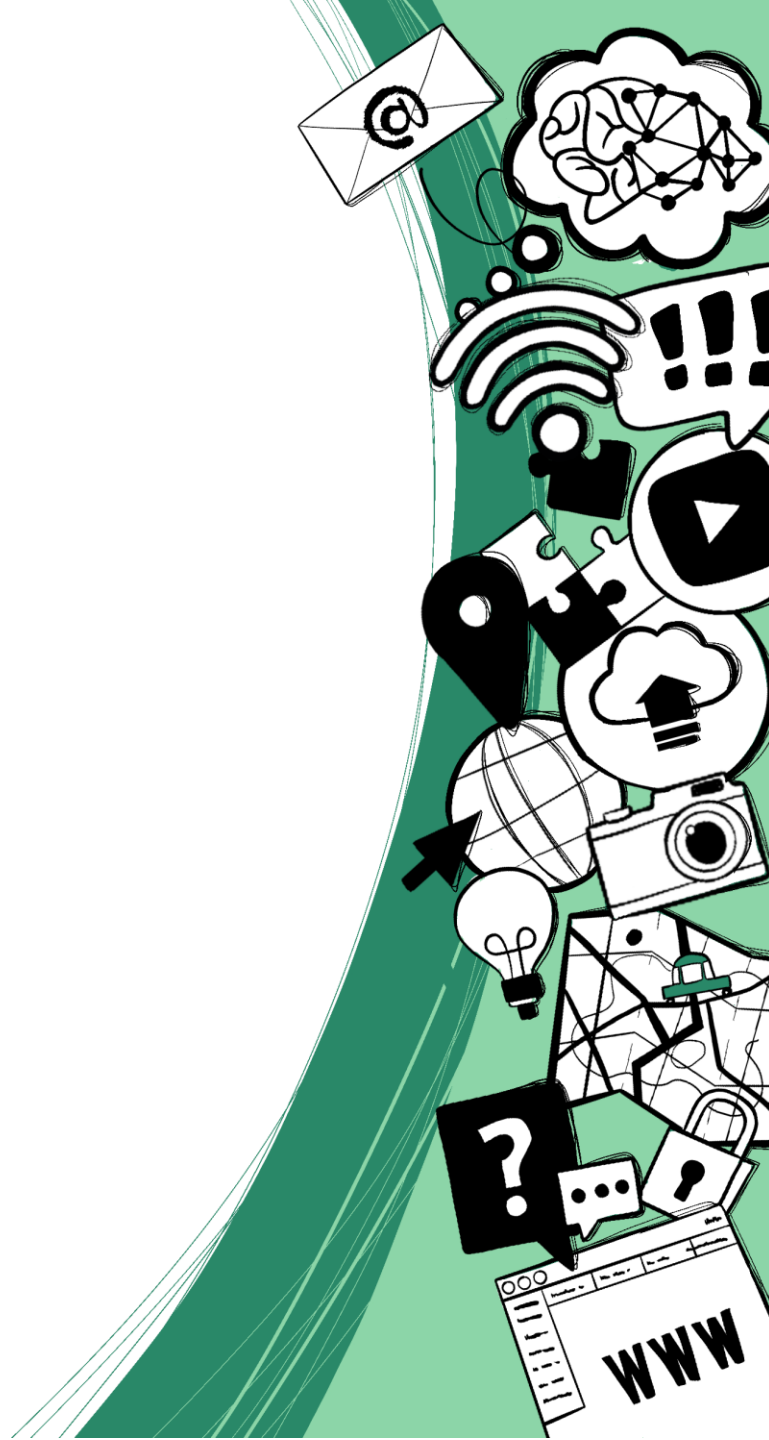




Reinforcement Learning



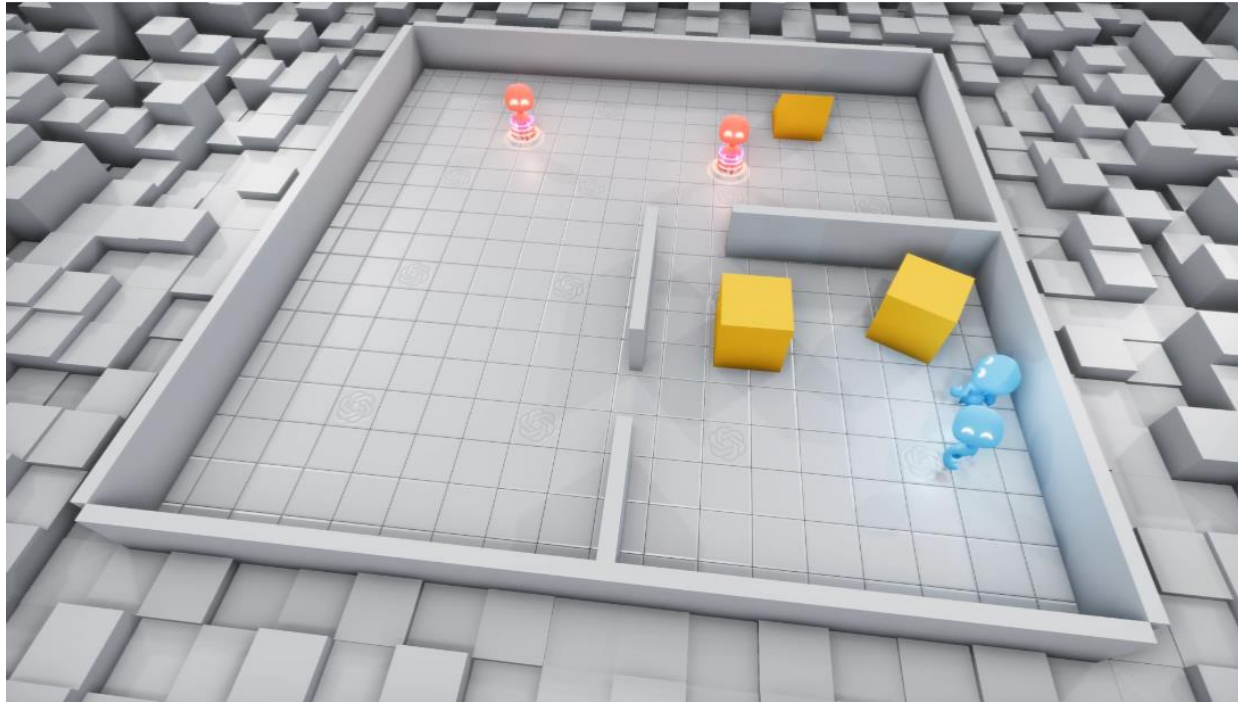
Übungen



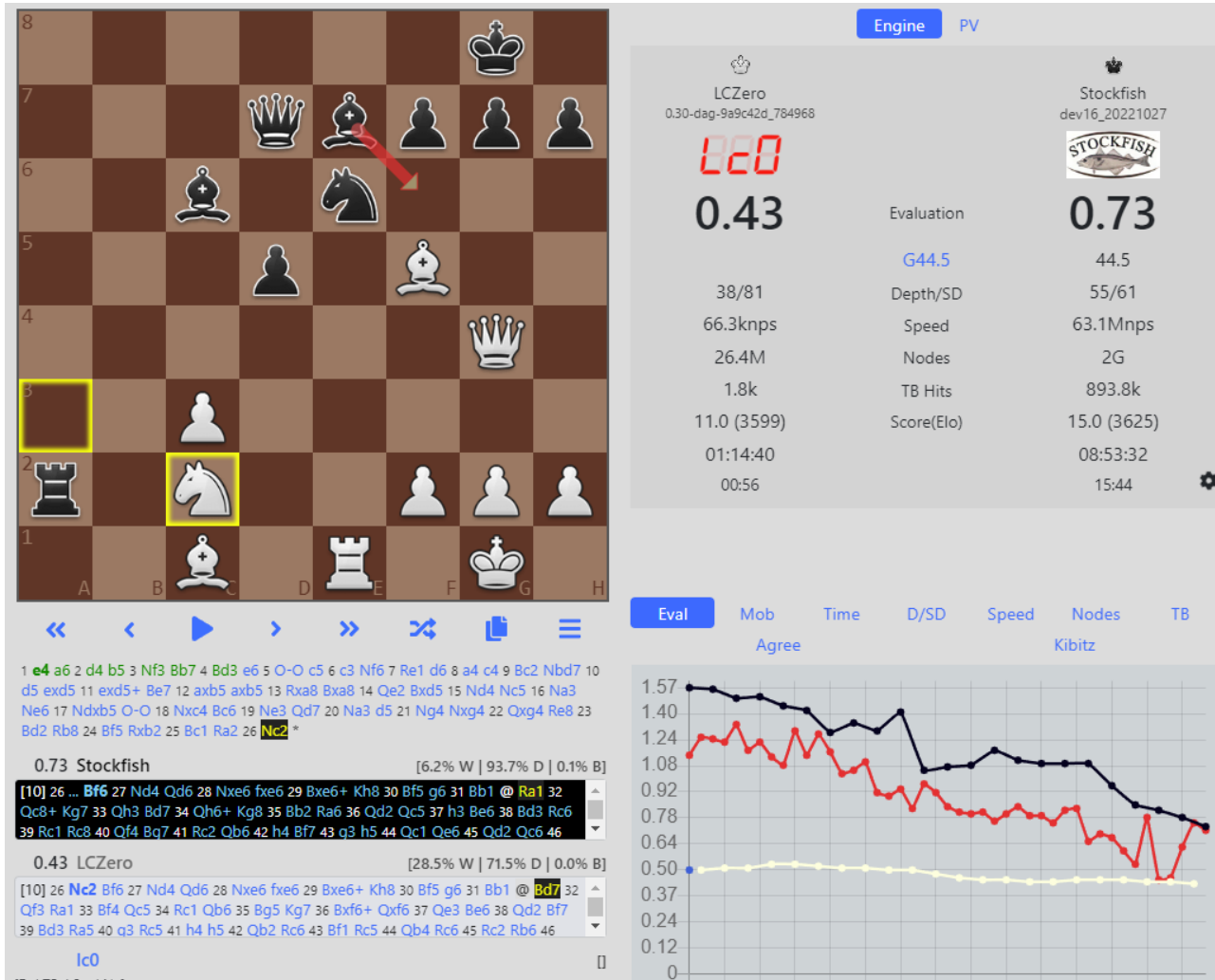


OpenAI Hide and Seek

<https://openai.com/blog/emergent-tool-use/>



Leela Chess Zero (Lc0)



The screenshot displays a chess engine interface with the following components:

- Chessboard:** Shows a chessboard with pieces. A red arrow points to a knight on e6, and a yellow box highlights a knight on b2.
- Engine Comparison:**

Engine	Score	Evaluation
LCZero (0.30-dag-9a9c42d_784968)	0.43	G44.5
Stockfish (dev16_20221027)	0.73	44.5
- Performance Metrics:**

Metric	LCZero	Stockfish
38/81	66.3knps	44.5
26.4M	Speed	55/61
1.8k	Nodes	63.1Mnps
11.0 (3599)	TB Hits	2G
01:14:40	Score(Elo)	893.8k
00:56		15.0 (3625)
- Performance Graph:** A line graph showing performance over time. The y-axis ranges from 0 to 1.57. The x-axis represents time. A red line (Stockfish) starts at ~1.4 and drops to ~0.7. A black line (LCZero) starts at ~1.5 and drops to ~0.4. A yellow line (LCZero) starts at ~0.5 and drops to ~0.4.
