

Battlesnake

Gruppe 19

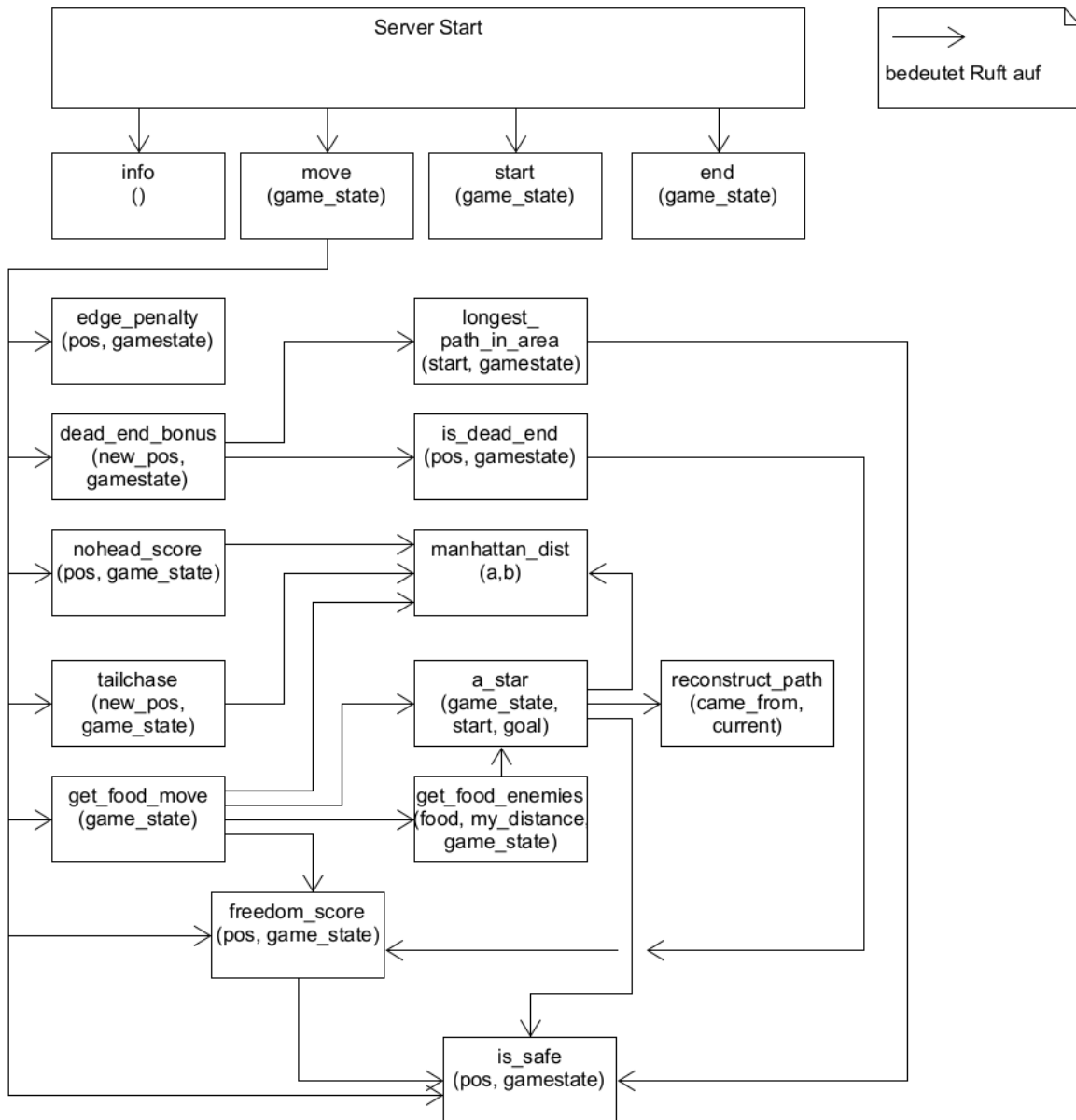
2025-07-04

URL des Bots: replit.dev
Github-Projekt: <https://github.com/Mattis-Brinker/KI>
Abgabetermin: 06.07.2025

Brinker, Mattis Paul	222200558
Rose, Lennert	222201353
Schlüter, Theo Peter	222201541
Schubert, Philipp	220200128

Aufgabe 1. Stellen Sie ihren Bot als UML Diagramm dar! Kennzeichnen Sie dabei welche Datei/Methode die Agentenfunktion ihres Bots enthält!

Im folgenden Diagramm sind die Methoden die beim Serverstart bzw. in jedem konsekutiven Zyklus abgerufen werden modelliert:



Es folgt eine Übersicht, wie die Funktionen in einzelne Python Dateien unterteilt sind.

