ANHANG ZU

AUTOMATISIERTE CHARAKTERE IN VIRTUELLEN WELTEN

von

Hauke Ernst

Informatik
Universität Bremen

zur Erlangung des akademischen Grades

Diplom Informatiker

vorgelegte Diplomarbeit

Gutachter:

1. Professor Dr. rer. nat. Wolfgang Coy
2. Professor Dr.-Ing. Friedrich-Wilhelm Bruns

Bremen, den 24. Juli 1997

INHALTSVERZEICHNIS

1 Der VRML-Prototyp VRRobot 1

1.1 Aufgabenstellung 1

1.2 Funktionsbeschreibung 1

1.3 Modularisierung 2

1.4 Definition des Prototypen in VRML2 2

1.5 Instanziierung des Prototypen 6

2 Das Modul VRMLInterface 7

2.1 Aufgabenstellung 7

2.2 Funktionsbeschreibung 7

2.3 Modularisierung 7

2.4 Die Klasse VRMLInterface 7

2.5 Die Klasse VRMLEventIn 17

2.6 Die Klasse VRMLEventOut 18

2.7 Die Klasse TextOutput 22

2.8 Die Klasse TextInput 23

2.9 Die Klasse LinkedList 24

2.10 Die Schnittstelle ResultProcessor 26

2.11 Die Schnittstelle VRMLController 26

2.12 Die Schnittstelle VRMLEvent 27

3 Das Modul VRRobot 28

3.1 Aufgabenstellung 28

3.2 Funktionsbeschreibung 28

3.3 Modularisierung 32

3.4 Die Klasse VRRobot 33

3.5 Die Klasse Procedure 58

3.6 Die Klasse Variable 61

3.7 Die Klasse Expression 63

3.8 Die Klasse Answer 69

3.9 Die Klasse FileIO 74

3.10 Die Klasse MainDialog 77

3.11 Die Klasse KonfigDialog 80

3.12 Die Klasse KonfigMotion 81

3.13 Die Klasse KonfigAnswers 81

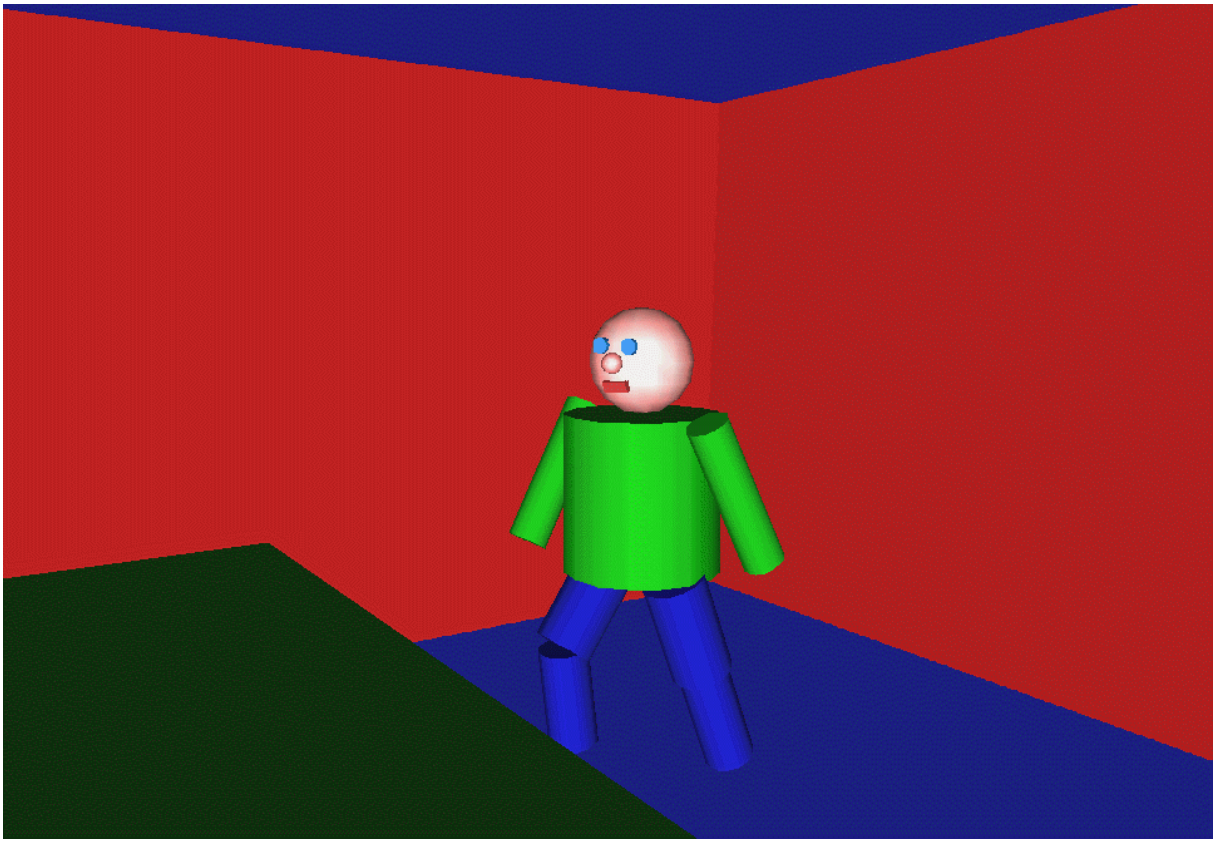
3.14 Die Schnittstelle Answering 82

4 Das Modul SchemeRobot 83
4.1 Aufgabenstellung 83
4.2 Funktionsbeschreibung 84
4.3 Modularisierung 85
4.4 Die Klasse SchemeRobot 86
4.5 Die Klasse Scheme 92
4.6 Die Klasse SchemeDialog 103
4.7 Die Klasse KonfigSchemeDialog 106
4.8 Die Schnittstelle SchemeContainer 108
5 Das Beispiel Museumsführer 109
5.1 Szenenbeschreibung in VRML2 109
5.2 Handlungsplan für den Museumsführer 111
5.3 Dialogkonfiguration für das Klee-Schema 115
5.4 Dialogkonfiguration für das Kandinsky-Schema 115
5.5 Dialogkonfiguration für das Matisse-Schema 116

1 Der VRML-Prototyp VRRobot

1.1 Aufgabenstellung

Die Aufgabe des Moduls „VRML-Prototyp VRRobot“ besteht in der Bereitstellung einer geeignetet 3D-Repräsentation des VRRobots, sowie der Implementation der Schnittstelle zu einem Java-Programm, welches die Verhaltenssteuerung übernimmt. In anderen VRML-Szenen kann dieser Prototyp dann instanziiert werden, wobei bestimmte Eigenschaften durch Parameterübergabe variiert werden können.



1.2 Funktionsbeschreibung

Die derzeitige Realisierung ist relativ einfach, Körper und Glieder des VRRobots bestehen nur aus einfachen Formen wie Zylindern, Quadern und Kugeln.

1.3 Modularisierung

Der beispielhafte VRML2-Prototyp VRRobot besteht aus mehreren VRML2-Knoten, deren Funktion direkt in der folgenden VRML2-Definition beschrieben ist.

1.4 Definition des Prototypen in VRML2

```
#VRML V2.0 utf8

# Beispielhafter Prototyp zur Demonstration der Javaklassen fuer die
# Handlungssteuerung automatisierter Charakter in virtuellen Welten.
# Autor: Hauke Ernst
# Version: 1.0
#Stand: 17.7.97

PROTO VRRobot [
# Parameter
    field SFVec3f translation 0 0 0
    field SFVec3f scale 1 1 1
     field SFRotation rotation 0 1 0 0
      field SFString ScriptFile "script/vrrobot.rcf"
      field MFString URL "../classes/VRRobot/vrrobot.class"
      field SFString Outs "robotpos,robotrot,headrot,
                        shank1rot,shank2rot,thigh1rot,thigh2rot,arm1rot,arm2rot,guide"
      field SFString Ins "move,clicked,proximitry,userpos,userrot,visibility,hunger_tick"
      field SFString Name "VRRobot"
   ] {
   DEF Robot Transform {
     translation IS translation
     scale        IS scale
     rotation     IS rotation

       children [
            # Zeitgeber fuer die Animation
        DEF HOURGLASS TimeSensor {
           loop TRUE
           enabled TRUE
           cycleInterval 0.1
           stopTime -1
         }
            # Zeitgeber fuer den Hunger-Zaehler
        DEF Hunger_Ticker TimeSensor {
           loop TRUE
           enabled TRUE
           cycleInterval 30
           stopTime -1
         }
            # Sensor fuer das Anklicken mit der Maus
        DEF RobotTouchSensor TouchSensor {}

            # Sensor fuer die Naehe des Benutzers
        DEF RobotProxSensor ProximitySensor { size 10 10 10 }

            # Sensor fuer die Sichtbarkeit durch den Benutzer
        DEF RobotVisSensor VisibilitySensor{}

            # Kamera-Definition, wenn der Benutzer die Welt aus Sicht des
            # VRRobot wahrnemen soll
        DEF TourGuide Viewpoint {
                  position 0 8.5 5
                  orientation 1 0 0  3.14
                  jump TRUE
                  fieldOfView 0.8
         }
            # Schnittstelle zur Einbindung der Javaklassen
         DEF RobotScript Script {
           url  IS URL
```

```
  scriptType "javabc"
  field SFString eventOuts IS Outs
  field SFString eventIns IS Ins
  field SFString KonfigFile IS ScriptFile
  field SFVec3f robotpos IS translation
    field SFString Name IS Name

  eventIn SFTime hunger_tick
  eventIn SFTime move
  eventIn SFBool clicked
  eventIn SFBool proximitry
  eventIn SFBool visibility
  eventIn SFVec3f userpos
  eventIn SFRotation userrot

  # Multiuser
  field SFNode robotobj USE Robot
  eventIn SFTime rpc_move
  eventIn SFString rpc_textin

  eventOut SFVec3f robotpos
  eventOut SFBool guide
  eventOut SFRotation robotrot
  eventOut SFRotation headrot
  eventOut SFRotation shank1rot
  eventOut SFRotation shank2rot
  eventOut SFRotation thigh1rot
  eventOut SFRotation thigh2rot
  eventOut SFRotation arm1rot
  eventOut SFRotation arm2rot
}
DEF Kopf Transform {
   translation 0 8.5 0
   children [
      Transform {
         children [
            Shape {
               appearance Appearance {
                  material Material { diffuseColor 1 0.5 0.5}
                }
                geometry Sphere {}
              }
            ]
         }
         DEF Nase Transform {
            translation 0 0 1
            children [
               Shape {
                  appearance Appearance {
                     material Material { diffuseColor 1 0.4 0.4}
                  }
                  geometry Sphere { radius 0.2 }
               }
            ]
         }
         DEF Auge1 Transform {
            translation 0.3 0.3 0.9
            rotation 1 0 0 1.57
            children [
               Shape {
                  appearance Appearance {
                     material Material { diffuseColor 0 0.5 1}
                  }
                  geometry Cylinder { radius 0.15 height 0.1 parts ALL}
               }
            ]
         }
         DEF Auge2 Transform {
            translation -0.3 0.3 0.9
            rotation 1 0 0 1.57
            children [
               Shape {
                  appearance Appearance {
                     material Material { diffuseColor 0 0.5 1}
                  }
                  geometry Cylinder { radius 0.15 height 0.1 parts ALL}
               }
            ]
         }
```

```
      DEF Mund Transform {
         translation 0 -0.4 0.82
         rotation 1 0 0 0.5
         children [
            Shape {
               appearance Appearance {
                  material Material { diffuseColor 1 0.1 0.1}
               }
               geometry Box { size 0.5 0.2 0.2}
            }
         ]
      }
   ]
}
DEF Oberkoerper Transform {
   translation 0 6 0
   children [
      Transform {
         children [
            Shape {
               appearance Appearance {
                  material Material { diffuseColor 0 1 0 }
               }
               geometry Cylinder{height 3 radius 1.5}
            }
         ]
      }
      DEF Arm1 Transform {
         center  0 1.5 0
         translation 1.9 0 0
         children [
            Shape {
               appearance Appearance {
                  material Material { diffuseColor 0 1 0 }
               }
               geometry Cylinder{height 3 radius .4}
            }
         ]
      }
      DEF Arm2 Transform {
        center  0 1.5 0
        translation -1.9 0 0
            children [
               Shape {
                  appearance Appearance {
                     material Material { diffuseColor 0 1 0 }
                  }
                  geometry Cylinder{height 3 radius .4}
               }
            ]
         }
      ]
   }
   DEF Unterkoerper Transform {
      translation 0 3.5 0
      children [
         DEF Bein1 Transform {
            center  0 1 0
            children [
               DEF Oberschenkel1 Transform {
                  translation 1 0 0
                  children [
                     Shape {
                        appearance Appearance {
                           material Material { diffuseColor 0 0 1 }
                        }
                        geometry Cylinder{height 2 radius 0.6}
                     }
                  ]
               }
               DEF Unterschenkel1 Transform {
                center  0 1 0
                translation 1 -2 0
                children [
                   Shape {
                       appearance Appearance {
                          material Material { diffuseColor 0 0 1 }
                       }
                       geometry Cylinder{height 2 radius 0.5}
```

```
                        }
                     ]
                  }
               ]
            }
            DEF Bein2 Transform {
              center  0 1 0
              children [
                 DEF Oberschenkel2 Transform {
                    translation -1 0 0
                    children [
                       Shape {
                          appearance Appearance {
                             material Material { diffuseColor 0 0 1 }
                          }
                          geometry Cylinder{height 2 radius 0.6}
                       }
                    ]
                 }
                 DEF Unterschenkel2 Transform {
                    center  0 1 0
                    translation -1 -2 0
                    children [
                       Shape {
                          appearance Appearance {
                             material Material { diffuseColor 0 0 1 }
                          }
                          geometry Cylinder{height 2 radius 0.5}
                       }
                    ]
                 }
              ]
           }
        ]
      }
        # EventIns/EventOut werden verknuepft
      ROUTE Hunger_Ticker.cycleTime               TO RobotScript.hunger_tick
      ROUTE HOURGLASS.cycleTime                   TO RobotScript.move
      ROUTE RobotScript.robotrot                  TO Robot.set_rotation
      ROUTE RobotScript.robotpos                  TO Robot.set_translation
      ROUTE RobotScript.headrot                   TO Kopf.set_rotation
      ROUTE RobotScript.thigh1rot                 TO Bein1.set_rotation
      ROUTE RobotScript.thigh2rot                 TO Bein2.set_rotation
      ROUTE RobotScript.shank1rot                 TO Unterschenkel1.set_rotation
      ROUTE RobotScript.shank2rot                 TO Unterschenkel2.set_rotation
      ROUTE RobotScript.arm1rot                   TO Arm1.set_rotation
      ROUTE RobotScript.arm2rot                   TO Arm2.set_rotation

      ROUTE RobotTouchSensor.isActive             TO RobotScript.clicked
      ROUTE RobotProxSensor.isActive              TO RobotScript.proximitry
      ROUTE RobotVisSensor.isActive              TO RobotScript.visibility
      ROUTE RobotScript.guide                     TO TourGuide.set_bind

       # Sensor zur Detektion der Benutzerposition
    DEF RobotUserPosSensor ProximitySensor { size 10000 10000 10000 }
      ROUTE RobotUserPosSensor.position_changed    TO RobotScript.userpos
      ROUTE RobotUserPosSensor.orientation_changed TO RobotScript.userrot
    }

    # Instanziierung mit den Standardparametern
    DEF Robot VRRobot {}
```

1.5 Instanziierung des Prototypen

Instanzen des Prototypen können mit Hilfe von EXTERNPROTO auch in anderen Weltdefinitionen erzeugt werden. Die Standardwerte der Parameter sind dabei nach Belieben ersetzbar:

```
#VRML V2.0 utf8
DEF CAMERA Viewpoint {
   position    0.0 5 20
   fieldOfView 0.8
}

EXTERNPROTO VRRobot [
      field SFString Name
    field SFVec3f translation
    field SFVec3f scale
     field SFRotation rotation
      field SFString ScriptFile
      field MFString URL
      field SFString Outs
      field SFString Ins
] "Proto.wrl#VRRobot"

VRRobot {
       translation 0 0 0
       scale 1 1 1
       ScriptFile "script/vrrobot.rcf"
       URL "../classes/VRRobot/vrrobot.class"
}
```

2 Das Modul VRMLInterface

2.1 Aufgabenstellung

Das Modul VRMLInterface soll die von den VRML-Browser-Herstellern bereitgestellte Klasse Script zur Ansteuerung von VRML-Objekten um weitergehende Funktionen erweitern. Insbesondere sollen Fuktionen zur Animation implementiert werden. Die Basisklasse Script enthält lediglich Funktionen zum Setzen einzelner Vektorwerte.

2.2 Funktionsbeschreibung

Auf der Grundlage eines Taktgebers, der als regelmäßiger EventIn („move“) realisiert ist, erlaubt das Modul VRMLInterface die Synchronisation seqentiell oder parallel ablaufender Bewegungen. Zum Starten und Anhalten linearer Rotationen und Translationen werden Funktionen bereitgestellt. Außerdem Die einzelnen Vektoren werden in einer dynamischen Schnittstellenverwaltung organisiert, wobei die zu verwaltenden Vektoren (EventIns und EventOuts) dem Modul beim Programmstart bekanntgemacht werden. Die aktuellen Werte aller Vektoren sind mit Hilfe entsprechender Funktionen abfragbar.

2.3 Modularisierung

Das Modul VRMLInterface besteht aus mehreren Klassen und Schnittstellen, die im folgenden einzeln beschrieben werden.