- Game History:**

1 e4 a6 2 d4 b5 3 Nf3 Bb7 4 Bd3 e6 5 O-O c5 6 c3 Nf6 7 Re1 d6 8 a4 c4 9 Bc2 Nbd7 10 d5 exd5 11 exd5+ Be7 12 axb5 axb5 13 Rxa8 Bxa8 14 Qe2 Bxd5 15 Nd4 Nc5 16 Na3 Ne6 17 Ndx5 O-O 18 Nxc4 Bc6 19 Ne3 Qd7 20 Na3 d5 21 Ng4 Nxc4 22 Qxg4 Re8 23 Bd2 Rb8 24 Bf5 Rxb2 25 Bc1 Ra2 26 **Nc2***

0.73 Stockfish [6.2% W | 93.7% D | 0.1% B]

[10] 26 ... Bf6 27 Nd4 Qd6 28 Nxe6 fxe6 29 Bxe6+ Kh8 30 Bf5 g6 31 Bb1 @ Ra1 32 Qc8+ Kg7 33 Qh3 Bd7 34 Qh6+ Kg8 35 Bb2 Ra6 36 Qd2 Qc5 37 h3 Be6 38 Bd3 Rc6 39 Rc1 Rc8 40 Qf4 Bg7 41 Rc2 Qb6 42 h4 Bf7 43 g3 h5 44 Qc1 Qe6 45 Qd2 Qc6 46

0.43 LCZero [28.5% W | 71.5% D | 0.0% B]

[10] 26 **Nc2** Bf6 27 Nd4 Qd6 28 Nxe6 fxe6 29 Bxe6+ Kh8 30 Bf5 g6 31 Bb1 @ **Bd7** 32 Qf3 Ra1 33 Bf4 Qc5 34 Rc1 Qb6 35 Bg5 Kg7 36 Bxf6+ Qxf6 37 Qe3 Be6 38 Qd2 Bf7 39 Bd3 Ra5 40 g3 Qc5 41 h4 h5 42 Qb2 Rc6 43 Bf1 Rc5 44 Qb4 Rc6 45 Rc2 Rb6 46

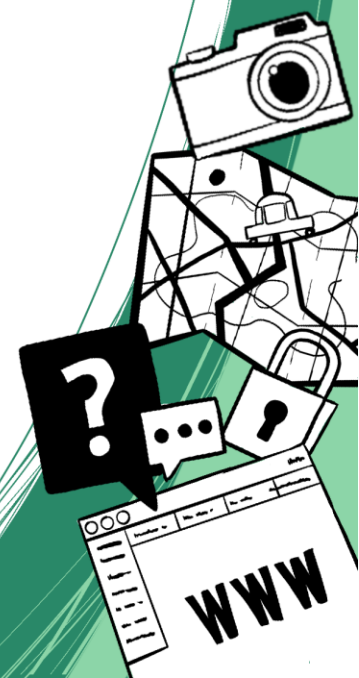


Alpha Star

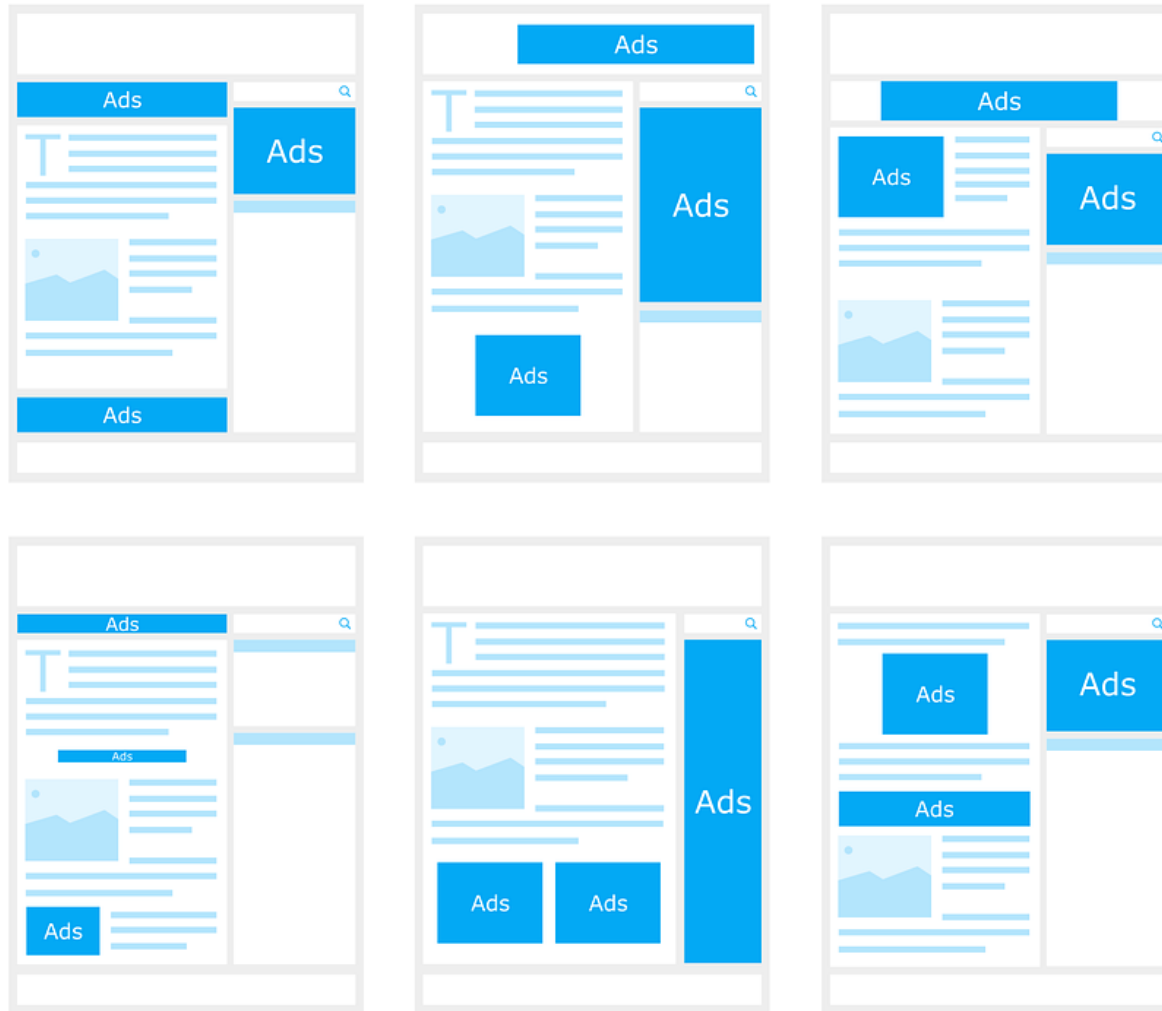


14:32
Catalyst LE

Player	Supply	Minerals	Gas	Workers	Army	APM	Production
AlphaStar	177 / 200	945 +2015	758 +873	64	113	940	2 1
LiquidTLO	147 / 172	335 +1595	442 +1030	61	86	1377	2 2



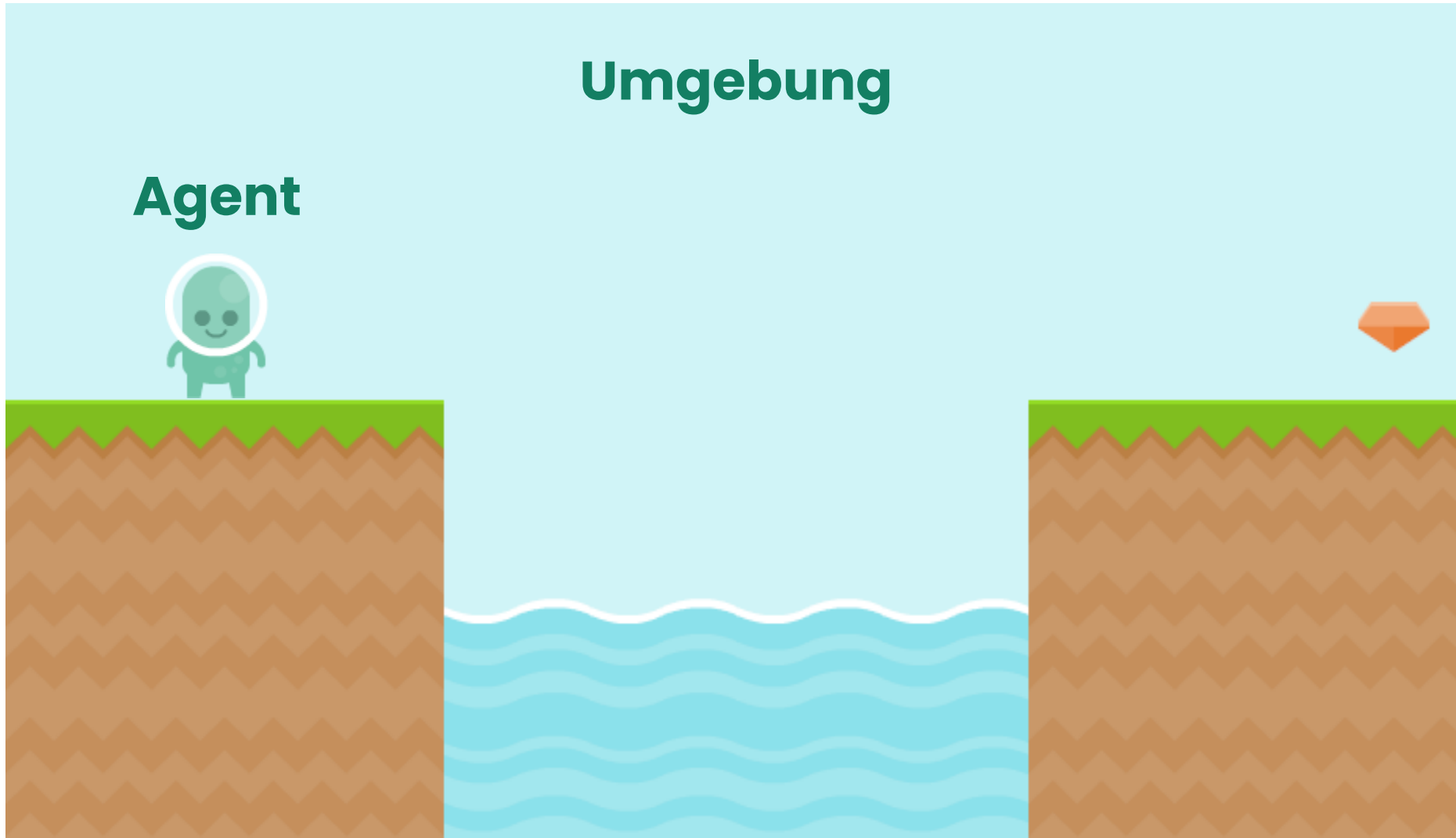
Benutzerdefinierte Werbung



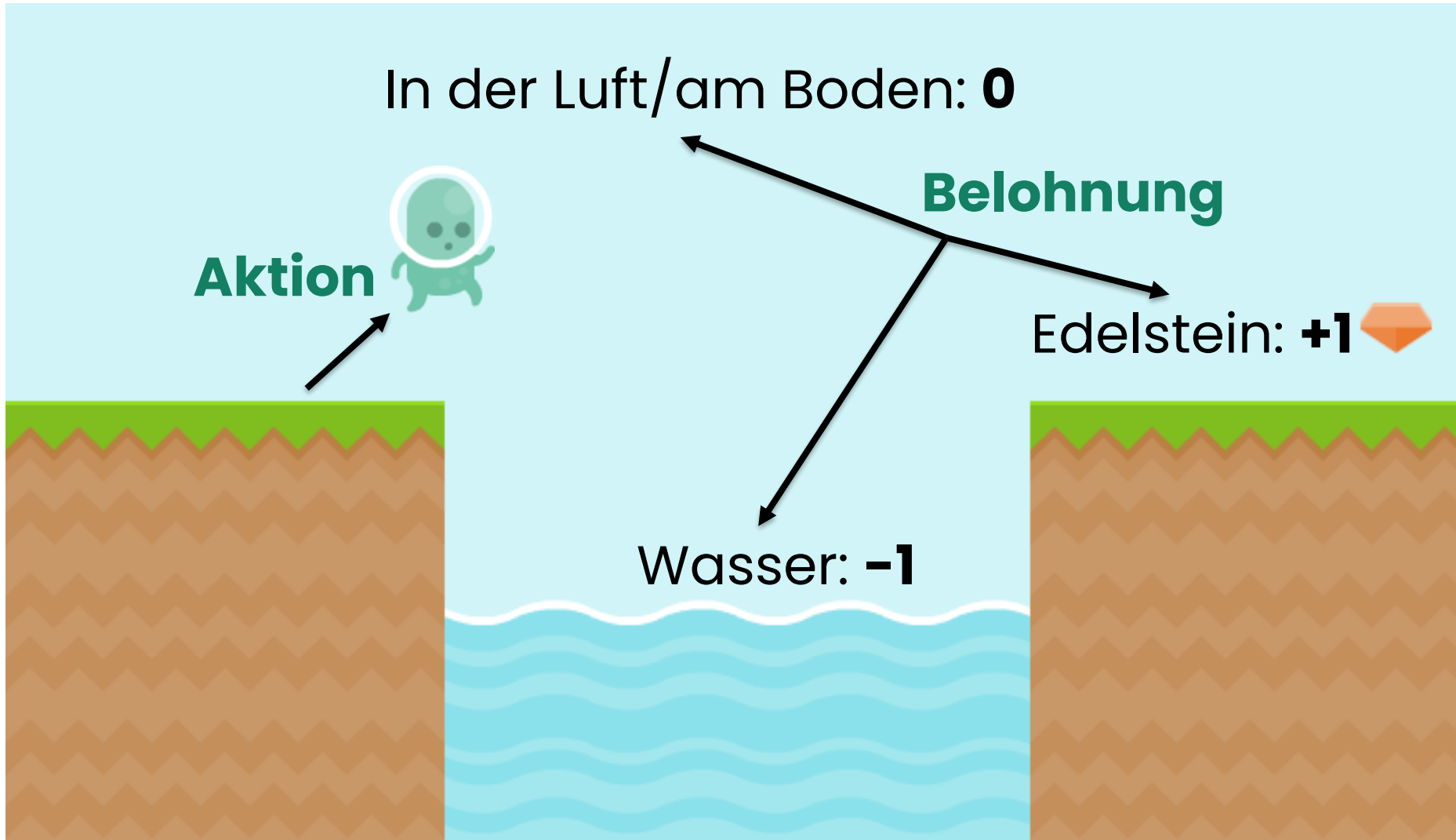
RL Basics



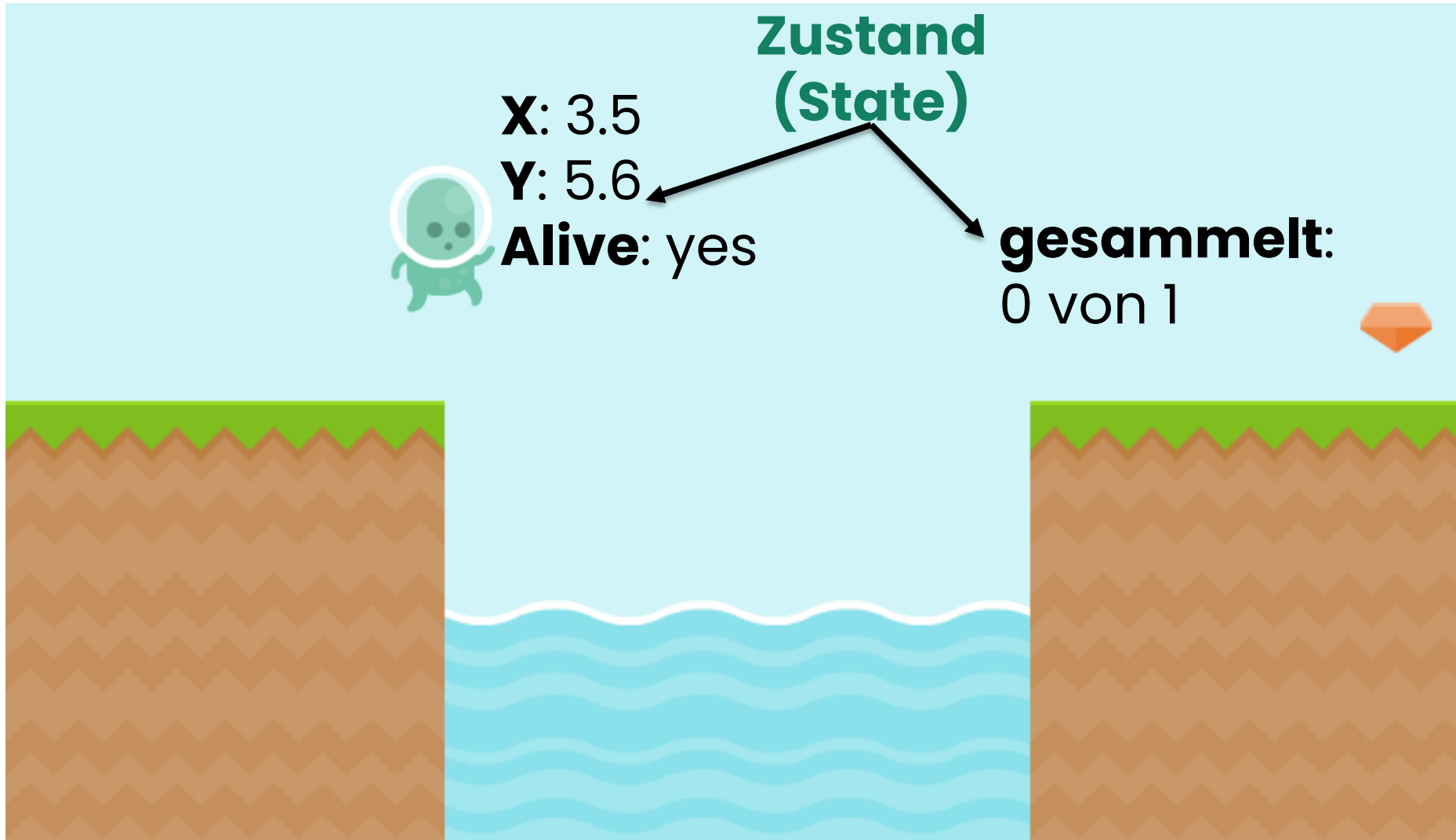
Was wir haben



Was wir haben



Was wir haben





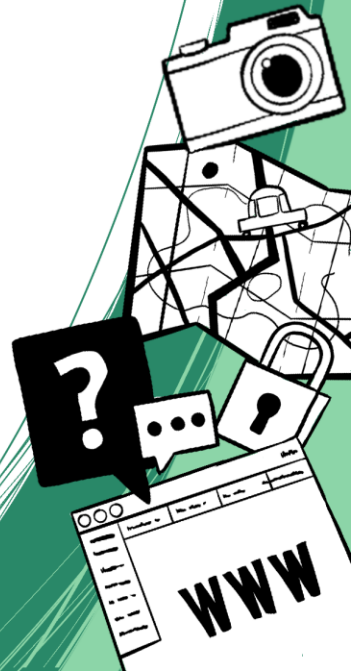
Beispiel - TicTacToe





Beispiel - TicTacToe

