Ghent University

Logisch programmeren

Verslag project "con-tax-tiks"

Tibo Vanheule

Academiejaar 2019-2020

Contents

1 Introductie **2**

1.1 Documentatie 2

1.2 Server: grafische weergave 2

1.3 Vraag vanuit de main 2

2 Interne bord voorstelling **3**

3 Algoritme **3**

4 Conclusie **3**

1 Introductie

Dit verslag voor het project "con-tax-tics" of "hex" bespreekt de implementatie van de parser, server, svg-generator en de simpele AI.

1.1 Documentatie

Heel het project is gedocumenteerd in de "java-doc"-stijl. Deze stijl is ondersteund door de pldoc-pacakage. Pldoc zorgt ervoor dat er HTML-bestanden gegeneerd worden. Op deze manier kan er snel en makkelijk een proxy server opgezet worden met eenvoudig navigeerbare documentatie-server.

Deze documentatie-server is bereikbaar via deze link: https://hex.tibovanheule.space/pldoc/.

1.2 Server: grafische weergave

Om het project iets interactiever te maken, is er een React-website gemaakt om het spel te spelen. Dit is bereikbaar via deze link: https://hex.tibovanheule.space.

1.3 Vraag vanuit de main

wat doet read_string(user_input,"\n","\r",End,Codes)?

Uit documentatie:

Read a string from Stream, providing functionality similar to split_string/4. The predicate performs the following steps:

1. Skip all charactershttps://www.overleaf.com/project/5e71d62cc1582f00010cbbac that match PadChars
2. Read up to a character that matches SepChars or end of file
3. Discard trailing characters that match PadChars from the collected input
4. Unify String with a string created from the input and Sep with the separator character read. If input was terminated by the end of the input, Sep is unified with -1.

The predicate read_string/5 called repeatedly on an input until Sep is -1 (end of file) is equivalent to reading the entire file into a string and calling split_string/4, provided that SepChars and PadChars are not partially overlapping.157 Below are some examples:

In eigen woorden: Leest een strings gedeeld door de separator '\n' en sla de '\r' over. (support voor windows). De string wordt gemaakt van de user input stream.

2 Interne bord voorstelling

Het bord word voorgesteld door een predicaat game(). Deze bevat datastructuur bevat Size, Turn, Number_of_tiles, Tiles, State en Oriëntatie. Bij tiles of tegels is wel op te merken dat we enkel de bezette tegels bijhouden. De vrije tegels zijn namelijk vrij gemakkelijk te bepalen via bagof, size en de bezette tegels.

Een voorbeeld van een spel bord voorgesteld in onze datastructuur: game(size(5,5), turn(Turn), number_of_tiles(5), tiles([tile(coordinate(X/Y),player(Player))], state(undecided), orientation(X,Y))

3 Algoritme

We hebben een heuristiek nodig voor het mini-max spel-boom. We kunnen natuurlijk niet de volledige spel boom doorlopen, maar stoppen wanneer we bijvoorbeeld diepte 3 hebben bereikt. We hebben dus een manier nodig om een spelbord te evalueren en een score aan toe te kennen, sinds we niet de simpele gewonnen/verloren toestand uit komen. Hiervoor gebruiken we een heuristiek die het korste pad kan vinden in een bord. Een simpel te implementeren algoritme is het dijkstra algoritme. Een simpel voorbeeld van waar dijkstra een korste pad vind ziet men in figuur 1. De score van het spelbord wordt bereknt volgens de inviduele score

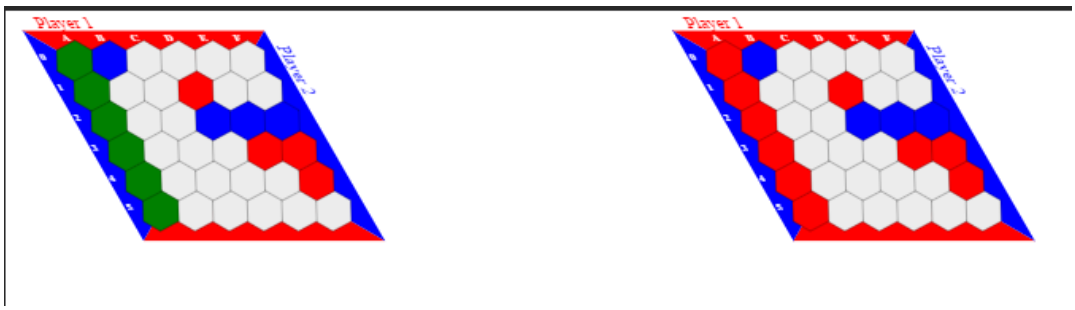

Figure 1: Links ziet men het door dijkstra gevonden groen pad, en rechts ziet u het normaal spelbord

van de spelers. Voor elke speler gebruiken we dijkstra om de korste route te bepalen via

4 Conclusie

Het dijkstra - en het minimax algoritme waren goed geïmplementeerd, maar persoonlijk is deze net iets te traag. Deze konden nog meer geoptimaliseerd worden met onder andere memoization en dijkstra specifieke optimalisaties.