Ghent University

Logisch Programmeren

Verslag Con-tac-tix project

Jan-Pieter Baert

Academiejaar 2019-2020

Inhoudsopgave

Jan-Pieter Baert
Bachelor of Sciences in Informatics
Stamnummer: 01703178

1 Inleiding

In dit verslag zal ik een korte bespreking geven over de implementatie van het spel Con-tac-tix en de hiervoor geschreven AI.

1.1 Korte speluitleg

Het spel con-tac-tix, soms ook wel hex genoemd is een spel gespeeld op een hexagonale 'rechthoek' bord, met arbitraire grootte.[1]
De bedoeling van het spel is een on-onderbroken pad te creëren van boven naar onder voor speler 1 (en van links naar rechts voor speler 2), de spelers mogen beurt om beurt een tegel leggen en het spel stop wanneer een speler zijn twee zijkanten verbonden heeft.

[1] Voor meer specifieke informatie over hex is wikipedia zeer handig

2 Interne bordvoorstelling

Ik heb gekozen om het bord voor te stellen als een feit dat de volgende elementen bevat:

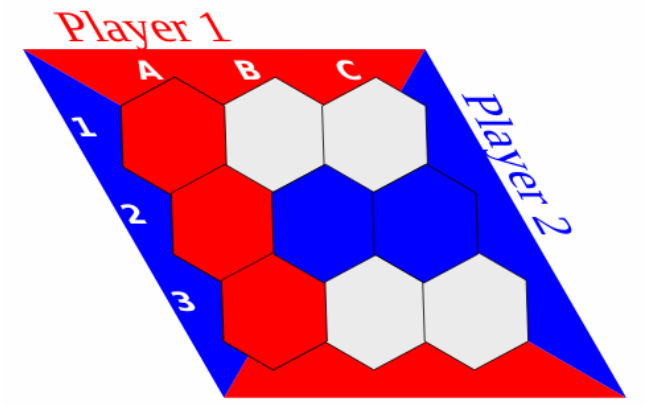
1. De grootte van het veld, opgeslagen als het paar `X/Y`
2. De kleur van de speler die aan zet is
3. De kleuren van beide spelers, opgeslagen als het paar `Player1/Player2`
4. De status van het spel. Het spel is ofwel bezig of gewonnen door één van de spelers, dit wordt voorgesteld als een getal, -100 als speler 1 gewonnen heeft en 100 als speler 2 gewonnen heeft, alle waarden ertussen zijn spelen die nog bezig zijn.
5. Een lijst van tegels die het bord voorstellen.
   Een tegel stellen we voor als een feit dat de volgende elementen bevat:
   - De coördinaten, opgeslagen als het paar `X/Y`
   - Het kleur van de tegel

2.1 Voorbeeld

Een mogelijke bordvoorstelling waarop speler 1 (hier in het rood gekleurd) reeds gewonnen heeft, kan dan voorgesteld worden als:

```
|board(3/3,blue,red/blue,-100,[tile(0/0,red),tile(0/1,red),tile(0/2,red),tile(1/1,blue),tile(2/1,blue)])
```

Dit is ook grafisch te zien op figuur 1



Figuur 1: Een 3x3 bord van con-tac-tix

3 Algoritme

Kort samengevat werkt het gebruikte algoritme voor de AI als volgt:

1. Genereer alle mogelijk zetten en kijk als dit tot een winnend bord leid.
2. Anders, gebruik een minimax boom (uitgelegd in 3.2) om te bepalen wat de beste volgende bord is. De minimax boom wordt geëvalueerd gebruikmakende van de heursitiek (uitgelegd in 3.3).