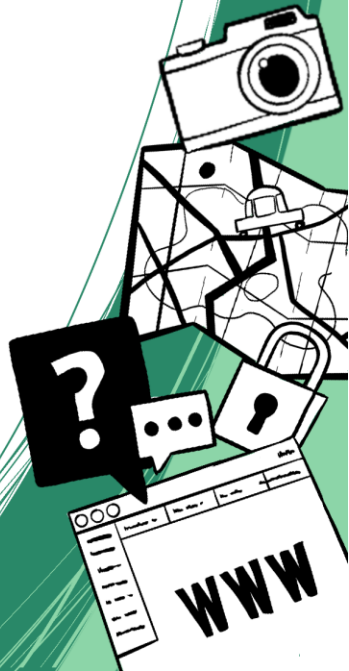
Agent



Beispiel - TicTacToe

Agent

- Spieler (X, O)

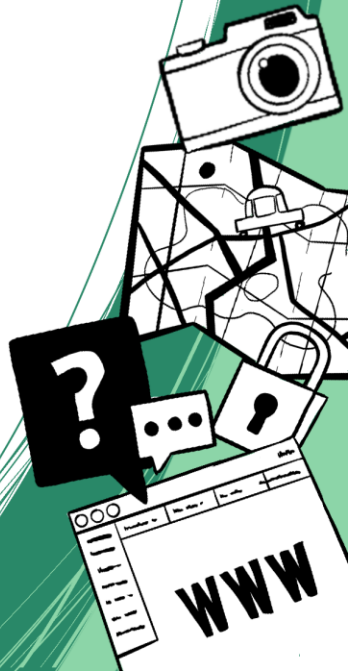


Bespiel - TicTacToe

Agent

- Spieler (X, O)

Zustand

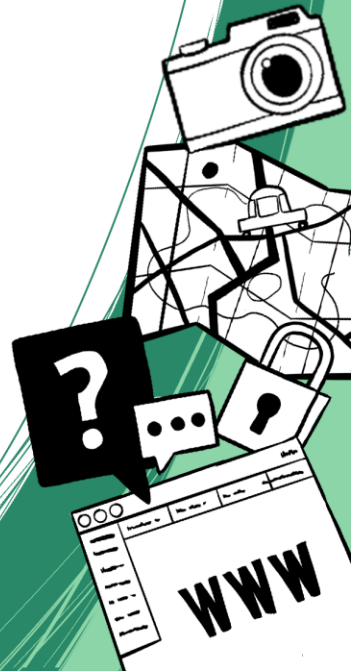
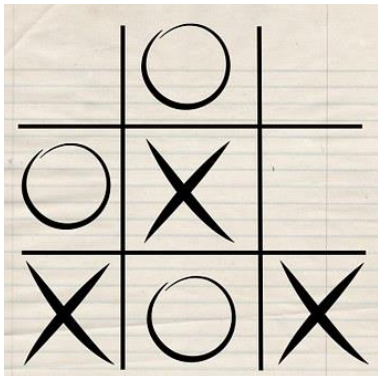


Bespiel - TicTacToe

Agent

- Spieler (X, O)

