Projekte der Veranstaltung "Fortgeschrittene funktionale Programmierung in Haskell"

Bestehenskriterien

Zum Bestehen der Veranstaltung müssen Sie insgesamt (Übung + Projekt) 220 Punkte erreichen.

Die Punkte sind unterteilt in Kategorien, aus denen nicht mehr als die angegebene Anzahl berücksichtigt wird:

- Parser (ohne binär): 75 Punkte
- div. Libraries: 90 Punkte
- "Core"-Haskell: 120 Punkte ("Core"-Haskell umfasst die Libraries `lens`,`mtl` bzw. `transformers`, `attoparsec` oder andere Parser,`gloss` oder andere GUI)
- Visualisierung: 100 Punkte
- Parallelism/Concurrency: 90 Punkte
- Binärparsing: 50 Punkte

Dies dient dazu, dass man sich nicht nur auf eine Sache konzentriert, sondern verschiedene Aspekte und deren Zusammenspiel verstanden haben muss.

Die im Folgenden beschriebenen Projekte sind alle offen gehalten. Wenn ihnen eine sinnvolle Erweiterung einfällt, sprechen sie die Tutoren an und diese Erweiterung wird in dieses Dokument aufgenommen und eine Bepunktung hierfür festgelegt.

Für alle Projekte gilt: **Bitte schreiben Sie ordentlichen Code. Organisieren Sie Ihren Code in sinnvollen Modulen. Geben Sie Typsignaturen mit an. Im Zweifelsfall dokumentieren Sie ihren Code. Liefern Sie außerdem eine Projektbeschreibung mit Bedienungsanleitung mit.**

Projekte

Von den Projekten "Sokoban" und "Snake" darf nur eines gewählt werden.

Sokoban (Punkte: 115 + 105 (optional))

Spielbeschreibung: Sokoban ist ein kachel- und rundenbasiertes Puzzlespiel (ähnlich wie `Game` aus den Übungen), in dem es darum geht, alle Kisten auf markierte Zielpunkte zu schieben (mit nur wenig Platz für Manöver). Hier finden Sie ein Beispiellevel, das alle Basisregeln deutlich macht.

Bepunktung

Das Spiel muss fehlerfrei spielbar sein (also: Siegbedingungen überprüfen, korrekte Kollisionsabfrage,..). Es sollen verschiedene spielbare Level vorliegen. Eine grafische Oberfläche ist Pflicht (Ausgabe auf der Konsole genügt nicht!), die Wahl der GUI-Bibliothek (gloss, not-gloss, sdl2, gtk2hs,...) steht Ihnen jedoch frei.

- Visualisierung: 30 Punkte (Visualisierung)
- Level-Parser: 25 Punkte (Parser)
- Benutzung von State oder RWST: 30 Punkte (Core)
- Implementation Spiellogik: 30 Punkte (Core)

Optional: Zusätzliche Gameplay-Features müssen in mitgelieferten Levels auftauchen.

- Verwendung von Lens: 20 Punkte (Core)
- Einlesen/Anzeigen von Bildern mittels Library: 10 Punkte (div. Libraries)
- Binärparser für Bitmaps und Anzeigen: 35 Punkte (Binärparsing)
- Gameplay-Features:
  - Türen (die man mit Schlüssel öffnen kann): 6 Punkte (Core) + 4 Punkte (Visual.)
  - Handschuhe (die eingesammelt Ziehen von Kisten ermöglichen): 6 Punkte (Core) + 4 Punkte (Visual.)
  - Undo/Redo: 10 Punkte (Core)
  - Speicherfunktion: 10 Punkte (Core)
  - Replayfunktion: 8 Punkte (Core) + 4 Punkte (Visual.)
  - ...

Snake (Punkte: 80 + 140 (optional))

Snake mit verschiedenen Level programmieren. Snake ist gut für eine Multiplayer-Implementation geeignet.

Spielbeschreibung: Wenn kein Essen auf dem Spielfeld zu sehen ist, dann erscheint auf einem freien Feld etwas zu Essen. Die Schlange kann gesteuert werden, läuft aber automatisch nach vorn, wenn keine Taste gedrückt wurde. Wenn der Kopf der Schlange Essen aufnimmt, dann verschwindet es und die Schlange beginnt zu wachsen. Ziel des Spiel ist es möglichst viele Punkte (Essen) zu sammeln, bevor die Schlange vor die Wand oder sich selbst läuft.

Level-Variante: Wenn ein Level an der Außenseite keine Wand hat, dann kommt die Schlange auf der anderen Seite des Levels wieder herein.

Bepunktung

Das Spiel ist gemäß der obigen Beschreibung fehlerfrei spielbar. Es gibt verschiedene Level (mit und ohne Wände), die beim Spielstart aus einer oder mehreren Textdateien geparst werden. Eine grafische Oberfläche ist Pflicht (Ausgabe auf der Konsole genügt nicht!), die Wahl der GUI-Bibliothek (gloss, not-gloss, sdl2, gtk2hs,...) steht Ihnen jedoch frei.

- Visualisierung: 25 Punkte (Visualisierung)
- Level-Parser: 25 Punkte (Parser)
- Implementation Spiellogik: 30 Punkte (Core)

Optional: Zusätzliche Gameplay-Features müssen in mitgelieferten Levels auftauchen.

- Benutzung von State oder RWST: 30 Punkte (Core)
- Verwendung von Lens: 20 Punkte (Core)
- Einlesen/Anzeigen von Bildern mittels Library: 10 Punkte (div. Libraries)
- Binärparser für Bitmaps und Anzeigen: 35 Punkte (Binärparsing)
- Highscore-Screen (angezeigt beim Pausieren/nach Spielende): 8 (Core) + 4 (Visualisierung)
- Items (beschleunigt, verbreitert o.ä. temporär...): 6 (Core) + 4 (Visualisierung)

Multiplayer: Jeder Spieler steuert eine Schlange. Tote Schlangen verschwinden vom Spielfeld. Wer zuletzt überlebt, hat gewonnen.

- Multiplayer an einem PC: 15 Punkte (Core)
- Multiplayer über Netzwerk: 30 Punkte (Concurrency) + 15 Punkte (Binärparsing) + 20 Punkte (Core)
- Dedizierter Multiplayer-Server: 60 Punkte (Concurrency)

Datamining (Punktesumme: 55 + 82 (optional))

Es wird ihnen eine CSV-Datei eines öffentlichen Haushaltsplanes gegeben. Lesen die Datei ein und visualisieren sie diese in einer geeigneten Weise. Für eine optionale, interagierbare Visualisierung gibt es die 45 optionalen Visualisierungspunkte.

Bepunktung

- Parsing von CSV-Daten mit `,` und `\n`: 15 Punkte (Parser)
- Verifikation der Daten: 5 Punkte (Parser)
- Erstellung der Visualisierung: 35 Punkte (Visualisierung)

Optional:

- Erkennen/Angeben von Feldtrennern und Zeilentrennern: 10 Punkte (Parser)
- Sinnvolle interaktive Visualisierung: 15 (Core) + 45 Punkte (Visualisierung) - z.b. wie Wuppertraler Haushaltsdaten
- Daten im JSON-Format annehmen: 12 Punkte (Parser) bei Selbst-Schreiben, 10 Punkte (div. Libraries) bei Nutzung von externen Libs.

Verteiltes Rechnen (Punktesumme: 45 + 132 (optional))

Dieses Projekt entspricht dem Betriebssysteme-Projekt des rechnenden Servers. Ihr Programm soll einen Port öffnen und dort Verbindungen akzeptieren. Diese schicken "Aufträge" (simple Arithmetik mit `+-*/`) an den Server, der einige Zeit brauchen soll, diese zu bearbeiten (2-30 Sekunden). Man kann mehrere "Jobs" starten, sich nach ihrem Zustand erkundigen, die Ergebnisse abholen oder auf Ergebnisse warten.

Bepunktung

- Minimal-Server: 45 Punkte (Parallelism/Concurrency)

Optional:

- Parsing der Eingaben mittels Parser: 12 Punkte (Parser)
- Verwendung von Lenses: 15 Punkte (Core)
- Verteiltes Rechnen: 35 Punkte (Parallelism/Concurrency) + 70 Punkte (div. Libraries) bei Verwendung von CloudHaskell - man rechnet nicht auf einem Server, sondern auf einer Cloud

Jabber-Bot (Punktesumme: 42 + 110 (optional))

Das XMPP-Protokoll (auch Jabber genannt) int eine offene Schnittstelle. Über die Techfak bekommen sie einen eigenen Jabber-Account (username@techfak.de). Schreiben sie einen Bot, der einfache Aufgaben in einem Jabber-Chat (oder Konferenz) übernimmt.

