Levi Vlasblom

l.vlasblom@student.avans.nl

Abstract

[Draw your reader in with an engaging abstract. It is typically a short summary of the document.   
When you’re ready to add your content, just click here and start typing.]

Pacman AI

Levi Vlasblom

Contents

[1. Inleiding 2](#_Toc200806008)

[2. Mogelijke benaderingen 2](#_Toc200806009)

[2.1 Search-based algoritmen 2](#_Toc200806010)

[2.2 Reinforcement Learning (RL) 2](#_Toc200806011)

[3. Vergelijking van benaderingen 3](#_Toc200806012)

[4. Gemaakte keuze: Search-based AI 3](#_Toc200806013)

[5. Mogelijke uitbreiding: Reinforcement Learning 3](#_Toc200806014)

[Bibliography 4](#_Toc200806015)

# 1. Inleiding

In dit hoofdstuk wordt onderbouwd welk AI-algoritme gekozen is voor de implementatie van een Pacman-agent. Verschillende algoritmische benaderingen zijn verkend, waaronder search-based algoritmes en reinforcement learning. De uiteindelijke keuze is gebaseerd op toepasbaarheid, uitlegbaarheid en de beschikbare functionaliteit binnen het bestaande Java-framework. <https://github.com/davidrobles/pacman-vs-ghosts>

# 2. Mogelijke benaderingen

## 2.1 Search-based algoritmen

Search-algoritmen benaderen het probleem door het spel te modelleren als een beslisboom. Omdat het spel deterministisch is, is het mogelijk om de gevolgen van acties te simuleren voordat ze daadwerkelijk worden uitgevoerd. De Game-klasse biedt ondersteuning voor deze aanpak via methoden als copy() en advanceGame(...). Dit framework is dus gericht om modellen te ondersteunen.

**Voorbeelden van toepasbare search-methodes:**

* Depth-limited lookahead
* Greedy Best-First Search
* Minimax (zonder tegenstander, of met neutrale ghosts)
* A\* voor padplanning naar doelen
* Breath-First Search

“Search-based methods are particularly effective in deterministic environments with perfect information”  
*(Russell & Norvig, 2021, Artificial Intelligence: A Modern Approach)*

## 2.2 Reinforcement Learning (RL)

Bij reinforcement learning leert een agent door beloningen te maximaliseren op basis van feedback. In het geval van Pacman kan de agent bijvoorbeeld beloond worden voor het verzamelen van punten en gestraft worden voor het sterven.

Hoewel RL theoretisch zeer geschikt is voor games, vereist het:

* Een lange trainingsperiode (veel simulaties)
* Een consistente representatie van de game state
* Reward shaping en episode management

“While powerful, RL methods often require thousands of episodes to converge on good policies”  
*(Sutton & Barto, 2018, Reinforcement Learning: An Introduction)*

# 3. Vergelijking van benaderingen

|  |  |  |
| --- | --- | --- |
| Eigenschap | Search-based AI | Reinforcement Learning |
| Deterministische simulatie | Direct toepasbaar via Game.copy() | Vereist training over episodes |
| Difficulty | Simuleerbare beslissingen | Moeilijker te interpreteren |
| Leercurve | Past bij huidig kennisniveau | Komt later in curriculum |
| Prestaties in beperkte tijd | Betrouwbaar en voorspelbaar | Afhankelijk van trainingskwaliteit |
| Frameworkintegratie | Volledig ondersteund | Niet standaard in Java-framework(Python Prefered) |

# 4. Gemaakte keuze: Search-based AI

Op basis van bovenstaande analyse is gekozen voor een search-based benadering. Deze sluit goed aan bij:

* De deterministische aard van het spel
* De mogelijkheid tot simulatie binnen de bestaande Game-architectuur
* De vereiste uitlegbaarheid van beslissingen binnen het verslag
* De tijdsbeperkingen van het project

De gekozen AI zal verschillende search mogelijkheden toepassen om vergelijking te onderscheiden en verder uitgewerkt worden in de ideaalste search benadering. Echter zal er wel naar Reinforcement Learning gekeken worden hoe het toegepast zou kunnen worden met het bestaande framework.

# 5. Mogelijke uitbreiding: Reinforcement Learning

Indien tijd en kennis het toelaten, kan reinforcement learning later worden toegepast als uitbreiding of vergelijkend experiment. Dit zou bijvoorbeeld inhouden:

* Een eenvoudige Q-learning agent
* Beloningen op basis van game.getScore() en game.gameOver()
* Vergelijking van prestatie, consistentie en interpretatie met de search-agent

# 6. Fase 1 Search implementations & Design

## 6.1 Greedy Search

Het doel van Greedy search is de snelste mogelijk kiezen omdat de agent weet waar het juiste resultaat is. Hiermee kan dus aangetoond worden dat de AI Agent de juiste richtingen opgaat binnen het Pac-man spel.

6.2 AI-feedback en validatie  
Om het gedrag van de AI-agent te verifiëren, is gekozen om per uitgevoerde zet diagnostische informatie zichtbaar te maken. In eerste instantie gebeurde dit via standaard console-uitvoer (System.out.println()), maar al snel is gekozen voor een geïntegreerde visuele overlay binnen het bestaande GameView-scherm. Hiermee worden tijdens het spel de gekozen zet, score-evaluatie en eventuele interne beslisparameters getoond, wat de interpretatie en validatie van het gedrag aanzienlijk vereenvoudigt.

# Bibliography

Course, A. (2025). Java Game Framework: Pacman. Breda.

Russle, s. &. (2021). *Artificial Intelligence: A Modern Approuch (4th ed.).* Pearson.

Sutton, R. S. (2018 ). *Reinforcement Learning: An Introduction (2nd ed.). .* MIT Press.