APP-Projekt 2017

Gruppenmitglieder

- Alexander Wähling
- Anastasiia Kysliak
- Kai Kulhmann
- Niklas Müller
- Robin Hundt (Gruppenleiter)

Tutor

Dominick Leppich

Anleitung

Das Spiel kann folgendermaßen kompiliert und gestartet werden:

1. Starten eines Terminals **`strg + alt + t`** (standarmäßig unter Unity)
2. Wechseln in das Verzeichnis, welches das tar Archiv enthält. Nach dem Download gewöhnlich unter `~/Downloads` zu finden.
3. Entpacken des tar Archivs:
   `tar xvf AgeOfTowers.tar`
4. Wechseln in das neue `AgeOfTowers` Verzeichnis
5. Automatisches Übersetzen des Quelltextes, erzeugen der JavaDoc Dokumentation und Erzeugung der `.jar` Datei durch Aufruf des `ant` Programms
6. Starten des Programms ist in zwei Modi möglich, über Kommandozeilenparamter oder einen interaktiven Modus:

Kommandozeilenparameter

- Einstellungen beginnen mit einem `-` und Schalter mit `--`

- `--help` zeigt die Hilfe des Spiels an mit kurzen Erklärungen zu allen möglichen Parametern

Nichtoptionale Einstellungen für lokale Spiele:

- Bei einem lokalen Spiel müssen die folgenden Einstellungen gesetzt werden:
- Der Spielertyp des roten und blauen Spielers (zu Spielertypen mehr unter **Spielertypen**)
  - **`-red`** {human, random, simple, adv1, adv2, remote}
  - **`-blue`** {human, random, simple, adv1, adv2, remote}
- `-size {4,..,26}` die Größe des Spielfeldes (mögliche Größe zwischen 4 und 26)

Ein Beispielaufruf des Spieles mit einem menschlichen Spieler als **RED** und einer zufälligen KI als **BLUE** auf einem Feld der Größe 8 könnte so aussehen:

```
java -jar AgeOfTowers.jar -red human -blue random -size 8
```

Anmerkung: Das Spiel startet Standardmäßig mit grafischer Ausgabe. Wenn Textausgabe gewünscht ist, kann die `-output` Einstellung zu `text` gesetzt werden. Möchte man das Spiel also mit textueller Ausgabe starten, würde der Aufruf wie folgt aussehen:

```
java -jar AgeOfTowers.jar -red human -blue random -size 8 -output text
```

Optionale Parameter (lokales Spiel):

- **`-delay <Zeit in ms>`** kann gesetzt werden um eine Verzögerung zwischen Zügen zu erzwingen. Besonders sinnvoll wenn zwei schnelle Computergegner miteinander spielen und man ihr Verhalten nachvollziehen möchte.
- **`-timeout <Anzahl an Zügen>`** kann gesetzt werden um Spiele nach einer gewissen Anzahl an Zügen abzubrechen
  - `-timeout 200` würde dafür sorgen, dass das Spiel nach 200 **gesamten** Zügen abgebrochen wird
- **`-games <Zahl größer 1>`** startet den Turniermodus mit der angegebenen Anzahl an Spielen
  - Während dem Turniermodus tauscht die Rolle der Spieler
  - Am Ende des Turniermodus werden die Ergebnisse der einzelnen Spiele den Spielern wieder in ihrer Startposition zugeordnet
  - Ist die Einstellung `-timeout` gesetzt, gilt diese pro im Turnier gespielten Spiel
  - Am Ende des Turniers erhält man eine Statistik über die Anzahl der gespielten Spiele, die Anzahl der gewonnenen Spiele pro Spieler, wie oft sie auf welche Art gewonnen haben und wie viele Züge sie im Schnitt benötigt haben
- **`--statistic`** bewirkt die kontinuierliche Ausgabe der Turnierergebenisse

Spielertypen

Die folgenden Spielertypen können entweder über `-offer <Spielertyp>` im Netzwerk angeboten werden (bis auf remote, siehe Abschnitt Netzwerkspiel) oder als lokaler Spieler durch bspw. `-red human` erstellt werden.

