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 gebruiken wij voor de beweging van de player. De lay-out van onze maze (beweegbare paden) slaan wij op in DFA. Deze DFA gebruiken wij ook voor het bewegen van de speler. Wij maken een aparte DFA om te zien of de player alle keys heeft opgepakt. De speler kan pas winnen (weglopen) als hij alle keys heeft en de uitgang bereikt.

### 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

RE zal gebruikt worden om random levels te genereren.

### RE naar ε-NFA

Hier zetten wij de RE van een level naar een ε-NFA.

### MSSC

Wij gebruiken MSSC om een ε-NFA om te zetten naar een DFA.

### TFA

Wij gebruiken de TFA om te zien of de DFA van de pre-made level en DFA van de level die door de level generator is gemaakt equivalent zijn met elkaar.

### Markov Chain Monte Carlo: Random walk algoritme

Wij gebruiken dit algoritme om onze vijand een bepaalde pad aan te leren, zodat die zelf kan kiezen welke richting het gaat na een bepaalde beweging.

Het zal een MCMC structuur hebben en zal de random walk algoritme erop toepassen.

### Dijkstra’s algoritme

Deze algoritme zal de kortste pad zoeken en zeggen hoeveel zetten die minimum nodig heeft om aan de finish te komen.

## Categorie: Platina

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.

We maken gebruik van de Markov Chain algoritme, om onze vijand zijn bewegingen aan te leren.

We hebben ook nog de Dijkstra’s algoritme om de kortste pad te vinden.

## Bronnen:

<https://en.wikipedia.org/wiki/Random_walk#Gaussian_random_walk>

<https://en.wikipedia.org/wiki/Gibbs_sampling>

<https://en.wikipedia.org/wiki/Markov_chain_Monte_Carlo>

<https://machinelearningmastery.com/markov-chain-monte-carlo-for-probability/>

<https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm>