Praktikum Programmiermethodik 2 (Technische Informatik)
WS 2016/2017, Hochschule für Angewandte Wissenschaften (HAW), Hamburg
Prof. Dr. Philipp Jenke, Kasperczyk-Borgmann

Für dieses Aufgabenblatt gelten die folgenden Regeln:

- Der Java Code Style (siehe EMIL) ist einzuhalten. Es werden keine Abgaben abgenommen, die diese Anforderungen nicht erfüllen.
- Pro Team muss ein gemeinsames Git-Repository für das Projekt vorhanden sein (z.B. auf GitHub, Bitbucket oder dem HAW-Informatik-Home-Verzeichnis).
- Der Code muss vollständig getestet sein.

Aufgabenblatt 4: Ein Simulator für Braitenberg-Vehikel

Aufgabe 4.0: Einarbeitung

Lernziele: Einarbeitung in bestehenden (Legacy)-Quellcode
Aufgabe:
Zusammen mit diesem Aufgabenblatt finden Sie eine Eclipse-Projekt. Dieses beinhaltet einen Simulator für Braitenberg-Vehikel und eine sehr einfache Darstellung. Machen Sie sich mit dem Konzept der Braitenberg-Vehikel vertraut (siehe z.B.: *https://de.wikipedia.org/wiki/Braitenberg-Vehikel*) und arbeiten Sie sich in die Code-Vorgaben ein. Sie müssen in der Lage sein, die grundlegenden Konzepte anhand des Quellcodes wiederzugeben (ausgenommen die mathematische Herleitung der Umrechnung der Signalstärke in die Vehikel-Bewegung). Machen Sie sich auch mit den verwendeten Koordinatensystemen (Weltkoordinatensystem der Vehikel und Bildkoordinatensystem im GUI) vertraut.

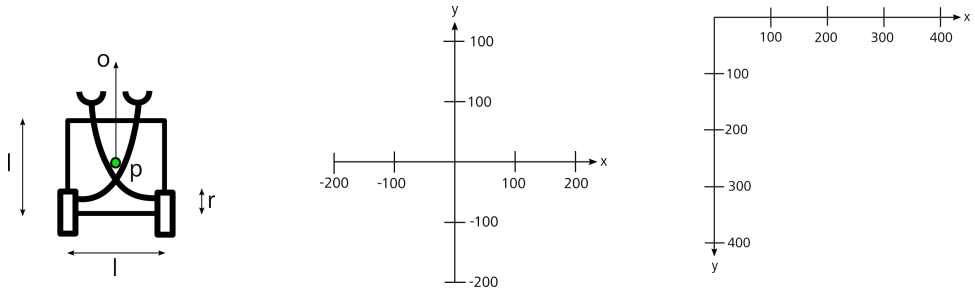

Abbildung 1: Braitenberg-Vehikel (links), Weltkoordinatensystem (Mitte), Bildkoordinatensystem (rechts).

Aufgabe 4.1: Steuerung

Lernziele: GUI-Elemente entwerfen, Ereignisverarbeitung
Aufgabe:

- Ergänzen Sie die Anwendung um eine grafische Benutzerschnittstelle mit zwei Komponenten: einem Knopf und einer Checkbox.
- Beim Druck auf den Knopf wird ein Simulationsschritt ausgeführt (`simulationsSchritt()`).
- Wird der Haken der Checkbox gesetzt, dann startet ein neuer Thread in dem wieder und wieder ein Zeitschritt der Simulation berechnet wird und 200ms gewartet wird. Wird der Haken wieder entfernt, dann wird der Thread beendet.
- Beide Komponenten sollen rechts neben der Zeichenfläche (`BVCanvas`) im Fenster erscheinen.

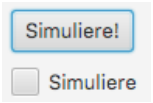

Abbildung 2: Knopf und Checkbox zur Kontrolle der Simulation.

Aufgabe 4.2: Observer

Lernziele: Observer-Muster anwenden
Aufgabe:

- Um den Inhalt der Simulation neu zu zeichnen, muss die Methode `zeichneSimulation()` von `BVCanvas` jedes Mal manuell aufgerufen werden. Durch das Observer-Muster soll dieser Vorgang automatisiert werden.

- Ergänzen Sie den Code um die notwendige Funktionalität, um die Vehikel und die Simulation zu Observables zu machen. Die Zeichenfläche wird als Beobachter umgesetzt und zeichnet die Szene neu, wenn sich bei den Observables etwas Relevantes geändert hat.

Aufgabe 4.3: Vehikel-Einstellungen
Lernziele: Dynamischer Aufbau von GUI-Komponenten
Aufgabe:

- Die Vehikel können unterschiedliche Bewegungsmuster haben. In der Vorgabe implementiert sind zwei Muster: Attraktion (hin zum Signal) und Abstoßung (weg vom Signal). Ergänzen Sie das GUI um je eine Schaltfläche für jedes Vehikel der Simulation.
- Auf jeder Schaltfläche soll es mindestens den Namen und das aktuelle Bewegungsmuster geben.
- Name: Anzeige des Namens des Vehikels.
- Bewegungsmuster: Auswahl (Combobox) des Bewegungsmusters. Wird ein neues Bewegungsmuster über das GUI ausgewählt, so muss das Verhalten selbstverständlich auch beim Vehikel gesetzt werden.

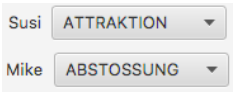

Abbildung 3: GUI-Elemente für Vehikel.

Aufgabe 4.4: Signal
Lernziele: Maus-Ereignisse
Aufgabe:

- Beim Klick auf die Zeichenfläche soll das Signal auf genau diese Stelle gesetzt werden.
- Dazu müssen Sie Maus-Ereignisse auf der Zeichenfläche abfangen und die Koordinaten eines Maus-Klicks auf die Zeichenfläche von Bildkoordinaten in Weltkoordinaten umrechnen.

Aufgabe 4.5: Darstellung Erweitern
Lernziele: Zeichenroutinen
Aufgabe:

- Ergänzen Sie die Darstellung der Vehikel auf der Zeichenfläche um die Angabe des Namens und um ein Icons für das aktuell gesetzte Bewegungsmuster.

Abbildung 4: Erweiterung der Darstellung der Vehikel durch Icons und Text.

Aufgabe X [freiwillig]: „Diese Aufgabe finde ich total toll und benötige Anregungen für weitere Teilaufgaben!"

- Visualisieren Sie die Funktionen, mit denen aus den Signalstärken die Motoransteuerungen berechnet werden.
- Implementieren Sie weitere Bewegungs-Verhalten (`BVBewegung`).
- Implementieren Sie eine 3D-Visualisierung.
- Bauen Sie ein reales Braitenberg-Vehikel, z.B. auf Basis eines Arduinos (bei Interesse gerne melden!)

Anlagen

- Quellcode-Vorgaben (Eclipse-Projekt)
- Beschreibung der Bewegungs-Berechnung (PDF)
- Video der Musterlösung unter *https://owncloud.informatik.haw-hamburg.de/index.php/s/H7VEaobQGRzCROx*