Zustand



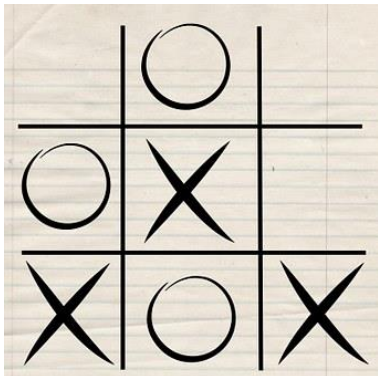
Beispiel - TicTacToe

Agent

- Spieler (X, O)

Aktionen

Zustand

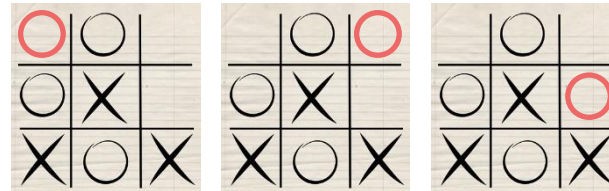


Beispiel - TicTacToe

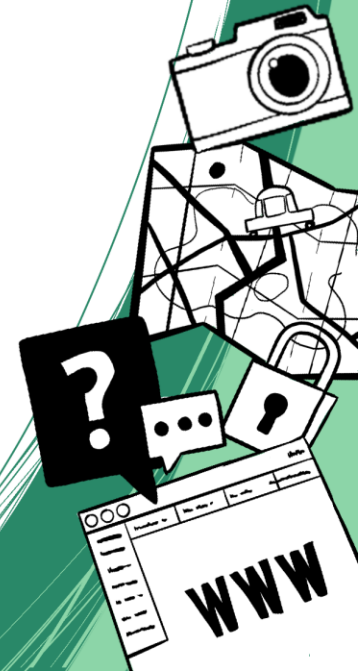
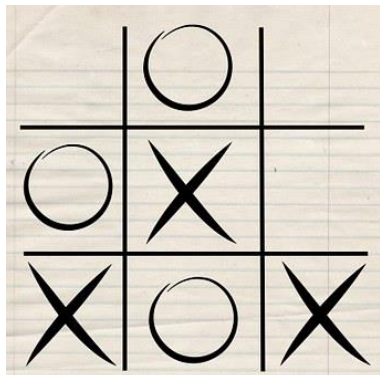
Agent

- Spieler (X, O)

Aktionen



Zustand

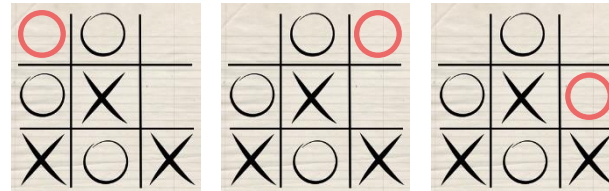


Beispiel - TicTacToe

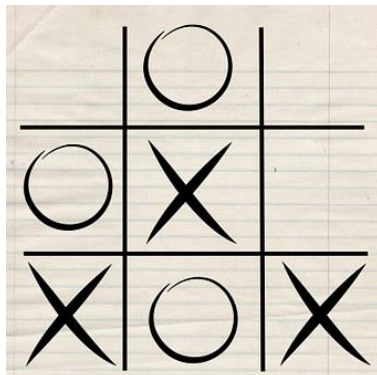
Agent

- Spieler (X, O)

Aktionen



Zustand



Belohnungen

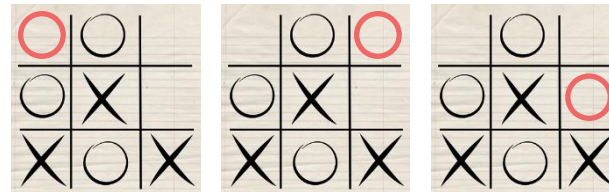


Beispiel - TicTacToe

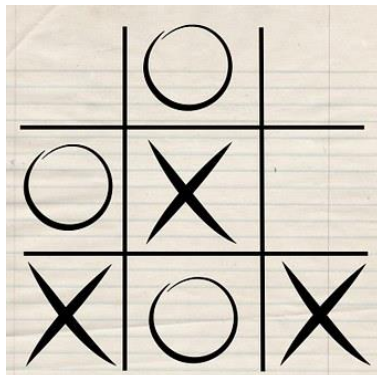
Agent

- Spieler (X, O)

Aktionen



Zustand



Belohnungen

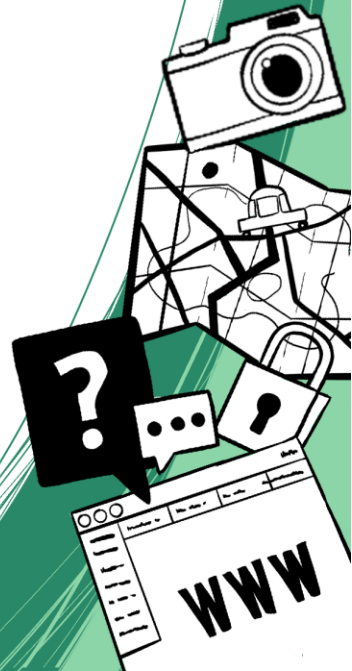
- gewonnen: **+1**
- verloren: **-1**
- sonst: **0**



Weitere Beispiele

Was sind **Agenten, States, Aktionen** und **mögliche Belohnungen** in...

