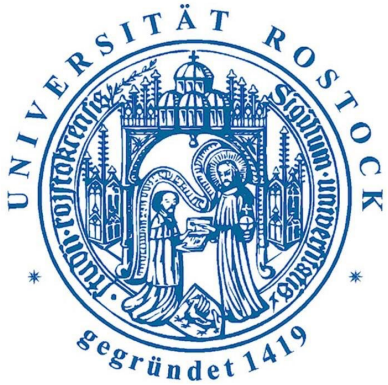



# KI Programmierprojekt Sommersemester 2025

**Gruppe 17**
Arian Akbari, 224100357
Hüseyin Kayabasi, 223201801
Johannes Langer, 221200970
Helin Oguz, 223202103
Cagla Yesildogan, 223201881

6. Juli 2025

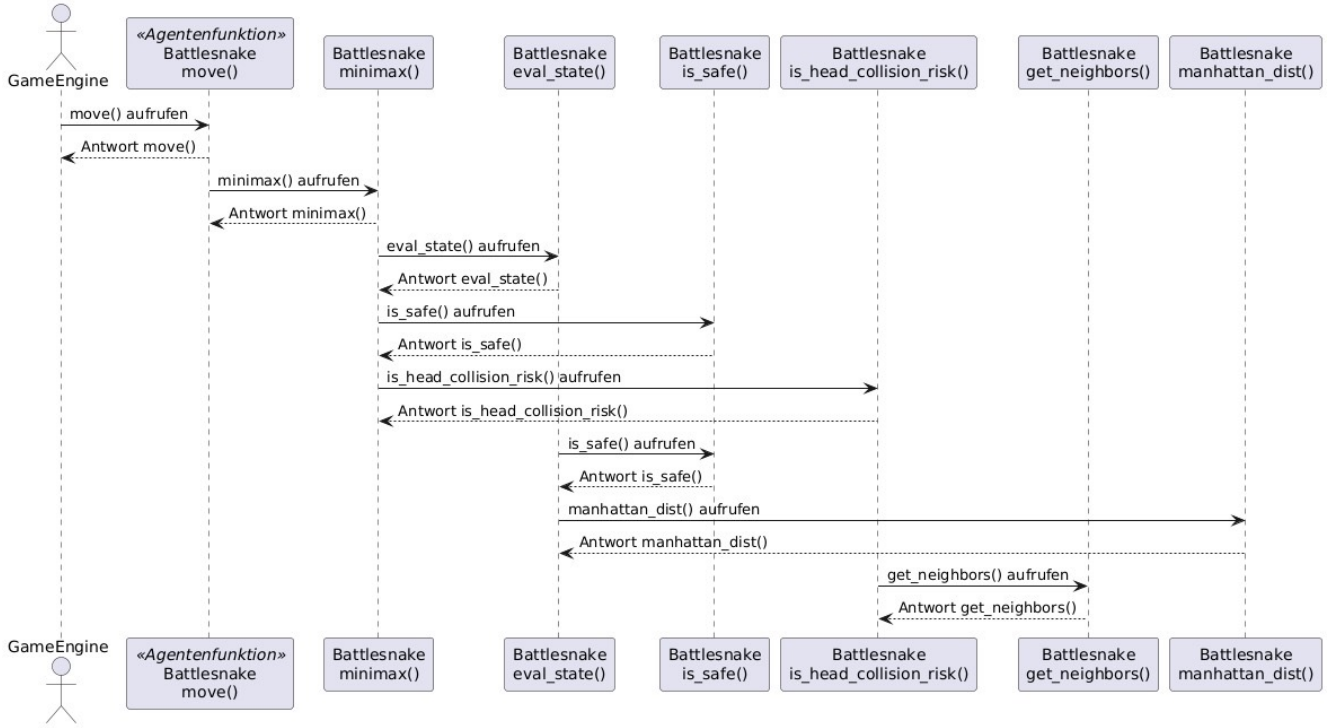
# 1 UML Diagramm

Wir haben uns für ein **Sequenzdiagramm** entschieden, da dieses Diagramm den zeitlichen Ablauf der Methodenaufrufe und die Agentenfunktion klar und verständlich darstellt.