2.4 Die Klasse VRMLInterface

```
/**********************************************************************
* @(#)VRMLInterface.java 1.0 97/06/23 Hauke Ernst
*/
package VRMLInterface;

import vrml.node.Script;
import vrml.node.Node;
import vrml.Field;
import vrml.field.SFNode;
import vrml.field.SFString;
import vrml.field.ConstSFString;
import vrml.field.SFVec3f;
import vrml.field.ConstSFVec3f;
import vrml.field.SFRotation;
import vrml.field.ConstSFRotation;
import vrml.field.SFBool;
import vrml.field.ConstSFBool;
```

```
import vrml.field.SFTime;
import vrml.field.ConstSFTime;
import vs.Vscp;
import vs.Transform;

import java.awt.*;
import java.util.*;
import java.io.*;
import java.lang.*;

import VRMLInterface.*;

/**
* Klasse zur Bereitstellung grundlegender Animationfunktionen für VRML2-Objekte.
* Beim Initialisieren wird aus den VRML-Feldern ("fields") namens "EventIns" und "EventOuts"
* die Schnittstellendefinition gelesen. Diese wird jeweils als durch Kommata
* getrennte Liste mit den Namen der zu verwaltenden EventIns bzw. EventOuts
* übergeben.
* Beispiel (VRML2): field SFString eventIns
      "move,clicked,proximitry,userpos,userrot,visibility,hunger_tick"
*
* @author  Hauke Ernst
* @version 1.0
*/
public class VRMLInterface extends Script
   implements VRMLController
{
   LinkedList m_EventOutList;
   LinkedList m_EventInList;
   double aRad;
   double aRadx360;
   double aRadx180;
   private boolean m_Initialized;
   SFNode RobotObjNode;
   Transform RobotObj;
   String m_EventOutDescr;
   SFString m_EventOutDescrNode;
   String m_EventInDescr;
   SFString m_EventInDescrNode;

/** Standard-Konstruktor */
   public VRMLInterface() {
      super();
      m_Initialized=false;
   }

/**
* Wird von der Basisklasse Script nach dem Erzeugen der Klasse aufgerufen.
* Aufgaben sind
* das Einlesen der Schnittstellendefinition aus den fields
* "EventIns" und "EventOuts"
* sowie allgemeine Initialisierung.
*/
   public  synchronized void initialize () {
      try {
         if(m_Initialized) return;
         aRad = (double) (Math.PI/180.0) ;
         aRadx180 = (double) Math.PI;
         aRadx360 = 2*(double)Math.PI;
         m_EventOutList=new LinkedList();
         m_EventOutList.reset();
         m_EventInList=new LinkedList();
         m_EventInList.reset();
         try {
            RobotObjNode=null;
            RobotObj=null;
            RobotObjNode = (SFNode)getField("robotobj");
            RobotObj = (Transform)RobotObjNode.getValue();
         }
         catch(Exception e1)
         {
   //         TextOutput.TextOut("Caught Exception in script initialisation (MultiUser Extensi-
      on): " + e1);
         }
         try {
            m_EventOutDescrNode = (SFString)getField("eventOuts");
            m_EventOutDescr = (String)m_EventOutDescrNode.getValue();
            StringTokenizer t = new StringTokenizer(m_EventOutDescr," \t,\n\r");
            while(t.hasMoreElements()) {
```

```
               String tok = t.nextToken();
               AddEventOut(tok);
            }
         }
         catch(Exception e2)
         {
         }
         try {
            m_EventInDescrNode = (SFString)getField("eventIns");
            m_EventInDescr = (String)m_EventInDescrNode.getValue();
            StringTokenizer t = new StringTokenizer(m_EventInDescr," \t,\n");
            while(t.hasMoreElements()) {
               String tok = t.nextToken();
               AddEventIn(tok);
            }
         }
         catch(Exception e3)
         {
         }
         m_Initialized=true;
      }
      catch(Exception e)
      {
        TextOutput.TextOut("Caught Exception in VRMLInterface initialisation: " + e);
      }
   }

/**
* Behandlung von VRML-Events.
* @param e Der VRML-Event
*/
    public void processEvent(vrml.Event e) {
      try {
         if(!m_Initialized) {
            initialize();
         }
         if(e == null) return;
         if(e.getValue() == null) return;
         String name = e.getName ();
         if(name == null) return;
// Multiuser Erweiterung von Sony: Nur der Master darf Events behandeln
         if(name.equalsIgnoreCase("move")){
            if(Vscp.amIMaster()) {
               Vscp.sendApplSpecificMsgWithDist(RobotObjNode, "rpc_"+name,
                                                "", Vscp.allClientsExceptMe);
             }
             else {
                Vscp.sendApplSpecificMsgWithDist(RobotObjNode, name, "",
                                                 Vscp.responderOnly);
             }
         }
////////////////////////////////////////////////
     }

     catch(Exception ex)
     {
       TextOutput.TextOut("Caught Exception in VRMLInterface.processEvent: " + ex);
     }
   }

/**
* Berechnet für alle EventOuts den nächsten Zustand und gibt
* diese an die VRML-Welt weiter.
* Mit Hilfe dieser Funktion wird für eine synchrone Animation
* der verwalteten EventOuts gesorgt.
*
* @return False im Fehlerfall
*/
   public synchronized boolean Tick()
   {
      try {
         if(!m_Initialized) {
            initialize();
            return false;
         }
         m_EventOutList.reset();
         while(m_EventOutList.hasMoreElements()) {
            VRMLEventOut ev = (VRMLEventOut) m_EventOutList.currentElement();
            if(ev!=null) {
```

```
               IncEvent(ev);
            }
            else TextOutput.TextOut("VRMLInterface: Leeres Element in EventOut-Liste!");
            m_EventOutList.nextElement();
         }
         Vscp.updateObject(RobotObjNode); // Multiuser
      }
      catch(Exception e)
      {
        TextOutput.TextOut("Caught Exception in VRMLInterface.Tick: " + e);
      }
      return true;
   }

/**
* Berechnet für einen ausgewählten EventOut den nächsten Zustand und gibt
* diese an die VRML-Welt weiter.
* @param ev Der EventOut (SFRotation oder SFVec3f)
* @return False im Fehlerfall
*/
   public boolean IncEvent(VRMLEventOut ev)
   {
      ev.IncTick();
    try {
        if(ev.GetAktTick()==ev.GetTicks()+1) ev.SetInc((LinkedList)null);
        Field f=ev.GetAktField();
        Field i=ev.GetInc();
        Field r=ev.GetRefField();
        if(f==null || i==null || r==null) return false;
        if(f instanceof SFRotation) {
           SFRotation rotf = (SFRotation) f;
           SFRotation roti = (SFRotation) i;
           SFRotation rotr = (SFRotation) r;
           float [] arrf = new float [4];
           float [] arri = new float [4];
           float [] arrr = new float [4];
           rotf.getValue(arrf);
           roti.getValue(arri);
           rotr.getValue(arrr);
           for(int n=0; n<4; n++) arrf[n]= arrr[n]+(float)ev.GetAktTick()*arri[n];
           rotf = (SFRotation)ev.GetField();
           ev.SetAktField(new SFRotation(arrf[0],arrf[1],arrf[2],arrf[3]));
           rotf.setValue(arrf);
        }
        else if(f instanceof SFVec3f) {
           SFVec3f rotf = (SFVec3f) f;
           SFVec3f roti = (SFVec3f) i;
           float [] arrf = new float [3];
           float [] arri = new float [3];
           rotf.getValue(arrf);
           roti.getValue(arri);
           for(int n=0; n<3; n++) arrf[n]+=arri[n];
           rotf = (SFVec3f)ev.GetField();
           ev.SetAktField(new SFVec3f(arrf[0],arrf[1],arrf[2]));
           rotf.setValue(arrf);
        }
     }
     catch(Exception e)
     {
       TextOutput.TextOut("Caught Exception in VRMLInterface.IncEvent: " + e);
       return false;
     }
     return true;
   }

/**
* Ordnet dem übergebenen Namen ein EventOut-Field zu.
* @param name Der gesuchte EventOut
* @return Das EventOut-Field
*/
   public Field getEventOutField(String name)
   {
      return super.getEventOut(name);
   }

/**
* Ordnet dem übergebenen Namen ein EventIn-Field zu.
* @param name Der gesuchte EventIn
* @return Das EventIn-Field
```

```
*/
   public Field getEventInField(String name)
   {
      return getEventIn(name);
   }

/**
* Ordnet dem übergebenen Namen ein Field zu.
* @param name Das gesuchte Field
* @return Das Field
*/
   public Field getVRMLField(String name)
   {
      return getField(name);
   }

/**
* Fragt den aktuellen Wert eines EventOut vom Typ SFVec3f ab.
* Diese Funktion wird von der Basisklasse "Script" nicht zur Verfügung
* gestellt (Nur-Lese-Zugriff auf EventOuts) und wird an dieser Stelle
* basierend auf detr eigenen Schnittstellen-Verwaltung implementiert.
* @param name Der Name des zu animierenden EventOut (vom VRML-Typ SFVec3f)
* @param curr Array, in dem das Ergebnis eingetragen wird
* @return False im Fehlerfall
*/
   public boolean GetTranslationValue(String name,float [] curr)
   {
     try {
        VRMLEventOut ev =  FindEventOut(name);
        if(ev==null) {
            TextOutput.TextOut("VRMLInterface GetTranslationValue: "+name+" wurde nicht gefun-
      den!");
            return false;
        }
        SFVec3f f = (SFVec3f) ev.GetAktField();
        if(f==null) f=new SFVec3f(0,0,0);
        f.getValue(curr);
      }
      catch(Exception e)
      {
        TextOutput.TextOut("Caught Exception in VRMLInterface.GetTranslationValue: " + e);
        return false;
      }
      return true;
   }

/**
* Fragt den aktuellen Wert eines EventOut vom Typ SFRotation ab.
* Diese Funktion wird von der Basisklasse "Script" nicht zur Verfügung
* gestellt (Nur-Lese-Zugriff auf EventOuts) und wird an dieser Stelle
* basierend auf detr eigenen Schnittstellen-Verwaltung implementiert.
* @param name Der Name des zu animierenden EventOut (vom VRML-Typ SFRotation)
* @param curr Array, in dem das Ergebnis eingetragen wird
* @return False im Fehlerfall
*/
   public boolean GetRotationValue(String name,float [] curr)
   {
     try {
        VRMLEventOut ev =  FindEventOut(name);
        if(ev==null) {
            return false;
        }
        SFRotation f = (SFRotation) ev.GetAktField();
        if(f==null) f=new SFRotation(0,0,0,0);
        f.getValue(curr);
      }
      catch(Exception e)
      {
        TextOutput.TextOut("Caught Exception in VRMLInterface.GetRotationValue: " + e);
        return false;
      }
      return true;
   }

/**
* Startet eine Translations-Animation.
* @param ev Der zu animierende EventOut (vom VRML-Typ SFVec3f)
* @param trans Zielzustand des EventOut
* @param ticks Anzahl der Zeiteinheiten für die Animation
```

```
* @param autostop Soll die Bewegung nach Ablauf der Zeiteinheiten angehalten werden?
* @return False im Fehlerfall
*/
   public boolean StartTranslation(VRMLEventOut ev, SFVec3f trans, int ticks, boolean autostop)
   {
     try {

        SFVec3f f = (SFVec3f) ev.GetAktField();
        if(f==null) f=new SFVec3f(0,0,0);
        float [] curr = new float [3];
        f.getValue(curr);
        float [] newtrans = new float [3];
        trans.getValue(newtrans);

        SetVec3(ev,curr[0],curr[1],curr[2]);
        SFVec3f inc = new SFVec3f((newtrans[0]-curr[0])/ticks,
                                  (newtrans[1]-curr[1])/ticks,
                                  (newtrans[2]-curr[2])/ticks);
        ev.SetInc(inc);
        LinkedList reflist=new LinkedList();
        reflist.append(trans);
        reflist.reset();
        ev.SetRefList(reflist);
        if(autostop)
           ev.SetTicks(ticks);
        else
           ev.SetTicks(-1);
        ev.SetAktTick(0);
     }
     catch(Exception e)
     {
        TextOutput.TextOut("Caught Exception in VRMLInterface.StartTranslation: " + e);
        return false;
     }
     return true;
   }

/**
* Startet eine Translations-Animation.
* @param name Der Name des zu animierenden EventOut (vom VRML-Typ SFVec3f)
* @param trans Zielzustand des EventOut
* @param ticks Anzahl der Zeiteinheiten für die Animation
* @param autostop Soll die Bewegung nach Ablauf der Zeiteinheiten angehalten werden?
* @return False im Fehlerfall
*/
   public boolean StartTranslation(String name, SFVec3f trans, int ticks, boolean autostop)
   {
     try {
        VRMLEventOut ev =  FindEventOut(name);
        if(ev==null)
            TextOutput.TextOut("VRMLInterface StartTranslation: "+name+" wurde nicht gefun-
      den!");
        else return StartTranslation(ev,trans,ticks,autostop);
     }
     catch(Exception e)
     {
        TextOutput.TextOut("Caught Exception in VRMLInterface.StartTranslation: " + e);
     }
     return false;
   }

/**
* Startet eine Rotations-Animation.
* @param ev Der zu animierende EventOut (vom VRML-Typ SFRotation)
* @param rotlist Zielzustand des EventOut. Sind mehrere Werte angegeben,
* so werden diese nacheinander angefahren.
* @param ticks Anzahl der Zeiteinheiten für die Animation
* @param autostop Soll die Bewegung nach Ablauf der Zeiteinheiten angehalten werden?
* @return False im Fehlerfall
*/
   public boolean StartRotation(VRMLEventOut ev, LinkedList rotlist, int ticks,boolean autostop)
   {
     try {
        SFRotation f = (SFRotation) ev.GetAktField();
        if(f==null) f=new SFRotation(0,0,0,0);
        float [] currRot = new float [4];
        f.getValue(currRot);
        rotlist.reset();
        LinkedList inclist = new LinkedList();
```

```
        int i=0;
        while(rotlist.hasMoreElements()) {
           float [] newRot = new float [4];
           SFRotation rot= (SFRotation) rotlist.currentElement();
           rotlist.nextElement();
           rot.getValue(newRot);
           while(currRot[3] >   aRadx180) currRot[3] -= aRadx360;
           while(currRot[3] <= -aRadx180) currRot[3] += aRadx360;
           while(newRot[3] >    aRadx180) newRot[3]  -= aRadx360;
           while(newRot[3] <=  -aRadx180) newRot[3]  += aRadx360;
           if(i==0) ev.SetAktField(new SFRotation(newRot[0],newRot[1],newRot[2],currRot[3]));
           i++;
           double angle=((double)newRot[3]-(double)currRot[3]);
           while(angle >    aRadx180) angle  -= aRadx360;
           while(angle <=  -aRadx180) angle  += aRadx360;
           SFRotation inc = new SFRotation(0,0,0,(float) (angle/(double)ticks));
           inclist.append(inc);
           currRot[0]=newRot[0];
           currRot[1]=newRot[1];
           currRot[2]=newRot[2];
           currRot[3]=newRot[3];
        }
        inclist.reset();
        ev.SetInc(inclist);
        rotlist.reset();
        f = (SFRotation) ev.GetAktField();
        rotlist.insert(f);
        rotlist.reset();
        ev.SetRefList(rotlist);
        ev.SetAktTick(0);
        if(autostop)
           ev.SetTicks(ticks);
        else
           ev.SetTicks(-ticks);
     }
      catch(Exception e)
      {
        TextOutput.TextOut("Caught Exception in VRMLInterface.StartRotation: " + e);
        return false;
      }
      return true;
   }

/**
* Startet eine Rotations-Animation.
* @param name Der Name des zu animierenden EventOut (vom VRML-Typ SFRotation)
* @param rotlist Zielzustand des EventOut. Sind mehrere Werte angegeben,
* so werden diese nacheinander angefahren.
* @param ticks Anzahl der Zeiteinheiten für die Animation
* @param autostop Soll die Bewegung nach Ablauf der Zeiteinheiten angehalten werden?
* @return False im Fehlerfall
*/
   public boolean StartRotation(String name, LinkedList rotlist, int ticks,boolean autostop)
   {
     try {
        VRMLEventOut ev =  FindEventOut(name);
        if(ev==null)
            TextOutput.TextOut("VRMLInterface StartRotation: "+name+" wurde nicht gefunden!");
        else return StartRotation(ev,rotlist,ticks,autostop);
     }
      catch(Exception e)
      {
        TextOutput.TextOut("Caught Exception in VRMLInterface.StartRotation: " + e);
      }
      return false;
   }

/**
* Stoppt jegliche Bewegung des übergebenen EventOuts.
* @param name Der Name des anzuhaltenen EventOut
* @return False im Fehlerfall
*/
   public boolean StopVector(String name)
   {

     try {
        VRMLEventOut e =  FindEventOut(name);
        if(e!=null) e.SetInc((LinkedList)null);
     }
```

```
      catch(Exception e)
      {
        TextOutput.TextOut("Caught Exception in VRMLInterface.Stop: " + e);
        return false;
      }
      return true;
   }

/**
* Fügt einen EventOut der Schnittstellenverwaltung hinzu.
* @param name Der Name des EventOut, gefolgt von einer Initialisierung (optional)
* @return False im Fehlerfall
*/
   public synchronized boolean AddEventOut(String name)
   {
     try {
        if(m_EventOutList==null) { TextOutput.TextOut("Interner Fehler "); return false;}
        m_EventOutList.reset();
        Field f=null;
        try {
         f=getEventOut(name);
        }
        catch(Exception e)
        {
            TextOutput.TextOut("Der Vektor "+name+" kann nicht gefunden werden");
        }
        if(f!=null) {
           VRMLEventOut e= new VRMLEventOut(name,this);
           if(e!=null)
           m_EventOutList.append(e);
        }
     }
      catch(Exception e)
      {
        TextOutput.TextOut("Caught Exception in VRMLInterface.AddEventOut "+name+": " + e);
        return false;
      }
      return true;
   }

/**
* Fügt einen EventIn der Schnittstellenverwaltung hinzu.
* @param name Der Name des EventIn
* @return False im Fehlerfall
*/
   public synchronized boolean AddEventIn(String name)
   {
     try {
        if(m_EventInList==null) TextOutput.TextOut("Uuuups");
        m_EventInList.reset();
        Field f=null;
        try {
         f=getEventIn(name);
        }
        catch(Exception e)
        {
            TextOutput.TextOut("Der Vektor "+name+" kann nicht gefunden werden");
        }
        if(f!=null) {
           VRMLEventIn e= new VRMLEventIn(name,this);
               if(e!=null) m_EventInList.append(e);
        }
     }
      catch(Exception e)
      {
        TextOutput.TextOut("Caught Exception in VRMLInterface.AddEventIn "+name+": " + e);
        return false;
      }
      return true;
   }

/**
* Sucht einen EventOut in der Schnittstellenverwaltung.
* @param name Der Name des EventOut
* @return der gesuchte EventOut
*/
   public VRMLEventOut FindEventOut(String name)
   {
      try {
```

```
         name=name.trim();
         LinkedList li = (LinkedList) GetEventOutList().clone();
         li.reset();
         while(li.hasMoreElements()) {
            VRMLEventOut e = (VRMLEventOut) li.currentElement();
            if(e.GetName().equals(name)) {
               return e;
            }
            li.nextElement();
         }
      }
      catch(Exception e)
      {
        TextOutput.TextOut("Caught Exception in VRMLInterface.FindEventOut: " + e);
      }
      return null;
   }

/**
* Gibt alle EventOuts der Schnittstellenverwaltung zurück.
* @return Die Liste aller EventOuts
*/
   public LinkedList GetEventOutList()
   {
      return m_EventOutList;
   }

/**
* Gibt alle EventIns der Schnittstellenverwaltung zurück.
* @return Die Liste aller EventIns
*/
   public LinkedList GetEventInList()
   {
      return m_EventInList;
   }

/**
* Setzt den Wert eines Rotations-EventOut (ohne Animation).
* @param name Der Name des EventOuts (vom Typ SFRotation)
* @param axisX x-Komponente der Drehrichtung
* @param axisY y-Komponente der Drehrichtung
* @param axisZ z-Komponente der Drehrichtung
* @param rotation Drehwinkel
* @return False im Fehlerfall
*/
   public boolean SetRotation(String name,float axisX, float axisY, float axisZ, float rotation)
   {
     try {
        VRMLEventOut ev =  FindEventOut(name);
        SetRotation(ev,axisX,axisY,axisZ,rotation);
     }
     catch(Exception e)
     {
       TextOutput.TextOut("Caught Exception in VRMLInterface.SetRotation: " + e);
       return false;
     }
     return true;
   }

   /**
* Setzt den Wert eines Rotations-EventOut (ohne Animation).
* @param ev Der EventOut (vom Typ SFRotation)
* @param axisX x-Komponente der Drehrichtung
* @param axisY y-Komponente der Drehrichtung
* @param axisZ z-Komponente der Drehrichtung
* @param rotation Drehwinkel
* @return False im Fehlerfall
*/
public boolean SetRotation(VRMLEventOut ev,float axisX, float axisY, float axisZ, float rotati-
      on)
   {
     try {
        SFRotation f = (SFRotation)ev.GetField();
        ev.SetAktField(new SFRotation(axisX,axisY,axisZ,rotation));
        f.setValue(axisX,axisY,axisZ,rotation);
     }
     catch(Exception e)
     {
       TextOutput.TextOut("Caught Exception in VRMLInterface.SetRotation: " + e);
```

```
      return false;
    }
    return true;
  }

/**
* Setzt den Wert eines Translations-EventOut (ohne Animation).
* @param name Der Name des EventOuts (vom Typ SFVec3f)
* @param x x-Komponente
* @param y y-Komponente
* @param z z-Komponente
* @return False im Fehlerfall
*/
  public boolean SetVec3(String name,float x, float y, float z)
  {
    try {
       VRMLEventOut ev = FindEventOut(name);
       SetVec3(ev,x,y,z);
    }
    catch(Exception e)
    {
      TextOutput.TextOut("Caught Exception in VRMLInterface.SetVec3: " + e);
      return false;
    }
    return true;
  }

/**
* Setzt den Wert eines Translations-EventOut (ohne Animation).
* @param ev Der EventOut (vom Typ SFVec3f)
* @param x x-Komponente
* @param y y-Komponente
* @param z z-Komponente
* @return False im Fehlerfall
*/
  public boolean SetVec3(VRMLEventOut ev,float x, float y, float z)
  {
    try {
       SFVec3f f = (SFVec3f)ev.GetField();
       ev.SetAktField(new SFVec3f(x,y,z));
       f.setValue(x,y,z);
    }
    catch(Exception e)
    {
      TextOutput.TextOut("Caught Exception in VRMLInterface.SetVec3: " + e);
      return false;
    }
    return true;
  }

/**
* Setzt den Wert EventOut vom Typ SFBool.
* @param name Der Name des EventOuts (vom Typ SFBool)
* @param b Der neue Wert des EventOut
* @return False im Fehlerfall
*/
  public boolean SetBool(String name,boolean b)
  {
    try {
       VRMLEventOut ev = FindEventOut(name);
       SetBool(ev,b);
    }
    catch(Exception e)
    {
      TextOutput.TextOut("Caught Exception in VRMLInterface.SetBool: " + e);
      return false;
    }
    return true;
  }

/**
* Setzt den Wert EventOut vom Typ SFBool.
* @param ev Der EventOut (vom Typ SFBool)
* @param b Der neue Wert des EventOut
* @return False im Fehlerfall
*/
  public boolean SetBool(VRMLEventOut ev,boolean b)
  {
    try {
```

```
         SFBool f = (SFBool)ev.GetField();
         ev.SetAktField(new SFBool(b));
         f.setValue(b);
      }
      catch(Exception e)
      {
        TextOutput.TextOut("Caught Exception in VRMLInterface.SetBool: " + e);
        return false;
      }
      return true;
    }

/**
* Setzt den Wert EventOut vom Typ SFString.
* @param ev Der EventOut (vom Typ SFString)
* @param s Der neue Wert des EventOut
* @return False im Fehlerfall
*/
    public boolean SetString(VRMLEventOut ev,String s)
    {
      try {
         SFString f = (SFString)ev.GetField();
         ev.SetAktField(new SFString(s));
         f.setValue(s);
      }
      catch(Exception e)
      {
        TextOutput.TextOut("Caught Exception in VRMLInterface.SetString: " + e);
        return false;
      }
      return true;
    }

/**
* Setzt den Wert EventOut vom Typ SFString.
* @param name Der Name des EventOuts (vom Typ SFString)
* @param s Der neue Wert des EventOut
* @return False im Fehlerfall
*/
    public boolean SetString(String name,String s)
    {
      try {
         VRMLEventOut ev = FindEventOut(name);
         SetString(ev,s);
      }
      catch(Exception e)
      {
        TextOutput.TextOut("Caught Exception in VRMLInterface.SetString: " + e);
        return false;
      }
      return true;
    }

/** Gibt eine Referenz auf das Basis-VRML-Objekt zurück.
* @return Referenz auf das Basis-VRML-Objekt
*/
    public SFNode GetMainObjNode()
      {
    return RobotObjNode;
      }
}
```

