaarheid van de kikker checkt.

procedurenaam	proceduretype	uitleg
maak-adt-spel	(Teken-ADT ->Spel-ADT)	Maakt Spel-ADT aan
toets-functie-tijdens-spel	(symbol ->/)	Logica toetsdruk
start	(/ ->/)	Spellus
collision-auto?	(Kikker-ADT, Pair, Teken-ADT ->/)	Collision check met auto
reset!	(/ ->/)	Herteken álle objecten op scherm
welkomst	(/ ->/)	Tekent welkomstscherm
dood!	(/ ->/)	Logica als kikker dood is
initialiseer-level!	(/ ->/)	Correct toewijzen van objecten aan variabelen bij start spel
set-autos!	(/->/)	Wijst auto-objecten toe aan variabelen
eind-spel	(Symbol ->/)	Correct logica be-eindigen spel (finish of game-over)

VI) Pil

Wanneer de kikker de pil eet, wordt hij blauw en onschendbaar. Onschendbaarheid garandeert veiligheid tegen auto's, maar niet tegen water. Het eten van de pil is noodzakelijk om de overkant te halen

Procedure-naam	Procedure-type	Uitleg
maak-adt-pil	(Positie-ADT ->Pil-ADT)	Maakt pil-ADT aan
verwijder!	(Teken-ADT, Score-ADT ->/)	Verbergt pil en updatet score
teken!	(Teken-ADT ->/)	Tekent pil
reset!	(Teken-ADT ->/)	Berekent nieuwe positie en hertekent pil voor nieuw level

VII) Score

Highscore bijhouden in tekstbestand. Deze blijft de hoogste score behaald. Er zijn 3 parameters: score, level-score en highscore. De eerste houdt de huidige score bij. De tweede houdt de score bij die werd behaald bij het begin van het level. Indien de kikker sterft, wordt de huidige score op deze gezet. De highscore zit in een tekstbestand en kan dus voor langere tijd opgeslagen blijven. Het spel kan dus zonder zorgen afgesloten worden (via de escape-knop) en weer opgestart worden zonder verlies van data.

procedurenaam	proceduretype	uitleg
maak-score-adt	(/ ->Score-ADT)	Maak Score-ADT aan
teken!	(Teken-ADT ->/)	Tekent huidige (high)score op scherm
volgend-level!	(Teken-ADT ->/)	Tekent nieuwe score.
reset-level!	(Teken-ADT ->/)	Zet score op level-score
update-score!	(Teken-ADT, Munt-/Pil-/Insect-ADT ->/)	Opeten van object verhoogt score
reset-highscore!	(Teken-ADT ->/)	Zet high-score op 0
einde-spel!	(Teken-ADT ->/)	Einde spel verhoogt highscore en zet score/level-score op 0

VIII) Auto

In deze fase ben ik erin geslaagd meerdere auto's per rijstrook te tekenen (zonder lage fps). Het gedrag van de auto's staat in de tabel hieronder vermeld. Wanneer de kikker op de auto's wandelt, verlies hij een leven.

Soort Auto	Gedrag
0	Normale auto
1	Auto die vertraagt wanneer kikker rijstrook betreedt
2	Auto die kikker volgt, maar niet over middenberm gaat
3	Auto die op random momenten vertraagt en versnelt

Gedrag auto's

Procedurenaam	Proceduretype	Uitleg
maak-adt-auto	(Number, Number ->Auto-ADT)	Maakt Auto-ADT aan
teken!	(Teken-ADT ->/)	Delegeert nieuwe score naar Teken-ADT
rij-links!	(Number ->/)	Beweegt de auto een bepaalde afstand links/rechts
update!	(Teken-ADT, Kikker-ADT ->/)	Verandert de positie van de auto afhankelijk van het type
herteken-bij-botsing!	(Number, Teken-ADT ->/)	Hertekent auto als er een botsing is
uit-het-scherm?	(/ ->/)	Zet verdwenen auto's terug op het scherm
reset!	(Teken-ADT ->/)	Zet auto's op oorspronkelijke positie

IX) Insect

De functionaliteit van de insecten is beperkt. Zij kunnen enkel de scoren helpen verhogen. Elk insect geeft een andere score terug, voor het aantal punten, zie de tabel hieronder.