3.1 Spelstatus bepalen

Om te bepalen of het spel is afgelopen door een overwinning van een speler, gebruiken we de volgende methode die gebaseerd is op het floodfill algoritme.[2]

We nemen de coördinaat (`-1/-1`) als beginpunt en we zoeken een weg naar de coordinaat (`X/Y`) waarbij `X` de hoogte van het speelveld is en `Y` de breedte van het speelveld, dit pad zoeken we langs tegels van hetzelfde kleur. We doen het floodfill algoritme met enkel de coördinaten van de tegels in het kleur van de speler waarvoor we nagaan of deze gewonnen is, dit is inclusief de coördinaten op de randen van het speelveld die die speler probeert te verbinden.

3.2 Minimax

Voor minimax gebruiken we de volgende interface:

```
minimax(Pos, BestNextPos, Val, Depth)
```

Hier is `Pos` het huidige bord, `BestNextPos` het meest optimale opvolgende bord, `Val` de waarde van de meest optimale direct opvolgende bord (namelijk `BestNextPos`) en `Depth` de maximale zoekdiepte tot waar de minimax boom gegenereerd wordt.[3]

Het minimax algoritme zal vertrekkende van een initieel bord alle opvolgers genereren tot een maximale diepte van `Depth` en dan evalueren, gebruikmakende van waarden van de heuristiek, welke directe opvolger de meest optimale is en welke waarde deze heeft.

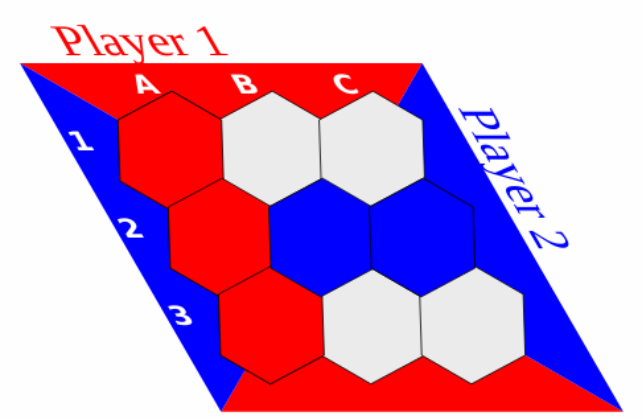
[2] Voor meer info is wikipedia handig.

[3] Deze minimax interface is gebaseerd op de minimax vanuit de les logisch programmeren (de code van hoorcollege 7)

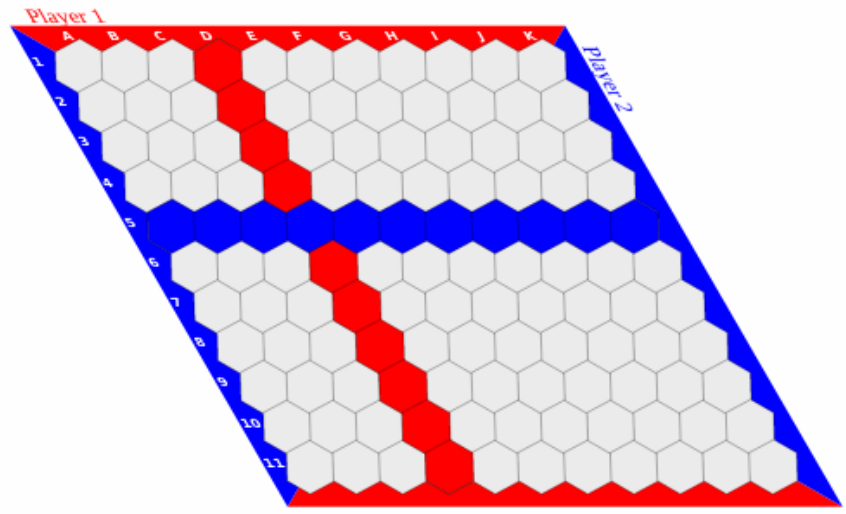
3.3 Gebruikte heuristiek

Als heuristiek nemen we een eenvoudige berekening: het verschil tussen `P2` en `P1` waar `P1` het aantal rijen is dat speler 1 bezet heeft en `P2` het aantal kolommen dat speler 2 bezet heeft. Dit geeft ons voor speler 1 een lage ideale score en voor speler 2 een hoge ideale score, hetgeen ideaal is voor het gebruik van een minimax boom. Hieronder zullen we een aantal voorbeelden geven om aan te tonen hoe de heuristiek werkt.

