Visualisierungshilfe: Die GUI zum Programmierwettbewerb

von: Dominik Honnef/Dominik Wagenführ

Da beim letzten Programmierwettbewerb einige Nutzer nach einer grafischen Oberfläche (GUI) fragten, können wir hier stolz eine präsentieren.

Beim letzten Programmierwettbewerb von **freies**Magazin im April dieses Jahres [1] fragten einige Nutzer nach einer grafischen Oberfläche (GUI), um das Geschehen besser verfolgen zu können. Aus diesem Grund hat Dominik Honnef diese für den anstehenden Wettbewerb umgesetzt. Die GUI ist dabei als Programmier- und Visualisierungshilfe gedacht und wird nicht zwingend für eine Teilnahme benötigt.

Anforderungen

Für die Umsetzung wird Ruby [2], Ruby-Gnome2 [3], Ruby-Gems [4] und die Gosu Game Development Library [5] benutzt. Daher muss man ein paar Ruby-Pakete (und deren Abhängigkeiten) installieren:

- **ruby** bzw. **ruby1.8**
- **rubygems**
- **libgtk2-ruby** bzw. **libgtk2-ruby1.8**

Daneben benötigt man noch einige Entwickler-Bibliotheken, um die Gosu GUI kompilieren zu können:

- **g++**
- **libgl1-mesa-dev**
- **libpango1.0-dev**
- **libboost-dev**
- **libsdl-mixer1.2-dev**
- **ruby1.8-dev**

Die Paketbezeichnung stammt von Debian/Ubuntu und kann/wird unter anderen Distributionen variieren. Unter Arch Linux gibt es beispielsweise keine extra dev-Pakete. Hier hilft ein Blick in die offizielle Installationsanleitung [6] weiter.

Nach der Installation der Pakete installiert man die Gosu GUI Library mit Root-Rechten über

```
# gem install gosu
```

Falls gefragt wird, welches gosu-Paket installiert werden soll, wählt man am besten **`gosu 0.7.14 (ruby)`** aus.

Falls eine Fehlermeldung der Art

```
ERROR:  While executing gem ... (Gem::GemNotFoundException)
    Could not find gosu (> 0) in any repository
```

kommt, sollte man die Installation einfach wiederholen, bis das Paket gosu gefunden wird.

Nach der Kompilierung von Gosu kann man die Entwicklerpakete wieder deinstallieren. Ruby & Co. müssen natürlich installiert bleiben.

GUI-Start

Die grafische Oberfläche besteht aus zwei Teilen. Zum einen aus der Oberfläche für alle Einstellungen des Spiels **`robots-gui-helper`** und zum anderen aus der Oberfläche für das Spielbrett **`robots-gui`**. Da die **`robots-gui`** nie direkt gestartet werden sollte, werden dessen Optionen hier nicht näher erläutert.

Den GUI-Starter ruft man per

```
$ ./robots-gui-helper
```

auf. Im Standardfall kann man nun bereits auf *„Spiel starten“* klicken und die Referenz-KI sollte den Roboter bewegen.

Die Einstellungen im Start sind alle selbsterklärend. Es soll hier nur auf die Optionen eingegangen werden, die nicht offensichtlich sind.

Über *„Neues Spiel starten“* kann man einstellen, ob der Roboter von seiner letzten Position aus der Datei **`bot.txt`** oder von der Startposition auf dem Spielbrett starten soll. Zusätzlich wird das Spielfeld und der Kartenstapel nicht neu eingelesen, wenn der Haken nicht gesetzt ist. So kann man ein unterbrochenes Spiel fortsetzen.

Entfernt man den Haken bei *„Spiel animieren“*, läuft der Roboter nicht über das Spielfeld, sondern das Spiel wird bis zum Ende berechnet, der Roboter also das Ziel gefunden hat oder zerstört wurde. Erst dann wird die Anzeige freigeschaltet und man sieht den Roboter an der Endposition.

Mit *„Spiel starten“* startet man das Spiel und kann dem Roboter bei der Bewegung zuschauen. Der Roboter hinterlässt ein transparente grüne Spur (Nein, er verliert kein Öl!), mit der man nachvollziehen kann, welche Felder er wie häufig betreten hat. Mit den Tasten [+] und [-] kann man die Bewegungsgeschwindigkeit des Roboters verändern.

Die „grüne Spur“, d. h. die Bewegungen des Roboters werden während des Ablaufes in der Sequenzdatei abgespeichert, die im zugehörigen Feld angegeben ist (als Standard ist die Datei **`globalseq.txt`** eingetragen). Diese Datei hat beispielsweise folgenden Aufbau, wobei jede Zeile einer Bewegung oder Drehung entspricht: