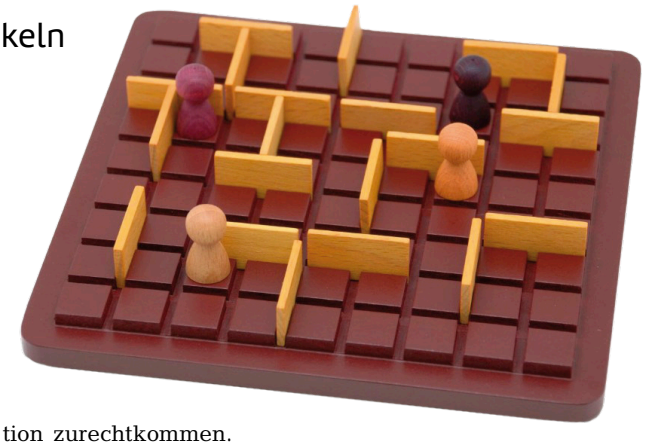
DOTNETPRO.CONTEST 1/2017 – AUSWERTUNG

Licht am Ende des Quoridors

Eine Bot-KI für das Brettspiel Quoridor entwickeln



Manchmal sieht man den Ausgang vor lauter Wänden nicht … oder etwa doch? Im letzten Programmierwettbewerb der dotnetpro wurde Taktikern und kühnen Planern die Aufgabe gestellt, eine KI zu entwickeln.

Diese KI sollte in der Lage sein, dass Brettspiel Quoridor gegen andere Computergegner zu spielen und natürlich im Idealfall auch gegen alle anderen Teilnehmer zu gewinnen. Insgesamt sieben kühne Strategen waren letztendlich angetreten, um sich den Sieg zu sichern.

Um zu ermitteln, wer sich denn nun den Titel des besten Quoridor Bots auf die Fahnen schreiben darf, traten alle Kontrahenten jeweils gegeneinander an. Dabei mussten sie je einmal mit der unteren und einmal mit der oberen Startposition zurechtkommen.

Bei sieben Teilnehmern ergaben sich also insgesamt 42 Spielpaarungen, die in Bild 1 visualisiert sind. In dieser Matrix kann man auch gleichzeitig sehen, wer das entsprechende Spiel gewonnen hat und warum. Die Spielbedingungen waren die gleichen wie in der Testumgebung PlayerVsBot: 60 Sekunden maximale Denkzeit und ein Spielabbruch nach 100 Zügen eines Bots.

Folgende Gründe für einen Sieg waren möglich:

- Regulärer Sieg: Der Bot hat seinen Kontrahenten regelkonform besiegt.
- Sieg durch invaliden Zug: Der Gegner hat einen nicht gültigen Zug als Antwort zurückgegeben.
- Sieg durch Timeout: Der Gegner brauchte zu lange, um seinen Zug zu übermitteln, sei es durch zu langes Rechnen oder durch eine Exception innerhalb des Bots.

Interessehalber wurden während des laufenden Turniers neben den Siegen und Niederlagen der Teilnehmer noch verschiedene Statistiken mitgeloggt. Diese sind in Tabelle 1 und Tabelle 2 zusammengefasst.

	CrazyBot	DonkeyBot	LukasBot	MMDuelBot	Qbot	QMinatorBot	StefanBoos
CrazyBot		CrazyBot Sieg regulär	LukasBot Sieg regulär	CrazyBot Sieg timeout	QBot Sieg regulär	QMinatorBot Sieg regulär	StefanBoos Sieg regulär
DonkeyBot	DonkeyBot Sieg invalider Zug		DonkeyBot Sieg regulär	DonkeyBot Sieg timeout	QBot Sieg regulär	QMinatorBot Sieg regulär	StefanBoos Sieg regulär
LukasBot	LukasBot Sieg regulär	LukasBot Sieg regulär		LukasBot Sieg timeout	LukasBot Sieg regulär	QMinatorBot Sieg regulär	LukasBot Sieg regulär
MMDuelBot	CrazyBot Sieg timeout	DonkeyBot Sieg timeout	LukasBot Sieg timeout		QBot Sieg timeout	QMinatorBot Sieg timeout	StefanBoos Sieg timeout
Qbot	CrazyBot Sieg regulär	DonkeyBot Sieg timeout	LukasBot Sieg regulär	QBot Sieg timeout		QMinatorBot Sieg regulär	QBot Sieg regulär
QMinatorBot	QMinatorBot Sieg regulär	QMinatorBot Sieg regulär	QMinatorBot Sieg regulär	QMinatorBot Sieg timeout	QMinatorBot Sieg regulär		StefanBoos Sieg regulär
StefanBoos	StefanBoos Sieg regulär	StefanBoos Sieg regulär	LukasBot Sieg regulär	StefanBoos Sieg timeout	StefanBoos Sieg regulär	QMinatorBot Sieg regulär	

Turnier: Matrix mit dem jeweiligen Spielergebnis (Bild 1)

Tabelle 1: Turnierstatistiken A

Bots	summed moving time	average moving time	sum placed walls	avg placed walls	avg placed walls (when winning)
QMinatorBotJochenBaier	16149 s	51,594 s	64	5,333	5,364
LukasBot	15,831 s	0,047 s	101	8,417	7,889
StefanBoos	2,902 s	0,008 s	100	8,333	7,500
DonkeyBot	0,524 s	0,002 s	86	7,167	3,800
Qbot	1068,433 s	3,370 s	76	6,333	5,800
CrazyBot	17792,909 s	59,508 s	60	5	3
MazeMakerDuelBot	109,626 s	9,135	0	0	–

Tabelle 2: Turnierstatistiken B

Bots	avg placed walls (when losing)	sum moves	avg moves per game	avg moves (when winning)	avg moves (when losing)
QMinatorBotJochenBaier	5	313	26,083	25,909	28
LukasBot	10	336	28	25,222	36,333
StefanBoos	10	370	30,833	26,500	39,500
DonkeyBot	9,571	318	26,500	15,400	34,429
Qbot	6,714	317	26,417	23,200	28,714
CrazyBot	6	297	24,750	18,750	27,750
MazeMakerDuelBot	0	12	1	–	1

Tabelle 3: Platzierungen der Teilnehmer

Platzierung	Bot	Siege
1	QMinatorBotJochenBaier	11
2	LukasBot	9
3	StefanBoos	8
4	DonkeyBot	5
5	Qbot	5
6	CrazyBot	4
7	MazeMakerDuelBot	0

Interessanterweise haben nicht alle Bots die maximal erlaubte Zugzeit ausgenutzt, um sich beispielsweise möglichst lange einen Spielbaum mit potenziell vorteilhaften Zügen zu errechnen. Die schnellste KI hatte hierbei eine mittlere Zugzeit von gerade einmal 0,002 Sekunden. Das andere Extrem dagegen war eine mittlere Zugzeit von 59,51 Sekunden.

Auch bei der Zuganzahl der verschiedenen Bots fanden sich große Unterschiede. So sind es knapp elf Züge Unterschied bei durchschnittlicher Zuganzahl bei siegreichen Spielen. Im ganzen Turnier wurden insgesamt 1963 Züge gespielt. Die letzten Kennzahlen, die noch mitprotokolliert wurden, verrieten etwas über den Einsatz der zu Verfügung stehenden Wände. Insgesamt wurden im Verlaufe des Wettkampfes 487 Wände platziert.

Doch genug der ganzen Statistiken und Theorie! Soweit man es nachvollziehen konnte, hielten sich alle Einsendungen an eine faire Spielweise. Kein Bot hat versucht, den gegnerischen Prozess zu beenden oder dem Gegner wertvolle Rechenzeit zu klauen.

Nachdem alle teilnehmenden Bots sich bis auf die letzte zu setzende Wall duelliert hatten, ergaben sich folgende Platzierungen (Tabelle 3): Den ersten Platz belegt der Bot *QMinatorBotJochenBaier* von Jochen Baier. Somit darf er sich über eine Freedev-Premium Lizenz für IncrediBuild freuen. IncrediBuild beschleunigt Builds, Testläufe, Paketierung und vieles mehr um das bis zu 30-Fache. An neuen Quoridor-Strategien darf fortan Lukas Slizik tüfteln, denn er sichert sich mit seinem virtuellen Strategen *LukasBot* den zweiten Platz und damit ein Quoridor Mini als Brettspiel. Den dritten Platz gewinnt Stefan Boos mit seinem nach ihm benannten Bot. Er erhält dafür ein Buch zum Thema Softwareentwicklung aus dem Fundus der Redaktion. Die Redaktion gratuliert den Gewinnern herzlich und sagt Danke für die tollen Einsendungen.

Im dotnetpro-Repository [1] des Contests findet sich jetzt eine *logHistory.txt*. In dieser Datei finden Sie alle Spiele des Turniers in Form von komprimierten Zeichenketten. Diese Strings können in den Replayviewer des OpenQuoridor Frameworks geladen werden. So können alle Interessierten noch einmal die Spiele nachverfolgen. Der Sourcecode des OpenQuoridor Frameworks lässt sich auch auf Github unter [2] finden. Die Autoren freuen sich, wenn sich der ein oder andere Teilnehmer dazu entschließen sollte, den entwickelten Bot per Pull Request in das Projekt mit einzubringen. ■

[1] OpenQuoridorFramework (OQF), www.dotnetpro.de/SL1704contestAuswertung1
[2] Hinweis für den dotnetpro-Contest 01/2017, www.dotnetpro.de/SL1704contestAuswertung2

Matthias Drescher
arbeitet in Nürnberg als Software Engineer bei Funkwerk Video Systeme GmbH. Dort ist er auf Sicherheitsanwendungen mit .NET und WPF spezialisiert. Zu erreichen unter **matthias.drescher@bytePassion.de**

Alexander Horn
arbeitet in Erlangen/Tennenlohe als Softwareentwickler bei der Astrum IT GmbH. Sein Schwerpunkt liegt auf der Entwicklung von Anwendungen mit .NET und WPF. Zu erreichen unter **alexander.horn@bytePassion.de**

dnpCode A1704contestAuswertung