Projeto de IA

Hugo Dias - 114142
Francisco Matos - 113726
Ruben Costa - 114190

OBJETIVO

DESENVOLVER UM AGENTE INTELIGENTE PARA JOGAR O JOGO DA SNAKE, PARA CAPTURAR ALIMENTOS E EVITAR COLISÕES COM PAREDES E OUTRAS SNAKES DE FORMA EFICIENTE

23/12/2024

Lógica e funcionamento do agente

- A cada iteração do jogo, o agente recebe o estado atualizado e recalcula suas prioridades de movimento.
  - O agente utiliza informações do estado atual do jogo fornecidas pelo servidor, incluindo a posição do corpo da cobra, o campo de visão (sight), e obstáculos, como paredes, outras cobras e frutas.
  - Baseado nesses dados, o agente avalia os movimentos possíveis e identifica posições seguras e perigosas.
  - Cada direção possível (w, a, s, d) recebe uma prioridade com base em fatores como proximidade de obstáculos, presença de comida, ou necessidade de evitar colisões.
  - Possui a capacidade de considerar movimentos futuros para minimizar riscos e garantir espaços seguros.
- Ele usa um algoritmo que cria um movimento em zigzag a partir das funções (def_map_limits, change_row), estratégias de evasão e priorização dinâmica para determinar a melhor direção.

Exemplo: Início → Movimenta na direção atual → Verifica os limites (def_map_limits) → Altera linha ou coluna (change_row) → Repete

Arquitetura

Características

- Deliberação e Planejamento: O agente não reage apenas de maneira reflexiva, mas analisa e planeja ações com base em seus objetivos e no conhecimento do ambiente.
- Atualização Contínua: Com cada nova interação com o ambiente, o agente ajusta seu estado interno e seus planos.
- Orientação por Objetivos: Todas as decisões são guiadas por um objetivo claro, permitindo ações mais direcionadas e estratégicas.

Benchmark do Agente

Testes

Fizemos testes com a sight>4 para ver se tinha uma taxa de sobrevivência maior e apanha mais fruta (a evitar super frutas)
Também fizemos para decidir quando comer frutas especiais, vimos que seria melhor comer só quando tiver muitas disponiveis. Assim no caso de o traverse == False podiamos reverter isso mais facilmente

Critérios Avaliados

A eficiência na coleta de Alimentos
A taxa de sobrevivência
Conforme comiamos super frutas sempre ou só em certos momentos (como falo anteriormente)

Funções para escolha de movimento

- order_directions_priorities (direction, priority) :

  Reordena as prioridades de direção.

  Exemplo: Pode priorizar ir para a direita (porque há comida) ou evitar ir para baixo (por causa de obstáculos).

- get_aim (pos, snake)

  Calcula a direção ideal para alcançar uma posição-alvo específica (geralmente comida).

- get_directions (pos)

  Lista as direções possíveis a partir de uma posição específica. É uma base para calcular movimentos válidos.

- choose_direction (body, sight)

  Ordena a lista de direções conforme as prioridades.

Funções para decisão do movimento

best_direction (snake, food, obstacles)

A função principal para tomar decisões. Analisa as informações coletadas e escolhe o movimento mais seguro e eficiente. Usa a lista de direções ordenadas por prioridade retornadas por choose_direction. Preve os aconteceimentos de movimentos futuros e apartir disso e cria listas com movimentos para cada possível acontecimento e seleciona conforme a lista que tem as direções ordenadas por prioridade.

Exemplo da lista de direções com as suas prioridades:

```
{'w': [1, 7462], 'd': [1, 7461], 's': [1, 7460], 'a': [1, 7458]}
```