2.5 Die Klasse VRMLEventIn

```
/*****************************************************************
* @(#)VRMLEventIn.java 1.0 97/06/23 Hauke Ernst
*/
package VRMLInterface;

import vrml.node.Script;
import vrml.node.Node;
import vrml.Field;
import vrml.field.SFNode;
import vrml.field.SFString;
import vrml.field.SFVec3f;
```

```
import vrml.field.SFRotation;
import vrml.field.SFBool;
import vrml.field.SFTime;
import vs.Vscp;
import vs.Transform;

import java.awt.*;
import java.util.*;
import java.io.*;
import java.lang.*;

import VRMLInterface.*;

/**
* Klasse zur Repräsentation eines EventIns für VRML2-Welten.
* Insbesondere wird die Abfrage aktueller Werte an die
* Funktionen von VRMLEvent angepasst.
*
* @author  Hauke Ernst
* @version 1.0
*/
public class VRMLEventIn
   implements VRMLEvent
{
   VRMLController m_Interface;
   String m_Name;
   Field m_Field;

/** Konstruktor.
* @param name Name des EventIn
* @param interf Referenz auf das VRMLInterface
*/
   public VRMLEventIn(String name, VRMLController interf)
   {
      try {
         m_Name=name;
         m_Interface=interf;
         m_Field=GetField();
      }
      catch(Exception e)
      {
        TextOutput.TextOut("Caught Exception in VRMLEventIn.constructor "+name+": " + e);
      }
   }

/** Gibt den Namen des EventIn zurück.
* @param der Name des EventIn.
*/
   public String GetName()
   {
      return m_Name;
   }

/** Gibt den aktuellen Wert des EventIn zurück.
* @return der aktuelle Wert des EventIn.
*/
   public Field GetField()
   {
      if(m_Interface!=null) m_Field=m_Interface.getEventInField(m_Name);
      return m_Field;
   }

/** Gibt den aktuellen Wert des EventIn zurück.
* @return der aktuelle Wert des EventIn.
*/
   public Field GetAktField()
   {
      return GetField();
   }
}
```

2.6 Die Klasse VRMLEventOut

```
/*****************************************************************
* @(#)VRMLEventOut.java1.0 97/06/23 Hauke Ernst
```

```
*/
package VRMLInterface;


import vrml.node.Script;
import vrml.node.Node;
import vrml.Field;
import vrml.field.SFNode;
import vrml.field.SFString;
import vrml.field.SFVec3f;
import vrml.field.SFRotation;
import vrml.field.SFBool;
import vrml.field.SFTime;
import vs.Vscp;
import vs.Transform;

import java.awt.*;
import java.util.*;
import java.io.*;
import java.lang.*;

import VRMLInterface.*;

/**
* Klasse zur Repräsentation eines EventOuts für VRML2-Welten.
* Insbesondere werden die Parameter der aktuellen Bewegung
* (Translation oder Rotation) verwaltet.
*
* @author  Hauke Ernst
* @version 1.0
*/
public class VRMLEventOut
   implements VRMLEvent
{
   String m_Name;
   Field m_Field;
   Field m_AktField;
   Field m_Inc;
   int m_Ticks;
   int m_AktTick;
   LinkedList m_IncList;
   LinkedList m_RefList;
   VRMLController m_Interface;

/** Konstruktor, der die Initialisierung des Objekts vornimmt.
* @param name Name des EventOuts
* @param interf Referenz auf das VRMLInterface
*/
   public VRMLEventOut(String name , VRMLController interf)
   {
      try {
         m_Name=name;
         m_Inc=null;
         m_Interface=interf;
       if(m_Interface!=null) m_Field=m_Interface.getEventOutField(m_Name);
          SetAktField(m_Field);
         try{
            // Falls ein EventIn mit Namen "set_<eventoutname>" existiert, kann der
            // aktuelle Wert aus der Welt genommen werden. Sonst wird der Wert
            // fuer den EventOut nicht initialisiert werden.
            if(m_Interface!=null) {
            Field f =m_Interface.getVRMLField(m_Name);
               if(f !=null && (f instanceof SFVec3f)) m_AktField=f;
              }
         }
         catch(Exception e) {System.out.println("Exception in GetAktField:"+e);}
         m_IncList=null;
         m_RefList=null;
      }
      catch(Exception e)
      {
        TextOutput.TextOut("Caught Exception in VRMLEventOut.constructor "+name+": " + e);
      }
   }

/** Gibt den Namen des EventOuts zurück.
* @return Der Name des EventOuts.
*/
   public  String GetName()
```

```
   {
      return m_Name;
   }

/** Setzt das Field-Objekt, mit dessen Hilfe die
* Steuerung des EventOuts in der VRML2-Welt möglich ist.
* @param i Das Field-Objekt (nur Schreib-Zugriff).
*/
   public  void SetField(Field i)
   {
      m_Field=i;
   }

/** Gibt das Field-Objekt, mit dessen Hilfe die
* Steuerung des EventOuts in der VRML2-Welt möglich ist, zurück.
* @return Das Field-Objekt (nur Schreib-Zugriff).
*/
   public  Field GetField()
   {
      return m_Field;
   }

/** Setzt das Field-Objekt, das den aktuellen Zustand des EventOuts
* enthält.
* @param i Das Field-Objekt (nur Lese-Zugriff).
*/
   public void SetAktField(Field i)
   {
      m_AktField=i;
   }

/** Gibt das Field-Objekt, das den aktuellen Zustand des EventOuts
* enthält, zurück.
* @return Das Field-Objekt (nur Lese-Zugriff).
*/
   public Field GetAktField()
   {
      return m_AktField;
   }

/** Setzt den Zähler, der den Fortschritt der gerade durchgeführten
* Bewegung des EventOuts enthält.
* @param i Der neue Zählerwert
*/
   public void SetAktTick(int i)
   {
      m_AktTick=i;
   }

/** Gibt den Zählerwert zurück, der den Fortschritt der gerade durchgeführten
* Bewegung des EventOuts enthält.
* @return Der aktuelle Zählerwert
*/
   public int GetAktTick()
   {
      return m_AktTick;
   }

/** Setzt die Anzahl von Zeitscheiben, die für die Animation
* verbraucht werden sollen.
* @param i Anzahl der Zeitscheiben für die Animation
*/
   public void SetTicks(int i)
   {
      m_Ticks=i;
   }

/** Gibt die Anzahl von Zeitscheiben zurück, die für die Animation
* verbraucht werden sollen.
* @return Anzahl der Zeitscheiben für die Animation
*/
   public int GetTicks()
   {
      return m_Ticks;
   }

/** Setzt die Liste der Stützstellen für die Animation.
* Diese Liste enthält feste Zustände (als Field-Objekte)
* innerhalb der Animation.
```

```
* Die Einbeziehung von Stützstellen dient der Vermeidung
* von Fehlern, die beim ständigen Inkrementieren entstehen
* würden.
* @param li Die Liste der Stützstellen als Liste von Field-Objekten.
*/
   public synchronized void SetRefList(LinkedList li)
   {
      if(li!=null) {
         li.reset();
      }
      m_RefList=li;
   }

/** Gibt die Liste der Stützstellen für die Animation zurück.
* @see #SetRefList
* @return Die Liste der Stützstellen als Liste von Field-Objekten.
*/
   public LinkedList GetRefList()
   {
      return m_RefList;
   }

/** Gibt die gerade aktuelle Stützstelle für die Animation zurück.
* @see #SetRefList
* @return Die Stützstelle als Field-Objekt.
*/
   public Field GetRefField()
   {
      LinkedList l = m_RefList;
      if(l==null) return null;
      return (Field) l.currentElement();
   }

/** Setzt Bewegungs-Inkremente.
* Jeder Eintrag der übergebenen Liste stellt das Inkrement
* zwischen zwei Stützstellen dar.
* @param li Liste von Bewegungs-Inkrementen als Field-Objekte.
* @see #SetInc(Field)
*/
   public synchronized void SetInc(LinkedList li)
   {
      m_Inc=null;
      if(li!=null && li.hasMoreElements()) {
         li.reset();
         m_Inc=(Field) li.currentElement();
      }
      m_IncList=li;
   }

/** Setzt ein Bewegungs-Inkrement.
* Diese Funktion kann für den Spezialfall verwendet werden,
* wenn nur eine Stützstelle vorhanden ist.
* @param i Das Bewegungs-Inkrement als Field-Objekt.
* @see #SetInc(LinkedList)
*/
   public synchronized void SetInc(Field i)
   {
      m_IncList=new LinkedList();
      m_IncList.reset();
      m_Inc=i;
   }

/** Gibt das gerade aktive Bewegungs-Inkrement zurück.
* @return Das gerade aktive Bewegungs-Inkrement.
*/
   public synchronized Field GetInc()
   {
      if(m_IncList!=null&&m_IncList.hasMoreElements()) m_Inc=(Field) m_IncList.currentElement();
      return m_Inc;
   }

/** Inkrementiert den Animations-Zähler.
* Wenn dieser seinen Maximalwert erreicht hat,
* wird die nächste Stützstelle und
* das nächste Bewegungs-Inkrement
* aktiviert.
* @see #SetTicks
* @see #SetRefList
* @see #SetInc(LinkedList)
```

```
*/
   public synchronized void IncTick()
   {
      m_AktTick++;
      if(m_IncList!=null)
         if(m_AktTick>Math.abs(m_Ticks))
         {
           if(m_IncList.hasMoreElements()) {
              m_IncList.nextElement();
              m_RefList.nextElement();
              if(m_IncList.hasMoreElements())  {
               m_AktTick=0;
           }
           else if( m_Ticks < 0) {
              m_IncList.reset();
              m_RefList.reset();
              if(m_IncList.hasMoreElements())  {
               m_AktTick=0;
              }
           }
         }
      }
   }
}
```

2.7 Die Klasse TextOutput

```
/*********************************************************************
* @(#)TextOutput.java1.0 97/06/23 Hauke Ernst
*/
package VRMLInterface;

import java.awt.*;
import VRMLInterface.ResultProcessor;

/** Dialogklasse zur Ausgabe eines einzeiligen Textes.
* Die Schaltfläche "Ok" schließt das Fenster.
*
*
* @author  Hauke Ernst
* @version 1.0
*/
public class TextOutput extends Frame
{
  private String m_String;
  private ResultProcessor m_ResultProcessor;
  private Font m_Font;
  private FontMetrics m_FontMetrics;
  private boolean m_FontSet=false;
  public TextOutput(ResultProcessor creator, String n)
   {
      Panel p = new Panel();
      p.add(new Button("Ok"));
      add("South",p);
      m_String=n;
      m_ResultProcessor=creator;
      m_Font = new Font("Helvetica", Font.BOLD, 14);
      setFont(m_Font);
   }
   public  boolean action(Event evt, Object arg)
   {
      if(arg.equals("Ok"))
      {
         dispose();
         if(m_ResultProcessor!=null)
         ((ResultProcessor)m_ResultProcessor).processResult(this,
             m_String);
         return true;
      }
      return false;
   }

   public boolean handleEvent(Event evt)
   {
      return super.handleEvent(evt);
```

```
   }
      public void setFonts(Graphics g)
     {  if (m_FontSet) return;
      m_FontMetrics = g.getFontMetrics(m_Font);
      m_FontSet = true;
   }

   public void paint(Graphics g)
   {  setFonts(g);
      int w = m_FontMetrics.stringWidth(m_String);
      Dimension d = size();
      Insets in = insets();
      int client_width = d.width - in.right - in.left;
      int client_height = d.height - in.bottom - in.top;
      int cx = (client_width - w) / 2;
      int cy = client_height / 2;
      g.drawRect(0, 0, client_width-1, client_height-1);
      g.setFont(m_Font);
      g.drawString(m_String, cx, cy);
   }

/** Gibt einen Text in einem Fenster der Klasse TextOutput aus.
* @param n Der auszugebende Text.
*/
   static public void TextOut(String n)
   {
      TextOutput f = new TextOutput(null,n);
      f.resize(300, 200);
      f.requestFocus();
      f.show();
      f.requestFocus();
   }
}
```

2.8 Die Klasse TextInput

```
/************************************************************************
* @(#)TextInput.java1.0 97/06/23 Hauke Ernst
*/
package VRMLInterface;

import java.awt.*;
import VRMLInterface.ResultProcessor;

/** Dialogklasse, die ein Texteingabefeld und
* eine Schaltfläche zum Schließen des Fensters
* enthält.
*
*
*
* @author  Hauke Ernst
* @version 1.0
*/
public class TextInput extends Frame
{
private TextArea ta;
  private ResultProcessor m_ResultProcessor;

  public TextInput(ResultProcessor creator)
   {
      Font myfont = new Font("Helvetica", Font.BOLD, 14);
      setFont(myfont);
      setTitle("Texteingabe");
      Panel p = new Panel();
      p.setLayout(new FlowLayout());
      p.add(new Button("Schliessen"));
      add("South", p);
      ta = new TextArea(8, 40);
      add("Center", ta);
      m_ResultProcessor=creator;
   }

   public boolean handleEvent(Event evt)
   {
      return super.handleEvent(evt);
```

```
   }

   public boolean action(Event evt, Object arg)
   {
      if (arg.equals("Schliessen"))
      {
         dispose();
         ((ResultProcessor)m_ResultProcessor).processResult(this,
            getText());
         return true;
      }
      else return super.action(evt, arg);
   }

   public String getText()
   {
      return ta.getText();
   }
}
```

2.9 Die Klasse LinkedList

```
/**********************************************************************
* @(#)LinkedList.java1.0 97/06/23 Hauke Ernst
*/
package VRMLInterface;

/** Listen-Klasse.
*
*
* @author  Hauke Ernst
* @version 1.0
*/
public class LinkedList implements Cloneable
{
   private Link head;
   private Link tail;
   private Link pre;
   private int len;

   public synchronized Object clone()
   {
      try { return super.clone();
      }
      catch (CloneNotSupportedException e) {
         return null;
      }
   }

   public synchronized void reset()

   {  pre = null;
   }

    public synchronized boolean hasMoreElements()

   {  return cursor() != null;
   }

   public synchronized String toString()
   {
      String s = new String("(");
      reset();
      while(hasMoreElements()) {
         s+= currentElement().toString()+",";
         nextElement();
      }
      return s+")";
   }

   public synchronized Object nextElement()
   {  if (pre == null) pre = head; else pre = pre.next;
      if (pre == null)
         throw new java.util.NoSuchElementException();
      return pre.data;
```

```
   }

   public synchronized Object currentElement()
   {  Link cur = cursor();
      if (cur == null)
         throw new java.util.NoSuchElementException();
      return cur.data;
   }

   public synchronized void insert(Object n)
   {  Link p = new Link(n, cursor());

      if (pre != null)
      {  pre.next = p;
         if (pre == tail) tail = p;
      }
      else
      {  if (head == null) tail = p;
         head = p;
      }

      pre = p;
      len++;
   };

   public synchronized void append(Object n)

   {  Link p = new Link(n, null);
      if (head == null) head = tail = p;
      else
      {  tail.next = p;
         tail = p;
      }
      len++;
   }

   public synchronized Object remove()
   {  Link cur = cursor();
      if (cur == null)
         throw new java.util.NoSuchElementException();
      if (tail == cur) tail = pre;
      if (pre != null)
         pre.next = cur.next;
      else
         head = cur.next;
      len--;
      return cur.data;
   }

    public synchronized int size()
   {  return len;
   }

    public synchronized java.util.Enumeration elements()
   {  return new ListEnumeration(head);
   }

   private synchronized Link cursor()
   {  if (pre == null) return head; else return pre.next;
   }
}

class Link
{  Object data;
   Link next;
   Link(Object d, Link n) { data = d; next = n; }
}

class ListEnumeration implements java.util.Enumeration
{
  private Link cursor;

  public ListEnumeration( Link l)
   {  cursor = l;
   }

    public synchronized boolean hasMoreElements()
   {  return cursor != null;
   }
```

```
    public synchronized Object nextElement()
   {  if (cursor == null)
      throw new java.util.NoSuchElementException();
      Object r = cursor.data;
      cursor = cursor.next;
      return r;
   }
}
```

2.10 Die Schnittstelle ResultProcessor

```
/*********************************************************************
* @(#)ResultProcessor.java 1.0 97/06/23 Hauke Ernst
*/
package VRMLInterface;

import java.awt.*;

/** Allen Benutzerdialogen in den Packages VRMLInterface,VRRobot
* und SchemeRobot ist ein ResultProcessor zugeordnet. Dies ist diejenige
* Klasse, die zur Behandlung von Dialog-Events durch die Funktion
* processResult aufgefordert wird.
* Alle Klassen, die Dialog-Events behandeln sollen, müssen daher
* das Interface ResultProcessor implementieren.
* Beispiel: "class SchemeRobot extents vrrobot implements ResultProcessor {...}"
* Der ResultProcessor wird den Dialogklassen jeweils im Kontruktor
* bekanntgemacht.
*
*
* @author  Hauke Ernst
* @version 1.0
*/
public interface ResultProcessor
{
   public void processResult(Frame source, Object obj);
}
```

2.11 Die Schnittstelle VRMLController

```
/*********************************************************************
* @(#)VRMLController.java 1.0 97/06/23 Hauke Ernst
*/
package VRMLInterface;

import vrml.Field;

/** Das Interface VRMLController umfaßt Klassen,
* die den öffentlichen Zugriff auf VRML2-EventIns und
* -EventOuts zur Verfügung stellen.
*
*
* @author  Hauke Ernst
* @version 1.0
*/
public interface VRMLController
{
/**
* Ordnet dem übergebenen Namen ein EventOut-Field zu.
* @param name Der gesuchte EventOut
* @return Das EventOut-Field
*/
   public Field getEventOutField(String name);

/**
* Ordnet dem übergebenen Namen ein EventIn-Field zu.
* @param name Der gesuchte EventIn
* @return Das EventIn-Field
*/
   public Field getEventInField(String name);
```

```
/**
* Ordnet dem übergebenen Namen ein Field zu.
* @param name Das gesuchte Field
* @return Das Field
*/
   public Field getVRMLField(String name);
}
```

2.12 Die Schnittstelle VRMLEvent

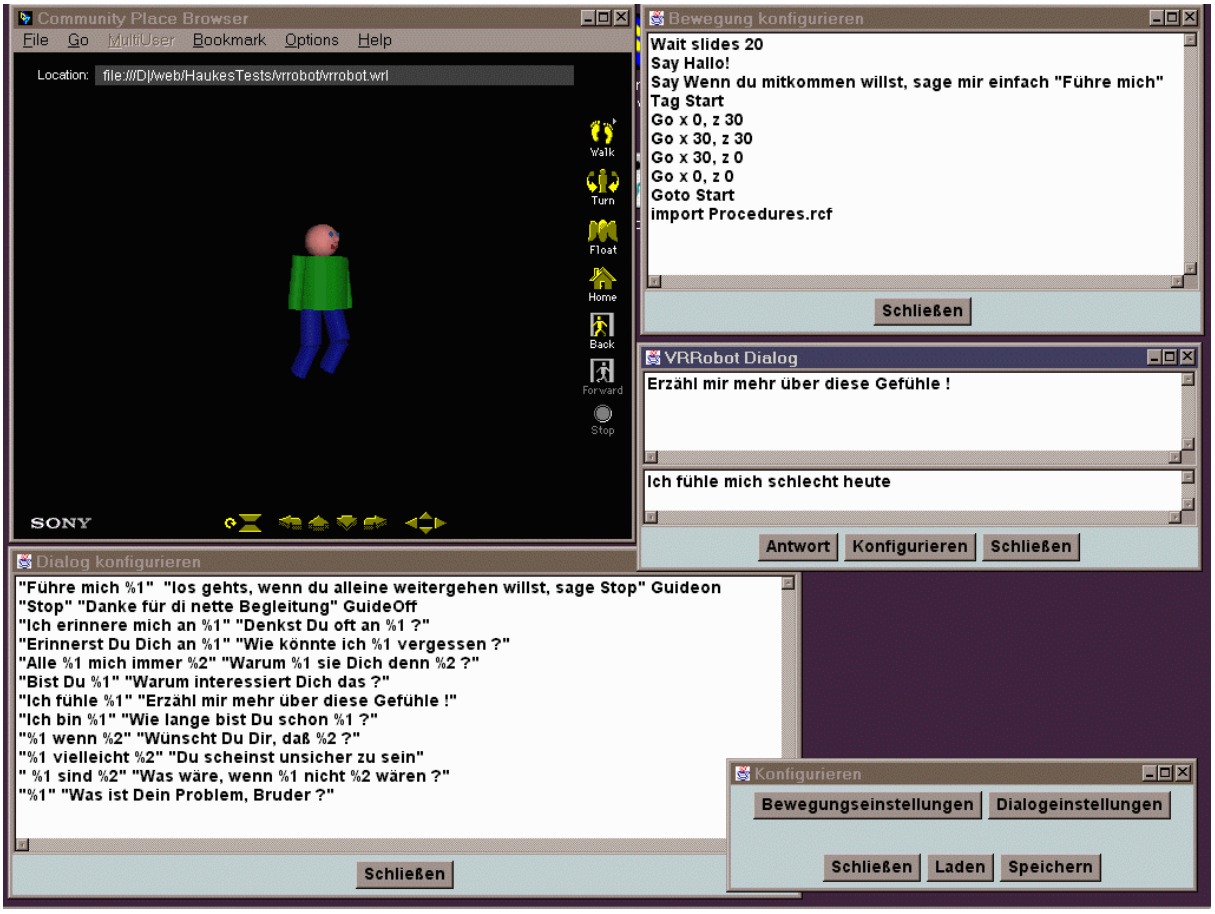
```
/*****************************************************************
* @(#)VRMLEvent.java1.0 97/06/23 Hauke Ernst
*/
package VRMLInterface;

import vrml.Field;

/** Das Interface VRMLEvent erlaubt einen gleichartige
* Zugriff auf VRMLEventIns und VRMLEventOuts.
* @see VRMLEventIn
* @see VRMLEventOut
*
*
* @author  Hauke Ernst
* @version 1.0
*/
public interface VRMLEvent
{
   public String GetName();
   public Field GetField();
   public Field GetAktField();
}
```

3 Das Modul VRRobot

3.1 Aufgabenstellung

Das Modul VRRobot dient der Steuerung der Bewegungen des VRML-Prototypen VRRobot und bietet die Möglichkeit, mit ihm in einen Textdialog zu treten.

Die Hauptaufgabe dieses Moduls besteht insbesondere darin, Basisfunktionalität für komplexere, abgeleitete Klassen zur Verfügung zu stellen.



3.2 Funktionsbeschreibung

Das Modul VRRobot stellt eine Skriptsprache zur Definition von Bewegungsabläufen und Dialogverhalten zur Verfügung. Außerdem werden Dialogfenster implementiert, mit deren Hilfe Skripts zusammengestellt werden können.

Der Befehlsumfang der Skriptsprache für die Bewegungssteuerung umfaßt bisher folgendes:

Rotate vector <name>, axisx <x>, axisy <y>, axisz <z>, direction <Winkel1 Winkel2...>, slides <Zeiteinheiten>, cyclic <Endlos?>, wait <Warten?>

⇒ Der durch „vector" angegebene Rotationsvektor wird angesteuert. Die Drehung wird durch „axisx", „axisy", axisz" und „direction" angegeben, wobei dem Parameter „direction" eine beliebige Folge von Werten zugewiesen werden kann. Die Zeit, die für den Vorgang verbraucht werden soll (und damit auch die Geschwindigkeit), wird durch den Parameter „slides" angegeben. „cyclic" gibt an, ob die Bewegung nach Erreichen des Zielzustandes weiterlaufen oder abgebrochen werden soll. Der Parameter „wait" bestimmt, ob das Skript sofort oder Beendigung der Bewegung weiter ausgeführt wird. „cyclic" und „wait" können die Zustände „true" oder „false" annehmen. Die Parameter können in beliebiger Reihenfolge notiert werden.

Beispiel:

```
Rotate vector arm2rot , axisx 1, axisy 0, axisz 0 ,direction (-40) 40,
       slides 10, wait false,cyclic true
```

Translate vector <name>, x <x>, y <y>, z <z>, slides <Zeiteinheiten>, cyclic <Endlos?>, wait <Warten?>

⇒ Der durch „vector" angegebene Translationsvektor wird angesteuert. Die Zielposition wird durch „x", „y" und „z" angegeben, zu den übrigen Parametern siehe Beschreibung von „Rotate".

Beispiel:

```
Translate vector robotpos , x 20, y 0, z 20 ,slides 10, cyclic false, wait
true
```

EventOut vector <Name>,

[axisx <x>, axisy <y>, axisz <z>, direction <Winkel>]

|[x <x>, y <y>, z <z>] |[value <Wert>]

⇒ Setzt den Wert von „vector". Ist dieser eine Rotation, so müssen gleichzeitig die Parameter „axisx", „axisy", „axisz" und „direction" gesetzt sein. Ist er eine Translation, werden „x", „y" und „z" angegeben. Handelt es sich um einen Vektor vom Typ Boolean oder String, so wird der Parameter „value" verwendet.

Beispiel:

```
EventOut vector guide, value true
```

Wait slides <Zeiteinheiten>

⇒ Der Kontrollfluß verharrt für „slides" Zeiteinheiten.

Beispiel:

```
Wait slides 30
```

LocalVar <Name> <Wert>

⇒ Der lokalen Variablen <Name> wird <Wert> zugewiesen. Nach diesem Befehl, aber nur innerhalb der aktuellen Prozedur, kann die Variable per %<Name> verwendet werden. Der gerade aktive Wert wird an der entsprechenden Stelle des Aufrufes eingesetzt.

Beispiel:

```
LocalVar warten 50
Wait slides %warten
```

GlobalVar <Name> <Wert>

⇒ Wie LokalVar, allerdings kann die Variable auch außerhalb der aktuellen Prozedur verwendet werden.

Beispiel:

```
GlobalVar warten 50
Wait slides %warten
```

Diejenigen Vektoren, die der Steuerung der VRML-Welt dienen (EventOuts und EventIns), sind als globale Variablen vordefiniert, so daß auf diesem Weg deren aktueller Wert abgefragt werden kann. Für jeden Translationsvektor <Transvek> sind die Variablen <Transvek>.x, <Transvek>.y und <Transvek>.z, für jeden Rotationsvektor <Rotvek> die Variablen <Rotvek>.axisx, <Rotvek>.axisy, <Rotvek>.axisz und <Rotvek>.direction definiert. Events vom Typ Boolean oder String stellen Variablen der Form <Eventname>.value zur Verfügung.

Beispiel:

```
Localvar xpos %robotpos.x
Localvar prox %proximitry.value
```

If <Bedingung> then <Befehl1> else <Befehl2>

⇒ Kann der boolsche Ausdruck <Bedingung> als „true“ ausgewertet werden, wird <Befehl1> ausgeführt, andernfalls <Befehl2>.

Beispiel:

```
If (%zposdif=0)&(%xposdif>=0) then Localvar angle 90
```

Say <Text>

⇒ <Text> wird im Dialogfenster ausgegeben. Dabei wird der Name des VRRobot dem auszugebenen Text vorangestellt.

Beispiel:

```
Say Wie geht’s ?
```

TextOut <Text>

⇒ <Text> wird im Dialogfenster ausgegeben.

Beispiel:

```
TextOut Wie geht’s ?
```

Tag <Bezeichner>

⇒ Ein Bezeichner, zu dem mit einem entsprechenden „Goto"-Befehl gesprungen werden kann, wird definiert.

Beispiel: siehe „Goto"

Goto <Bezeichner>

⇒ Das Skript wird von vorne nach hinten nach „Tag <Bezeichner>" durchsucht. Von der gefundenen Einsprungstelle aus wird die Ausführung des Skriptes fortgeführt.

Beispiel:

```
Tag Start
.....
Goto Start
```

Procedure <Name> <Parameterliste>
begin
<Befehle>
end

⇒ Die Prozedur <Name> wird als parametrisierbares Unterprogramm eingeführt. Dabei besteht die Parameterliste aus einer durch Kommata getrennte Folge von Parametern, die jeweils wieder aus Name und Standardwert bestehen. Bei jedem Aufruf der Prozedur mit konkreten Parametern wird die Befehlsfolge <Befehle> abgearbeitet und danach die aufrufende Prozedur weiterverarbeitet.

Beispiel:

```
procedure Gehehinundher weg 30
begin
Localvar x %robotpos.x
Localvar z %robotpos.z
TurntoPlace xpos %x, zpos %z+%weg
GotoPlace   xpos %x, zpos %z+%weg
TurntoPlace xpos %x, zpos %z
GotoPlace   xpos %x, zpos %z
end
```

Besondere Prozeduren sind die sog. Callbacks/Eventhandler, die automatisch beim Auftreten eines bestimmten VRML-Ereignisses (EventIns) aufgerufen werden. Ist zu einem EventIn eine gleichnamige Prozedur definiert, so wird diese sofort abgearbeitet. Prozeduren können auch dynamisch zur Laufzeit erzeugt werden:

```
procedure MyRuntimeProc\nbegin\nSay Es klappt!\nend
```

Import <Dateiname>

⇒ Der Inhalt der Datei <Dateiname> wird an dieser Stelle des Skriptes eingesetzt.

Beispiel:

```
Import Procedures.rcf
```

Kommentare /* <Kommentar> */ oder // <Kommentar>

⇒ Der von /* und */ eingeschlossene Text bzw. der Text hinter // bis zum nächsten Zeilenende wird als Kommentar interpretiert und ansonsten nicht ausgewertet.

Beispiel:

```
KeinKommentar
/*
Kommentar
*/
KeinKommentar // Kommentar
```

Dialogverhalten

Im Skript zur Definition des Dialogverhaltens können Einträge der folgenden Form gemacht werden:

„<Eingabestring>“ „<Ausgabestring>“ <Befehl>

Der Interpreter versucht bei jeder Texteingabe des Benutzers, eine passende Antwort anhand der Skript-Einträge zu finden. Dabei durchsucht er das Skript von vorne nach hinten und vergleicht jeweils die Benutzereingabe mit <Eingabestring> des Eintrages. Der Eingabestring darf Wildcards der Form %<Nummer> (also zum Beispiel %1, %2 usw.) enthalten. Die für Wildcards gefundenen Einsetzungen können in <Ausgabestring> referenziert werden. <Befehl> wird als Prozeduraufruf ausgewertet und kann ebenfalls die Wildcards verwenden.

Passen zu einer Texteingabe des Benutzers mehrere Muster, so wird zufällig, aber unter Berücksichtigung einer bestimmten Gewichtung, einer der in Frage kommenden Einträge ausgewählt. Durch diese Möglichkeit redundanter Dialogdefinitionen wird das Antwortverhalten unvorhersehbar und damit interessanter.

