TA Tog project idea

# Titel: Untitled13

## Beschrijving

Het is een maze-game met een user als input (input string). Het zal een karakter besturen (bepaalt current state) die als objective alle munten moet oprapen. Het thema is (waarschijnlijk) iets rond het labyrinth van Minos (mythologisch).

## Features

* 1.1: Implementatie UI (start, stop, pauze, …)
* 1.2: Implementatie extra opties voor de speler (levelkeuze, moeilijkheidsgraad,. …)
* 2.1: Inlezen van levels
* 2.2: Inlezen van levelspecificaties (zoals vijanden en munten)
* 3.1: Bewegingsfunctionaliteit voor de speler (registreren van input en updaten van positie)
* 3.2: Bewegingsfunctionaliteit voor de vijanden volgens een vast pad met een bepaalde snelheid
* 3.3+: Extra speciale opties zoals fog of war, playertracking, …
* 4.1: Grafische voorstelling (van een level) met QT.
* 4.2: Scorebord (tekstbestand)
* 4.3+: Eigen design: textures voor speler, vijanden, munten, …

## Motivatie

Wij hebben voor een spel gekozen omdat we dit een leuke afwisseling vonden van serieuzere onderwerpen. Dit wil echter niet zeggen dat spellen geen nut hebben, integendeel spellen kunnen zeer verlossend werken en zijn daarom van onschatbare waarde in onze moderne samenleving.  
We kozen meer specifiek voor een doolhofspel vanwege de nostalgie natuurlijk 😉 maar ook omdat dit zeer goed past binnen de automatentheorie (input/output).

## Algoritmen en structuren

### DFA

DFA’s vormen de basisstructuren van ons spel. We gaan deze gebruiken voor zowel levellayouts als voor het bijhouden van collectibles (munten en eventueel anderen). Een final state wordt in een level bereikt op ofwel een bepaalde plek (een uitgangsdeur) of op elke plek maar enkel als de collectible-DFA ook final is (productautomaat?). De collectible DFA wordt final na het behalen van alle collectibles.

### ε-NFA

Deze kunnen ook de layout van een level weergeven. Ons eerste idee is om bij ε-NFA’s gebruik te maken van de epsilontransities om portalen (teleporteren van 1 vak naar een ander zonder aparte input) gemakkelijk te implementeren.

### RE

Met RE’s kunnen we op een gemakkelijke manier de waarden die de gebruiker koos als extra opties doorgeven naar het level. Door ons eigen algoritme te gebruiken kunnen we deze RE omzetten naar een DFA die de plaatsen van de collectibles en de eventuele uitgang aanduidt.

### Product automaat

De productautomaat wordt gebruikt om de level DFA te combineren met de collectible DFA of andere DFA’s. Zo kunnen we alles in een keer bijhouden en moeten we op het einde slechts eenmalig checken of aan alle voorwaarden voldaan is (final state van de productautomaat).

### SSC en MSSC

De SSC en de MSSC gebruiken we allebei om grotere zekerheid en compatibiliteit te garanderen. We maken het mogelijk om levels die in de vorm van een NFA of een ε-NFA gedefinieerd zijn ook te gebruiken in algoritmen die een DFA vragen.

### RE naar DFA

Zoals eerder vermeld dient dit algoritme dat we zelf zullen ontwerpen ervoor om RE’s (die zullen voorkomen als definitie van de levelspecificaties) om te zetten naar een DFA. Dit bespaart ons een tussenstap die we zouden maken bij het omzetten naar ε-NFA om die vervolgens naar een DFA om te zetten.

## Categorie: goud

We implementeren onze eigen structuren en algoritmen en we maken bovendien gebruik van ons eigen “RE naar DFA”- algoritme dat niet in de cursus voorkwam.