Technische Dokumentation zum Simulator

Verhalten kybernetischer Vehikel
Informatikprojekt I00-2

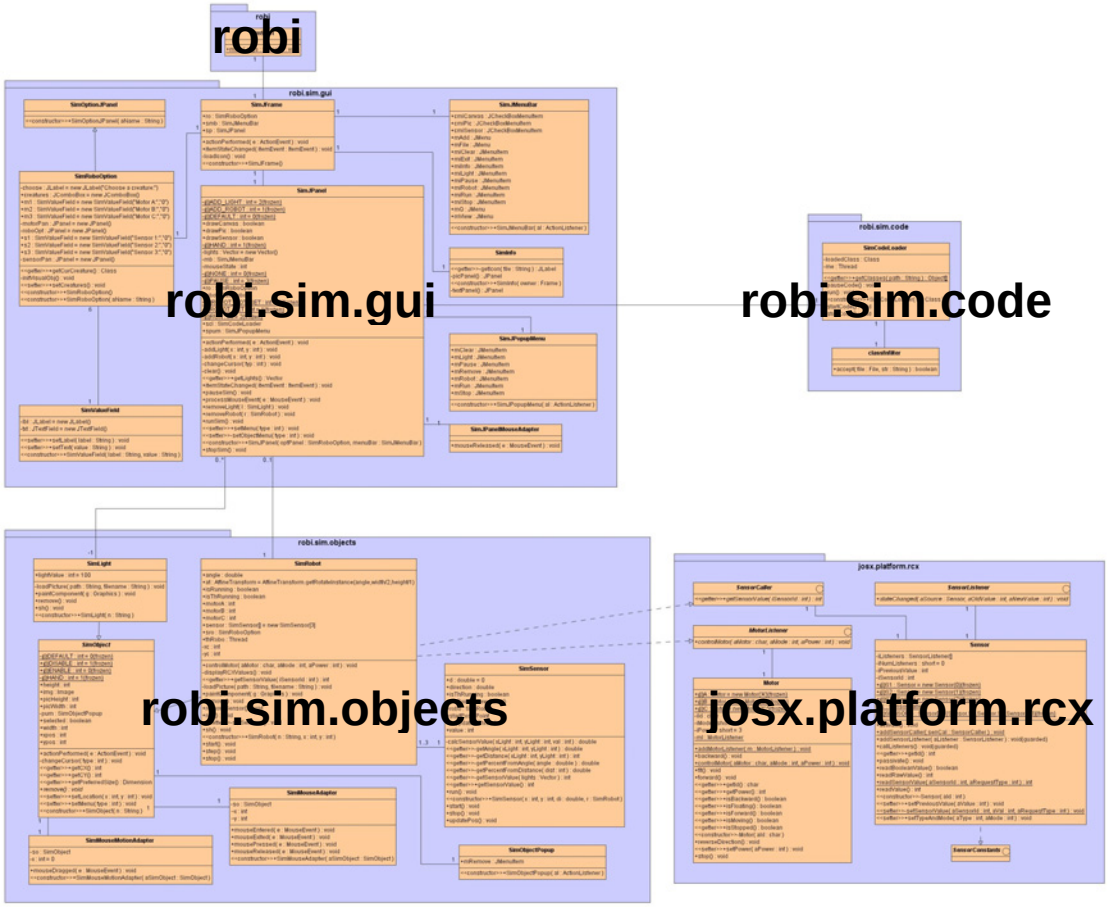
Reto Witschi
Andreas Gafner
Lukas Reusser
Pascal Fleury

HTA Bern I00-2

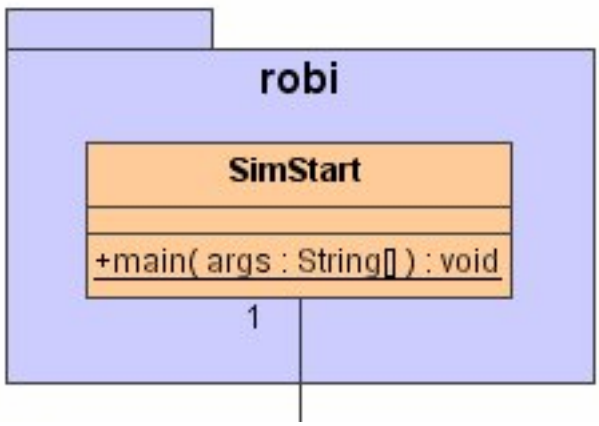
Inhaltsverzeichnis

1 Verfeinertes Klassendiagramm

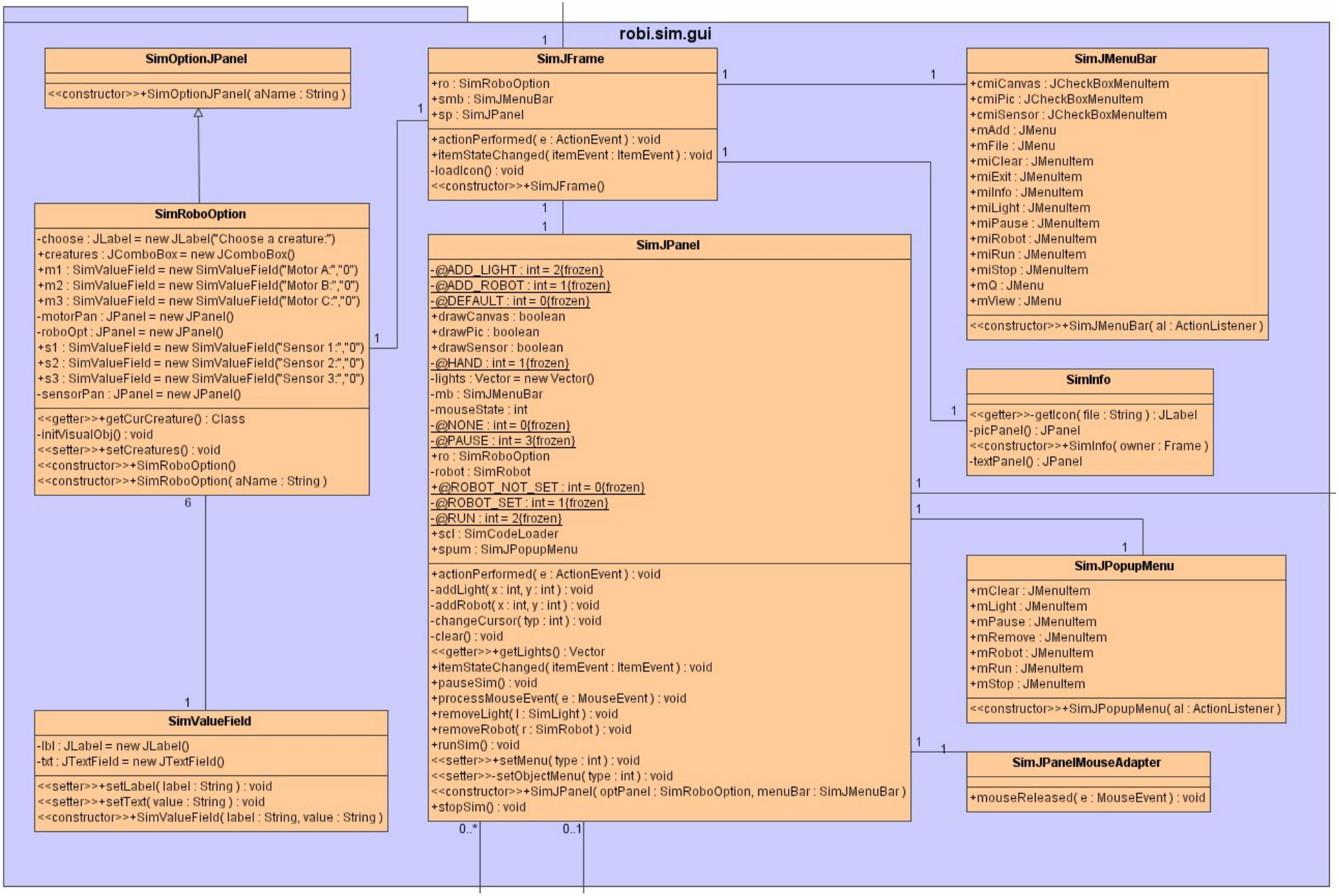
1.1 Übersicht



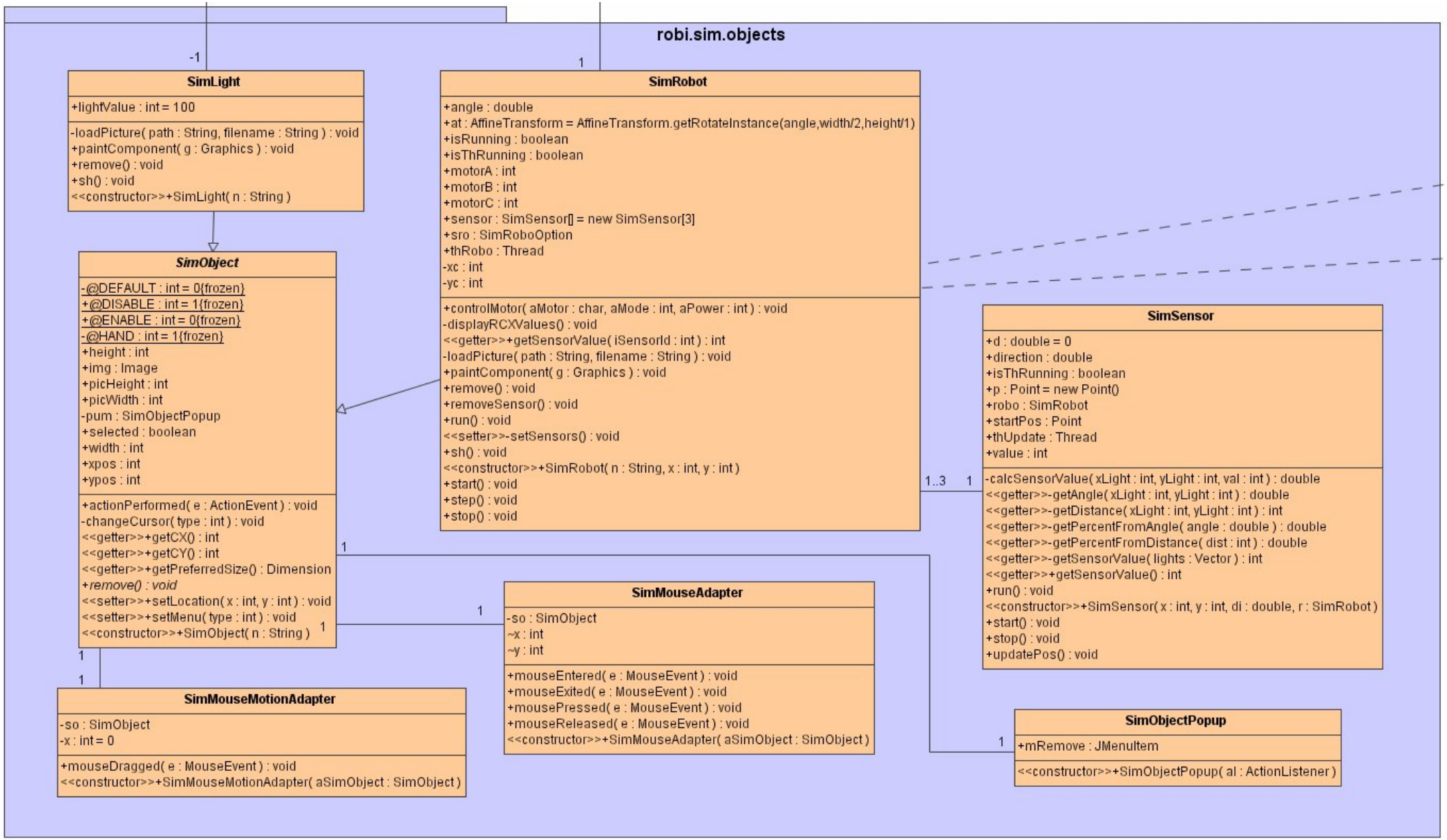
1.2 robi



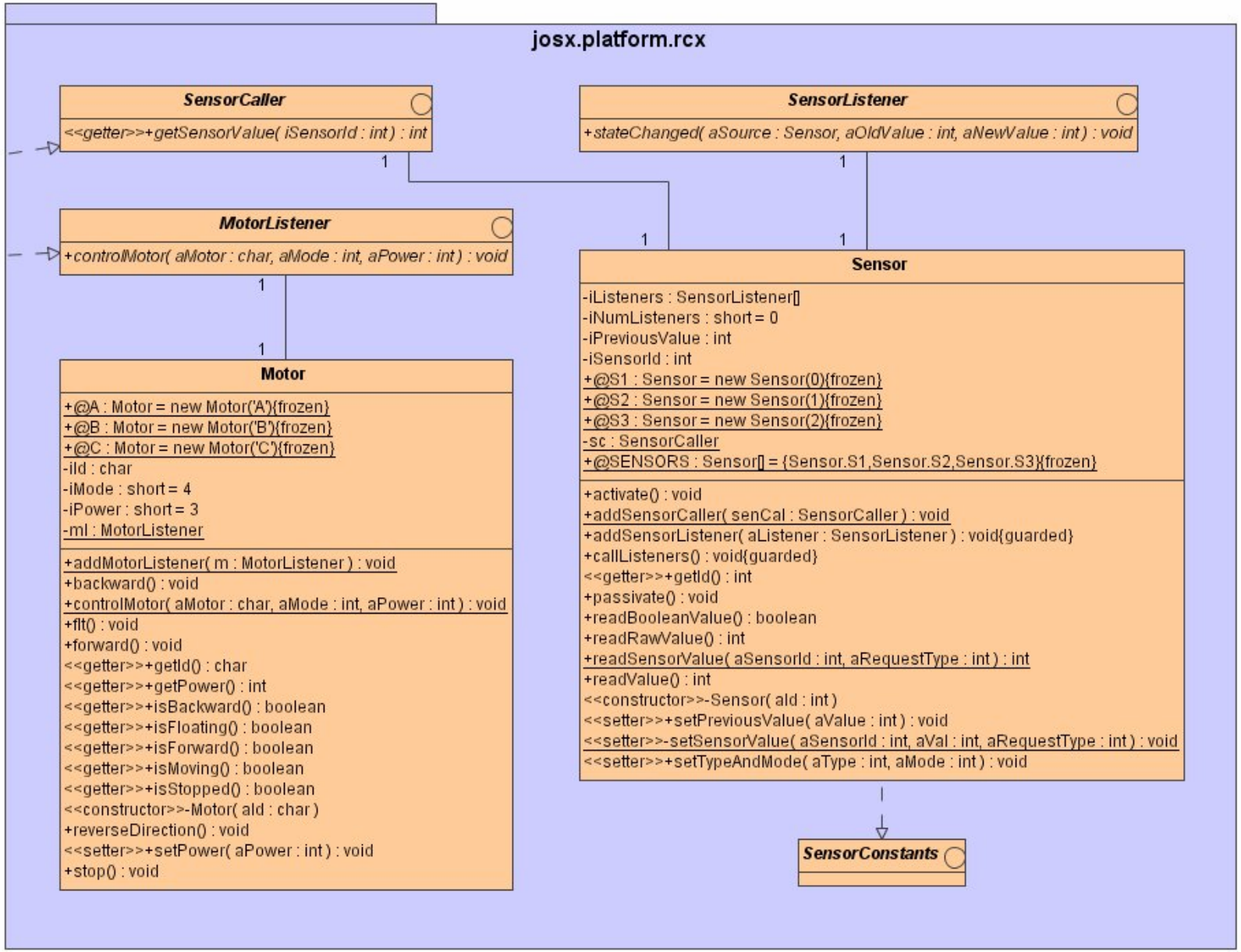
1.3 robi.sim.gui



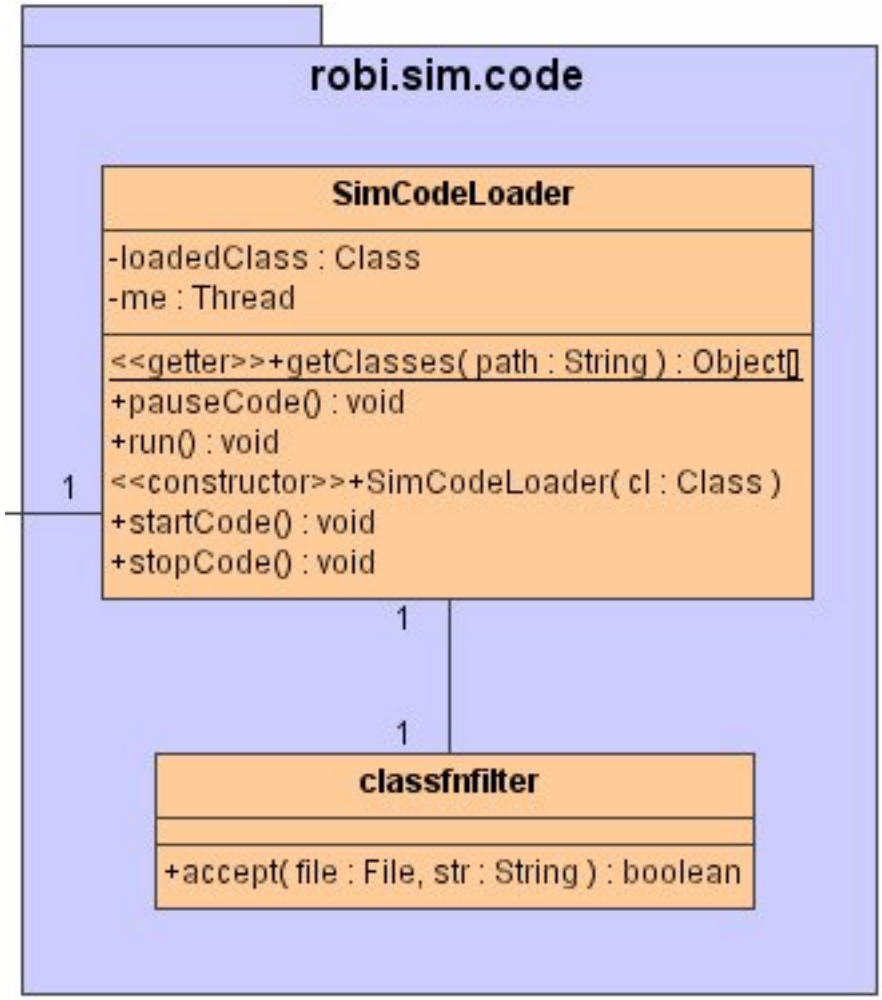
1.4 robi.sim.objects



1.5 josx.platform.rcx



1.6 robi.sim.code



2 Aufteilung der Pakete

2.1 robi

In diesem Paket befindet sich die Start Klasse *SimStart.*

2.2 robi.code

In diesem Paket befinden sich die Klassen die auch auf dem Roboter lauffähig sind. Aus diesem Paket werden alle Klassen mit einer Main Methode ausgelesen und im DropDown Feld für die Programmwahl angezeigt.

2.3 robi.sim.code

In diesem Paket befindet sich die Klasse *SimCodeLoader* welche den LeJos Code lädt und dann in einem eigenen Thread auch ausführt.

2.4 robi.sim.gui

In diesem Paket befinden sich alle Klassen die etwas grafisch darstellen.

SimInfo	Beinhaltet das Info Fenster (Version, Credits, ...)
SimJFrame	Dies ist das Hauptfenster. Hier wird das *SimJPanel* und das *SimOptionJPanel* aufgesetzt
SimJMenuBar	MenuBar des Hauptfensters
SimJPanel	Dieses Panel stellte die simulierte Welt dar. Hier spielt sich alles ab.
SimJPoupupMenu	Das PopupMenu im *SimJPanel*. Von hier wird zum Beispiel die Simulation gestartet.
SimOptionJPanel	Der linke Teil des Simulators. Von dieser Klasse werden die Panels abgeleitet welche Informationen zu den Objekten in der Simulation anzeigen.
SimRoboOption	In diesem Panel werden die Informationen zum Roboter während der Simulation angezeigt.
SimValueField	Eine Anzeigekomponente für *SimRoboOption*

2.5 robi.sim.objects

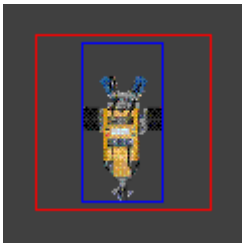
In diesem Paket befinden sich die Objekte die sich in der simulierten Welt befinden.

SimObject	Abstrakte Klasse welche ein Simulationsobjekt darstellt. Von dieser Klasse werden alle Simulationsobjekte abgeleitet.
SimLight	Ein Licht in der Simulation. Abgleitet von *SimObject*
SimRobot	Der Roboter in der Simulation. Abgleitet von *SimObject*
SimSensor	Ein Sensorobjekt in der Simulation.
SimObjectPopup	Das PopupMenu das über einem Simulationsobjekt angezeigt wird.

3 Implementation des Roboters

3.1 Allgemein

Für den Roboter wurde eine eigene JComponent entwickelt. In ihr befindet sich das Bild des Roboters. Die Länge und die Breite das JComponent ergibt sich aus der Diagonale des Roboterbildes. (damit man hinterher das Bild auch um 360 Grad drehen kann ohne das ein Teil abgeschnitten wird)

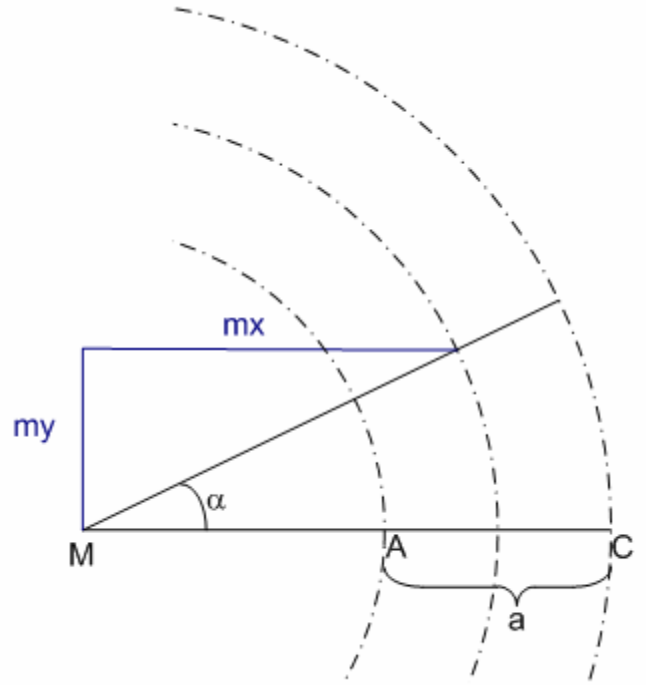


Der rote Rahmen kennzeichnet die Grösse der Roboterkomponente.
Der blaue Rahmen kennzeichnet die Grösse des effektiven Bildes.

In der Simulation hat der Roboter 2 Motoren und 2 Sensoren. Von der Klassenstruktur her könnte er aber 3 Motoren und 3 Sensoren haben (wie bei der Lego RCX vorgesehen.

3.2 Bewegung

Die Drehung wird in der Komponente erledigt (immer um den Mittelpunkt des Bildes). Damit sich nun der Roboter z.B. um das linke Rad drehen kann (Motor A:0, Motor C:7) muss sich aber zusätzlich auch noch die ganze Komponente verschieben. Bei allen Drehungen (Motor A != Motor C) kommt folgende Formel zum Einsatz:



Der Winkel Alpha bestimmt die Geschwindigkeit mit welcher der Roboter eine Drehung macht. Die Verschiebung geschieht dann folgendermassen:

- Momentane Position wird festgehalten
- Drehung um Winkel Alpha
- Berechnung der neuen Position
- Verschiebung um die Differenz

Formeln zur Berechung der Position

$$mx = \left(\frac{a \cdot |MA|}{|MC| - |MA|} + \frac{a}{2} \right) \cdot \cos(\alpha)$$

$$my = \left(\frac{a \cdot |MA|}{|MC| - |MA|} + \frac{a}{2} \right) \cdot \sin(\alpha)$$

wobei

a = Radabstand = Bildbreite

$|MA|$ kann mit der Geschwindigkeit von Motor A gleichgestellt werden

$|MC|$ kann mit der Geschwindigkeit von Motor C gleichgestellt werden

ist.

3.3 Aktualisieren der Roboterposition

Die neue Berechnung der Roboterposition wird in einem eigenen Thread (Methode *run()* in der Klasse SimRobot) angestossen. Die effektive Berechung findet in der Methode *step()* in der Klasse SimRobot statt.

4 Implementation der Sensoren

Für die Sensoren wurde eine eigene Klasse einwickelt, so ist es auch problemlos möglich ein bis drei Sensoren zum Roboter hinzuzufügen und diese dann interaktiv reagieren zu lassen.

4.1 Berechnung der Sensorwerte

Da die Lichtsensoren sowohl auf die Lichtstärke als auch auf den Einfallswinkel reagieren, sind einige Berechnungen nötig bevor die effektiven Sensorwerte ausgelesen werden können. Zudem gibt es sowohl zur Distanz (macht Lichtstärke aus) als auch zum Einfallswinkel gemessene Kennlinien die in beiden Fällen nicht linear sind.

4.1.1 Distanz zu einer Lichtquelle

Die Distanz zu einer Lichtquelle wird folgendermassen berechnet:

$$dist = \sqrt{\left(\left|sx - lx\right|\right)^2 + \left(\left|sy - ly\right|\right)^2}$$

sx, sy : Position des Sensors
lx, ly : Position des Lichts
$dist$: Distanz zum Licht

4.1.2 Winkel relativ zur Lichtquelle

Der Winkel zur Lichtquelle wird folgendermassen berechnet:

$$x = lx - sx$$

$$y = -ly + sy$$

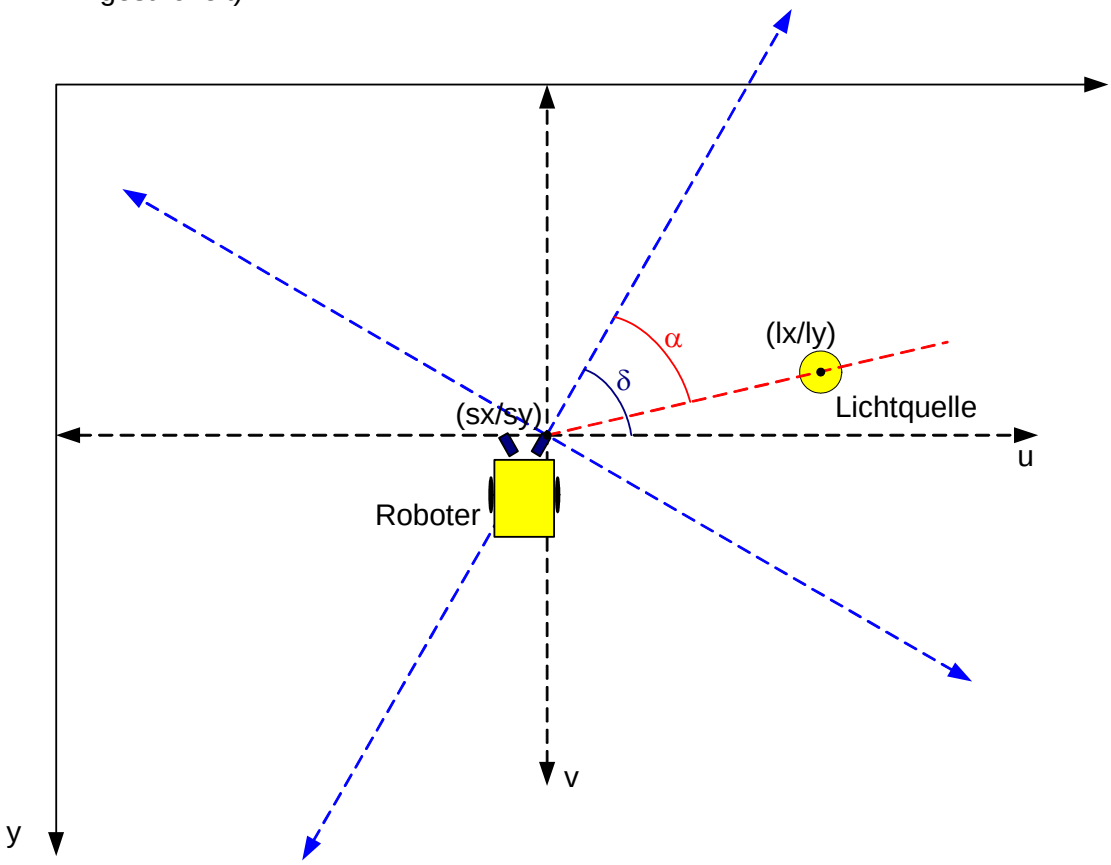
$$\alpha = \tan\left(\frac{y\cos(\delta) - x\sin(\delta)}{y\sin(\delta) + x\cos(\delta)}\right)$$

sx, sy : Position des Sensors
lx, ly : Position des Lichts
δ : Richtung des Sensors
α : Winkel der Lichtquelle relativ zum Sensor

wobei natürlich gilt
Quadrant I: α
Quadrant II: $\alpha + \pi$
Quadrant III: $\alpha - \pi$
Quadrant IV: α

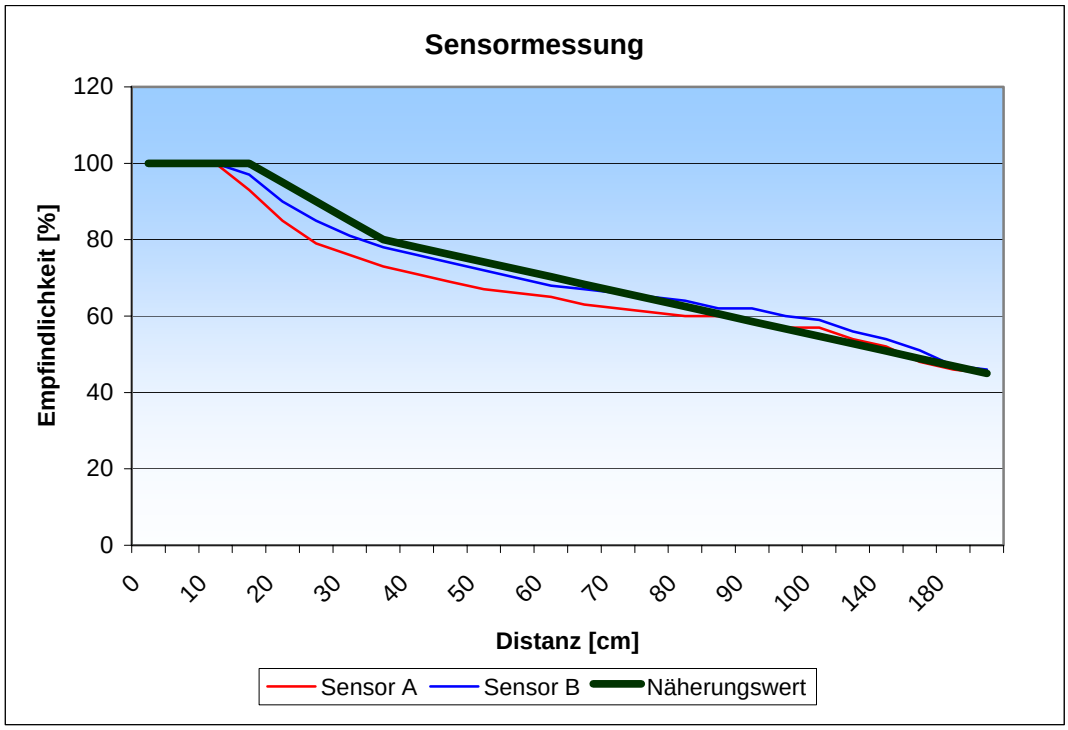
Folgendes geschieht:

1. Das Koordinatensystem wird verschoben, so dass der Ursprung neu beim Sensor ist (schwarz gestrichelt).
2. Das Koordinatensystem wir um den Winkel des Sensors gedreht (blau gestichelt).
3. Der Winkel zur Lichtquelle (jetzt relativ zum Sensor) wird berechnet (rot gestrichelt).