Beispiele:

```
"Ich erinnere mich an %1" "Denkst Du oft an %1 ?"
"Erinnerst Du Dich an %1" "Wie könnte ich %1 vergessen ?" Grin
"Alle %1 mich immer %2" "Warum %1 sie Dich denn %2 ?"
"Bist Du %1" "Warum interessiert Dich das ?"
"Ich fühle %1" "Erzähl mir mehr über diese Gefühle !"
"Ich bin %1" "Wie lange bist Du schon %1 ?"
"%1 wenn %2" "Wünscht Du Dir, daß %2 ?"
"%1 vielleicht %2" "Du scheinst unsicher zu sein"
"%1 sind %2" "Was wäre, wenn %1 nicht %2 wären ?"
"%1" "Was ist Dein Problem, Bruder ?" Shrug
```

3.3 Modularisierung

Das Modul VRRobot besteht aus mehreren Klassen und Schnittstellen, die im folgenden einzeln beschrieben werden.

3.4 Die Klasse VRRobot

```
/*********************************************************************
* @(#)vrrobot.java1.0 97/06/23 Hauke Ernst
*/
package VRRobot;

import vrml.node.Script;
import vrml.node.Node;
import vrml.Field;
import vrml.field.SFNode;
import vrml.field.SFString;
import vrml.field.ConstSFString;
import vrml.field.SFVec3f;
import vrml.field.ConstSFVec3f;
import vrml.field.SFRotation;
import vrml.field.ConstSFRotation;
import vrml.field.SFBool;
import vrml.field.ConstSFBool;
import vrml.field.SFTime;
import vrml.field.ConstSFTime;
import vs.Vscp;
import vs.Transform;

import java.awt.*;
import java.util.*;
import java.io.*;
import java.lang.*;

import VRMLInterface.*;
import VRRobot.*;
/**
* Klasse zur Steuerung von automatisierten Charakteren.
* Ihre Funktion ist die
* Bereitstellung und Umsetzung einer Skriptsprache zur
* Verhaltens-Festlegung von Charakteren, die VRML2-Welten bewohnen.
* Die Skriptsprache und die Steuerung sind unabhängig von der
* Gestalt des Charakters und dem Aufbau der VR-Welt. In die Steuerung
* können beliebige EventIns und EventOuts einbezogen werden, diese
* müssen lediglich VRML-seitig in den Feldern "EventIns" und "EventOut"
* deklariert werden (siehe VRMLInterface).
* Um die Steuerung zu erleichtern, bietet die Skript-Sprache die
* Möglichkeit zur Hierarchiebildung mit Hilfe von parametriesierbaren
* Prozeduren, die auch zur Laufzeit vom Skript selbst verändert und erzeugt
* werden können (selbstmodifizierend!), zur Verwendung von globalen und lokalen
* Variablen, zur Auswertung mathematischer Ausdrücke und erlaubt Kommentare sowie
* die Einbindung von Prozeduren aus anderen Dateien.
* Wichtig ist, dass in der aufrufenden VRML2-Datei ein TimeSensor
* einen Aufruftakt vorgibt und regelmäßig den EventIn "move" erzeugt.
*
*
* @author  Hauke Ernst
* @version 1.0
* @see     VRMLInterface
*/
public class vrrobot extends VRMLInterface
   implements ResultProcessor, Answering
{
         /** Konstante zur Konvertierung von Winkeln in Bogenmaß */
   double aRad;
         /** Ausrufstack für Prozeduraufrufe */
   Stack m_ProcStack;
         /** Liste aller bekannter Prozeduren */
    LinkedList m_Procedures;
       /** Liste aller globalen Variablen */
   LinkedList m_GlobalVars;
       /** Liste der aktiven Befehlsfolge */
   LinkedList m_ModifierList;
       /** Liste der möglichen Antwort-Zuordnungen */
   LinkedList m_DialogList;
       /** Wird die Maustaste über dem Charakter gedrückt (EventIn "clicked")?*/
   boolean m_LastClickState;
       /** Ist der Betrachter in der Nähe (EventIn "proximitry")? */
   boolean m_LastProxState;
       /** Ist der Charakter vom Benutzer aus sichtbar (EventIn "visibility")? */
   boolean m_LastVisState;
       /** Sind alle Befehle abgearbeitet? */
   boolean m_nomore_MotionCommands;
```

```
      /** Wurde die Funktion "initialize()" schon aufgerufen? */
  boolean m_Initialized=false;
     /** Soll beim naechsten Aufruf von SetMotionCommands die
     *Skriptsprachenprozedur "Initialize" aufgerufen werden? */
boolean m_InitProc=false;
      /** Hauptfenster: Textdialog   */
     MainDialog m_MainDialog;
      /** Konfigurationsfenster für Textdialoge */
  KonfigAnswers m_KonfigAnswers;
      /** Konfigurationsdialog für den Handlungsplan */
  KonfigMotion m_KonfigMotion;
      /** Konfigurationsdialog gesamt */
  KonfigDialog m_KonfigDialog;
      /** Zuletzt vom Benutzer eingegebener Text im Hauptfenster */
  String m_Text;
      /** Vergangene vom Benutzer eingegebene Texte*/
  String m_TalkHist;
      /** aktiven Befehlsfolge als Text */
  String m_MotionCommands;
      /** aktive Dialogeinträge als Text */
  String m_DialogCommands;
      /** Semaphor zum selektiven Verwerfen von "move"-EventIns */
  int m_LockEvents=1;
      /** Semaphor zum selektiven Verwerfen von EventIns */
  int m_LockAllEvents=1;
      /** aktive Prozedur */
  Procedure m_aktProcedure;
      /** Name der Konfigurationsdatei */
  String m_KonfigFile;
      /** Name der Konfigurationsdatei aus VRML*/
  SFString m_KonfigFileNode;
      /** Name des VRRobot */
  String m_Name;
      /** Name des VRRobot aus VRML*/
  SFString m_NameNode;

  /** Standard-Konstruktor */
  public vrrobot() {
     super();
     m_Initialized=false;

  }

/**
* Wird von der Basisklasse Script nach dem Erzeugen der Klasse aufgerufen.
* Aufgaben sind
*    Dateninitialisierung und
*    Einlesen von Konfigurationsdaten aus der Datei, deren Namen in dem Feld
*    "KonfigFile" festgelegt sein sollte. Existiert dieses Feld nicht, wird
*    als Name "Klassenname.rcf" angenommen.
*/
public  synchronized void initialize () {
   try  {
     if(m_Initialized) return;
     m_LockAllEvents=1;
     m_LockEvents=1;
     super.initialize();
     m_Initialized=false;
     int i;
     aRad = (double) (Math.PI/180.0) ;
     m_ProcStack=null;
     m_Procedures=new LinkedList();
     m_GlobalVars=new LinkedList();
     m_aktProcedure=null;
     m_TalkHist=new String();
     try {
        m_NameNode = (SFString)getField("Name");
        m_Name = (String)m_NameNode.getValue();
     }
     catch(Exception e4){m_Name=new String("VRRobot")}

     try {
        m_KonfigFileNode = (SFString)getField("KonfigFile");
        m_KonfigFile = (String)m_KonfigFileNode.getValue();
     }
     catch(Exception e2)
     {
        m_KonfigFileNode=null;
        m_KonfigFile=getClass().getName()+".rcf";
```

```
      }
      m_LastClickState=false;
      m_LastProxState=false;
      m_LastVisState=true;
      m_MainDialog=null;
      m_KonfigAnswers=null;
      m_KonfigDialog=null;
      m_KonfigMotion=null;
      m_Text=new String("");
      m_ModifierList = new LinkedList();
      m_DialogList = new LinkedList();
      m_nomore_MotionCommands=false;
      m_MotionCommands=new String("");
      m_ProcStack=new Stack();
      m_Initialized=true;
      ReadKonfig(m_KonfigFile);
     }
     catch(Exception e3)
     {
       TextOutput.TextOut("Caught Exception in script initialisation: " + e3);
     }
      m_LockAllEvents=0;
      m_LockEvents=0;
   }

   /** Der Aufruf dieser Funktion bewirkt, daß bei nächster Gelegenheit
   * die Skriptsprachenprozedur "Initialize aufgerufen wird.
   * Zu diesem Zweck wird ein Flag gesetzt, was von der Funktion SetMotionCommands
   * ausgewertet wird.
    */
   public void CallInitProc()
     {
      m_InitProc=true;
     }


/**
* Behandlung von VRML-Events.
* "move"-Events werden verworfen, wenn die Bearbeitung des letzten Events noch nicht abgeschlos-
      sen ist.
* Ansonsten werden die spezialisierten Funktionen "clicked", "proximitry", "visibility" und
      "move"
* aufgerufen.
* @param e Der VRML-Event
*/
    public void processEvent(vrml.Event e) {
      try {
         if(m_LockAllEvents>0) return;
         if(!m_Initialized) {
            initialize();
         }
         if(e == null) return;
         if(e.getValue() == null) return;
         String name = e.getName ();
         if(name == null) return;
         m_LockEvents++;
         if(name.equalsIgnoreCase("shutdown")){
            shutdown();
         }
         if(name.equalsIgnoreCase("clicked")){
            clicked(((ConstSFBool)e.getValue()).getValue());
         }
         else if(name.equalsIgnoreCase("proximitry")){
            proximitry(((ConstSFBool)e.getValue()).getValue());
         }
         else if(name.equalsIgnoreCase("visibility")){
            visibility(((ConstSFBool)e.getValue()).getValue());
         }
         else if(name.equalsIgnoreCase("move")){
            if(Vscp.amIMaster()) {
               if(m_LockEvents==1) {
                     super.processEvent(e);
                     move();
               }
             }
         }
         else if(name.equalsIgnoreCase("rpc_move")){
            if(m_LockEvents==1) {
                  super.processEvent(e);
```

```
                move();
            }
         }
         else if(name.equalsIgnoreCase("rpc_textin")){
            Say(Answer(((ConstSFString)e.getValue()).getValue()));
         }
         CallProcedure(name,e);
      }
     catch(Exception ex)
     {
      TextOutput.TextOut("Caught Exception in vrrobot.processEvent: " + ex);
     }
     m_LockEvents--;
   }

 /** Behandlung des EventIn "move".
 * Sorgt für die Abarbeitung von Befehlen und Durchführung von
 * Bewegungen mit Hilfe der Funktion "Apply". Bei Bedarf werden Prozeduren vom Stack geholt. */
   public synchronized void move()
   {
      try {
        if(!Vscp.amIMaster()) return;
           m_nomore_MotionCommands=false;
        if(m_nomore_MotionCommands) return;
        if(m_aktProcedure==null) return;
        if(m_aktProcedure.GetStep() >= m_aktProcedure.GetSteps()) {
          int steps=0;
          boolean applied=false;
          String entry=null;
          while(!applied && !m_nomore_MotionCommands){
             if(m_ModifierList != null && m_ModifierList.hasMoreElements()) {
               entry=(String) (m_ModifierList.currentElement());
               m_ModifierList.nextElement();
               steps = Apply(entry);
               if(steps>0) applied=true;
               if(m_aktProcedure!=null){
                  m_aktProcedure.SetStep(0);
                  m_aktProcedure.SetSteps(steps);
               }
             }
             else {
               if(m_ProcStack==null) return;
               if(m_ProcStack.empty()) {
                  m_nomore_MotionCommands=true;
               }
               else {
                   Procedure poppedproc=(Procedure) m_ProcStack.pop();
                   steps=0;
                   if(poppedproc!=null) {
                     m_aktProcedure=poppedproc;
                     m_ModifierList=(LinkedList) m_aktProcedure.GetCom();
                     steps=m_aktProcedure.GetSteps()-m_aktProcedure.GetStep();
                     if(steps>0) applied=true;
                   }
                }
             }
           }
        }
        if(Tick()) {
           if(m_aktProcedure!=null) m_aktProcedure.IncStep();
        }
      }
      catch(Exception e)
      {
         TextOutput.TextOut("Caught Exception in vrrobot.move: " + e);
      }
   }

/** Interpretation und Ausführung eines Befehls.
* Vorher werden Variablen durch ihre aktuellen Werte ersetzt.
* Außerdem werden alle Vorkommen von "\n" durch Zeilenumbrüche ersetzt,
* falls diese nicht von eckigen Klammern ([...]) eingeschlossen sind.
* Besteht der Befehl dann aus mehreren Zeilen, wird für die Ausführung
* dieser Befehlsfolge eine temporäre Prozedur eingerichtet.
* <pre>
* Beispiel:
*  LocalVar commands [Say Das Herstellen neuer Prozeduren...\nSay ... zur Laufzeit]
*  LocalVar commands [%commands\nSay ...FUNKTIONIERT!]
*  LocalVar proc [procedure newproc\nbegin\n %commands\nend]
```

```
*  %proc
*  newproc
* </pre>
* @param s Der auszuführende Befehl
* @return Gibt zurück, wieviele Ticks der Befehl zur Ausführung braucht.
* Nach Ablauf dieser Ticks (EventIn "move") kann der nächste Befehl ausgeführt
* werden.
*/
   public int Apply(String s)
   {
      if(s==null) return 0;
      s=s.trim();
      if(s.length()<=0) return 0;
      int ret=0;
       Procedure aktProcedure = m_aktProcedure;
      try {
         s = AssignEventOutVals(s);
         s = AssignEventInVals(s);
         if(aktProcedure!=null)
            s = aktProcedure.AssignLocalVars(s);
         s = Procedure.AssignVars(s,m_GlobalVars);
         s = AssignHistory(s);
         s = Expression.EvalStringExpression(s);

         //wenn mehrere Zeilen, dann
         //   temporäre Prozedur erzeugen
         if(Expression.Contains(s,'\n'))
         {
            DefProcedure("Temp","",s);
            return CallProcedure("Temp","");
         }
         StringTokenizer t = new StringTokenizer(s," ");
         String Param;
         String Command=new String();
         Param=new String();
         if(t.hasMoreElements()) Command = t.nextToken();
         while(t.hasMoreElements())
            Param += t.nextToken() + " ";
         Param = Param.trim();
         if(Command.equalsIgnoreCase("Localvar")) {
            if(aktProcedure!=null)
               aktProcedure.Assign(new Variable(Param));
            ret=0;
         }
         else if(Command.equalsIgnoreCase("Globalvar")) {
            Procedure.Assign(new Variable(Param),m_GlobalVars);
            ret=0;
         }
         else if(Command.equalsIgnoreCase("Rotate")) ret=Rotate(Param);
         else if(Command.equalsIgnoreCase("Wait")) ret=Wait(Param);
         else if(Command.equalsIgnoreCase("Translate")) ret=Translate(Param);
         else if(Command.equalsIgnoreCase("Stop")) ret=Stop(Param);
         else if(Command.equalsIgnoreCase("EventOut")) ret=EventOut(Param);
         else if(Command.equalsIgnoreCase("If")) ret=If_Statement(Param);
         else if(Command.equalsIgnoreCase("Tag")) ret=0;
         else if(Command.equalsIgnoreCase("Goto")) ret=Goto(Param);
         else if(Command.equalsIgnoreCase("Procedure")) ret=DefRuntimeProcedure(Param);
         else if(Command.equalsIgnoreCase("Answer")) {
            ret=0;
            Say(Answer(Param));
         }
         else if(Command.equalsIgnoreCase("Say")) {
            SendToSharedObj("rpc_textin",Param);
            Say(Param);
            ret=0;
         }
         else if(Command.equalsIgnoreCase("TextOut")) {
            TextOut(Param);
            ret=0;
         }
         else ret = CallProcedure(Command,Param);
      }
      catch(Exception e)
      {
         TextOutput.TextOut("Caught Exception in vrrobot.Apply: " + e);
      }
      return ret;
   }
```

```
/** Ausführung des Befehls "Rotate" (Rotationsbewegung).
* Eine Rotationsbewegung für den gegebenen Rotationsvektor wird erzeugt.
* <pre>
* Skript-Parameter:
*   vector    (String) -> Name des Rotationsvektors
*   axisx     (Float)  -> x-Richtung des Vektors, um den gedreht werden soll
*   axisy     (Float)  -> y-Richtung des Vektors, um den gedreht werden soll
*   axisz     (Float)  -> z-Richtung des Vektors, um den gedreht werden soll
*   direction ([Float])-> Eine durch Leerzeichen getrennte Liste von Richtungen, in die gedreht
      werden soll
*   slides    (Float)  -> Anzahl der für die Bewegung benötigten Ticks
*   cyclic    (Bool)   -> Soll sich die Bewegung endlos (bis zur explizieten Beendigung) wieder-
      holen?
*   wait      (Bool)   -> Soll der Kontrollfluß bis zum Ende der Bewegung warten?
* Beispiel:
*   Rotate vector arm1rot , axisx 1, axisy 0, axisz 0 ,direction 40 (-40), slides 10, wait fal-
      se,cyclic true
* </pre>
* @param param Die Parameterliste (Komma als Trennzeichen) des Befehls als String
* @return Gibt zurück, wieviele Ticks der Befehl zur Ausführung braucht.
* Nach Ablauf dieser Ticks (EventIn "move") kann der nächste Befehl ausgeführt
* werden.
*/
   public int Rotate(String param)
   {
      int steps=0;
      try {
         String tok;
         StringTokenizer t= new StringTokenizer(param,",");
         LinkedList angles=null;
         Variable v=null;
         String p=null;
         float x=0,y=0,z=0;
         String name=new String();
         boolean autostop = true;
         boolean wait=true;
         boolean absolute=false;
         while(t.hasMoreElements()) {
            v=new Variable(t.nextToken());
            p=v.GetName();
            if(p.equalsIgnoreCase("slides"))
               steps = v.intValue();
            else if(p.equalsIgnoreCase("direction"))
               angles=v.floatListValue();
            else if(p.equalsIgnoreCase("axisx"))
               x=v.floatValue();
            else if(p.equalsIgnoreCase("axisy"))
               y=v.floatValue();
            else if(p.equalsIgnoreCase("axisz"))
               z=v.floatValue();
            else if(p.equalsIgnoreCase("vector"))
               name=v.GetValue();
            else if(p.equalsIgnoreCase("cyclic"))
               autostop=!(v.boolValue());
            else if(p.equalsIgnoreCase("autostop"))
               autostop=v.boolValue();
            else if(p.equalsIgnoreCase("wait"))
               wait=v.boolValue();
         }
         angles.reset();
         LinkedList li = new LinkedList();
         while(angles.hasMoreElements()) {
             double angle= ((Float)angles.currentElement()).doubleValue();
             li.append(new SFRotation(x,y,z,(float)(angle*aRad)));
             angles.nextElement();
         }
         StartRotation(name,
            li,
            (int)steps,autostop);
         if(!wait) steps=0;
      }
      catch(Exception e)
      {
         TextOutput.TextOut("Caught Exception in script in Rotate: " + e);
      }
      return steps;
   }

/** Ausführung des Befehls "Translate" (Translationsbewegung).
```

```
* Eine Translationsbewegung für den gegebenen Translationsvektor wird erzeugt.
* <pre>
* Skript-Parameter:
*   vector    (String) -> Name des Translationsvektors
*   x     (Float)  -> x-Koordinate des Zielortes
*   y     (Float)  -> y-Koordinate des Zielortes
*   z     (Float)  -> z-Koordinate des Zielortes
*   slides    (Float)  -> Anzahl der für die Bewegung benötigten Ticks
*   cyclic    (Bool)   -> Soll sich die Bewegung endlos (bis zur explizieten Beendigung) wieder-
      holen?
*   wait      (Bool)   -> Soll der Kontrollfluß bis zum Ende der Bewegung warten?
* Beispiel:
*   Translate vector robotpos , x (%xpos), y 0, z (%zpos) ,slides %ticks
* </pre>
* @param param Die Parameterliste (Komma als Trennzeichen) des Befehls als String
* @return Gibt zurück, wieviele Ticks der Befehl zur Ausführung braucht.
* Nach Ablauf dieser Ticks (EventIn "move") kann der nächste Befehl ausgeführt
* werden.
*/
   public int Translate(String param)
   {
      int steps=0;
      try {
         String tok;
         StringTokenizer t= new StringTokenizer(param,",");
         Variable v=null;
         String p=null;
         float x=0,y=0,z=0;
         String name=new String();
         boolean autostop = true;
         boolean wait=true;
         while(t.hasMoreElements()) {
            v=new Variable(t.nextToken());
            p=v.GetName();
            if(p.equalsIgnoreCase("slides"))
               steps = v.intValue();
            else if(p.equalsIgnoreCase("x"))
               x=v.floatValue();
            else if(p.equalsIgnoreCase("y"))
               y=v.floatValue();
            else if(p.equalsIgnoreCase("z"))
               z=v.floatValue();
            else if(p.equalsIgnoreCase("vector"))
               name=v.GetValue();
            else if(p.equalsIgnoreCase("cyclic"))
               autostop=!(v.boolValue());
            else if(p.equalsIgnoreCase("autostop"))
               autostop=v.boolValue();
            else if(p.equalsIgnoreCase("wait"))
               wait=v.boolValue();
         }
         StartTranslation(name,
            new SFVec3f(x,y,z),
            (int)steps,autostop);
         if(!wait) steps=0;
      }
      catch(Exception e)
      {
         TextOutput.TextOut("Caught Exception in script in Translate: " + e);
      }
      return steps;
   }

/** Ausführung des Befehls "Wait" (Warten, Anhalten des Kontrollflusses).
* Der Kontrollfluß wird fur die angegebene Zeit angehalten.
* <pre>
* Skript-Parameter:
*   slides    (Float)  -> Anzahl der zu wartenden Ticks
* Beispiel:
*   Wait slides 30
* </pre>
* @param param Die Parameterliste (Komma als Trennzeichen) des Befehls als String
* @return Gibt zurück, wieviele Ticks der Befehl zur Ausführung braucht.
* Nach Ablauf dieser Ticks (EventIn "move") kann der nächste Befehl ausgeführt
* werden.
*/
   public int Wait(String param)
   {
      int steps=0;
```

```
      try {
         String tok;
         StringTokenizer t= new StringTokenizer(param,",");
         Variable v=null;
         String p=null;
         while(t.hasMoreElements()) {
            v=new Variable(t.nextToken());
            p=v.GetName();
            if(p.equalsIgnoreCase("slides"))
               steps = v.intValue();
         }
      }
      catch(Exception e)
      {
         TextOutput.TextOut("Caught Exception in script in Wait: " + e);
      }
      return steps;
   }

/** Ausführung des Befehls "Stop" (Anhalten eines Vektors).
* Der Kontrollfluß wird fur die angegebene Zeit angehalten.
* <pre>
* Skript-Parameter:
*   vector    (String)  -> Name des anzuhaltenden Vektors
* Beispiel:
*   Stop vector arm1rot
* </pre>
* @param param Die Parameterliste (Komma als Trennzeichen) des Befehls als String
* @return Gibt zurück, wieviele Ticks der Befehl zur Ausführung braucht.
* Nach Ablauf dieser Ticks (EventIn "move") kann der nächste Befehl ausgeführt
* werden.
*/
   public int Stop(String param)
   {
      int steps=0;
      try {
         String tok;
         StringTokenizer t= new StringTokenizer(param,",");
         Variable v=null;
         String p=null;
         String name=new String();
         steps = 0;
         while(t.hasMoreElements()) {
            v=new Variable(t.nextToken());
            p=v.GetName();
            if(p.equalsIgnoreCase("vector"))
               name = v.GetValue();
         }
         StopVector(name.trim());
      }
      catch(Exception e)
      {
         TextOutput.TextOut("Caught Exception in script in Stop: " + e);
      }
      return steps;
   }

/** Ausführung des Befehls "Goto" (Sprung-Befehl).
* Der Kontrollfluß springt innerhalb der gerade aktiven Prozedur bis zur angegebenen
* Sprungstelle.
* <pre>
* Skript-Parameter:
*   (String)  -> Name der Sprungstelle
* Beispiel:
*   Tag Start
*   ...
*   Goto Start
* </pre>
* @param param Die Parameterliste (Komma als Trennzeichen) des Befehls als String
* @return Gibt zurück, wieviele Ticks der Befehl zur Ausführung braucht.
* Nach Ablauf dieser Ticks (EventIn "move") kann der nächste Befehl ausgeführt
* werden.
*/
   public synchronized int Goto(String param)
   {
      try {
         String tag = param.trim();
         m_ModifierList.reset();
         boolean found=false;
```

```
         while(m_ModifierList.hasMoreElements() && !found)
         {
            String s = (String) m_ModifierList.currentElement();