Astar.py:

- get_food_move
- get_food_enemies
- a_star
- reconstruct_path

Dead_End.py:

- dead_end_bonus
- is_dead_end
- longest_path_in_area

Helper.py:

- get_new_position
- get_direction
- manhatton_dist
- tailchase
- freedom_score

Scoring.py:

- edge_penalty
- no_head_score


Main.py:

- info
- start
- end
- move

Aufgabe 2. Welche Art von Agent stellt ihr Bot dar? Beschreiben Sie welche Konzepte aus der Vorlesung Sie für ihren Bot nutzen und begründen Sie, warum Sie sich für diese Konzepte entschieden haben!

Das Programm stellt einen nutzen-basierten Agenten dar. Dazu wurde der A*-Algorithmus implementiert um den kürzesten bzw. bezüglich der Kosten optimalen Pfad zum Essen zu finden. Der Abstand fließt dann in die Berechnung der Scores für die erlaubten Bewegungsrichtungen ein.

Neben A* wurde u.A. auch ein Freedomscore berücksichtigt, mit dem überprüft werden kann in welche Richtung sich die Schlange bewegen muss um am Sichersten von konkurrierenden Schlangen zu sein. Des Weiteren wird die Position der nächsten „Enemy“-Bots berücksichtigt um potenzielle Gefahren oder Gefährdungssituation optimal zu nutzen. Da das Feld endlich ist und es Ecken gibt an denen die Handlungsmöglichkeit des Bots weiter eingeschränkt sind vermeidet der Bot in diese Positionen zu gelangen.

Diese und weitere Parameter fließen in den Score ein. Die Schlange bewegt sich entsprechend in die Richtung mit dem höchsten Score. 

Aufgabe 3. Welchen weiteren Überlegungen sind ihren Bot eingeflossen bzw. welche zusätzlichen Strategien verwendet ihr Bot? Warum haben Sie sich für die Verwendung dieser Strategien entschieden?

Zusätzlich wird mit freedom_score ein floodfill-Algorithmus zur Bestimmung aller erreichbaren Felder implementiert. Dieser dient zur Vermeidung von Sackgassen. Er berücksichtigt allerdings nicht, ob auch ein Pfad durch alle berechneten Felder möglich ist.

Zudem wird, wenn der Abstand zwischen Kopf und Schwanz 1 beträgt durch tailchase ein score berechnet. Das bedeutet, wenn die Schlange direkt an ihrem Schwanz ist, wird zu einem gewissen Maß die Verfolgung des Schwanzes priorisiert, da dies solange man durch keine andere Schlange bedroht wird, eine sichere Option ist.


Die Funktion no_head_score überprüft, ob sich Gegner in dem nächsten Zug auf dasselbe Feld bewegen können bewertet solch einen Zug für einen größeren oder gleich großen Gegner negativ und für kleinere Gegner positiv. Die Funktion dient also zum Teil zum Angriff kleiner Gegner und Kollisionsvermeidung bei größeren Schlangen.

Generell sind zentralere Felder sicherer, da es potenziell mehr Bewegungsoptionen gibt bzw. die Schlange schlechter in Sackgassen gezwungen werden kann. Dies berücksichtigt edge_penalty, indem weiter außen gelegene Felder leicht bestraft werden.

Wenn sich die Schlange in einer Sackgasse befindet ist, es sinnvoll einen Pfad möglichst großer Länge zu berechnen. Das ist die Aufgabe von dead_end_bonus. Die Funktion berechnet solch einen Pfad und belohnt den ersten Schritt des Pfades.

Aufgabe 4. Betrachten Sie die Umgebung von Battlesnake genauer! Welche Eigenschaften bringt Sie mit? Nutzen Sie dazu die in der Vorlesung und Übung behandelten Konzepte und Begriffe!

Die Eigenschaften der Umgebung der Battlesnakes sieht wie folgt aus:

Observabel:	Das gesamte Spielfeld ist einsehbar.
Nicht-deterministisch:	Die Apfelpositionen sind zufällig. (Es kann jedoch auch für deterministisch argumentiert werden, da die Generierung der Äpfel nach gewissen Spielregeln erfolgt) 
Sequentiell:	Ein Zug folgt auf den anderen. Durch den vorherigen Zug werden darauffolgende Optionen eingeschränkt. Ein Zug nach rechts führt dazu, dass daraufhin ein Zug nach links zum Tod führt.
Statisch:	Alle Aktionen der Schlangen erfolgen parallel zueinander.
Diskret:	Es gibt eine endliche Anzahl von Feldern, Schlangen, und Zugoptionen pro Schlange.
Multi-Agent:	Es gibt mehrere Gegnerschlangen, welche alle als Agenten fungieren.