Game AI

Wetenschappelijke toepassingen

Robbe De Geyndt – Odisee 2015-2016

Inhoud

[Opdracht: 2](#_Toc452155978)

[Onderzoek: 2](#_Toc452155979)

[Werking van programma: 3](#_Toc452155980)

[Besluit en reflectie: 3](#_Toc452155981)

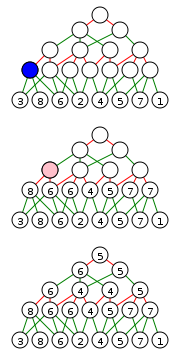
[Bronnen: 4](#_Toc452155982)

Game AI

# Opdracht:

De opdracht is het programmeren van een AI, die vier-op-een-rij kan spelen. Deze speler moet zich intelligent gedragen. Voor deze AI wordt er gebruik gemaakt van de minimax strategie.

# Onderzoek:

Om de volgende zet intelligent te kunnen bepalen maken we gebruik van de minimax strategie. [1] Dit betekend dat er wordt berekend op welke manier de speler zijn eigen winst kan maximaliseren en de tegenstanders winst kan minimaliseren. Dit gebeurt door elke zet af te lopen en een rank te geven. Ook worden de volgende stappen bekeken om te kunnen zien welke stap er het meest zal opleven. Dit programma kan 3 tot 4 stappen vooruit bereken en performant spelen tegelijkertijd.

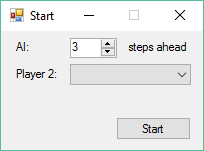
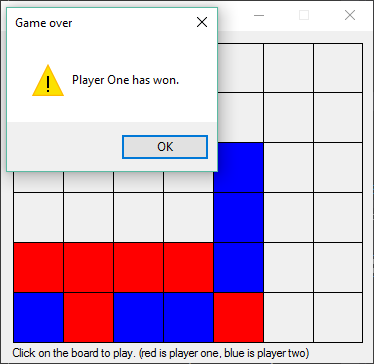
1 Figuur A

Elke beurt uit bestaat 7 mogelijkheden, het aantal mogelijkheden exponentieel toenemen met het aantal stappen dat er vooruit gekeken wordt. Er worden 3 punten toegekend aan een stap die onmiddellijke winst opleveren 2 voor 3 op een rij en 1 voor 2 op een rij. Het omgekeerde wordt toegekend aan de stappen die de tegenspeler iets opleveren. De waarde van een stap die pas de volgende beurt iets opleveren worden gehalveerd.

Voor deze applicatie is er veel gebruik gemaakt van parallel lopende threads [2] om het systeem performant te maken. De verschillende mogelijkheden kunnen namelijk gemakkelijk worden afgegaan in verschillende threads.

Voor het spelbord wordt er gebruik gemaakt van een bitmap die gegenereerd wordt na elke beurt en daarna wordt weergegeven in de UI. De keuze van de gebruiker wordt bepaald door een click op deze afbeelding.

# Werking van programma:



Wanneer de applicatie wordt gestart wordt aan de gebruiker gevraagd om een te kiezen hoeveel stappen de AI mag vooruitdenken, er wordt ook de optie gegeven om de AI tegen een ander AI te laten spelen en hoeveel stappen deze mag bereken. De gebruiker kan spelen door op de juiste kolom op het bord te klikken.

Indien de 2 AI’s tegen elkaar spelen kan de gebruiker op het bord klikken om te starten. Wanneer een speler gewonnen heeft, zal er een boodschap verschijnen. Ook wanneer her bord vol is en er geen winnaar is zal dit worden aangegeven.

# Besluit en reflectie:

Het moeilijkste onderdeel van deze opgave was het schrijven van de stappen die de AI aflegt. Dit wordt recursief gedaan waardoor het moeilijk is om de logica soms te volgen.

Een ander probleem met de applicatie is dat ze in Visual Studio zeer langzaam is, waardoor het debuggen met de ingebouwde debugger van Visual Studio niet vaak niet mogelijk was. De applicatie loopt vrij performant buiten Visual Studio maar eist wel erg veel CPU-tijd op.

Het resultaat is dat het Algoritme vrij moeilijk te verslaan is. Als men 2 instanties van de AI tegen elkaar laat spelen, duurt het vaak ook vrij lang tot één van de spelers 4 op een rij heeft, vaak eindigt het spel ook zonder een winnaar. Ook heeft de AI die meer stappen vooruit berekent, meer kans om te winnen van een AI met minder stappen.

Een mogelijk manier om de applicatie nog te verbeteren zou het toevoegen van algoritmes die rekenwerk kunnen besparen door bepaalde mogelijkheden, die tot niks leiden, weg te laten.

# Bronnen:

Figuur A.: <https://nl.wikipedia.org/wiki/Minimax>

1. <https://github.com/ethangunderson/minimax>
2. <https://msdn.microsoft.com/en-us/library/ee256691(v=vs.110).aspx>