            StringTokenizer t = new StringTokenizer(s,"| \t\n\r(){}[]");
            String Param;
            String Command;
            Param=new String();
            if(t.hasMoreElements()) {
               Command = t.nextToken().trim();
               if(Command.equalsIgnoreCase("Tag")) {
                  while(t.hasMoreElements())
                     Param += t.nextToken() + " ";
                  if(Param.trim().equalsIgnoreCase(tag)) found=true;
               }
            }
            if(m_ModifierList.hasMoreElements() && !found) m_ModifierList.nextElement();
         }
         if(!found) TextOutput.TextOut("Fehler in Bewegungskonfiguration: " + param + " nicht
      gefunden");
      }
      catch(Exception e)
      {
         TextOutput.TextOut("Caught Exception in Goto: " + e);
      }
      return 1;   // Sonst Gefahr von Deadlock
   }

/** Gibt die Liste der globalen Variablen zurück.
* @return Liste der globalen Variablen
*/
   public LinkedList GetVars()
   {
      return m_GlobalVars;
   }

/** Sucht die benannte Variable in der Liste der globalen Variablen
* und gibt diese zurück.
* @param name Name der gesuchten Variable
* @ return die gefundene Variable
*/
   public Variable GetVar(String name)
   {
      Variable v=null;
       LinkedList li=(LinkedList) GetVars().clone();
      li.reset();
      while(li.hasMoreElements()) {
         v = (Variable) li.currentElement();
         if(v.GetName().equalsIgnoreCase(name)) {
            return v;
         }
         else li.nextElement();
      }
      return v;
   }

/** Referenzen auf die Talk-History auflösen.
* Ersetzt im übergebenen Ausdruck alle Vorkommen von "history(n)"
* durch die letzten n Texteingaben.des Benutzers.
* Bei n<0 wird die gesamte History genommen.
* @param expr Der zu bearbeitende Ausdruck
* @return Gibt den bearbeiteten Ausdruck zurück
*/
   public String AssignHistory (String expr)
   {
      String ret = new String(expr);
      try {
         int posstart = Expression.FindSubString(ret,"history");
         int posend=posstart;
         int len = ret.length();
         if(posstart != -1) {
            posend=posstart+7;
            String hist;
            while(ret.charAt(posend)== ' ') posend++;
            if(posend <= len && ret.charAt(posend) == '(') {
               char c;
               String linesstr=new String();
               posend++;
               for(c=ret.charAt(posend); posend <= len && c != ')'; linesstr=linesstr+c, po-
      send++, c=ret.charAt(posend));
```

```
                hist=TalkHistory(Expression.intValue(linesstr));
                posend++;
                ret = ret.substring(0,posstart)+hist+ret.substring(posend,len);
             }
          }
       }
       catch(Exception e)
       {
          TextOutput.TextOut("Caught Exception in vrrobot.AssignHistory: " + e);
       }
       return ret;
    }


/** Gibt die letzten Einträge in der Talk-History zurück.
* @param  lines Anzahl der gewünschten Einträge in der Talk-History
* @return Talk-History als String
*/
    public String TalkHistory (int lines)
    {
       String hist = new String();
       try {
          StringTokenizer t= new StringTokenizer(m_TalkHist,"\n\t\r");
          while(t.hasMoreElements()&&lines!=0) {
             hist = hist + t.nextToken() +" ";
             lines--;
          }
       }
       catch(Exception e)
       {
          TextOutput.TextOut("Caught Exception in vrrobot.TalkHistory: " + e);
       }
       return hist;
    }

/** Referenzen auf die aktuellen Werte aller EventOuts auflösen.
* Dabei wird die Funktion AssignEventVal zu Hilfe genommen.
* @see #AssignEventVal
* @see #AssignEventInVals
* @param expr Der zu bearbeitende Ausdruck
* @return Gibt den bearbeiteten Ausdruck zurück
*/
    public String AssignEventOutVals (String expr)
    {
        if(!Expression.Contains(expr,'%')) return expr;
       String ret = new String(expr);
       try {
          LinkedList evouts = (LinkedList) GetEventOutList().clone();
          evouts.reset();
          while(evouts.hasMoreElements())
          {
             VRMLEventOut ev = (VRMLEventOut) evouts.currentElement();
             ret=AssignEventVal(ret,ev);
             evouts.nextElement();
          }
       }
       catch(Exception e)
       {
          TextOutput.TextOut("Caught Exception in vrrobot.AssignEventOutVals: " + e);
       }
       return ret;
    }

/** Referenzen auf die aktuellen Werte von EventIns auflösen.
* Dabei wird die Funktion AssignEventVal zu Hilfe genommen.
* @see #AssignEventVal
* @see #AssignEventOutVals
* @param expr Der zu bearbeitende Ausdruck
* @return Gibt den bearbeiteten Ausdruck zurück
*/
    public String AssignEventInVals (String expr)
    {
        if(!(Expression.Contains(expr,'%')&&Expression.Contains(expr,'.'))) return expr;
       String ret = new String(expr);
       try {
          LinkedList evins = (LinkedList) GetEventInList().clone();
          evins.reset();
          while(evins.hasMoreElements())
          {
```

```
            VRMLEventIn ev = (VRMLEventIn) evins.currentElement();
            ret=AssignEventVal(ret,ev);
            evins.nextElement();
         }
      }
      catch(Exception e)
      {
         TextOutput.TextOut("Caught Exception in vrrobot.AssignEventInVals: " + e);
      }
      return ret;
   }
/** Referenzen auf die aktuellen Werte aller EventOuts auflösen.
* Ersetzt im übergebenen Ausdruck alle Vorkommen von "%Eventname.Komponentenname" durch den je-
      weiligen aktuellen Wert"
* Je nach Typ des gerade betrachteten Events kommen für Komponentenname unterschiedliche Werte
      in Frage.
* Diese sind im einzelnen:
* <pre>
* Für Events vom Typ SFRotation:
*  axisX
*  axisY
*  axisZ
*  direction
* Für Events vom Typ SFVec3f:
*  x
*  y
*  z
* Für Events vom Typ SFString:
*  value
* Für Events vom Typ SFBool:
*  value
* </pre>
* @param expr Der zu bearbeitende Ausdruck
* @param ev Der zu ersetzende VRMLEvent (EventIn oder EventOut)
* @return Gibt den bearbeiteten Ausdruck zurück
*/
   public String AssignEventVal(String expr,VRMLEvent ev)
   {
       if(Expression.FindSubString(expr,ev.GetName())<0) return expr;
      String ret = new String(expr);
      try {
         if(ev==null) return ret;
         vrml.Field f = ev.GetAktField();
         if(f==null) return ret;
         if(f instanceof SFRotation) {
            float [] curr= new float[4];
            ((SFRotation)f).getValue(curr);

      ret=Expression.ReplaceSubExpr(ret,"%"+ev.GetName()+".axisX",Float.toString(curr[0]));

      ret=Expression.ReplaceSubExpr(ret,"%"+ev.GetName()+".axisY",Float.toString(curr[1]));

      ret=Expression.ReplaceSubExpr(ret,"%"+ev.GetName()+".axisZ",Float.toString(curr[2]));

      ret=Expression.ReplaceSubExpr(ret,"%"+ev.GetName()+".direction",Double.toString((double)cu
      rr[3]/aRad));
         }
         else if(f instanceof SFVec3f) {
            float [] curr= new float[3];
            ((SFVec3f)f).getValue(curr);
            ret=Expression.ReplaceSubExpr(ret,"%"+ev.GetName()+".x",Float.toString(curr[0]));
            ret=Expression.ReplaceSubExpr(ret,"%"+ev.GetName()+".y",Float.toString(curr[1]));
            ret=Expression.ReplaceSubExpr(ret,"%"+ev.GetName()+".z",Float.toString(curr[2]));
         }
         else if(f instanceof SFString)

      ret=Expression.ReplaceSubExpr(ret,"%"+ev.GetName()+".value",((SFString)f).getValue());
         else if(f instanceof SFBool)
            ret=Expression.ReplaceSubExpr(ret,"%"+ev.GetName()+".value",new Boole-
      an((((SFBool)f).getValue())).toString());
      }
      catch(Exception e)
      {
         TextOutput.TextOut("Caught Exception in vrrobot.AssignFieldVal: " + e);
      }
      return ret;
   }

/** Ausführung des Befehls "EventOut" (Setzen von EventOut-Werten).
```

```
* Mit Hilfe dieses Befehls können einzelne Werte dierekt gesetzt werden, was zum Beispiel
* zur Realisierung von "Beamen" oder zum Binden der Benutzersicht (Guided Tour) verwendet werden
     kann.
* <pre>
* Skript-Parameter:
*   vector    (String)  -> Name des EventOuts
*   Je nach Typ von "vector" kommen zusätzlich verschiedene Parameter hinzu.
*   Diese sind im einzelnen:
*   Für Events vom Typ SFRotation:
*    axisX
*    axisY
*    axisZ
*    direction
*   Für Events vom Typ SFVec3f:
*    x
*    y
*    z
*   Für Events vom Typ SFString:
*    value
*   Für Events vom Typ SFBool:
*    value
*
* Beispiel:
*   procedure GuideOn
*   begin
*     EventOut vector guide, value true
*   end
* </pre>
* @param param Die Parameterliste (Komma als Trennzeichen) des Befehls als String
* @return Gibt zurück, wieviele Ticks der Befehl zur Ausführung braucht.
* Nach Ablauf dieser Ticks (EventIn "move") kann der nächste Befehl ausgeführt
* werden.
*/
   public int EventOut(String param)
   {
      try {
         String tok;
         StringTokenizer t= new StringTokenizer(param,",");
         Variable v=null;
         String p=null;
         float x=0,y=0,z=0,axisx=0,axisy=0,axisz=0,angle=0;
         String value=new String();
         String name=new String();
         while(t.hasMoreElements()) {
            v=new Variable(t.nextToken());
            p=v.GetName();
            if(p.equalsIgnoreCase("x"))
               x=v.floatValue();
            else if(p.equalsIgnoreCase("y"))
               y=v.floatValue();
            else if(p.equalsIgnoreCase("z"))
               z=v.floatValue();
            else if(p.equalsIgnoreCase("vector"))
               name=v.GetValue();
            else if(p.equalsIgnoreCase("direction"))
               angle=v.floatValue();
            else if(p.equalsIgnoreCase("axisx"))
               axisx=v.floatValue();
            else if(p.equalsIgnoreCase("axisy"))
               axisy=v.floatValue();
            else if(p.equalsIgnoreCase("axisz"))
               axisz=v.floatValue();
            else if(p.equalsIgnoreCase("value"))
               value=v.GetValue();
         }
        VRMLEventOut ev = FindEventOut(name);
        if(ev==null) TextOutput.TextOut("Fehler in EventOut: "+name+" nicht gefunden");
        else {
           if(ev.GetField() instanceof SFBool)
              SetBool(ev,Expression.boolValue(value));
           else if(ev.GetField() instanceof SFString)
              SetString(ev,value);
           else if(ev.GetField() instanceof SFVec3f)
              SetVec3(ev,x,y,z);
           else if(ev.GetField() instanceof SFRotation)
              SetRotation(ev,axisx,axisy,axisz,angle);
            else TextOutput.TextOut("Fehler in EventOut: unbekannter Typ "+name);
         }
      }
```

```
      catch(Exception e)
      {
         TextOutput.TextOut("Caught Exception in script in EventOut: " + e);
      }
      return 0;
   }

/** Ausführung des Befehls "If" (Bedingte Ausführung).
* Der Befehl "If" bitet die Möglichkeit, die Ausführung eines anderen Befehls von einer boolsche
      Bedingung
* abhängig zu machen.
* <pre>
* Syntax:
*   If Bedingung then Befehl1 else Befehl2
*   Wenn der boolsche Ausdruck Bedingung den Wert "true" hat, wird Befehl1 ausgeführt, ansonsten
      Befehl2.
* Beispiel:
*   if (%zposdif>0) then Localvar angle (atan (%xposdif/%zposdif))
* </pre>
* @param param Die Parameterliste (Komma als Trennzeichen) des Befehls als String
* @return Gibt zurück, wieviele Ticks der Befehl zur Ausführung braucht.
* Nach Ablauf dieser Ticks (EventIn "move") kann der nächste Befehl ausgeführt
* werden.
*/
   public int If_Statement(String param)
   {
      String com=new String("");
      try {
         int cond_pos=0;
         int then_pos=Expression.FindSubExpr(param,"then");
         int else_pos=Expression.FindSubExpr(param,"else");
         if(else_pos == -1) else_pos=param.length();
         if(then_pos!=-1
            && Expression.boolValue(param.substring(cond_pos,then_pos)))
                        com  = param.substring(then_pos+4,else_pos);
         else if(else_pos+4 <param.length())
            com  = param.substring(else_pos+4,param.length());
         if(com.length()>0)  return Apply(com.trim());

      }
      catch(Exception e)
      {
         TextOutput.TextOut("Caught Exception in script in If_Statement "+param+": " + e);
      }
      return 0;
   }

/** Ausführung des Befehls "Procedure" (Definieren einer Prozedur zur Laufzeit).
* Mit Hilfe dieses Befehls können jederzeit neue Prozeduren erzeugt und schon vorhandene über-
      schrieben werden.
* <pre>
* Syntax:
*   Procedure Prozedurname Parameterliste
*   begin
*     Befehl1
*     Befehl2
*     ...
*   end
*
* Beispiel:
*   procedure MyRuntimeProc\nbegin\nSay Es klappt!\nend

* </pre>
* @param param Die Parameterliste (Komma als Trennzeichen) des Befehls als String
* @return Gibt zurück, wieviele Ticks der Befehl zur Ausführung braucht.
* Nach Ablauf dieser Ticks (EventIn "move") kann der nächste Befehl ausgeführt
* werden.
*/
   public synchronized int DefRuntimeProcedure(String param)
   {
      try {
         String procname=null;
         StringTokenizer t2 = new StringTokenizer(param," \t\r\n");
         if(t2.hasMoreElements()) procname=t2.nextToken();
         String procparam=new String();
         while(t2.hasMoreElements()) procparam+=t2.nextToken()+" ";
         LinkedList proccom=new LinkedList();
         int comsection = 0;
         String tok=null;
```

```
         while(m_ModifierList.hasMoreElements() && comsection<=0) {
            tok = ((String)(m_ModifierList.currentElement())).trim();
            m_ModifierList.nextElement();
            if(tok.toLowerCase().startsWith("begin")) comsection++;
            else procparam+=tok+" ";
         }
         while(m_ModifierList.hasMoreElements() && comsection>0) {
            tok = ((String)(m_ModifierList.currentElement())).trim();
            m_ModifierList.nextElement();
            if(tok.toLowerCase().startsWith("begin")) comsection++;
            else if(tok.toLowerCase().startsWith("end")) comsection--;
            if(comsection>0) {
               proccom.append(tok);
            }
         }
         DefProcedure(procname,procparam.trim(),proccom);

      }
      catch(Exception e)
      {
         TextOutput.TextOut("Caught Exception in script in DefRuntimePRocedure "+param+": " +
      e);
      }
      return 0;

   }


/** Aufruf einer selbstdefinierten Prozedur.
* Die gerade aktive Prozedur wird auf dem Stack abgelegt.
* <pre>
* Syntax:
*   Prozedurname Parameterliste
*
* Beispiel:
*   GotoPlace xpos %x, zpos %z+30
* </pre>
* @param name Name der aufzurufenden Prozedur
* @param param Die Parameterliste (Komma als Trennzeichen) des Befehls als String
* @return Gibt zurück, wieviele Ticks der Befehl zur Ausführung braucht.
* Nach Ablauf dieser Ticks (EventIn "move") kann der nächste Befehl ausgeführt
* werden.
*/
   public synchronized int CallProcedure(String name, String param)
   {
      boolean found =false;
      try {
         m_Procedures.reset();
         Procedure p=null;
         while(m_Procedures.hasMoreElements()&& !found)
         {
            p = (Procedure) m_Procedures.currentElement();
            if(p.GetName().equalsIgnoreCase(name)) found = true;
            else m_Procedures.nextElement();
         }
         if(found && p!=null) {
            m_ProcStack.push(m_aktProcedure);
            m_aktProcedure=p;
            m_ModifierList=p.GetCom(param);
            if(m_ModifierList==null) return 0;
            m_ModifierList.reset();
            m_aktProcedure.SetSteps(0);
            if(m_ModifierList.hasMoreElements()) {
               String entry = (String)  (m_ModifierList.currentElement());
               m_ModifierList.nextElement();
               return Apply(entry);
            }
            else return 0;
         }
      }
      catch(Exception e)
      {
         TextOutput.TextOut("Caught Exception in vrrobot.CallProcedure: " + e);
      }
      return 0;
   }

/** Aufruf einer selbstdefinierten Callback-Prozedur.
* Callback-Prozeduren werden von processEvent automatisch aufgerufen, wenn VRML-Events ankommen.
```

```
* Im Skript können solche Callbacks wie normale Prozeduren definiert werden, nur daß der
* Prozedurname dem Namen des jeweiligen EventIns entsprechen muß.
* <pre>
* Beispiel:
*   procedure visibility value true
*   begin
*     Say Visibility=%value
*   end
* </pre>
* @see #DefProcedure
* @param name Name der aufzurufenden Prozedur
* @param e Der VRML-Event
* @return Gibt zurück, wieviele Ticks der Befehl zur Ausführung braucht.
* Nach Ablauf dieser Ticks (EventIn "move") kann der nächste Befehl ausgeführt
* werden.
*/
   public int CallProcedure(String name, vrml.Event e)
   {
      try {
         String param=new String();
         vrml.ConstField cf=e.getValue();
         if(cf instanceof ConstSFString)
            param = new String("value "+((ConstSFString)cf).getValue());
         else if(cf instanceof ConstSFBool)
            param = new String("value "+((ConstSFBool)cf).getValue());
         else if(cf instanceof ConstSFRotation) {
            float [] rot = new float[4];
            ((ConstSFRotation)cf).getValue(rot);
            param = new String("axisx "+rot[0]+",axisy "+rot[1]+",axisz "+rot[2]+",direction
      "+rot[3]);
         }
         else if(cf instanceof ConstSFVec3f) {
            float [] vec = new float[3];
            ((ConstSFVec3f)cf).getValue(vec);
            param = new String("x "+vec[0]+",y "+vec[1]+",z "+vec[2]);
         }
         return CallProcedure(name,param);
      }
      catch(Exception ex){
         TextOutput.TextOut("Caught Exception in vrrobot.CallProcedure: " + ex);
      }return 0;
   }

/** Gibt die gerade aktive Prozedur zurück
* @return Die aktive Prozedur
*/
   public Procedure AktProcedure()
   {
      return m_aktProcedure;
   }

/** Definiert eine Prozedur.
* Ist eine gleichnamige Prozedur bereits vorhanden, wird diese zuvor gelöscht.
* Prozeduren können beim Einlesen der Konfigurationsdatei(en) oder auch zur Laufzeit definiert
      werden.
* <pre>
* Syntax:
*  procedure Prozedurname Parameterliste
*  begin
*    Befehl1
*    Befehl2
*    ...
*  end
* Die Parameterliste besteht dabei aus durch Kommata getrennte Paare von Parameternamen und De-
      faultwerten
* Beispiel:
*  procedure GotoPlace xpos 0 ,zpos 0, ticks 30
*  begin
*    TurnToPos xpos (%xpos), zpos (%zpos), ticks %ticks
*    StartWalk
*    Translate vector robotpos , x (%xpos), y 0, z (%zpos) ,slides %ticks
*    StopWalk
*  end
* </pre>
* @param name Name der zu definierenden Prozedur
* @param param Parameterliste
* @return Erfolg.
*/
   public synchronized boolean DefProcedure(String name, String param,LinkedList com)
```

```
   {
      try {
         m_Procedures.reset();
         while(m_Procedures.hasMoreElements()) {
            Procedure proc = (Procedure) m_Procedures.currentElement();
            if(proc.GetName().equalsIgnoreCase(name)) m_Procedures.remove();
            else m_Procedures.nextElement();
         }
         m_Procedures.append(new Procedure(name,param,com));
      }catch(Exception e){
         TextOutput.TextOut("Caught Exception in vrrobot.DefProcedure: " + e);
      }return true;
   }

/** Definiert eine Prozedur.
* Ist eine gleichnamige Prozedur bereits vorhanden, wird diese zuvor gelöscht.
* Prozeduren können beim Einlesen der Konfigurationsdatei(en) oder auch zur Laufzeit definiert
      werden.
* <pre>
* Syntax:
*  procedure Prozedurname Parameterliste
*  begin
*    Befehl1
*    Befehl2
*    ...
*  end
* Die Parameterliste besteht dabei aus durch Kommata getrennte Paare von Parameternamen und De-
      faultwerten
* Beispiel:
*  procedure GotoPlace xpos 0 ,zpos 0, ticks 30
*  begin
*    TurnToPos xpos (%xpos), zpos (%zpos), ticks %ticks
*    StartWalk
*    Translate vector robotpos , x (%xpos), y 0, z (%zpos) ,slides %ticks
*    StopWalk
*  end
* </pre>
* @param name Name der zu definierenden Prozedur
* @param param Durch Kommata getrennte Parameterliste als String
* @return Erfolg.
*/
   public boolean DefProcedure(String name, String param,String com)
   {
      try {
         LinkedList li = new LinkedList();
         StringTokenizer t = new StringTokenizer(com,"\n");
         while(t.hasMoreElements())
            li.append(t.nextToken());
         DefProcedure(name,param,li);
      }catch(Exception e){
         TextOutput.TextOut("Caught Exception in vrrobot.DefProcedure: " + e);
      }return true;
   }

/** Durchsucht den übergebenen String nach alleinstehenden Prozeduren.
* Prozeduren, die innerhalb von anderen Prozeduren definiert werden, werden dabei
* ausgespart, da diese wegen ihrer potentiellen Laufzeitabhängigkeit erst zur Laufzeit aufgebaut
      werden können.
* Die gefundenen Prozeduren werden mit Hilfe von DefProcedure interpretiert. Danach wird die
* Hauptprozedur namens "VRRobotMain" aufgerufen.
* @see #DefProcedure
* @param com Einzulesender String, dieser enthält die Prozedurdefinitionen.
* @return Liste aller Zeilen, die keiner Prozedur zugeordnet werden konnten.
*/
   public LinkedList ParseProcedures(String com)
   {
      LinkedList li=new LinkedList();
      try {
         m_Procedures=new LinkedList();
         String procname=null;
         String procparam=null;
         LinkedList proccom=null;
         StringTokenizer t = new StringTokenizer(com,"\n");
         String tok;
         int i=0;
         while(t.hasMoreElements()) {
            tok = t.nextToken();
            procname=null;
            if(Expression.SubStringCheck(tok,"Procedure")) {
```

```
               // Die Deklaration einlesen
               StringTokenizer t2 = new StringTokenizer(tok," \t\r\n");
               if(t2.hasMoreElements()) t2.nextToken();
               if(t2.hasMoreElements()) procname=t2.nextToken();
               procparam=new String();
               while(t2.hasMoreElements()) procparam+=t2.nextToken()+" ";
               proccom=new LinkedList();
               int comsection = 0;
               while(t.hasMoreElements() && comsection<=0) {
                  tok = t.nextToken().trim();
                  if(tok.toLowerCase().startsWith("begin")) comsection++;
                  else procparam+=tok+" ";
               }
               while(t.hasMoreElements() && comsection>0) {
                  tok = t.nextToken().trim();
                  if(tok.toLowerCase().startsWith("begin")) comsection++;
                  else if(tok.toLowerCase().startsWith("end")) comsection--;
                  if(comsection>0) proccom.append(tok);
               }
               DefProcedure(procname,procparam.trim(),proccom);
            }
            if(procname==null) li.append(tok);
         }
         li.reset();
//         DefProcedure("VRRobotMain","",li);
         CallProcedure("VRRobotMain","");