Aufgaben:

- Remember-me: Der Bot reagiert auf ein Kommando der Form "remind me at 13:45: go eat in Mensa" und um 13:45 schreibt der Bot den Absender mit der Nachricht "go eat in Mensa" an.
- Wikipedia-suche: Der Bot reagiert auf ein Kommando der Form "wiki universität bielefeld" und sucht auf Wikipedia nach "universität bielefeld", nimmt den ersten Treffer und antwortet mit dem ersten Absatz des Artikels
- Kalender-Plugin: Das ekVV stellt den persönlichen Kalender im caldav-Format zur Verfügung. Laden sie diesen zum Start/auf Wunsch ein und senden sie vor jedem Event eine Nachricht an den jeweiligen User.
- Verteiltes-Rechnen-Plugin: Wenn sie das Projekt "verteiltes Rechnen" gemacht haben, dann sollen sie auch in der Lage sein über ihren Bot mit dem jeweiligen Server zu reden (und so Aufträge absenden, abholen etc.).
- Webprojekt-Plugin: Wenn das Webprojekt ebenfalls gemacht wurde, dann soll dieser Bot in der Lage sein private Nachrichten an den jeweiligen User weiterzuleiten.

Bepunktung

- Aufsetzen des Bots: 40 Punkte (div. Libraries)
- Remember-Me-Aufgabe: 12 Punkte (Core)

Optional:

- Wikipedia-Suche-Aufgabe: 25 Punkte (Core) + 15 Punkte (div. Libraries) bei Verwendung von hexpat-lens zum Antwort-Parsen
- Kalender-Plugin:
  - Parser für ekvv-Kalender: 20 Punkte (Parser) oder 20 Punkte (div. Libraries)
  - Implementation Erinnerungsfunktion: 12 Punkte (Core)
- Verteiltes Rechnen-Plugin:
  - Bot baut eine Verbindung zum Projekt "Verteiltes Rechnen" auf und leitet Input/Output durch: 10 Punkte (Parser) + 12 Punkte (Core)
- "Webprojekt"-Plugin:
  - Notifications vom Forum per Jabber verstehen und an den angegeben User weiterleiten: 16 Punkte (Core)

Webprojekt (Punktesumme: 65 + 105 (optional))

Erstellen sie ein simples Forum. Die Registrierung kann entweder über Email/Passwort oder auch per Plugins, wie "OAuth" (ekvv, Google, Facebook, Github, Twitter, Twitch, ...) ermöglicht werden.

Das Admin-Panel soll umfassen: - Hinzufügen/Entfernen von Foren - Moderations-Fähigkeiten - Rechteverwaltung (Benennen von "Moderatoren", einschränken der Rechte auf Unterforen)

Bepunktung

- Minimal-Forum: 42 Punkte (div. Libraries) + 15 Punkte (Core)
- Registrierung: 8 Punkte (div. Libraries)

Optionale Zusätze:

- Gute, responsive Visualisierung mittels Bootstrap (o.ä.): 45 Punkte (Visualisierung)
- Upload von Avataren und deren Darstellung: 12 Punkte (div. Libraries)
- Admin-Panel: 24 Punkte (Core)
- Notifications per E-Mail: 12 Punkte (Core)
- Notifications per Jabber: 12 Punkte (div. Libraries)

Abgabedatum: 18.09.2017, 10 Uhr.

Projekte werden vorzugsweise als Projekt auf Github oder über euer **Remote-Folder** abgegeben. Jede Abgabe umfasst eine Projektbeschreibung mit Bedienungsanleitung. Die Projekte sind in sinnvoller Weise in Modulen organisiert. Typsignaturen sind mit angegeben und wichtige/komplizierte Stellen sind dokumentiert. Bitte vermeiden Sie es Ihrem git Repository temporäre Dateien (insb. das `.stack-work`-Verzeichnis) hinzuzufügen. Legen Sie eine `.gitignore` an, damit sowas nicht zufällig passiert.