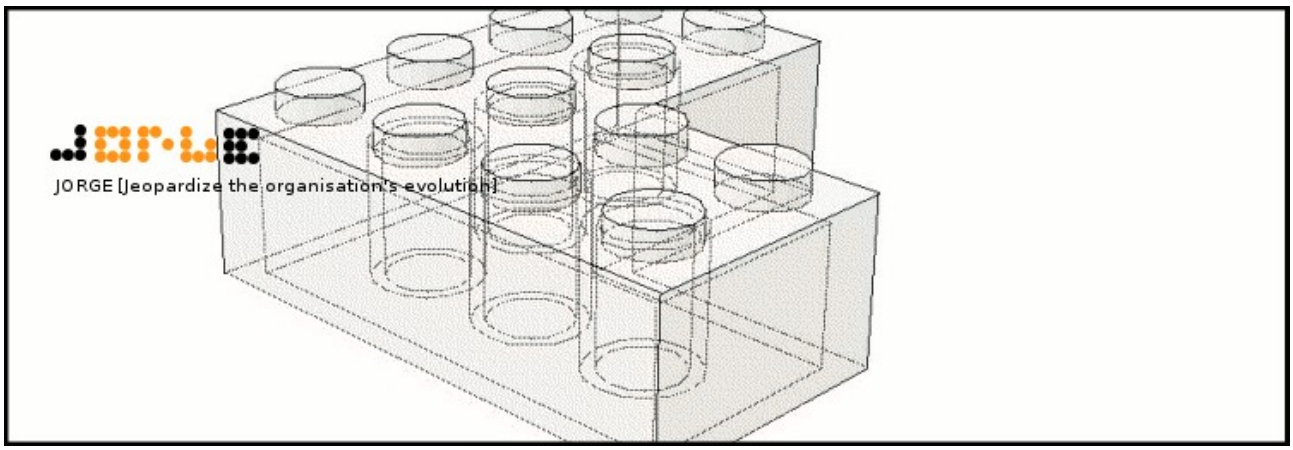
Berner Fachhochschule
Hochschule für Technik und Informatik HTI



Benutzerhandbuch

[Jorge - Lego MindStorms]

Autoren	Nik Lutz [I3STW, lutzn@hti.bfh.ch]
	Stefan Feissli [I3STW, feiss@hti.bfh.ch]
	Christof Seiler [I3STW, seilc@hti.bfh.ch]
Version	1.05
Datum	23.06.2005
Status	Freigegeben
Webseite	jorge.hta-bi.bfh.ch

Inhaltsverzeichnis

1 Einleitung

1.1 Zweck des Dokuments

Diese Dokument dient als Gebrauchsanweisung und beschreibt alle erforderlichen Schritte um erfolgreich eine Simulation auszuführen.

2 Vor dem Start

Um die Applikation laufen müssen zuerst alle Installationsschritte durchführt werden, dazu verweisen wir auf unsere Installationanweisung, die auf http://jorge.hta-bi.bfh.ch heruntergelanden werden kann. Ausserdem muss LeJos (http://lejos.sourceforge.net) installiert sein.

3 LeJos Unterstützung

- Zur Programmierung kann das ganze LeJos-API (http://lejos.sourceforge.net/apidocs/) genutzt werde
- Der Motor A steuert die linken Räder des Roboters
- Der Motor C steuert die rechten Räder des Roboters
- Stösst der linke Fühler gegen ein Hindernis wird der Berührungs-Sensor S1 ausgelöst
- Stösst der rechte Fühler gegen ein Hindernis wird der Berührungs-Sensor S3 ausgelöst
- Alle anderen Anweisungen (wie z.B. Motor B, Sensor S2 oder Anweisungen fürs LCD-Panel) werden vom Roboter ignoriert. Die Aktion wird aber im Overlay “Emulator Log“ angezeigt.