      }
      catch(Exception e)
      {
         TextOutput.TextOut("Caught Exception in vrrobot.ParseProcedures: " + e);
      }
      return li;
   }

/** Führt die erste Stufe der Vorverarbeitung von Skriptprogrammen aus.
* Dabei werden Kommentare (in Java/C Syntax) entfernt und Datei-Imports der Form "import Datei-
      name" aufgelöst.
* @param str Zu verarbeitender String
* @return Verarbeiteter String.
*/
   public String Preprocess(String str)
   {
      try {
         int start=0;
         int end=0;
         int len = str.length();
         start=Expression.FindSubString(str,"/*");
         if(start!=-1) {
           end=start+Expression.FindSubString(str.substring(start,len),"*/");
           if(end!=-1)
            return Preprocess(str.substring(0,start)
                              +str.substring(end+2,len));
         }
         start=Expression.FindSubString(str,"//");
         if(start!=-1) {
           end=start+Expression.FindSubString(str.substring(start,len),"\n");
           if(end!=-1)
            return Preprocess(str.substring(0,start)
                              +str.substring(end,len));
         }
         int i,pos;
         if((pos=Expression.FindSubString(str,"import")) != (-1)) {
            StringBuffer filenamebuf = new StringBuffer();
            for(i=pos+6;str.charAt(i)!='\n'; i++) {
               filenamebuf.append(str.charAt(i));
            }
            String filename = new String(filenamebuf).trim();
            String path = FileIO.GetPath(m_KonfigFile);
            if(path!=null && !path.equalsIgnoreCase(""))
               filename = path+"/"+filename;
            String importtext=FileIO.ReadFromFile(GetWorldPath(),filename);
            return Preprocess(str.substring(0,pos)
                              +importtext
                              +str.substring(i,len));
         }
      }
      catch(Exception e)
      {
         TextOutput.TextOut("Caught Exception in vrrobot.Preprocess: " + e);
```

```
      }
      return str;
   }

/** Ermittelt, ob noch Befehle auszuführen sind.
* Dies bezieht Prozeduren auf dem Stack mit ein, schliesst aber evetuell auftretene Callbacks
      aus.
* @return Gibt zurück, ob noch Befehle auszuführen sind..
*/
   public boolean Motion_Ended()
   {
      return m_nomore_MotionCommands;
   }

/** Gibt die komplette Talk-History zurück.
* @return Die Talk-History.
*/
   public String GetText()
   {
      return m_TalkHist;
   }

/** Gibt die Position in der aktuellen Befehlsfolge zurück.
* @return Position der aktuellen Befehlsfolge.
*/
   public synchronized int GetMotionPos()
   {
      LinkedList l = (LinkedList) m_ModifierList.clone();
      l.reset();
      boolean found=false;
      int i=0;
      while(l.hasMoreElements() && !found)
      {
         if(m_ModifierList.currentElement() == l.currentElement()) found=true;
         else {
            i++;
            l.nextElement();
         }
      }
      return i;
   }

/** Ersetzt die aktuelle Befehlsfolge durch eine neue.
* @param com Befehlsfolge als String (jede Zeile wird als Befehl interpretiert)
* @param motionpos Einsprungposition: Legt fest, mit welchem Befehl die Ausführung begonnen wer-
      den soll
*/
   public synchronized void SetMotionCommands(String com,int motionpos)
   {
      try {
         if(com == null) com =new String("");
         if(m_MotionCommands == null) m_MotionCommands =new String("");
         m_MotionCommands = new String(com);
         m_nomore_MotionCommands=true;
         m_ModifierList=new LinkedList();
         m_ModifierList.reset();
         m_ProcStack = new Stack();
         com = Preprocess(com);
         ParseProcedures(com);
         int i=0;
         while(m_ModifierList.hasMoreElements() && ((i++) < motionpos))
            m_ModifierList.nextElement();
         m_nomore_MotionCommands=false;

         if(m_aktProcedure!=null) {
         m_aktProcedure.SetStep(0);
         m_aktProcedure.SetSteps(0);
         }
            if(m_InitProc) {
               CallProcedure("initialize","");
               m_InitProc=false;
             }
         move();

      }
      catch(Exception e)
      {
         TextOutput.TextOut("Caught Exception in vrrobot.SetMotionCommands: " + e);
      }
```

```
   }

/** Ersetzt die aktuelle Befehlsfolge durch eine neue.
* @param com Befehlsfolge als String (jede Zeile wird als Befehl interpretiert)
*/
   public void SetMotionCommands(String com)
   {
      try {
         SetMotionCommands(com,0);
      }
      catch(Exception e)
      {
         TextOutput.TextOut("Caught Exception in vrrobot.SetMotionCommands: " + e);
      }
   }

/** Ersetzt die Definition des Dialogverhaltens.
* @param com Liste von Einträgen als String (jede Zeile wird als Eintrag interpretiert).
* Aus jedem dieser Einträge wird ein Objekt vom Typ "Answer" erstellt; die Summe dieser Einträge
      legt
* das Dialogverhalten fest.
* Ein von der Klasse Answer interpretierbarer Eintrag ist folgendermaßen aufgebaut:
* <pre>
* Syntax:
*   "Muster" "Antwort" Befehl
* Muster legt fest, bei welchen Texteingaben des Benutzers dieser Eintrag zur Verwendung kommt.
* Muster kann dabei bis zu 10 durch %Nummer gekennzeichnete variable Anteile beinhalten, die im
      Antworttext
* Antwort oder im Befehl Befehl wiederverwendet werden können.
* Beispiele:
*   "Ich erinnere mich an %1" "Denkst Du oft an %1 ?"
*   "Führe mich" "los gehts, wenn du alleine weitergehen willst, sage Stop" Guideon
* </pre>
* @see Answer
*/
   public synchronized void SetDialogCommands(String com)
   {
      if(com == null) com =new String("");
      m_DialogCommands=new String(com);
      m_DialogList.reset();
      while(m_DialogList.hasMoreElements())
         m_DialogList.remove();
      com = Preprocess(com);
      StringTokenizer t = new StringTokenizer(com,"\n");
      while(t.hasMoreElements())
         m_DialogList.append(new Answer(t.nextToken()));
   }

/** Gibt die Definition des Bewegungsverhalten zurück.
* @return Die Definition des Bewegungsverhaltens als String
* @see SetMotionCommands
*/
   public String GetMotionCommands()
   {
     return m_MotionCommands;
   }

/** Gibt die Definition des Dialogverhaltens zurück.
* @return Die Definition des Dialogverhaltens als String (Jede Zeile repräsentiert einen
* von der Klasse "Answer" interpretierbaren Eintrag)
* @see Answer
*/
   public String GetDialogCommands()
   {
      return m_DialogCommands;
   }

/** Mausklick-Callback.
* Wird aufgerufen, wenn der Benutzer mit seiner Maus auf den Charakter klickt (oder die Mausta-
      ste wieder losläßt).
* @param val Gibt an, ob der Charakter angeklickt oder losgelassen wurde.
*/
   public void clicked( boolean val) {
      if (val == false && m_LastClickState == true)
      {
           OpenMainDialog();
      }
      m_LastClickState = val;
   }
```

```
/** Nähe-Callback.
* Wird aufgerufen, wenn der Benutzer sich in die Nähe des Charakters begibt (oder diese ver-
     läßt). Der Nähe-Radius
* ist durch den entprechenden ProximitrySensor in der VRML-Definition festgelegt.
* @param val Gibt an, ob der Charakter der Nähe des Benutzers ist.
*/
   public void proximitry( boolean val) {

      if (val == true && m_LastProxState == false)
      {
      }
      m_LastProxState = val;
   }

/** Sichtbarkeits-Callback.
* Wird aufgerufen, wenn der Charakter ins Sichfeld des Benutzers kommt (oder dieses verläßt).
* @param val Gibt an, ob der Charakter sichtbar ist oder nicht.
*/
   public void visibility( boolean val) {
      if (val == true && m_LastVisState == false)
      {
      }
      m_LastVisState = val;
   }

/** Gibt zurück, ob der Charakter in der Nähe des Benutzers ist.
* @return Gibt zurück, ob der Charakter in der Nähe des Benutzers ist.
*/
   public boolean GetProxState()
   {
      return m_LastProxState;
   }

/** Gibt zurück, ob der Charakter durch den Benutzer sichtbar ist.
* @return Gibt zurück, ob der Charakter durch den Benutzer sichtbar ist.
*/
   public boolean GetVisState()
   {
      return m_LastVisState;
   }

/** Behandlungsfunktion für Dialogereignisse.
* Diese Funktion ist eine Funktion des Interface "ResultProcessor", sie ruft für jeden Dialog
* eine spezialisierte Behandlungsfunktion auf.
* @param source Der aufrufende Dialog.
* @param obj Das betreffende Dialogelement.
* @see ResultProcessor
* @see #HandleKonfigMotion
* @see #HandleKonfigDialog
* @see #HandleMainDialog
* @see #HandleKonfigAnswers
* @see #HandleDialog
*/
   public void processResult(Frame source, Object obj)
   {
     try {
      if(source instanceof KonfigMotion)
         HandleKonfigMotion(source,obj);
      else if(source instanceof KonfigDialog)
         HandleKonfigDialog(source,obj);
      else if(source instanceof MainDialog)
         HandleMainDialog(source,obj);
      else if(source instanceof KonfigAnswers)
         HandleKonfigAnswers(source,obj);
      else HandleDialog(source,obj);
    }
     catch(Exception e)
     {
      TextOutput.TextOut("Caught Exception in processResult: " + e);
     }
   }

/** Behandlungsfunktion für Dialogereignisse in Kindklassen.
* Diese Funktion ist in dieser Klasse leer und für die Überlagerung in Kindklassen gedacht.
* @param source Der aufrufende Dialog.
* @param obj Das betreffende Dialogelement.
* @see #processResult
*/
```

```
   public void HandleDialog(Frame source, Object obj)
   {
   }

/** Öffnet ein Fenster der Klasse KonfigMotion zum Anzeigen und Verändern der Bewegungsbefehle.
* @see KonfigMotion
* @see #HandleKonfigMotion
*/
   public void KonfigMotion()
   {
      KonfigMotion d = m_KonfigMotion;
      if(d==null) {
         d = new KonfigMotion(this);
         d.resize(400, 300);
      }
      d.requestFocus();
      d.setText(GetMotionCommands());
      d.show();
      d.requestFocus();
      m_KonfigMotion = d;
   }

/** Behandlungsfunktion für Dialogereignisse aus der Dialogklasse KonfigMotion.
* @param source Der aufrufende Dialog.
* @param obj Das betreffende Dialogelement.
* @see #processResult
* @see #KonfigMotion
* @see KonfigMotion
*/
   public synchronized void HandleKonfigMotion(Frame source, Object obj)
   {
      try {
         if(((String)obj).equalsIgnoreCase("Schließen"))
         {
            source.hide();
            SetMotionCommands(((KonfigMotion)source).getText());
         }
      }
      catch(Exception e){
         TextOutput.TextOut("Caught Exception in HandleKonfigMotion: " + e);
      }
   }

/** Öffnet ein Fenster der Klasse KonfigDialog.
* Dieser Dialog beinhaltet Schaltflächen zum Laden, Speichern sowie zum Öffnen der Dialoge
* KonfigMotion und KonfigDialog
* @see KonfigDialog
* @see #HandleKonfigDialog
* @see #KonfigMotion
* @see #KonfigDialog
*/
   public void KonfigDialog()
   {
      KonfigDialog d = m_KonfigDialog;
      if(d==null) {
         d = new KonfigDialog(this);
         d.resize(400, 150);
      }
      d.requestFocus();
      d.show();
      d.requestFocus();
       m_KonfigDialog = d;
   }

/** Behandlungsfunktion für Dialogereignisse aus der Dialogklasse KonfigDialog.
* @param source Der aufrufende Dialog.
* @param obj Das betreffende Dialogelement.
* @see #processResult
* @see #KonfigDialog
* @see KonfigDialog
*/
   public void HandleKonfigDialog(Frame source, Object obj)
   {
      try {
         if(((String)obj).equalsIgnoreCase("Schließen"))
         {
            source.hide();
         }
         if(((String)obj).equalsIgnoreCase("Bewegungseinstellungen"))
```

```
         {
            KonfigMotion();
         }
         if(((String)obj).equalsIgnoreCase("Dialogeinstellungen"))
         {
            KonfigAnswers();
         }
         if(((String)obj).equalsIgnoreCase("Laden"))
         {
            FileDialog d = new FileDialog(source,"Konfiguration laden",
                           FileDialog.LOAD);
//            d.setFilenameFilter("*.rcf");
            d.setDirectory(GetWorldPath());
            d.setFile(getClass().getName()+".rcf");
            d.show();
            String filename = d.getFile();
            if(filename!=null)
               ReadKonfig(d.getDirectory()+File.separator+filename);
         }
         if(((String)obj).equalsIgnoreCase("Speichern"))
         {
            FileDialog d = new FileDialog(source,"Konfiguration speichern",
                           FileDialog.SAVE);
//            d.setFileNameFilter("*.rcf");
            d.setDirectory(GetWorldPath());
            d.setFile(getClass().getName()+".rcf");
            d.show();
            String filename = d.getFile();
            if(filename!=null)
               SaveKonfig(d.getDirectory()+File.separator+filename);
         }
      }
      catch (Exception e){
         TextOutput.TextOut("Caught Exception in HandleKonfigDialog: " + e);
      }
   }

/** Öffnet ein Fenster der Klasse KonfigAnswers zur Definition des Dialogverhaltens.
* @see KonfigAnswers
* @see #HandleKonfigAnswers
*/
   public void KonfigAnswers()
   {
       KonfigAnswers d = m_KonfigAnswers;
      if(d==null){
         d = new KonfigAnswers(this);
         d.resize(500, 300);
      }
      d.requestFocus();
      d.setText(GetDialogCommands());
      d.show();
      d.requestFocus();
       m_KonfigAnswers=d;
   }

/** Behandlungsfunktion für Dialogereignisse aus der Dialogklasse KonfigAnswers.
* @param source Der aufrufende Dialog.
* @param obj Das betreffende Dialogelement.
* @see #processResult
* @see #KonfigAnswers
* @see KonfigAnswers
*/
   public void HandleKonfigAnswers(Frame source, Object obj)
   {
      try {
         if(((String)obj).equalsIgnoreCase("Schließen"))
         {
            source.hide();
            SetDialogCommands(((KonfigAnswers)source).getText());
         }
      }
      catch (Exception e){
         TextOutput.TextOut("Caught Exception in HandleKonfigAnswers: " + e);
      }
   }

/** Öffnet ein Fenster der Klasse MainDialog zur Durchführung von Textdialogen.
* Der Dialog bitet zusätzlich die Möglichkeit zum Aufruf des Konfigurationsdialoges.
* @see MainDialog
```

```
* @see #HandleMainDialog
* @see KonfigDialog
*/
   public void OpenMainDialog()
   {
      MainDialog d = m_MainDialog;
      if(d==null) {
         d  = new MainDialog(this);
         d.resize(600,300);
      }
      d.requestFocus();
      d.show();
      d.toFront();
      d.requestFocus();
      m_MainDialog = d;
   }

/** Behandlungsfunktion für Dialogereignisse aus der Dialogklasse MainDialog.
* @param source Der aufrufende Dialog.
* @param obj Das betreffende Dialogelement.
* @see #processResult
* @see #OpenMainDialog
* @see MainDialog
*/
   public void HandleMainDialog(Frame source, Object obj)
   {
      try {
         if(obj==null) return;
         if(((String)obj).equalsIgnoreCase("Schließen"))
         {
            source.hide();
         }
         else if(((String)obj).equalsIgnoreCase("Antwort"))
         {
            String text = ((MainDialog)source).TextIn().replace('\n',' ');
            ((MainDialog)source).TextIn("");
            SendToSharedObj("rpc_textin",text);
            if(text!=null && !text.equals("")) Say(Answer(text.trim()));
            m_Text=text;
         }
         else if(((String)obj).equalsIgnoreCase("Konfigurieren"))
         {
            KonfigDialog();
         }
      }
      catch(Exception e){
         TextOutput.TextOut("Caught Exception in HandleMainDialog: " + e);
      }
   }

/** Textausgabe im Hauptfenster, unter anderem als Implementation des Skript-Befehls "Say".
* Dem Text wird der Name des VRRobot vorangestellt.
* <pre>
* Skript-Parameter:
*   (String) -> Ausgabetext
* Beispiel:
*   Say Hallo!
* </pre>
* @param s Der angegebene Text wird im Hauptfenster (MainDialog) ausgegeben.
*/
   public void Say(String s)
   {
      try {
         if(s!=null) {
            OpenMainDialog();
            if(m_MainDialog != null) m_MainDialog.RobotTextOut(m_Name,s);
         }
      }
      catch(Exception e){
         TextOutput.TextOut("Caught Exception in vrrobot.Say: " + e);
      }
   }

/** Textausgabe im Hauptfenster, unter anderem als Implementation des Skript-Befehls "TextOut".
*
* <pre>
* Skript-Parameter:
*   (String) -> Ausgabetext
* Beispiel:
```

```
*   TextOut Hallo!
* </pre>
* @param s Der angegebene Text wird im Hauptfenster (MainDialog) ausgegeben.
*/
   public void TextOut(String s)
   {
      try {
         if(s!=null) {
            OpenMainDialog();
            if(m_MainDialog != null) m_MainDialog.TextOut(s);
         }
      }
      catch(Exception e){
         TextOutput.TextOut("Caught Exception in vrrobot.TextOut: " + e);
      }
   }

/** Schickt einen String an alle anderen Instanzen in einer Mehrbenutzerwelt.
* Um gemeinsam benutzte Objekte in einer Mehrbenutzerwlt zu synchronisieren,
* müssen die Benutzereingaben an alle anderen Clients weitergegeben werden.
* Dieser Mechanismus ist biher spezifisch für den Sony Community Browser, d.h. er ist
* nicht im VRML2-Standard enthalten.
* @param ev Name der Schnittstelle (EventIn)
* @param p Der zu verschickende Text.
*/
   public void SendToSharedObj(String ev,String p)
   {
      try {
         Vscp.sendApplSpecificMsgWithDist(GetMainObjNode(),ev,
            p, Vscp.allClientsExceptMe);
      }
      catch(Exception e){
         TextOutput.TextOut("Caught Exception in vrrobot.SendToSharedObj: " + e);
      }
   }

/** Beantwortet eine Texteingabe des Benutzers aufgrund einer Liste von Objekten der Klasse
      "Answer".
* Aus dieser Liste wird mit Hilfe eines Pattern-Matching Verfahrens ein passender Eintrag ausge-
      wählt
* und zurückgegeben. Ist dem ausgewählten Eintrag zusätzlich ein Befehl zugeordnet,
* so wird dieser ausgeführt.
* @see Answer
* @see #SetDialogCommands
* @see MainDialog
* @param userstring Der vom Benutzer eingegebene Text.
* @return Der Antworttext
*/
   public synchronized String Answer(String userstring)
   {
      String s=null;
      try {
         if( userstring==null
          || userstring.equals("")
          || userstring.equals("\n")
          || userstring.equals("\n\n"))
            return null;
         m_TalkHist = userstring + "\n" + m_TalkHist;
         m_DialogList.reset();
         Answer a=null;
         LinkedList matchlist=new LinkedList();
         while(m_DialogList.hasMoreElements())
         {
            a = (Answer)(m_DialogList.currentElement());
            s = a.Match(userstring);
            if(s!=null)
               matchlist.append(a);
            m_DialogList.nextElement();
         }
         // Liste bzgl. ihrer Wertung sortieren
         int matchrating_sum=0;
         LinkedList new_matchlist=new LinkedList();
         int match_cnt=matchlist.size();
         while(matchlist.hasMoreElements()) {
            a = (Answer)(matchlist.currentElement());
            if(a!=null) {
               new_matchlist.reset();
               matchrating_sum += a.GetMatchRating(match_cnt);
               while(new_matchlist.hasMoreElements()&&a!=null) {
```

```
                if(a.GetMatchRating(match_cnt) >
      ((Answer)new_matchlist.currentElement()).GetMatchRating(match_cnt)) {
                   new_matchlist.insert(a);
                   a=null;
                }
                else new_matchlist.nextElement();
             }
             if(a!=null) new_matchlist.append(a);
          }
          matchlist.nextElement();
       }
       if(matchlist.size()!=0) {
          // Aus der Liste der passenden Einträge wird zufällig,
          // aber mit Gewichtung der Wertung, einer ausgewählt.
          int r = (int) (Math.random()*(double)matchrating_sum);
          matchlist.reset();
          matchrating_sum=0;
          while(matchlist.hasMoreElements() && matchrating_sum < r) {
             a= (Answer) matchlist.currentElement();
             matchrating_sum += a.GetMatchRating(match_cnt);
             matchlist.nextElement();
          }
          if(a!=null)
             s=a.Match(userstring);
       }
       OpenMainDialog();
       if(m_MainDialog != null) m_MainDialog.UserTextOut(userstring);
       if(s==null || a==null) s = new String("Keine Ahnung");
       else {
          int steps = Apply(a.GetRestString());
                if(m_aktProcedure!=null) {
              m_aktProcedure.SetStep(0);
           m_aktProcedure.SetSteps(steps);
         }
       }
    }
    catch(Exception e){
       TextOutput.TextOut("Caught Exception in vrrobot.Answer: " + e);
    }
    return s;
 }

/** Speichert alle Konfigurationsdaten in einer Datei.
* @see #ReadKonfig
* @param filename Der Dateiname.
*/
 public void SaveKonfig(String filename)
 {
    String str=new String();
    str += "Begin_Motion\n";
    str += GetMotionCommands()+"\n";
    str += "End_Motion\n";
    str += "Begin_Dialog\n";
    str += GetDialogCommands()+"\n";
    str += "End_Dialog";
    FileIO.SaveToFile(GetWorldPath(),filename,str);
 }

/** Liest Konfigurationsdaten aus einer Datei.
* @see #SaveKonfig
* @param filename Der Dateiname.
*/
 public void ReadKonfig(String filename)
 {
    try {
       String str = FileIO.ReadFromFile(GetWorldPath(),filename);
       String tok;
       String motionstring;
       String dialogstring;
       if(str!=null) {
          StringTokenizer t = new StringTokenizer(str,"\n");
          while(t.hasMoreElements())
          {
             tok = t.nextToken();
             if(tok.equalsIgnoreCase("Begin_Motion"))
             {
                motionstring=new String();
                tok = t.nextToken();
                while(t.hasMoreElements() && !tok.equalsIgnoreCase("End_Motion"))
```

```
                {
                   motionstring += tok+"\n";
                   tok = t.nextToken();
                }
                     CallInitProc();
                SetMotionCommands(motionstring);
             }
             if(tok.equalsIgnoreCase("Begin_Dialog"))
             {
                dialogstring=new String();
                tok = t.nextToken();
                while(t.hasMoreElements() && !tok.equalsIgnoreCase("End_Dialog"))
                {
                   dialogstring += tok+"\n";
                   tok = t.nextToken();
                }
                SetDialogCommands(dialogstring);
             }
          }
       }
      }
      catch(Exception e)
      {
         TextOutput.TextOut("Caught Exception in ReadKonfig: " + e);
      }
   }

/** Gibt den Pfad der zum VRRobot gehörenden
* virtuellen Welten (bzw. der VRML2-Datei) zurück.
* @return Der Pfad der VRML2-Datei.
*/
   public String GetWorldPath()
   {
      try {
         return FileIO.GetPath(getBrowser().getWorldURL());
      }
      catch(Exception e)
      {
         TextOutput.TextOut("Caught Exception in GetWorldPath: " + e);
      }
      return null;
   }
}
```