- **human**
  - menschlicher Spieler Typ der es dem User ermöglicht das Spiel selbst über die textuelle oder graphische Ein-/Ausgabe zu spielen
- **random**
  - Computer Spieler der jeden seiner Züge zufällig aus allen seiner möglichen Züge zu diesem Zeitpunkt auswählt
- **simple**
  - Computer Spieler der seine Züge nach einer einfachen Strategie (siehe Seite 17 der Spielbeschreibung) bewertet und einen zufälligen Zug mit maximaler Bewertung auswählt
- **adv1**
  - Computer Spieler welcher seine Züge nach einer tendenziell einfacheren Bewertungsstrategie auswählt als der simple Player, durch eine bessere Gewichtung der verschiedenen Aktionen im Schnitt jedoch bessere Ergebnisse erziehlt
- **adv2**
  - Erweiterter Computer Spieler der seine Züge mit Hilfe des Monte Carlo Tree Search Algorithmus auswählt
  - Für diesen Spieler existieren eine Vielzahl an veränderbaren Parametern die genutzt werden können um seine Spielweise anzupassen (siehe Abschnitt **Monty Carlo Player**)
- **remote**
  - Ist einer der angegebenen Spielertypen beim Programmstart vom Typ `remote` so wird im Netzwerk nach einem Spieler gesucht mit dem das Spiel gespielt werden kann (siehe Abschnitt Netzwerkspiel->Spieler finden)

Netzwerkspiel

Es gibt zwei Arten von Netzwerkspiel:

1. Spieler anbieten:

- Möchte man einen Spieler im Netzwerk anbieten muss die Einstellung `-offer <Player type>` und `-name <Name des Netzwerkspielers>` gesetzt sein
- Möchte man einen Netzwerkspieler auf einem bestimmten Port anbieten, so kann dies über die Einstellung `-port <Port>` erreicht werden (Standardmäßig werden Spieler auf dem default Java RMI Port 1099 angeboten)
- Beim Anbieten eines Spielers im Netzwerk ist Standardmäßig die graphische Ausgabe aktiviert. Dies kann ebenfalls über die `-output {graphic, text, none}` Einstellung geändert werden

2. Spieler finden

Möchte man einen im Netzwerk angebotenen Spieler finden, so muss mindestens für

einen der beiden Spieler Einstellungen der Spielertyp `remote` sein (Er kann auch für beide `remote` sein). Über die Einstellungen
`-host <hostname oder ip>`
`-port <port>` und
`-name <Name des Spielers>` kann ein im Netzwerk angebotener Spieler gefunden werden. Diese drei Parameter sind jedoch optional.
Wird kein Host Name angegeben wird automatisch auf dem Lokalhost nach einem Spieler gesucht. Beim Weglassen des Port Parameters wird der standard Java RMI Port 1099 genutzt. Lässt der Benutzer ebenfalls den Namens Parameter weg, so wird auf dem spezifizierten Host (bzw. im default Fall dem localhost) nach allen möglichen Spielern gesucht, deren Namen dem Spieler ausgegeben und er wird interaktiv nach dem Spieler gefragt gegen den er sich messen möchte.
Sucht man nach Spielern, so läuft im nachfolgenden Spiel die gesamte Spiellogik auf dem eigenen Rechner ab. Dies bedeutet auch, dass derjenige der einen oder mehrere Remote Spieler sucht, dafür verantwortlich ist die Spielfeldgröße, den Delay, oder die Turnierparameter festzulegen.

Aufgeben eines Spiels

Einem menschlichen Spieler ist es möglich das Spiel aufzugeben. In der graphischen Oberfläche geschieht dies über den Surrender Button in der oberen rechten Ecke des Fensters.
In der Text Ein-/Ausgabe wird das Aufgeben dadurch realisiert, dass entweder `surrender` oder ein leerer Zug eingeben wird. Im zweiten Fall wird der Benutzer aufgefordert das Aufgeben zu bestätigen um ungewünschtes Aufgeben zu vermeiden.

Speichern und Laden eines Spiels

Speichern

Das Spiel bietet eine Speicher- und Lademechanik, die es ermöglicht, ein Spiel in eine Textdatei zu speichern. Spielt man AgeOfTowers mit der Graphik-Ausgabe, so kann man einfach auf `Save and Exit` klicken. Dann öffnet sich ein Dialog, in dem der Textdatei noch ein Name gegeben werden muss. Ist dies erfolgt, so wird dies mit `Save` bestätigt. Nach einem abschließenden Zug wird das Spiel beendet.

In der Konsolenausgabe ist der Pfad zur Speicherdatei im `.aot`-Format zu lesen. Dieser führt in das Home-Verzeichnis des Users mit dem Unterordner `AOT_Saves`. Dies ist auch wärend eines Bot-Spiels möglich.