Die folgende Abbildung zeigt den Aufruf- und Entscheidungsablauf des Bots:

- Der Game Engine ruft die `move()`-Methode auf.
- Die `move()`-Methode ist die Agentenfunktion, da hier die Entscheidung getroffen wird, in welche Richtung sich die Schlange bewegen soll.
- Von `move()` wird `minimax()` aufgerufen, um die beste mögliche Bewegung zu berechnen.
- Die Methode `minimax()` ruft dabei mehrere Hilfsmethoden auf:
  - `eval_state()`: Bewertet die aktuelle Situation.
  - `is_safe()`: Prüft, ob ein Feld sicher ist.
  - `is_head_collision_risk()`: Prüft das Risiko einer Head-to-Head-Kollision.
  - `get_neighbors()`: Liefert benachbarte Felder.
  - `manhattan_dist()`: Berechnet die Entfernung zu Zielen (z.B. Nahrung).

Die Agentenfunktion ist also die `move()`-Methode. Diese ist im Sequenzdiagramm klar gekennzeichnet.



# 2 Agent

Unser Bot ist ein **Goal-Based Agent**. Der Bot hat das Ziel, so lange wie möglich zu überleben. Dabei passt er sein Verhalten dynamisch an:

- Bei niedriger Gesundheit priorisiert er das Erreichen von Nahrung.
- Wenn die Schlange groß genug ist, konzentriert sie sich auf das Überleben und vermeidet enge Räume.

Wir haben uns für dieses Agentenmodell entschieden, weil es in Battlesnake entscheidend ist, flexibel auf die Situation zu reagieren und Ziele situativ zu ändern. Ein reiner Reflex-Agent wäre nicht ausreichend, da komplexere Entscheidungsschritte (z.B. Minimax, Sicherheitsprüfung) benötigt werden.

# 3 Überlegungen zum Bot

Unser Bot verwendet folgende zusätzliche Strategien, die über einfache Bewegungsregeln hinausgehen:

- **Minimax-Algorithmus mit Alpha-Beta-Pruning:**
  Der Bot berechnet die möglichen zukünftigen Züge in mehreren Ebenen und versucht, den bestmöglichen Zug zu wählen, indem er potenzielle Gefahren minimiert und Vorteile maximiert.
  *Warum?* Dadurch kann der Bot besser planen und frühzeitig riskante Situationen vermeiden.
- **Head-to-Head-Kollisionsvermeidung:**
  Der Bot berücksichtigt, ob gegnerische Schlangen in der Nähe sind und vermeidet bewusst Felder, auf die größere oder gleich große Schlangen als nächstes ziehen könnten.
  *Warum?* Um unnötige Kopf-an-Kopf-Kollisionen zu vermeiden und das Überleben zu sichern.
- **Dynamische Futter-Priorisierung:**
  Der Bot priorisiert das Fressen von Nahrung nur bei niedrigem Gesundheitswert oder kleiner Körpergröße. Ab einer bestimmten Körpergröße ignoriert der Bot bewusst Nahrung und konzentriert sich darauf, in sicheren offenen Bereichen zu bleiben.
  *Warum?* Große Schlangen neigen dazu, sich selbst einzukesseln. Die Begrenzung der Körpergröße erhöht die Überlebenschance.

Diese Strategien machen den Bot flexibler und deutlich überlebensfähiger im Vergleich zu einfachen Reflex-Agenten.

# 4 Umgebung von Battlesnake

Die Umgebung von Battlesnake lässt sich folgendermaßen charakterisieren:

- **Dynamisch:**
  Die Umgebung verändert sich ständig durch die Bewegungen der Schlangen und das Erscheinen von Nahrung. Jeder Zug beeinflusst die zukünftige Spielsituation.
- **Teilweise beobachtbar (partiell beobachtbar):**
  Obwohl das Spielbrett sichtbar ist, sind die zukünftigen Bewegungen der Gegner unbekannt. Man sieht den aktuellen Zustand, aber nicht die zukünftigen Aktionen der anderen Agenten.
- **Deterministisch:**
  Die Spielregeln sind klar definiert und jede Aktion hat eine eindeutige Folge, ohne Zufallselemente. Wenn man die Züge kennt, kann man das Ergebnis exakt vorhersagen.
- **Mehragentensystem:**
  Es gibt mehrere Agenten (Schlangen), die unabhängig voneinander agieren und konkurrieren. Die Entscheidungen eines Agenten beeinflussen die anderen.

Diese Eigenschaften machen Battlesnake zu einem anspruchsvollen Umfeld für die Entwicklung eines Agenten. Man benötigt Strategien, die mit unvollständiger Information und dynamischen Veränderungen umgehen können.