- ...Leela Chess Zero
- ...OpenAI Hide and Seek
- ...Custom Advertisement



Q-Lernen

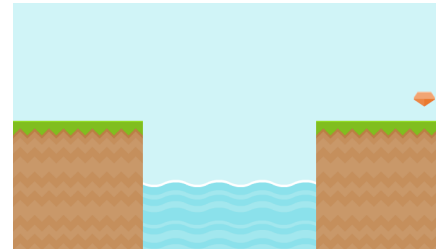


RL-Schleife

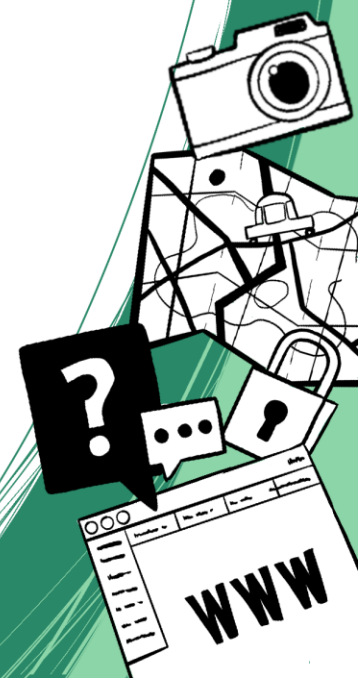
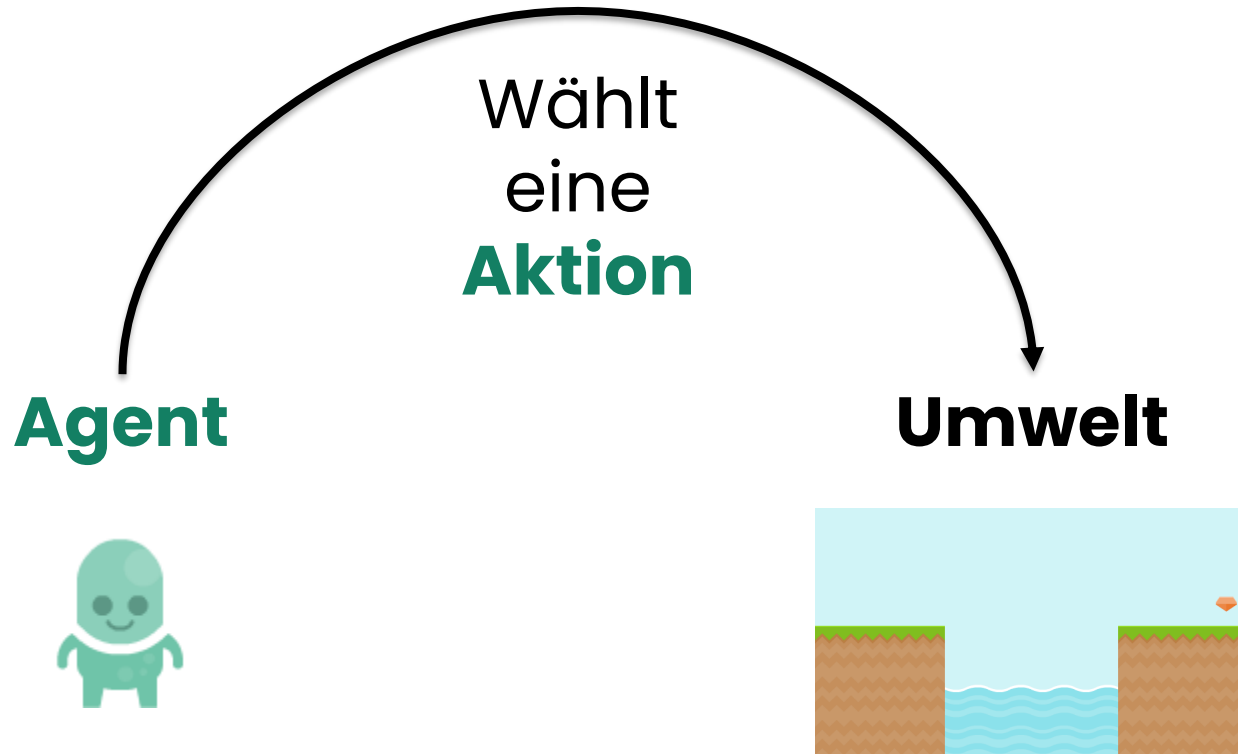
Agent



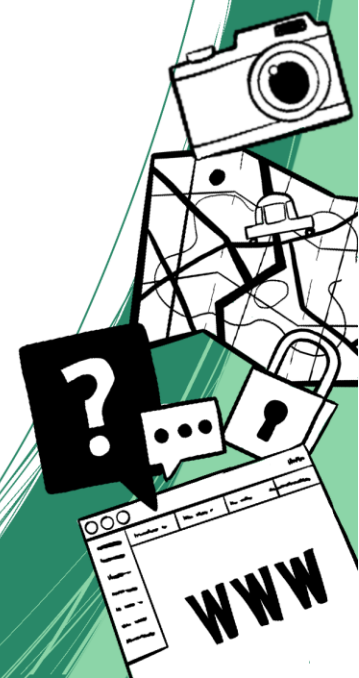
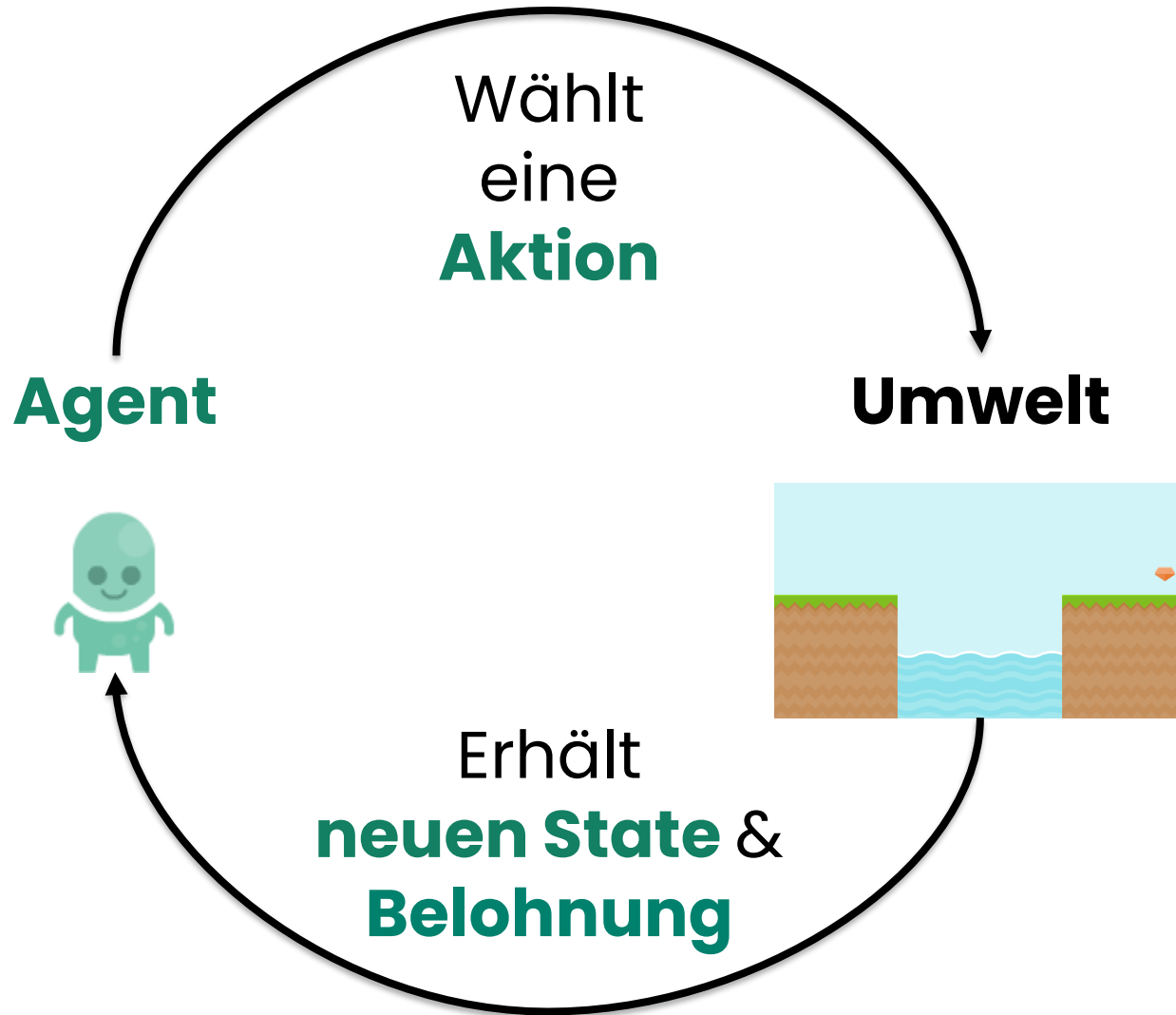
Umwelt



