1. Übungsblatt

Ausgabe: 29.10.09
Abgabe: 12.11.09

1 *Polymorpher Speicher* *5 Punkte*

Die in der Vorlesung gezeigte Implementation eines Speichers ist leider nur monomorph. Implementieren Sie einen Speicher mit polymorphen Referenzen, der folgende Signatur besitzt (vergleichbar mit `Data.IORef`):

```
type Ref a -- Referenzen
type State -- Zustand
type Mem   = ST State -- Zustand

newRef :: Typeable a⇒ a→ Mem (Ref a)
readRef ::  Typeable a⇒ Ref a→ Mem a
writeRef ::  Typeable a⇒ Ref a→ a→ Mem ()

run :: Mem a→ a
```

Hierbei ist ST der aus der Vorlesung bekannte Zustandsübergangsmonade.

Ist der so implementierte Speicher typsicher, oder läßt sich damit 'durch die Hintertür' eine Funktion `coerce :: a→ b` implementieren, die beliebige Konversion des Argumentes zuläßt?

Hinweis: Die Klasse Typeable stammt aus dem Modul `Data.Dynamic`, das Ihnen bei der Lösung der Aufgabe nützlich sein kann.

2 *Lost in Space* *15 Punkte*

Wir schreiben das Jahr 3039. Die Menschheit fliegt inzwischen zum Nachmittagskaffee nach Alpha Centauri, und man kann Wochenendaufenthalte im Orionnebel buchen, aber eines hat sich nicht gebessert: die Leute werfen immer noch leere Flaschen einfach aus dem Raumschiff — mit dem Pfand dafür könnte man sich eine goldenen Nase verdienen.

In dieser Aufgabe werden wir den beliebten PI3-Raumschiffsimulator um einige Zusatzfunktionalitäten anreichern. Es soll durch einen Weltraum fliegen, indem es Asteroiden, einzusammelndes Leergut und gefrorene Wasserstoffklumpen, nützlich als Treibstoff, gibt, beim Fliegen soll es Wasserstoff verbrauchen, und wenn es mit Hindernissen kollidiert—kabumm!

Im einzelnen sollen folgende Zusatzfunktionalitäten implementiert werden:

1. Die Welt soll prinzipiell beliebig groß (insbesondere größer als der Bildschirm) sein.
2. Die Visualisierung im Raumschiffsimulator soll geändert werden: das Raumschiff soll immer in der Mitte des Bildschirms sein, lediglich die Drehung wird visualisert, und die Umgebung bewegt sich.
3. Neben dem Raumschiff soll es auch Hindernisse geben:

   - Asteroiden (wesentlich größer als das Raumschiff);
   - Wasserstoffklumpen (kleiner als das Raumschiff);
   - Pfandleergut (etwa Größe der Wasserstoffklumpen).

   Asteroiden können sich langsam bewegen, die anderen Hindernisse sind stationär. Das Raumschiff interagiert mit den Hindernissen: an Asteroiden zerschellt es, Wasserstoffklumpen und Leergut werden eingesammelt. Wasserstoff wird zu Treibstoff, das Leergut wird gesammelt.
4. Bei der Bewegung des Raumschiffs wird der Treibstoffverbrauch berücksichtigt: Beschleunigung benötigt Treibstoff, Drehung auch (aber weniger). Ohne Treibstoff treibt das Raumschiff ziellos.

   Der Zustand des Raumschiffs soll neben Geschwindigkeit, Richtung und Schub auch noch Treibstoff und Punkte (für eingesammeltes Leergut) enthalten.
5. Eine weitere Zusatzfunktionalität für das Raumschiff ist Bremssschub, der einfach rückwärts wirkt (und natürlich auch Treibstoff kostet).
6. Am Anfang muss die Welt zufällig initialisiert werden; insbesondere die zufällige Berechnung von Asteroiden als hinreichen konvexe Asteroiden ist nicht ganz trivial.

   Wenn alles Leergut eingesammelt ist, ist das Spiel zu Ende — oder es wird eine neue Welt, mit mehr Asteroiden, ausgewürfelt ('next level').

Dies ist Revision 594 vom 2009-10-29.