3.1 LeJos Beispielprogramm

Der folgende Beispiel-Code definiert das Verhalten eines einfachen Roboters: Solange keiner der zwei Sensoren ausgelöst wird, fährt der Roboter vorwärts. Stösst der Roboter mit dem rechten Fühler gegen ein Hindernis, weist das Program den Roboter an, während 3 Sekunden rückwärts zu fahren, danach während 2 Sekunden nach links zu drehen, und dann wieder forwärts weiterzufahren. Für den linken Fühler ist das Verhalten analog.

Robot.java

```
import josx.platform.rcx.*;

public class Robot {
    public static int LEFT_SENSOR = 1;
    public static int RIGHT_SENSOR = 3;


    Sensor touchSensorL = Sensor.S1;
    Sensor touchSensorR = Sensor.S3;

    public Robot (){
         touchSensorL.setTypeAndMode(SensorConstants.SENSOR_TYPE_TOUCH,
                                     SensorConstants.SENSOR_MODE_BOOL  );
         touchSensorR.setTypeAndMode(SensorConstants.SENSOR_TYPE_TOUCH,
                                     SensorConstants.SENSOR_MODE_BOOL  );

                touchSensorL.addSensorListener(new TouchBackListener(this, LEFT_SENSOR));
         touchSensorR.addSensorListener(new TouchBackListener(this, RIGHT_SENSOR));
     }

// TURN
    public void go_right(){
      Motor.A.forward();
      Motor.C.backward();
    }

    public void go_left(){
      Motor.A.backward();
      Motor.C.forward();
    }

// Drive
    public void go_forward(){
      Motor.A.forward();
      Motor.C.forward();
    }

    public void go_backward(){
      Motor.A.backward();
      Motor.C.backward();
    }

// Stop
    public void stop(){
      Motor.A.stop();
      Motor.C.stop();
    }

    public void pause(int delay){
      try {
          Thread.sleep(delay);
          stop();
      }catch (InterruptedException ie) { }
    }

// Start

    public void go(){
        this.go_forward();
        while (true) {  // forward forever... }
    }

    public static void main(String[] args){
        Robot robo = new Robot();
        robo.go();
    }
}
```

TouchBackListener.java

Der TouchBackListener verarbeitet Sensor-Ereignisse und benutzt die in Robot.java definierten Funktionen zum Steuern des Roboters.

```
import josx.platform.rcx.Sensor;
import josx.platform.rcx.SensorListener;

public class TouchBackListener implements SensorListener {
        private Robot touchRobot;
        private int side= -1;


            public TouchBackListener(Robot touchRobot, int side) {
                this.touchRobot = touchRobot;
                this.side = side;
        }

        public void stateChanged(Sensor s, int oldValue, int newValue){
        if (s.readBooleanValue()){
                        if(this.side == Robot.LEFT_SENSOR){
                                touchRobot.go_backward();
                                touchRobot.pause(3000);

                                touchRobot.go_right();
                                touchRobot.pause(2000);

                                touchRobot.go_forward();
                        }
                        if(this.side == Robot.RIGHT_SENSOR){
                                touchRobot.go_backward();
                                touchRobot.pause(3000);

                                touchRobot.go_left();
                                touchRobot.pause(2000);

                                touchRobot.go_forward();
                        }
           }
        }
}
```

4 Vom Code zur Simulation – Ein Schritt für Schritt Beispiel

In diesem Abschnitt wird Schritt für Schritt ein Beispiel durchlaufen, um so die Hauptfunktionen von JORGE aufzuzeigen. Wir nutzen dabei das im Abschnitt 3 Beschriebene LeJos-Program.

4.1 Beispielprogram Kompilieren

LeJos-Programme müssen mit LeJos kompiliert werden, damit sie von JORGE ausgeführt werden können. Dafür muss LeJos (http://lejos.sourceforge.net) installiert sein.

4.1.1 CLASSPATH setzen

Für den Compiler muss in der Kommandozeile (Terminal/Konsole) die Umgebungsvariable CLASSPATH im minimum das Installations-Verzeichnis von LeJos und den aktuellen Ordner enthalten. In diesem Beispiel gehen wir davon aus, dass LeJos in "/home/jorge/apps/"bzw. in "c:\apps\" installiert ist, wenn sie LeJos auf ihrem System in einem anderen Ordner installiert haben, müssen sie die Beispiele entsprechend anpassen.

Linux (bash-shell):

```
export CLASSPATH=/home/jorge/apps/lejos_2_1_0/:.
```

Windows (cmd.exe):

```
set CLASSPATH=c:\apps\lejos_2_1_0/;.
```

4.1.2 Kompilieren

Zum Kompilieren nutzt man den LeJos-Compiler und gibt ihm die Java-Datei an, welche die Main-Methode enthält:

Linux (bash-shell):

```
/home/jorge/apps/lejos_2_1_0/bin/lejosc Robot.java
```

Windows (cmd.exe):

```
c:\apps\lejos_2_1_0/bin/lejosc.exe Robot.java
```

4.1.3 Die Klassen zu einer Datei Zusammenführen (linken)

Da Java-Programme meistens aus mehreren Klassen bestehen, und beim kompilieren mehrere .class-Dateien entstehen, müssen diese zu einer Datei verschmolzen werden.

Linux (bash-shell):

```
/home/jorge/apps/lejos_2_1_0/bin/emu-lejos Robot -o Robot.emu
```

Windows (cmd.exe):

```
c:\apps\lejos_2_1_0/bin/emu-lejos.exe Robot -o Robot.emu
```

4.2 Ausführen des Beispielprogramms

Dateien die wie im Abschnitt X:Y erzeugt wurden, können in die Simulation geladen werden. Hierfür muss der Ort der .emu-Datei JORGE beim Start als Argument übergeben werden.
Wie oben beschrieben kann ein eigenes LeJos Programm erstellt und mit JORGE ausgeführt werden. In diesem Beispiel werden wir das Programm aus Abschnitt 3 ausführen.

Linux:

./jorge <pfad zum program>/Robot.emu

Auf Linux müssen nun die grafischen Einstellungen direkt an der Kommandozeile eingestellt werden. Wir empfehlen:

- Renderer: OpenGL Rendering SubSystem
- FSAA: 0
- Full Screen: Yes
- Video Mode: Je nach Auflösung ihres Bildschirm ist diese Einstellung unterschiedlich. Es kann die höchste verfügbare Auflösung ausgewählt werden.

Windows (cmd.exe):

jorge.exe <pfad zum program>\Robot.emu

Wenn sie diesen Befehl ausführen, werden sie über ein Dialogfenster gefragt, welche grafischen Einstellung sie vornehmen möchten. In unserem Fall können sie die Einstellung wie folgt vornehmen:

- Rendering Subsystem: Direct3D9 Rendering SubSystem
- Anti aliasing: None
- Floating-point mode: Fastest
- Rendering Device: [Erkannte Grafikkarte]
- VSync: Yes

- Video Mode: Je nach Auflösung ihres Bildschirm ist diese Einstellung unterschiedlich. Es kann die höchste verfügbare Auflösung ausgewählt werden.

4.2.1 Positionieren des Roboters

Nun startet die eigentliche Applikation und man kann dem Roboter durch die Umgebung navigieren. Mit den Tasten “I“ und “R“ können nun die Motoren eingeschaltet und mit “K“ und “D“ wieder ausgeschaltet werden. Vor dem Einschalten des Emulators empfehlen wir den Roboter zuerst in Richtung Labyrinth zu navigieren, so wird der Roboter auch auf entsprechende Fühler-Events reagieren.

4.2.2 Ausführen des Beispielprogramms

Nachdem sich der Roboter an der gewünschten Stelle befindet, kann der Benutzer mit der Taste “O“ die Simulation starten. Zum Beenden der Applikation drücken sie einfach die “Esc“ Taste.