4.1.3 Kennlinie für die Distanz

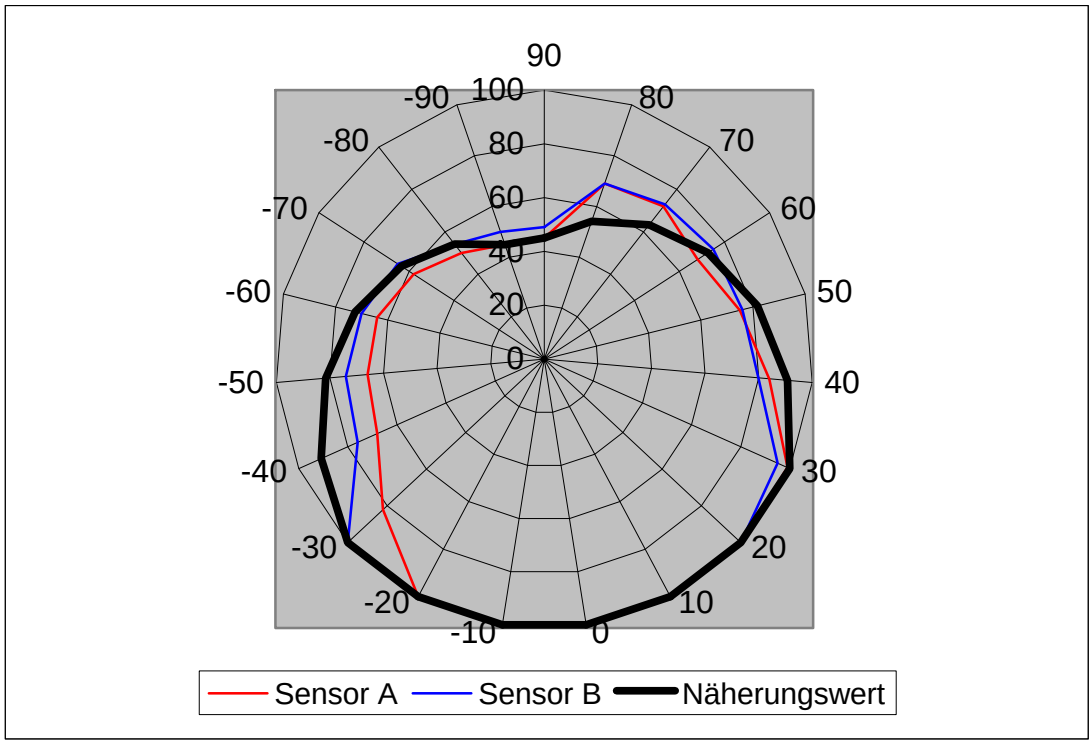
Die Empfindlichkeit im Verhältnis zur Distanz ist nicht linear. Es wurde eine Annäherung zur gemessenen Kennlinie gewählt.



Die rote und blaue Linie kennzeichnen je die Messung des 1. und 2. Sensors. Die schwarze Linie ist die für die Simulation gewählte Annäherung.

4.1.4 Kennlinie für den Einfallswinkel

Die Empfindlichkeit im Verhältnis zum Einfallswinkel ist ebenfalls nicht ganz linear. Es wurde auch hier eine Annäherung zur gemessenen Kennlinie gewählt.



Die rote und blaue Linie kennzeichnen je die Messung des 1. und 2. Sensors. Die schwarze Linie ist die für die Simulation gewählte Annäherung.

4.1.5 Mehrere Lichtquellen

Mit all den oben erwähnten Berechnungen ergibt sich nun der Sensorwert für eine Lichtquelle. Wenn nun der Sensor unter dem Einfluss von mehreren Lichtquellen steht, werden die Werte addiert. Der Sensorwert beträgt jedoch niemals mehr als 100.

4.2 Aktualisieren der Sensorwerte

Die Sensorwerte werden in einem eigenen Thread (Methode *run()* in der Klasse SimSensor) pro Sensor berechnet. Damit ist sichergestellt, dass die Sensorwerte immer aktuell sind. Der Refresh der Daten geschieht mit einer doppelt so hohen Kadenz als der Refresh der Roboterbewegung (in der Klasse SimRobot).

5 Verwendete Tools und Java Version

Diese Simulation wurde in Eclipse 2.1 unter Java 1.3.1 entwickelt.