3.3.1 Voorbeelden



Figuur 2: Bord met heuristiek -1 (2-3)



Figuur 3: Bord met heuristiek 1 (11-10)

3.4 Versnelling door $\alpha - \beta$ snoeien

Als versnelling voegen we $\alpha - \beta$ snoeien toe aan de minimax boom, hiervoor hebben we gebruik gemaakt van dynamische feiten in prolog om de waarden voor α en β aan te passen.

Tabel 1 toont het verschil in uitvoeringstijd van de AI, op een leeg bord waar de hoogte en breedte dezelfde is. We zien duidelijk dat $\alpha - \beta$ snoeien de AI enorm versnelt, hierdoor kunnen we ook de maximale diepte op -1 zetten om de AI tot ieder blad te laten gaan zonder dat dit de snelheid zodanig vertraagt dat het spel virtueel onspeelbaar wordt.

Grootte	Uitvoeringstijd zonder	Uitvoeringstijd met
2	0.198s	0.201s
3	23.116s	0.213s
4	2500s	0.200s

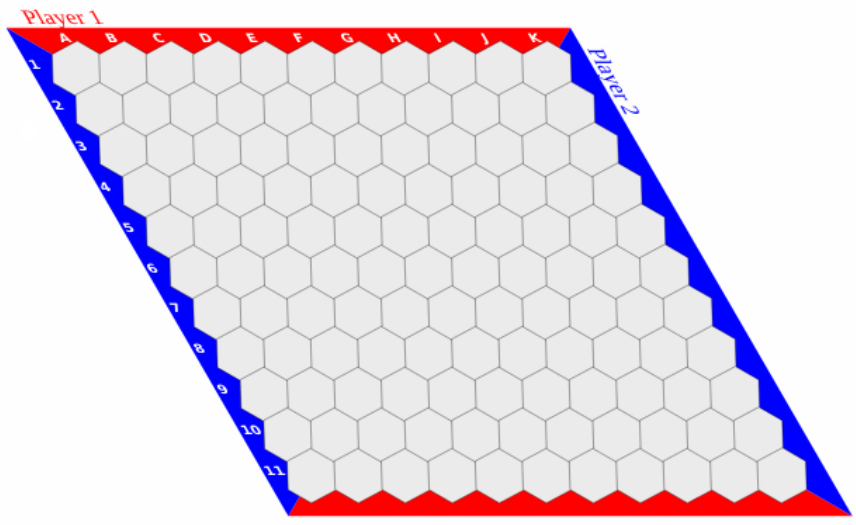
Tabel 1: Vergelijking van uitvoeringstijd zonder en met $\alpha - \beta$ snoeien

4 Interactieve versie

De interactieve versie van deze AI maakt gebruik van de unicode versie van de output. De speler kiest eerst de grootte van het veld, daarna wordt er random beslist of de speler of de bot begint. De speler is altijd de eerste speler, met het rode kleur en de bot is steeds de tweede speler met het blauwe kleur.

4.1 Voorbeeld:

In volgende figuur zien we het initieel bord als de speler 11 als grootte neemt.



Figuur 4: Een 11x11 bord van de interactieve con-tac-tix

5 Conclusie

Hoewel het algemene algortime van minimax met $\alpha - \beta$ snoeien goed geimplenteerd is, zijn er nog aanpassingen/verbeteringen mogelijk.

1. Een andere, meer uitgebreide heuristiek zou kunnen helpen om de AI beter te maken.
2. Bij het genereren van opvolgers van een bord en nagaan of een spel gewonnen is zouden nog aanpassingen kunnen gemaakt worden zodat deze sneller gebeuren.

Dit alles zou moeten zorgen voor een nog betere spelervaring.