Soort Insect	Aantal punten
0	50
1	75
2	100
3	150

Punten Insect

Procedurenaam	Proceduretype	Uitleg
maak-adt-insect	(Positie-ADT ->Insect-ADT)	Maakt Insect-ADT aan
teken!	(Teken-ADT ->/)	Tekent Insect-ADT
verwijder!	(Teken-ADT, Score-ADT ->/)	Verwijdert Insect-ADT
reset!	(Teken-ADT ->/)	Tekent insect op nieuwe random positie

B) Nieuwe ADT's

In deze fase zijn er weinig adt's toegevoegd. Er werd een level-systeem toegevoegd. De oude adt's zijn vooral uitgebreid (om de moeilijkere levels te maken).

I) Level

Dit is het enige ADT toegevoegd in deze fase. Het zorgt voor correcte upgrades van levels en houdt deze dus ook bij. Wat zijn de moeilijkheden? Er zijn meer objecten naarmate de levels vorderen. Ook gaan de auto's vlugger. Als alle levels voltooid zijn, verschijnt er finished game en start het spel opnieuw.

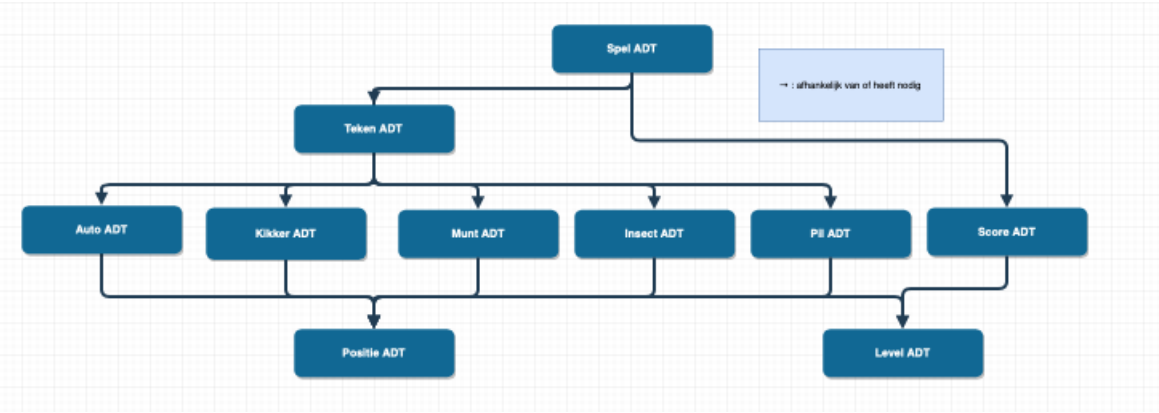
procedurenaam	proceduretype	uitleg
maak-adt-level	(/ ->/)	Maakt Level-ADT aan
teken-level!	(Teken-ADT ->/)	Tekent huidig level op scherm
volgend-level!	(Teken-ADT ->/)	Zet level naar het volgende
reset!	(Teken-ADT ->/)	Zet level op 1
struik-x-pos	(/ ->pair)	Geeft x-posities terug om struiken te tekenen
auto-refresh-rate	(/ ->Number)	Geeft snelheid auto's terug

3 Afhankelijkheidsdiagram

Veel is er niet veranderd ten opzichte van vorige fase. Het Spel-ADT heeft alle andere ADT's, want het spel heeft ze nodig voor de spellogica. Het Teken-ADT tekent alle objecten op het scherm met hun correcte positie, dus heeft beide (objecten en positie) ook nodig.

De meeste objecten die getekend worden zoals zichtbaar in het diagram, hebben nood aan een positie-ADT. Enkel Score-ADT heeft geen Positie-ADT nodig, want de positie ervan verandert niet gedurende het spel en bevindt zich niet in de spellogica.

Nu is er ook een Level-ADT. Dit Alle andere objecten hebben het Level-ADT nodig, want anders weten ze niet welke er getekend moeten worden. Het score-ADT is afhankelijk van het level-ADT om zijn highscore/score aan te passen als het level verhoogt.



Afhankelijkheidsdiagram

4 Conclusie

In deze fase heb ik abstracties veel dieper gesnapt en refactoring ook. De moeite die ik heb met abstraheren is weg. Er waren momenten dat ik echt in 'the flow' zat. Het moeilijkst aan deze fase, was de correcte logische volgorde van level-opbouw (wanneer game-over toevoegen, wanneer level toevoegen, hoe objecten verwijderen en toevoegen). Dankjewel aan de VUB om dit project te laten maken, ik kijk er met een glimlach op achteruit.

5 Bibliografie

Racket Documentation (https://docs.racket-lang.org) - Geraadpleegd voor case, assoc...
Assets library (https://www.spriters-resource.com/arcade/frogger/sheet/11067/) geraadpleegd december 2018, bevat meeste van mijn assets