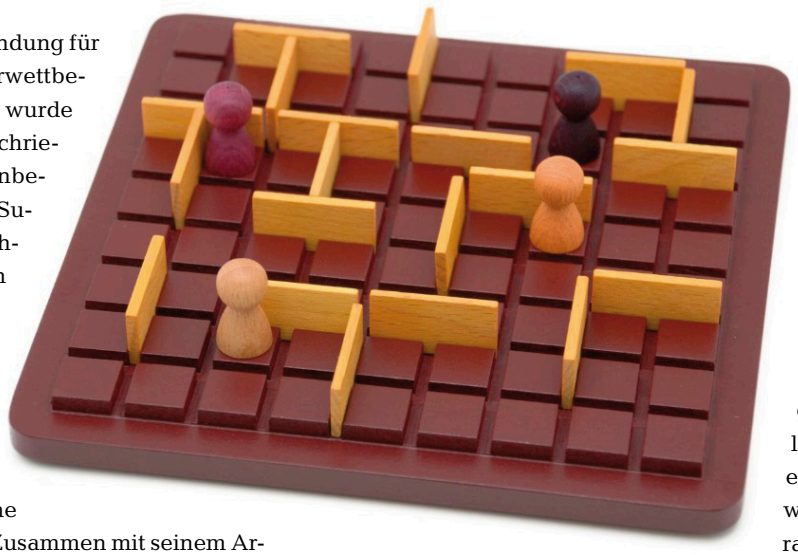
DOTNETPRO.CONTEST 01/2017 – AUFGABE

Der Weg durch den Quoridor

Eine Bot-KI für das Brettspiel Quoridor entwickeln.

Über die Aufgabenfindung für den Programmierwettbewerb in der dotnetpro wurde schon des Öfteren geschrieben. Die vielen Nebenbedingungen machen die Suche knifflig. Doch manchmal fügt sich das Leben in ganz wunderbarer Weise. So geschehen vor einigen Wochen, als die Nachricht von Matthias Drescher die Redaktion erreichte: Er habe eine Ideen für den Contest. Zusammen mit seinem Arbeitskollegen Alexander Horn entwickelte er daraufhin die Software und schrieb auch den Artikel zu diesem Contest. Die Redaktion bedankt sich sehr herzlich für die Initiative und freut sich auf besonders viele Teilnehmer. Schließlich kommt die Aufgabe von Mannen aus den eigenen Reihen. Jetzt zieht sich die Redaktion zurück und lässt Matthias Drescher sprechen – nein, schreiben.



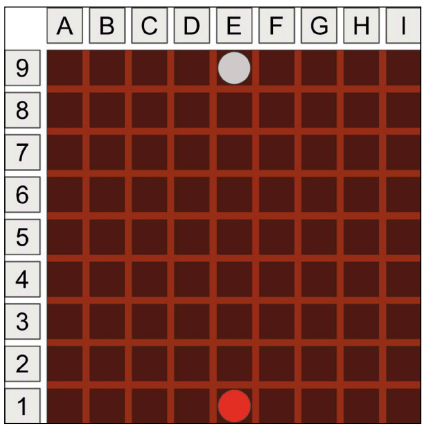
Quoridor Mini in Hardware, das dem zweiten Gewinner als Preis winkt (Bild 1)

In diesem Contest ist unsere Perspektive verdreht. Normalerweise lesen wir die Contests und versuchen uns an dem einen oder anderen. Doch dieses Mal stellen wir die Aufgabe für den Wettbewerb.

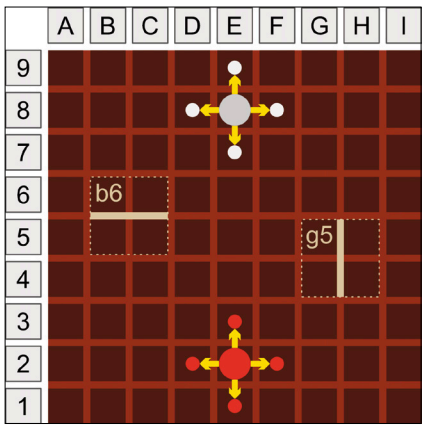
Vor einigen Monaten ist uns das Brettspiel Quoridor untergekommen, und wir waren auf Anhieb begeistert. So einfach, so genial und so herausfordernd! Nachdem wir die ersten Partien gegen erfahrenere Spieler allesamt verloren hatten, wurde unser Kampfgeist erst recht geweckt. Leider mussten wir feststellen, dass sich im Netz nur sehr wenig zum Thema „Strategie bei Quoridor" finden lässt. Und auch die vereinzelt anzutreffenden KIs waren keine wirkliche Herausforderung.