In der Text Ein-/Ausgabe kann nur gespeichert, wenn ein menschlicher Spieler beteiligt ist. Dieser muss, wenn er am Zug ist statt eines normalen Zuges `save` eingeben. Nach einer Rückfrage des Programms muss wieder der Name der zu erstellenden Speicherdatei eingegeben werden. Enter bestätigt den Speichervorgang, beendet das Spiel und gibt wieder den Pfad zu Datei aus.

Laden

Die Lademechanik wird beim aufrufen der `.jar` -Datei benutzt. Man übergibt den Namen einer `.aot` - Datei hinter der Setting `-load`. Dabei ist zu beachten, dass nun keine Setting `-size` mehr gemacht werden muss, da die Größe des gespeicherten Boards geladen wird.

Quick Reference

Parameter	Optionen	Beschreibung
`-size`	Zahl von 4 bis einschließlich 26	- Setzt die Feldgröße auf den spezifizierten Wert
`-red`	Einer der Spielertypen beschrieben unter Spielertypen	- Setzt den roten Spieler auf den spezifizierten Typ
`-blue`	SpielerTyp des blauen Spielers	- Setzt den blauen Spieler auf den spezifizierten Spieler Typ
Optional		
`-delay`	Verzögerung in Millisekunden	- Zeit die zwischen den Zügen der Spieler vergehen soll
`-output`	none, text, graphic	- Art der Ausgabe des Spiels
`-timeout`	Zuganzahl	- Anzahl der maximalen Züge pro Spiel bevor dieses abgebrochen wird - Kann für normale Spiele sowie für Turniere angegeben werden
`-games`	Anzahl (n > 1) an Spielen im Turnier	- Startet ein Turnier mit der angegebenen Anzahl an Spielen
`--statistic`	Flag	- Kontinuierliche Anzeige des Turnierstatus während eines Turniers
`-load`	Name der Datei, die geladen werden soll "Name.aot"	- Lädt die angegebene Save-File als Spielstand
Remote		
`-offer`	Angebotener Spieler Typ	- Bietet einen Spieler dieses Types im Netzwek an
`-name`	Name des Netzwerkspielers	- Spezifiziert den Namen unter dem ein Netzwerkspieler angeboten werden soll
`-port`	Gültiger Port	- Port auf dem entweder ein Netzwerkspieler angeboten oder gesucht werden soll

Parameter	Optionen	Beschreibung
`-host`	Host (Bsp. CIP-Rechner Adresse oder IP)	- Host auf dem nach einem Netzwerkspieler gesucht werden soll
Debug		
`--debug`	Flag	- Wenn gesetzt werden alle LEVEL1 Debug Ausgaben ausgegeben
`-dlevel`	Debug Level von 1 bis 7 (Zahl)	- Wenn gesetzt werden alle Debug Nachrichten die dem gesetzten Level oder niedriger ausgegeben
`-dsource`	Ort der Debug Nachrichten: board, io, main, network, player	- Wenn gesetzt werden alle Debug Nachrichten aus dem spezifizierten Bereich ausgegeben - Kann in Verbindung mit `-dlevel` genutzt werden um Debug aus einem Bereich mit einem spezifizierten Level oder niedriger zu erhalten
Monty Carlo Player		
`-thinktime`	Zeit pro Zug in ms	- Setzt die Zeit die die erweiterte KI pro Runde verwenden wird um den bis dahin besten Zug zu berechnen
`--fair`	Flag	- Wenn gesetzt wird die KI immer so viel Zeit für die Berechnung ihres Zuges verwenden wie der Gegner zuvor gebraucht hat
Advanced AI Settings		
`-pstrategy`	PlayOut Strategie: light (l), heavy (h), dynamic (d)	- PlayOut Strategie die der MCTS Algorithmus in der Simulationsphase verwendet
`-tstrategy`	TreeSelection Strategie: max (m), robus (r)	- TreeSelection Strategie welche der MCTS Algorithmus verwendet um den

Parameter	Optionen	Beschreibung
		bestmöglichen Zug auszuwählen - Max ist der Zug mit der höchsten win / game ratio und robust der Zug mit den meisten simulierten Spielen
`-parallel`	Ungefähre maximale Anzahl an nebenläufigen Threads der KI	- Kann gesetzt werden, um der KI es zu ermöglichen eine höhere Anzahl an Threads zur Simulierung von Spielen zu verwenden
`-bias`	Bias Faktor in der UCB1 Formel	- Kann gesetzt werden um den bias Faktor zur Berechnung der UCB1 Werte von Nodes im MCTS zu verändern (Erfahrungsgemäß sollten Werte zwischen 0,4 und 2,5 gewählt werden)