Fortgeschrittene Techniken der Funktionalen Programmierung WS 04/05

Christoph Lüth <cxl>

# Semesterübungsblatt

Ausgabe: 09.02.05

Abgabe: 01.04.05

## 4 *Haskell Racer 2000* *60 Punkte*

In der Semesteraufgabe soll es darum gehen, das einfache Fahrsimulationsprogram `Car.hs` aus der Veranstaltung Praktischen Informatik 3 zu einem vollwertigen Simulator und/oder Autorennspiel weiterzuentwickeln (die Grenzen sind sicherlich fließend).

Die Erweiterung geschieht in drei Stufen:

1. Da der Simulator in einer Vorlesung für Anfänger der funktionalen Programmierung vorgestellt wurde, wurde die Programmstruktur möglichst einfach gehalten. Inbesondere wurde der inhärente Zustandsübergang sehr direkt durch direkte Übergabe modelliert.

   In einem ersten Schritt sollten Sie ihre neugewonnen Kentnisse in der wunderbaren Welt der funktionalen Programmierung nutzen, um hier eine etwas anspruchsvollere Programmstruktur einzuführen. Insbesondere sollten Sie

   - den Zustandsübergang des Autos durch `Arrow` modellieren, und
   - das Programm über der Art des Autos parametrisieren — d.h. wir wollen sowohl Autos mit gelenktem Vorderradantrieb (wie gegeben) modeliieren, als auch Autos mit Differentialantrieb.

   Als Grafikbücherei können Sie entweder die Hugs Graphics Library `HGL` nehmen, aber zu bevorzugen wäre `wxHaskell` zusammen mit `hOpenGL`, soweit verfügbar.

2. In einem zweiten Schritt wird jetzt der Funktionsumfang entsprechend der 6. Aufgabe in der Praktischen Informatik 3 erweitert: wir fügen eine Umgegend hinzu, die Autos und andere Hindernisse enthält, sowie Tests, ob das Auto mit den Hindernissen kollidiert.

   Außerdem sollte die Bewegung des Autos (insbesondere Beschleunigung und Bremsverhalten) realitätsnäher gestaltet werden. Wer richtig gut ist, berücksichtigt auch die Bodenbeschaffenheit und Schlupf.

   Ferner wollen wir Sensoren berücksichtigen, zum Beispiel Ultra-Schall- oder Lasersensoren. Wie können wir diese modellieren, welche Werte geben sie zurück?

   Wenn die Modellierung aus der ersten Phase richtig war, können wir jetzt das Lenken des Autos durch eine Modellierung in Arrows darstellen. Das einfache, durch den Benutzer gelenkte Verhalten ist dann sozusagen der triviale Arrow (Eingabe = Ausgabe). Anspruchsvolleres Verhalten (Hindernissvermeidung etc.) sind dann nicht mehr trivialere Arrows.

3. In der letzten Phase wird das Spiel um Mehrbenutzerfähigkeiten und (noch wichtiger) Konfigurierbarkeit erweitert. Insbesondere sollen mehrere, auf verteilten Rechnern laufende Simulatoren gleichzeitig dieselbe Welt modellieren (und im Idealfall sich die Berechnung teilen). Außerdem sollte ein Simulator die Umgebung laden, und eine Fahrt in dieser Umgebung abspeichern und später wieder laden können. Datenaustauch mit anderen Programmen, insbesonder SimRobot, ist dabei erwünscht.

Das Spiel kann dann so funktionieren, dass der Spieler ein mit einem selbstgelenkten Auto gegen andere Autos eine Rennen fahren kann. Eigentliches Ziel ist aber die Implementation des Rahmenwerkes, in dem dieses Spiel dann stattfindet.

Dies ist Version 1.0 vom 2005/02/09 16:16:29.