IPASS RAPPORTAGE

PEPIJN DEVUE 1821614

PROBLEEMBESCHRIJVING

Dammen is een bekend maar niet al te populair, meestal als gezin- of huisgenoten een potje willen dammen is er niemand die een potje mee wilt doen. Zelfs als je een zusje, vriend of moeder kan vinden om een potje te doen is diegene meestal ook nog eens een stuk beter of slechter dan jij bent en is er helemaal geen lol aan beide kanten. En ook is er, tegenovergesteld aan schaken, geen online platform en/of gemeenschap om mee te spelen, bonden of leren. Kortom, het is moeilijk om te kunnen genieten van een passie voor dammen en het is moeilijk om het spel te leren of beter te worden.

OPDRACHTBESCHRIJVING

Mijn vader, een beginnende dammer, wilt beter worden met het spel. Het probleem is alleen dat er niet altijd iemand is die met hem wilt dammen en zelfs als er iemand is dat diegene meestal niet het zelfde niveau beheerst.

Een minimax algoritme denkt een gegeven aantal stappen vooruit en bepaalt zo een zet om te doen, hoe meer stappen er vooruit wordt gedacht, hoe moeilijker. Zo heeft mijn vader altijd een tegenstander die niet alleen op verschillende niveaus kan spelen maar ook nog eens consistent die niveaus kan nabootsen.

EISEN

Mijn vader wilt een applicatie die makkelijk te gebruiken, het liefst met zo min mogelijk nodige uitleg nodig. In deze applicatie kan hij een potje dammen tegen de computer. De applicatie mag geen illegale zetten toelaten om verwarring te voorkomen. Daarnaast wilt mijn vader zoveel mogelijk begeleiding krijgen van de computer, zo wilt hij dat je een stuk kan selecteren om te zien welke mogelijke zetten gemaakt kunnen worden met dat stuk.

ALGORITME

Het gekozen algoritme voor een dam-bot is minimax [2]. Minimax is een zoek- en backtrack algoritme met het doel om de beste zet in een zero-sum perfect-info game. Het algoritme kijkt naar alle mogelijke zetten die een speler kan maken en onthoud de status van het bord na die zetten. Vervolgens kijkt het algoritme bij elk van deze borden weer naar alle mogelijke zetten die de volgende speler kan zetten, dit gaat door tot een maximum aantal iteraties, ook wel ply genoemd. Als het algoritme zijn ply heeft bereikt of vroegtijdig een spelstatus tegenkomt waarin een speler wint heeft gevonden kent hij een score (door middel van heuristiek) toe aan die eind-node. Vervolgens volgt hij zijn pad terug om te kijken welke zet het beste is, dit wordt gedaan met de aanname dat beide spelers proberen te winnen. Ik heb voornamelijk voor dit algoritme gekozen omdat het vaak gebruikt wordt bij spellen zoals dammen, maar ook omdat dit ook is hoe een goede dammer zou spelen, scenario's voorbeelden en zo kijken wat de beste zet is. Ook is alpha-beta pruning bij het algoritme toegepast [3], er kan worden herkend dat de spelstatus waar op een gegeven moment naar gekeken wordt sowieso niet

het beste is, op zo'n moment kapt het algoritme die tak af om een kortere afspeelsnelheid te hebben. Dit heb ik gedaan omdat dit een extreem positief effect heeft op efficiëntie van het algoritme.

EVALUTATIE

Het gekozen algoritme past top bij het gegeven probleem, ik heb nog niemand gevonden die heeft kunnen winnen van een 3-ply bot, terwijl het programma gerust doorgaat tot 6 zonder al te lang hoeven te wachten. Het enige nadeel wat ik gevonden heb is het volgende scenario, het is het einde van een 6-ply spel en wit heeft 2 koningen en nog een dam op de een-na-laatste rij. In elke zet die wit doet geeft goede score omdat wit in de toekomst altijd de zet kan doen om de 3de koning te maken. Omdat hij het later ook kan doen doe hij dat niet en probeert hij te winnen met 2 koningen, terwijl het makkelijker zou zijn met 3 koningen. Voor efficiëntie heb ik niet veel kunnen doen, ik heb de functies getPieces() en estimateScore() ietsje kortere kunnen maken door alleen te loopen door de zwarte vakken op het bord in plaats van alle alle vakken. En natuurlijk heb ik ook alpha-beta pruning toegepast wat zo'n 20% van de handelingen van een zet kan afkappen. Er zijn natuurlijk geen ethische overwegingen nodig in dit project

BRONNEN

1. Jonathan Schaeffer et al. ,Checkers Is Solved.Science317,1518-1522(2007).DOI:10.1126/science.1144079
2. Strong, G. S. (2011). The minimax algorithm. Minimax-notes. http://users.encs.concordia.ca/~kharma/coen352/Project/Minimax-notes.pdf
3. GeeksforGeeks. (2023). Minimax Algorithm in Game Theory Set 4 Alpha Beta Pruning. GeeksforGeeks. https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-4-alpha-beta-pruning/
4. Constants — NumPy v2.0.dev0 Manual. (n.d.). https://numpy.org/devdocs/reference/constants.html
5. GeeksforGeeks. (2021). Default arguments in Python. GeeksforGeeks. https://www.geeksforgeeks.org/default-arguments-in-python/
   Python Classes. (n.d.). https://www.w3schools.com/python/python_classes.asp
6. GeeksforGeeks. (2022). PYGLET On Draw Event. GeeksforGeeks. https://www.geeksforgeeks.org/pyglet-on-draw-event/
7. pyglet Documentation — pyglet v2.0.7. (n.d.). https://pyglet.readthedocs.io/en/latest/index.html#