Functioneel programmeren: Star Routes

Lowie Van den Bossche

23 december 2025

1 Inleiding

Star Routes is een strategisch puzzelspel ontwikkeld in Haskell. De speler navigeert een ruimteschip tussen planeten met als doel een specifieke bestemming te bereiken binnen een tijdslimiet. Het spel vereist zorgvuldig beheer van brandstof en bescherming (shield), aangezien routes kosten met zich meebrengen en gevaren (hazards) schade toebrengen.

2 Technische Vereisten

Het project maakt gebruik van de volgende bibliotheken:

- **Gloss:** Voor de grafische interface en de interactieve game-loop.
- **Parsec:** Voor de recursieve afdaling-parser die de `.star` configuratiebestanden inleest.
- **HUnit:** Voor de unit testing van zowel de parser als de spellogica.
- **System.Random:** Voor het genereren van willekeurige brandstofkosten en nebula-effecten.

3 Spelcomponenten & Logica

De architectuur volgt een functionele benadering waarbij de `GameState` bij elke frame of interactie wordt getransformeerd:

- **Planeten:** Knooppunten met optionele `Effecten` (`Fuel`, `Repair`, `None`). Het systeem houdt via `planetVisited` bij of een effect reeds is toegepast.
- **Routes:** Verbindingen met een `Direction` (`Forward`, `Backward`, `BiDirectional`). Voor bidirectionele routes worden parallelle lijnen getekend om de rijrichting te verduidelijken.
- **Gevaren:** `Asteroid`, `Pirates`, `Nebula` en `Radiation`. De logica controleert via `intersectsRoute` of een schip tijdens zijn reis door een gevaar beweegt.

4 Gebruik van Monads

In de implementatie zijn monads essentieel voor een veilige en schone code:

- **Maybe Monad:** Wordt intensief gebruikt in `Logic.hs` (bijv. in `initGameState` en `confirmRoute`) om veilig om te gaan met opzoekacties die kunnen falen, zonder dat het programma crasht.
- **Parser Monad:** Parsec-combinators maken het mogelijk om de grammatica van het `.star` bestand op een declaratieve wijze te definiëren.
- **IO Monad:** Wordt uitsluitend gebruikt aan de randen van het systeem (Main/Lib) voor bestandstoegang en de initiële random seed.

5 Overzicht van Testen

De robuustheid van het programma is geverifieerd met een uitgebreide test-suite in `HUnit`. De volgende onderdelen zijn gedekt:

5.1 Parser Tests

- `testParsePlanet`: Validatie van namen, coördinaten en de drie types effecten.
- `testParseRoute`: Controle op richtingen (`-->`, `<->`, `<--`), brandstofranges en reistijd.
- `testParseHazard`: Correcte parsing van alle types, inclusief specifieke velden zoals `damage` en `fuelLoss`.
- `testParseMission`: Validatie van start/eind-nodes en tijdslimieten.
- `testParseFullConfig`: Een integratietest die een compleet bestand met commentaar en lege regels verwerkt.

5.2 Game Logic Tests

- `testInitGameState`: Verificatie van de startwaarden van het schip.
- `testCanAffordRoute`: Controle of reizen geweigerd wordt bij onvoldoende brandstof of tijd.
- `testHazardIntersection`: Wiskundige controle op de botsing tussen scheepspaden en gevarenzones.
- `testApplyHazardEffect`: Validatie van specifieke effecten op shield en fuel (inclusief random nebula verlies).
- `testPlanetEffects`: Garantie dat planeet-bonussen slechts één keer worden toegepast.
- `testWin/LoseConditions`: Controle op `MissionSuccess` en de verschillende `FailReason` types.

6 Zelfontworpen Level: "The Gauntlet"

Dit level test resource management en risico-inschatting onder tijdsdruk.

6.1 Opzet & Uitdagingen

Het schip moet van *Launch Bay* naar *Destination Prime* binnen 50 seconden.

- **Pad A (Gevaarlijk):** Via *Fuel Station Alpha*, *Junction Crossroads* en *Repair Bay*. Bevat elke soort hazard 1 keer.
- **Pad B (Omweg):** Je neemt een kleine omweg, maar deze route is veel veiliger. Bevat 1 Asteroid Field en Pirates

6.2 Oplossing

Pad A: 1. Reis naar *Fuel Station Alpha* (incasseer 25 shield damage, herstel fuel). 2. Via *Junction* naar *Repair Bay*. Hier verlies je 20 fuel door piraten, maar herstel je het shield. 3. De laatste etappe naar de *Goal* gaat door een `Nebula`. Door de eerdere fuel-stop heb je net genoeg marge om het willekeurige verlies op te vangen. Totaal verbruik: ∼90 fuel en ∼22 seconden reistijd.

Pad B: Reis naar *Safe Harbor*, je verliest hier shield maar deze wordt weer 100 in *Safe Harbor*. Reis dan verder via *Emergency Depot* naar *Destination Prime*. Je passeert nog piraten onderweg maar in totaal heb je meer dan genoeg fuel en tijd om deze reis te overleven.

7 Reflectie

Het project toont aan hoe Haskell's sterke typesysteem fouten in complexe logica (zoals hazard-intersecties) vroegtijdig kan vangen. Een belangrijk leerpunt was de rendering; door de `Scale` en `Translate` functies in de juiste volgorde toe te passen, kon de kaart gecentreerd worden zonder de UI-elementen te verstoren. Ook het oplossen van de animatie-teleportatie (door de startpositie dynamisch te bepalen op basis van de huidige planeet) verbeterde de spelervaring aanzienlijk.