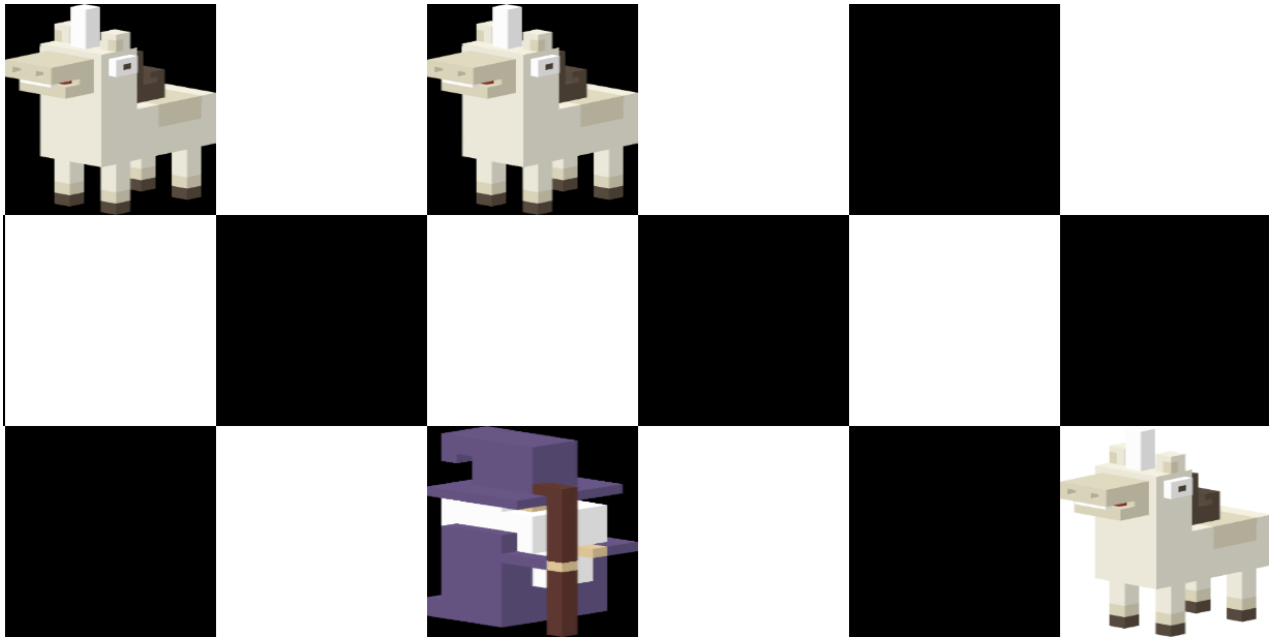
Lesbrief: AI in Games



Gandalf vs Unicorns

Garry Kasparov stond in zijn carrière vanaf 1986 tot hij in 2005 stopte 225 van de 228 maanden nummer 1 van de wereld, en wordt daarom door velen gezien als de grootste schaakmeester ooit. Toch werd ook hij in 1994 verslagen door een computer (https://www.youtube.com/watch?v=3EQA679DFRg). Dit kwam niet door machine learning of big data technieken, maar door (relatief) eenvoudige AI-algoritmes en een voor die tijd snelle computer met voldoende geheugen. Doordat de computers daarna alleen maar sneller werden heeft sinds 2005 nooit meer een mens kunnen winnen van een computer met voldoende rekenkracht (https://deepchess.org/blog/f/human-against-computer-chess-matches).

Een AI-algoritme als Minimax speelt in geheugen alle mogelijke paden voor een spel, en kiest het pad naar een zekere winst. Problemen met dit soort algoritmes zijn echter geheugen en tijd. Die zijn namelijk voor de meeste computers op voordat ze alle mogelijke spelsituaties bekeken hebben. De truc om hiermee om te gaan is om niet tot in het oneindige door te gaan, maar slechts een paar zetten vooruit te kijken. Met 4 zetten vooruit versla je een beginner, met 8 een sterke speler en met 12 zetten Kasparov. De moeilijkheid zit hem er hier wel in dat je in moet zien te schatten wat een goede spelsituatie is (dichter bij de winst) waar je AI naartoe wilt spelen.

Uitzoeken

- Wat is Breadth first search (hoef je niet uit je hoofd te kennen, maar je moet het kunnen vergelijken met minimax)
- Minimax
- Game state value
- Search depth
- Verschil AI algoritmes (zoals minimax) en Machine learning (zoals neural networks)

Opdracht

Gandalf wil oversteken naar de overkant van het schaakbord, maar de 3 eenhoorns proberen hem tegen te houden. Aan jou de taak om de eenhoorns te programmeren.
Gandalf beweegt als een koning uit het schaakspel, en de eenhoorns als paarden.

Demonstreer:

- Werkende Gandalf vs Unicorns (AI) game
    - Implementeer minimax
    - Verbeter de evaluatie functie
- Verschil tussen zoekdieptes (aantal zetten vooruit kijken)
- Verschillende bordgroottes / aantal tegenstanders
- Optioneel: verbeter performance van de minimax met alpha beta pruning
- Optioneel: extra karakters

Leerdoelen

- Ik begrijp het verschil tussen traditionele AI en Machine Learning technieken
- Ik begrijp het verschil tussen gewone search (breadth first) en adversarial search (minimax).
- Ik begrijp de tijd-complexiteit van een zoekalgoritme
- Ik kan MiniMax uitleggen en toepassen
- Ik begrijp hoe ik een evaluatie functie moet maken

Toetsing

Om deze lesbrief te behalen moet je de volledig uitgewerkte opdracht kunnen demonstreren (zie boven) en moet je vragen kunnen beantwoorden over het uitzoek-werk (zie boven).

Tools

- Typescript

Startcode

- https://github.com/HR-CMGT/PRG09-AI

Tips

Je hebt een echt bord met spelers (de gamestate) waarop je speelt. De AI die vooruit kijkt voor het minimax algoritme mag hier natuurlijk niet op spelen, daarom moet je een kopie maken van de gamestate (`gameStateCopy = gamestate.copy()`).

Als je de beweging van een speler opvraagt, bijv. `king.getMoves()` dan krijg je standaard de mogelijke moves in de **echte** game. Als je de moves in de **kopie** (voor minimax) wilt weten moet je zelf de positie van de speler meegeven, bijv. `king.getMoves(gameStateCopy.kingPos)`.

Bronnen

Minimax

- https://www.youtube.com/watch?v=l-hh51ncgDI
- https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/
- https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-2-evaluation-function/
- https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-3-tic-tac-toe-ai-finding-optimal-move/

Alpha-beta pruning (optioneel)

- https://en.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning