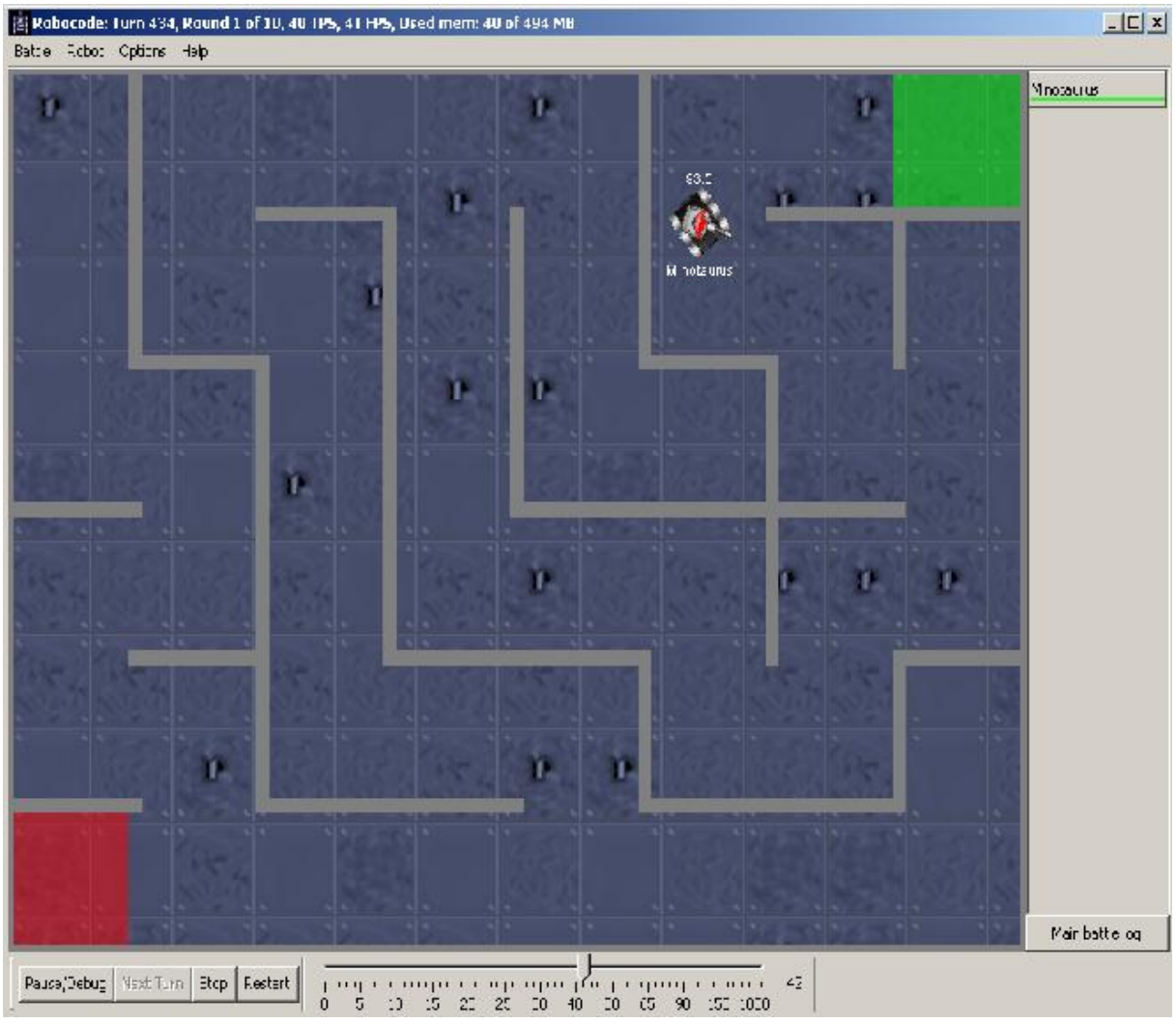
Problemlösen durch Suchen: A*-Suche

Ausgangssituation

Der Agent befindet sich in einem Labyrinth und soll vom Start- zum Zielfeld gelangen.

Es gelten die folgenden Bedingungen:

- Das Startfeld ist immer in der rechten oberen Ecke
- Das Zielfeld ist immer in der linken unteren Ecke
- Informationen über das Labyrinth sind abrufbar
- Einsatz von Radar und Geschütz sind nicht notwendig
- Die Wände können nicht mit dem Geschütz zerstört werden
- Das Rammen von Wänden kostet Energie
- Das Wände im Labyrinth sind immer rechtwinklig angeordnet
- Lücken im Labyrinth sind immer ausreichend groß, damit der Agent diese passieren kann.



Um die Aufgabe bearbeiten zu können, erstellen Sie ein neues Spiel mit dem Spieltyp *Maze* und fügen Sie Ihren Agenten hinzu. Zum Testen können Sie auch erst einmal einen Agenten vom Typ *sample.Minotaurus* hinzufügen und selber einmal ausprobieren, aus dem Labyrinth heraus zu finden.

Aufgabe

Implementieren Sie einen A*-Algorithmus, um auf dem kürzesten Weg vom Start- zum Zielfeld zu gelangen. Versuchen Sie die Arbeitsweise des Algorithmus grafisch auf dem Spielfeld darzustellen. Überlegen Sie sich anschließend:

- Wie groß ist die Komplexität der Suche?
- Wie effizient wäre ein uninformierter Suchalgorithmus?
- Was muss noch getan werden, damit der Agent das Ergebnis der Suche umsetzen kann?

Die Bearbeitung der Aufgabe soll in Einzelarbeit erfolgen. Sie können sich natürlich zusammensetzen und beraten, aber die Implementierung soll jeder selber für sich durchführen.

Bei der Abnahme wird Ihre Implementierung an einer Kartengröße von 1200x1000 Pixeln getestet.

Denken Sie an eine software- und programmiertechnisch korrekte Form der Implementierung. Denken Sie auch den angemessenen Einsatz von Datenstrukturen und Algorithmen. Zu einer tadelosen Lösung gehört auch eine aussagekräftige und vollständige Kommentierung des Quelltextes. Die Kommentierung sollte im Idealfall *javadoc*- oder *doxygen*-fähig sein.

Packen Sie Ihre Lösung, also alle zu Ihrem Agenten gehörenden Klassen in eine Zip-Archiv und benennen Sie es nach folgendem Schema `vorname.nachname-ISys1.zip` und laden Sie das Archiv rechtzeitig im Moodle-Kurs hoch.

Beispielsweise würde das Archiv mit der Lösung von Michael Breuker `michael.breuker-ISys1.zip` heißen.

Hinweise

Auch wenn das Spielfeld kontinuierlich erscheint, ist es diskretisiert und besteht aus Pixeln bzw. Kacheln. Jeden einzelnen Pixel zu betrachten, wird aber nicht besonders hilfreich sein, da der Suchraum dadurch zu groß wird. Eine Reduktion des Suchraums kann durch eine gröbere Kachelung des Spielfelds erzielt werden. Desweiteren könnten auch nur befahrbare Bereiche in die Suche mit einbezogen werden.

Schauen Sie sich die Packages `robocode.Rules` und `robocode.BattlefieldMap` genauer an. Dort finden Sie Informationen zur Kachelgröße und zur Navigation im Labyrinth. Die beiden Packages finden Sie nur in der *javadoc*-Dokumentation Ihrer Robocode-Installation.