So war unsere Idee geboren, selbst ein kleines Test-Framework zu entwickeln, um damit Bots schreiben zu können. Nach einer Weile dachten wir uns: Warum nicht andere an diesem Spaß teilhaben lassen? Wenige E-Mails mit Tilman Börner später war der Quoridor-Contest aus der Taufe gehoben und wir Feuer und Flamme für die vor uns liegende Arbeit.

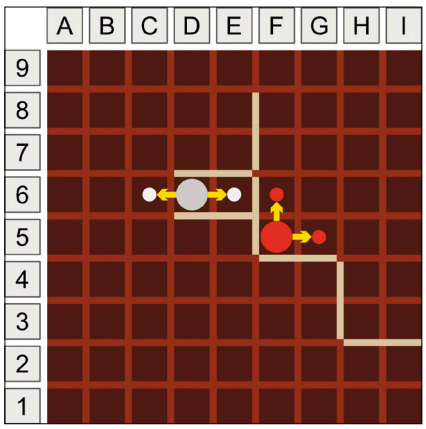
Aber was ist Quoridor überhaupt für ein Spiel? Quoridor ist ein Strategiespiel für zwei oder vier Spieler (Bild 1). Mirko Marchesi hat das Spiel erfunden. Aktuell wird Quoridor vom französischen Spieleverlag Gigami vertrieben.



Die Startaufstellung (Bild 2)



Erstes Beispielszenario mit allen Zugmöglichkeiten und Wand-Notations-Visualisierung (Bild 3)



Zweites Beispielszenario mit allen Zugmöglichkeiten (Bild 4)

Im Rahmen dieses Contests wird allerdings nur das Zwei-Spieler-Spiel betrachtet. Unser Test-Framework ist inzwischen zu einem Open-Source-Projekt mit dem Namen Open-QuoridorFramework (OQF) herangewachsen. Dieser Contest markiert gewissermaßen den Startschuss für das Projekt.

Die Spielregeln

Jeder Quoridor-Spieler bekommt eine Figur und zehn Wandelemente, die jeweils zwei Felder lang sind. Das Ziel des Spiels ist es, mit der eigenen Figur auf die andere Seite des Spielfelds zu kommen. Dabei ist es egal, welches der neun Felder der gegenüberliegenden letzten Reihe betreten wird.

Zu Beginn werden die Spielfiguren wie auf Bild 2 zu sehen aufgestellt. Der untere Spieler (in Rot auf Feld *e1*) beginnt mit dem ersten Zug. Immer wenn ein Spieler an der Reihe ist, muss er entweder die Figur um ein Feld bewegen oder ein Wandelement platzieren. Hat der Spieler keine Wandelemente mehr, muss er die Figur bewegen.

Gezogen werden kann nur horizontal oder vertikal (Bild 3), nicht über Wände hinweg (Bild 4) und nicht aus dem Spielfeld heraus. Die Sonderregeln zur Bewegung werden in Bild 5 veranschaulicht. Wie dort zu sehen ist, kann ein Spieler den anderen überspringen, wenn die beiden so in angrenzenden Feldern stehen, dass der Ziehende sich auf das Feld des Gegners stellen könnte. Ist ein Überspringen nicht möglich, weil eine Wand den Weg versperrt, kann nach links oder rechts ausgewichen werden, sofern man nicht auch dort von einer Wand behindert wird.

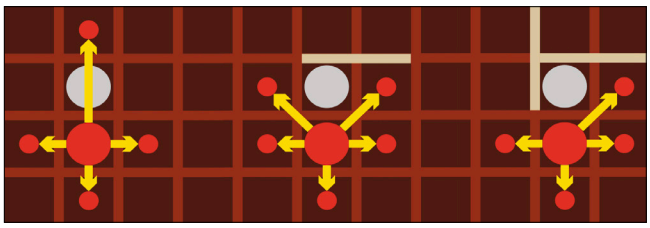
Beim Setzen eines Wandelements ist darauf zu achten, dass diese sich immer zwischen den Feldern befinden müssen, sich nicht überschneiden dürfen, nicht aus dem Brett herausragen dürfen und dem Gegner immer ein Weg zu einem der neun Ziel-Felder offengehalten werden muss.

Quoridor-Notation

Da es keine offizielle Quoridor-Notation gibt [1], wurde das Sinnvollste aus allen vorhandenen Notationen kombiniert. Die Reihen des Spielbretts werden danach durch Zahlen und die Spalten durch Buchstaben bezeichnet (siehe etwa Bild 2). Hieraus ergeben sich 81 Spielfelder von *a1* bis *i9*. Die Position einer Spielfigur kann damit eindeutig bezeichnet werden. Für ein Wandelement, das immer zwischen vier Feldern platziert werden muss, reicht es, die Koordinate eines dieser

Listing 1: Das IQuoridorBot-Interface

```
public interface IQuoridorBot
{
  event Action<Move> NextMoveAvailable;
  event Action<string> DebugMessageAvailable;
  void Init (PlayerType yourStartPosition,
    GameConstraints gameConstraints);
  void DoMove(BoardState currentState);
}
```



Bewegungs-Sonderregeln mit allen Zugmöglichkeiten (Bild 5)

Felder und die Orientierung (*h* für horizontal; *v* für vertikal) anzugeben. Hier wird dafür das Feld links oben herangezogen. Die zwei Wandelemente in Bild 3 befinden sich folglich an den Positionen *b6h* und *g5v*.

Die Aufgabe

Schreiben Sie einen Bot, der alle anderen Gegner im Contest schlagen kann! Dazu müssen Sie das Interface *IQuoridorBot* der *OQF.Bot.Contracts* implementieren (Listing 1). Eine Übersicht aller Elemente der *OQF.Bot.Contracts* sind in Tabelle 1 zu finden. Im Quellcode sind alle diese Klassen, Strukturen und Enumerationen mit XML-Doku-Kommentaren versehen. ▶

Tabelle 1: Elemente der OQF.Bot.Contracts

Name	Typ	Bedeutung
XField	Enum	Eindeutige Zuordnung der Spalten des Spielbretts (A – I)
YField	Enum	Eindeutige Zuordnung der Reihen des Spielbretts (Nine – One)
FieldCoordinate	Struct	Eindeutige Position auf dem Spielbrett
PlayerType	Enum	Die zwei möglichen Startpositionen der Spieler (TopPlayer / BottomPlayer)
WallOrientation	Enum	Die zwei möglichen Orientierungen der Wandelemente (Horizontal / Vertical)
Player	Klasse	Spielinvariante Spielerinformationen (Startposition und Name)
PlayerState	Klasse	Aktueller Zustand des Spielers
Wall	Klasse	Auf dem Spielfeld platziertes oder zu platzierendes Wandelement
Boardstate	Klasse	Alle spielrelevanten Informationen zu einem bestimmten Zeitpunkt (inkl. der Spielhistorie)
Move	Klasse	Abstrakte Oberklasse für die drei folgenden Moves
FigureMove	Klasse	Spielzug, bei dem die Spielfigur bewegt wird
WallMove	Klasse	Spielzug, bei dem ein Wandelement gesetzt wird
Capitulation	Klasse	Spielzug, bei dem der Spieler aufgibt
GameConstraints	Klasse	Spielinvariante Rahmenbedingungen
IQuoridorBot	Interface	Bot-Interface, das zu implementieren ist, um einen Bot in das OQF zu laden

Bot-Implementierung

Um nachvollziehen zu können, was zu tun ist, wird im Folgenden ein Spielablauf aus der Sicht eines Bots dargestellt:

Vor Spielbeginn wird die Methode *Init(PlayerType, GameConstraints)* aufgerufen. Hier wird dem Bot mitgeteilt, auf welcher Seite des Bretts er startet (*PlayerType*) und wie die Rahmenbedingungen (*GameConstraints*) gestaltet sind.

Es gibt zwei Rahmenbedingungen: die maximale Bedenkzeit, die dem Bot während eines Zuges gewährt wird, und die maximale Anzahl an Zügen, bevor das Spiel abgebrochen wird. In letzterem Fall verliert der Spieler mit dem ersten Zug.

Sobald ein Bot an der Reihe ist, wird die Methode *DoMove(BoardState)* aufgerufen. Ab diesem Zeitpunkt hat der Bot innerhalb der angegebenen Zeit die Möglichkeit, das Event *NextMoveAvailable* zu feuern. Im Parameter *BoardState* wird der aktuelle Zustand des Bretts übergeben. Der gesamte Spielverlauf ist darüber auch nachvollziehbar.

Um allzu aufwendige Berechnungen vor dem Spielstart zu unterbinden, muss auch die Abarbeitung der Init-Methode innerhalb der Zeit geschehen, die in den *GameConstraints* festgelegt wurde.

Zusätzlich kann der Bot jederzeit zu Debug-Zwecken das Event *DebugMessageAvailable* feuern, wodurch eine Nachricht abgeschickt werden kann, die daraufhin in der Oberfläche der Testanwendung (*PlayerVsBot*) dargestellt wird.

Weitere Tipps zur Bot-Implementierung sind in der Hilfe von PlayerVsBot zu finden. Außerdem findet sich in den Quelldateien des OpenQuoridorFrameworks eine Beispiel-Implementierung eines kompletten Bots: der SimpleWalkingBot. Wie der Name schon vermuten lässt, beschränkt er sich nur auf das Laufen. Er wird also niemals ein Wandelement setzen. Eine Dummy-Lösung, die als Ausgangspunkt für Ihre Lösung dienen kann, ist ebenfalls vorhanden.

Die Auswertung

Um einen Sieger in diesem Contest zu ermitteln, werden alle abgegebenen Bots in einem mehrstufigen Turnier gegeneinander antreten, bis die Plätze 1 bis 3 eindeutig ausgespielt sind. Vor jedem einzelnen Spiel in diesem Turnier werden die Startpositionen ausgelost.

Ein Bot verliert ein einzelnes Spiel, wenn er entweder die maximale Bedenkzeit oder die maximale Anzahl an Zügen überschreitet, eine Exception auftaucht, er einen ungültigen Zug macht oder nicht als Erster auf die gegenüberliegende Seite des Bretts kommt.

Ein ungültiger Zug wäre zum Beispiel das Setzen einer Wand, sodass dem Gegner kein Weg mehr zum Ziel bleibt.

Mitmachen, gewinnen

Mitmachen kann jeder. Voraussetzung ist .NET 4.5.2 und die *OQF.Bot.Contracts* des OpenQuoridorFrameworks. Holen Sie sich dazu das OpenQuoridorFramework über das Git-Repository unter *https://github.com/dotnetpro/contest012017* oder über das ZIP auf der Heft-CD.

Schicken Sie uns ausschließlich die DLL *<botName>.dll* als ZIP gepackt mit dem Betreff „dotnetpro contest 01/2017“ an die E-Mail-Adresse *contest@dotnetpro.de*. Einsendeschluss ist der 16. Januar 2017. Wir bestätigen den Erhalt Ihrer Lösung per E-Mail – allerdings nicht tagesaktuell. Die Ergebnisse werden in der dotnetpro 4/2017 bekannt gegeben. Der Rechtsweg ist ausgeschlossen. Die Gewinne können nicht ausbezahlt werden.

Wer sich für die aktuelle Entwicklung des OpenQuoridorFrameworks interessiert, findet dieses unter *https://github.com/bytePassion/OpenQuoridorFramework*. Es sei aber noch einmal darauf hingewiesen, dass Letzteres für den Contest keine Relevanz hat. ■

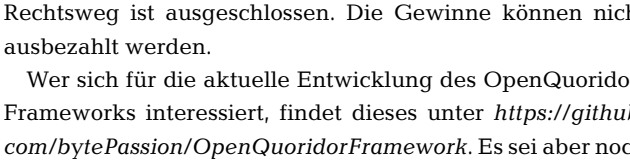
[1] www.dotnetpro.de/SL1701contestAufgabe1
[2] www.dotnetpro.de/SL1701contestAufgabe2

Das können Sie gewinnen

1. Preis: Eine Freedev-Premium-Lizenz für IncrediBuild im Wert von rund 1500 Dollar. IncrediBuild beschleunigt Builds, Testläufe, Paketierung und vieles mehr um das bis zu 30-Fache. Das schafft es, indem es die Aufgaben auf die Rechner im lokalen Netzwerk verteilt. IncrediBuild ist eine Plug-and-Play-Lösung, die out-of-the-box auf Ihrer vorhandenen Hardware läuft. Sie befähigt über 100 000 Anwender, die Zeit bis zum Verkauf zu reduzieren. So spart die Software Entwicklern Stunden an Arbeitszeit und reduziert damit die HPC-Kosten. Freie Version unter [2].

2. Preis: Ein Quoridor Mini als Brettspiel (Bild 1).

3. Preis: Ein Buch zur Softwareentwicklung aus dem Fundus der Redaktion.



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

dnpCode A1701contestAufgabe