5 Übersicht

5.1 LeJos Programme

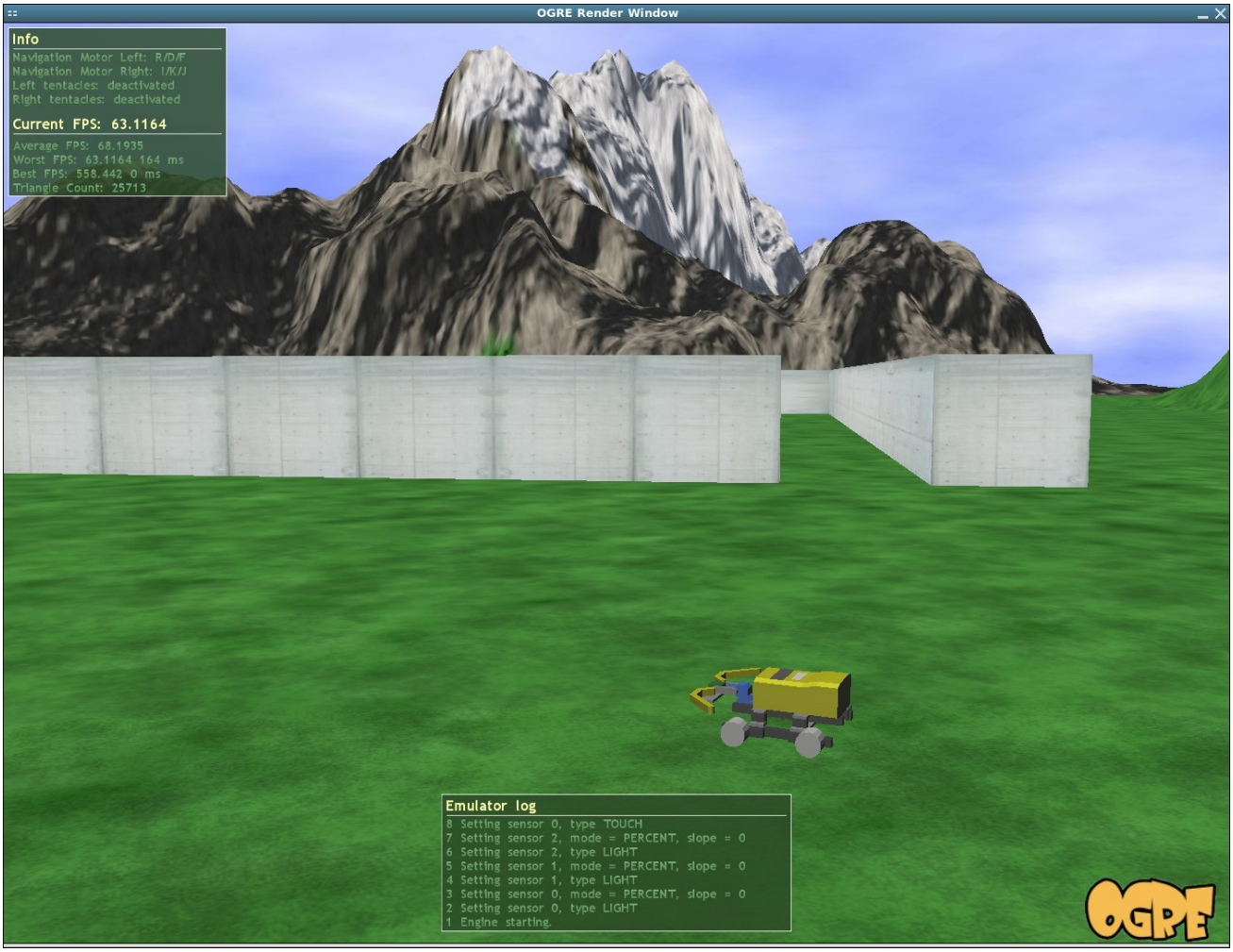
Zum Kompilieren kann der Kompilier von LeJos benutzt werden. D.h. es können die für LeJos üblichen Verfahren zum Kompilieren angewendet werden. Diese sind auf der Hompage von LeJos dokumentiert. Zum Linken der .class Dateien sollte *emu-lejos* benutzt werden, das sie ebenfalls im bin/ Verzeichnis der LeJos-Installation finden.

5.2 Program Starten

Beim Start von JORGE muss der vollständige Pfad eines mit *emu-lejos* kompilierten Programs übergeben werden.

5.3 Das JORGE Fenster

- Oben rechts im Abschnitt “Info“ werden Informationen über den Zustand der Motoren und Sensoren des Roboters angezeigt.
- “Current FPS“ (Frames Per Seconds) steht für die aktuelle Anzahl von Bildern die pro Sekunde angezeigt werden. Direkt darunter finden sie eine kleine Statistik über den Verlauf der FPS und die Anzahl angezeigter Dreiecke (aus denen das Bild aufgebaut wird).
- Das Feld “Emulator log“ zeigt die letzten acht Anweisungen die ihr Program dem Roboter gegeben hat. Dort sehen sie z.B. wie ihr Program die Sensoren initialisiert, wann Sensor-Daten gelesen werden, welcher Motoren eingeschaltet wird und vieles mehr.



5.3.1 Navigationsmodus

Der Navigationsmodus ist der Default Modus wenn das Programm gestartet wird. In diesem Modus ist es möglich mit der Tastatur den Roboter direkt durch die Landschaft zu navigieren. Jeder Tastendruck wird sogleich ausgeführt, und führt zu einer Reaktion auf dem Bildschirm.

5.3.2 Simulationsmodus

In den Simulatinosmodus wechselt man mit der Taste “o“, um ein vorher programmiertes LeJos Programm auszuführen. Sobald vom Navigations- in den Simulationsmodus gewechselt wird, beginnt die Simulation. Um das Programm

wieder zu beenden kann entweder auf das Ende des LeJos-Programmes gewartet werden oder man drückt die Taste ESC.