RL-Schleife

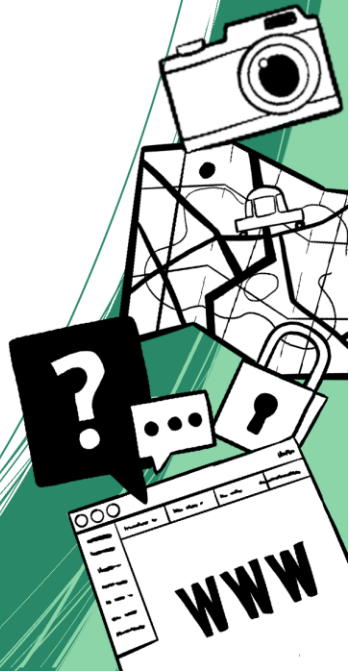


RL-Schleife



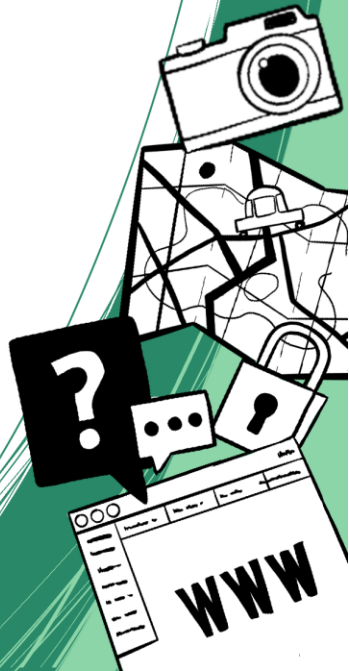
Q-Lernen

- Für jeden **Zustand** speichern wir einen **Qualitätswert (Q-Wert)** für **jede Aktion** (wie gut die Aktion dem Zustand gegeben wird)



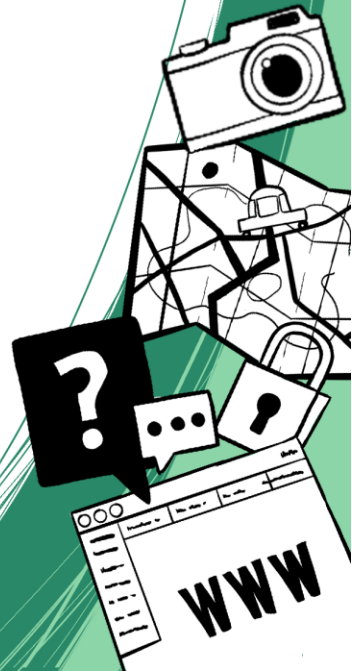
Q-Lernen

- Für jeden **Zustand** speichern wir einen **Qualitätswert (Q-Wert)** für **jede Aktion** (wie gut die Aktion dem Zustand gegeben wird)
- Wenn dann ein **Zustand** erreicht ist, wählen man die Aktion mit dem höchsten **Q-Wert**



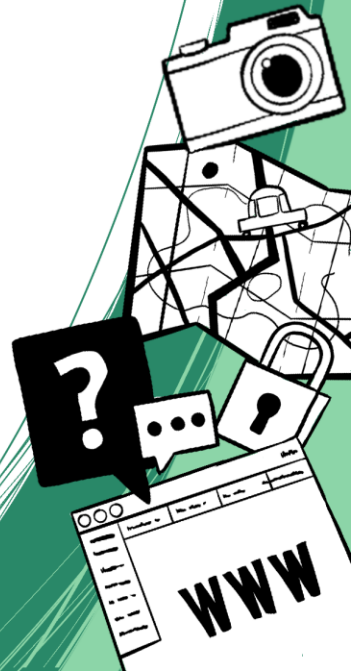
Q-Lernen

- Für jeden **Zustand** speichern wir einen **Qualitätswert (Q-Wert)** für **jede Aktion** (wie gut die Aktion dem Zustand gegeben wird)
- Wenn dann ein **Zustand** erreicht ist, wählen man die Aktion mit dem höchsten **Q-Wert**
- Schließlich erhöhen / verringern man den **Q-Wert** in Bezug auf die **Belohnung**, nachdem die **Aktion** ausgeführt wurde

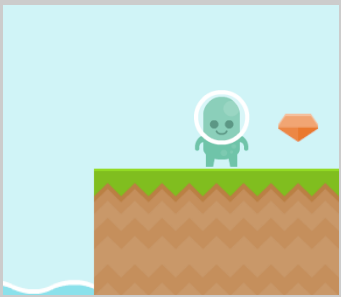
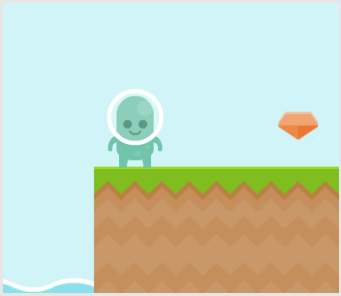


Q-Lernen

- Für jeden **Zustand** speichern wir einen **Qualitätswert (Q-Wert)** für **jede Aktion** (wie gut die Aktion dem Zustand gegeben wird)
- Wenn dann ein **Zustand** erreicht ist, wählen man die Aktion mit dem höchsten **Q-Wert**
- Schließlich erhöhen / verringern man den **Q-Wert** in Bezug auf die **Belohnung**, nachdem die **Aktion** ausgeführt wurde
- Für kleine Szenarien kann dies in einer Tabelle (**Q-Table**) gespeichert werden

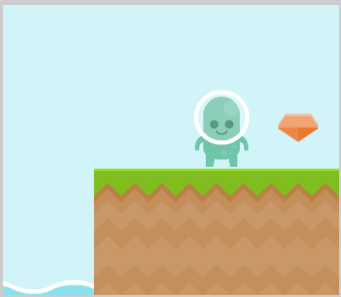
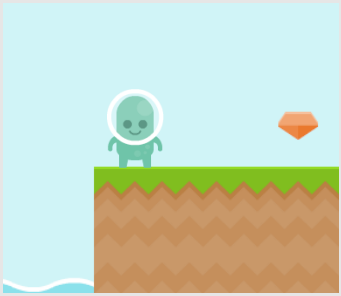


Q-Table Beispiel

Zustand	Schritt links	Schritt rechts	Sprung links	Sprung rechts
	0	1	-1	0
	-1	0	-1	1
...				



Q-Table Beispiel

Zustand	Schritt links	Schritt rechts	Sprung links	Sprung rechts
	-0.2	1	-1	0
	-1	0.5	-1	1
...				



Münzspiel

Probiere es an einem **echten**
Beispiel aus!

