# Plan van aanpak: vier op een rij

### Probleembeschrijving

Er moet voor Casper de Vries een applicatie gemaakt worden die intelligente zetten voor vier op een rij kan maken. De ai in de applicatie moet meerdere stappen vooruit kunnen denken om het zo concurrerend mogelijk te maken om tegen te spelen en, mag de AI niet te lang doen over een zet om de gebruiker een betere speel ervaring te geven. Mogelijk kan er ook vier op een rij gespeelt kunnen worden in de applicatie zelf. zodat Casper in ze eentje vier op een rij kan spelen voor als er niemand anders is.

### Algoritme beschrijving

Het totale algoritmen word een zoekalgoritmen die zoekt naar de best mogelijke zet door meerdere beurten vooruit te rekenen. Hieronder staan de onderdelen beschreven waaruit het algoritmen gaat bestaan.

Om te beginnen met het algoritmen is de basis een MinMax algoritmen. Dit houd in dat wanneer het de AI zijn beurt is dat er een zet gekozen word die de score maximaliseert, en als het de persoon zijn beurt is kiest die een zet kiest die de score minimaliseert. Dit algoritme upgrade we dan naar een Alpha-Beta algoritme, hiermee word het aantal zoekpaden die volledig doorzocht worden vermindert en kan er dieper gezocht worden. Dat resulteert in een betere maximalisatie. Dit word gedaan door bijvoorbeeld als je een score weet die beter is dan een andere scores, is er geen rede om zetten te ontdekken met die lagere scores.

De score word dan bepaalt als volgende: (21 aantal beurten per speler + 1) - (aantal gespeelde beurten + aantal beurten voor mogelijke winst). Om score is 0 met gelijkspel, positief met winst en negatief met verlies. Met dit score systeem kunnen ze met elkaar vergeleken worden om te kijken welk het best is.

Zie de volgende bron.

Pascal, P. (2019). Gamesolver.org. Solving connect 4: how to build a perfect AI. Part 2, 3 en 4. http://blog.gamesolver.org/solving-connect-four/03-minmax/

Om de score te bepalen van een zet moet eerst de positie bekend zijn. Dit gaat gedaan worden met een Numpy array van 7 kolommen en 6 rijen die het speelbord symboliseert. Voor de stenen van speler 1 staat het nummer 1, de stenen van speler 2 het nummer 2 en lege plekken 0. Wat ook van belang is, is het herkennen van winst bij een positie. Hiervoor gaat het speelbord gescand worden op alle 69 mogelijke win manieren.

Zie de volgende bron.

Medad, N. (2020). Roboticsproject.com. connect four robot. Connect 4 algorithm, setup functions.

https://roboticsproject.readthedocs.io/en/latest/ConnectFourAlgorithm.html#setup-functions

afhankelijk van hoe de AI op dit punt presteert kunnen er nog een paar heuristieken upgrades gedaan worden. deze zijn 1: Voor elke positie word gekeken hoeveel win mogelijkheden dat oplevert en hoeveel win mogelijkheden voor de tegenstander. De score die hier uitkomt is de twee

winmogelijkheden min elkaar. 2: is een functie die kijkt of een pad van de tegenstander geblokkeerd kan worden. 3: kijken hoeveel stenen er op een rij staan bij een positie. Hoe meer stenen er op een rij staan hoe beter die positie word beoordeeld.

Zie de volgende bron.

Faris A. (2012-04-27). Sourceforge.net. a connect 4 game with a smart AI.

https://sourceforge.net/projects/connect4ai/

Uiteindelijk is de bedoeling dat de AI binnen 5 seconden een slimme zet moet vinden.

**Taken**

1. Score van posities algoritmen bouwen. | noodzakelijk
2. MinMax algoritmen bouwen. | noodzakelijk
3. Upgrade naar Alpha-Beta algoritmen. | noodzakelijk
4. Testen en verbeteren | noodzakelijk
5. Heuristieken upgrades bouwen | optioneel
6. GUI maken waarin tegen de AI gespeeld word | optioneel
7. Presentatie voorbereiden | noodzakelijk
8. Rapportage van project maken | noodzakelijk

**Planning**

De taken staan opvolgorde. Van eerst naar laatst

| **Week 1** | **Week 2** | **Week 3** |
|---|---|---|
| Taak 1<br>Taak 2<br>Taak 3<br>Taak 4 | Taak 4<br>Taak 5 | Taak 6<br>Taak 8<br>Taak 7 |

De week voor het project zal er ook al begonnen worden met taak 1 en taak 2.

**Risico's**

- Score algoritmen bouwen lijkt mij het moeilijkst en kan dus potentieel veel tijd kosten.
  Om het te voorkomen zal taak 1 hier hoogste prioriteit krijgen zodat ik het mezelf zo makkelijk mogelijk maak met taak 2.
  Als het toch het geval is dat, zal de planning iets opgeschoven moeten worden en zal dat te koste gaan van taak 6.

- Mijn solver geeft wel goede resultaten, maar doet er te lang over om comfortabel in real-time tegen te kunnen spelen. Om dit te voorkomen moet de code zo geschreven worden dat het op de meest efficiënte manier kan runnen. Als het toch voor doet zal er een extra upgrade aan het al bestaande algoritmen gemaakt worden of een alternatieve algoritmen gebruikt worden.