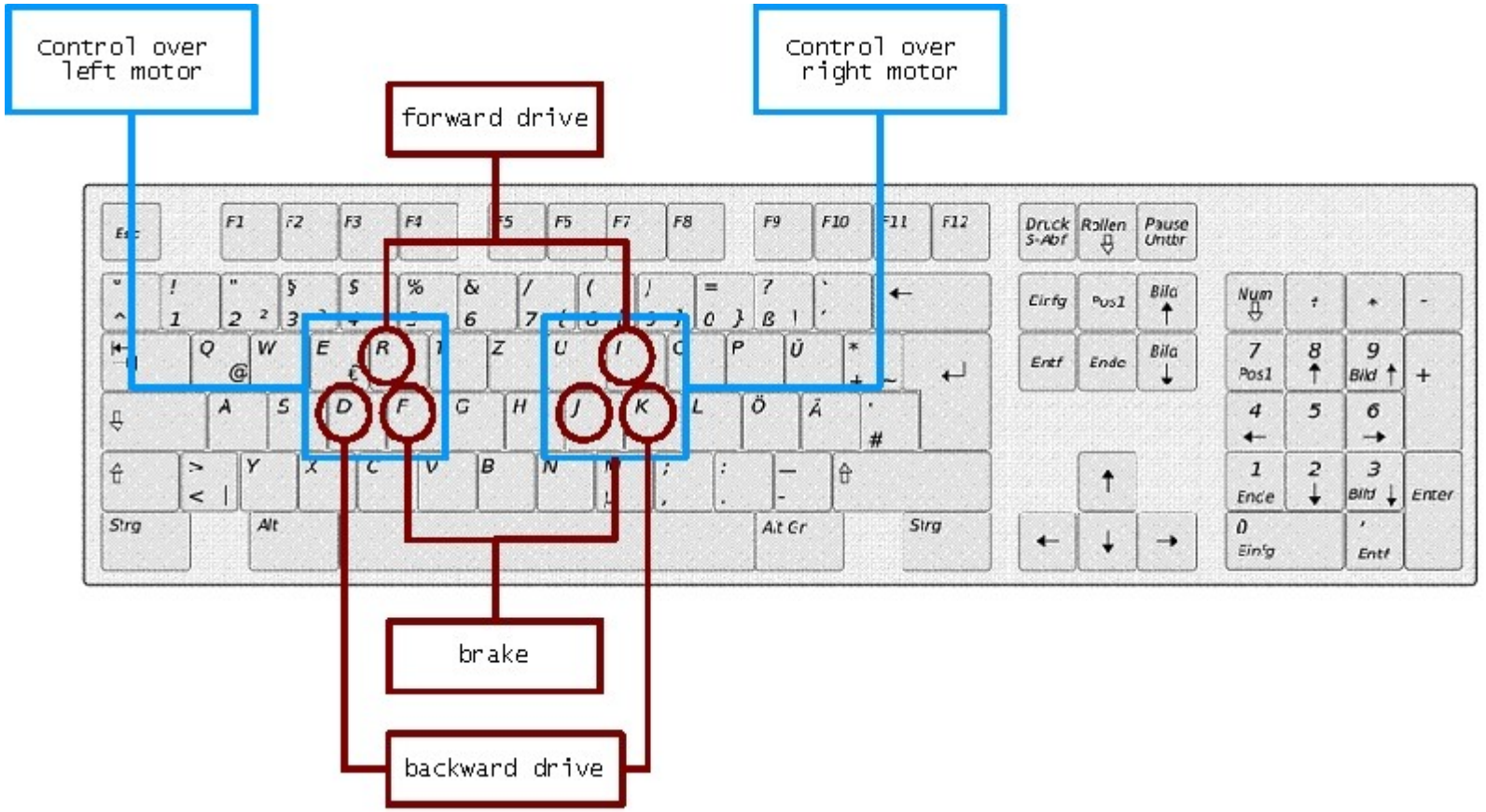
5.3.3 Tastenverteilung

Die Simulation kann durch Inputs vom Keyboard kontrolliert werden. Die folgende Tabelle zeigt alle verfügbaren Funktionen:

Key	Funktion
Q	Welchselt in drei verschiedene Darstellungsmodi: SOLID, WIREFRAME und POINTS. Defaultwert ist SOLID.
W	Schaltet die Infofenster auf dem Bildschrim ein und aus.
E	Zeigt die Boundingboxes des Roboters an.
R	Der rechte Motor schaltet den Vorwärtsgang ein.
A	Schwenkt die Kamera nach links.
S	Schwenkt die Kamera nach rechts.
D	Der linke Motor stoppt.
F	Der linke Motor schaltet den Rückwärtsgang ein.
R	Der linke Motor schaltet den Vorwärtgang ein.
K	Der rechte Motor stoppt.
J	Der rechte Motor schaltet den Rückwärtsgang ein.
I	Der rechte Motor schaltet den Vorwärtgang ein.
O	Umschalten von Navigations- in den Simulationsmodus und umgekehrt.

Zusätzlich kann mit den Pfeiltasten und der Maus die Kamera bewegt werden.

In der Grafik auf der nächsten Seite sind die wichtigsten Tasten zur Navigation des Roboters abgebildet. Eine besondere Rolle kommt der Taste “o“ zu, welche zur Umschaltung in den Simulationsmodus dient. Der Simulationsmodus führt dann, das per Kommandozeile geladene LeJos Progamm, aus.


Control over left motor
forward drive
Control over right motor
brake
backward drive

6 Kontakt

Bei Fragen oder Bugreports zögern sie nicht uns per Email zu kontaktieren. Die Adressen finden Sie auf der ersten Seite.

Viel Erfolg und viel Spass beim Ausprobieren.

Team Jorge