3.5 Die Klasse Procedure

```
/*********************************************************************
* @(#)Procedure.java1.0 97/06/23 Hauke Ernst
*/
package VRRobot;

import java.awt.*;
import java.util.*;
import java.io.*;
import java.lang.*;

import VRMLInterface.LinkedList;
import VRMLInterface.ResultProcessor;

/**
* Die Klasse Procedure enthält alle Daten und Funktionen,
* die zur Verwaltung einer Prozedur der Skriptsprache des
* VRRobot benötigt werden.
*
*
* @author  Hauke Ernst
* @version 1.0
*/
public class Procedure
{
   String m_Name;
   String m_Param;
   LinkedList m_Com;
   LinkedList m_LocalVars;
   int m_Step;
```

```
   int m_Steps;

/** Konstruktor.
* @param name Name der Prozedur
* @param param Durch Kommata getrennte Parameterliste der Prozedur.
*        Jeder Parameter besteht aus einem Parameternamen und einen
*        Default-Wert.
* @param com Liste der Befehle, die die Prozedur enthalten soll
*/
   public Procedure(String name, String param, LinkedList com)
   {
      m_Name=name;
      m_Param=param;
      m_Com=com;
      m_Step=0;
      m_Steps=0;
      m_LocalVars= new LinkedList();
      String tok;
      StringTokenizer t= new StringTokenizer(param,",");
      Variable v=null;
      while(t.hasMoreElements()) {
         v=new Variable(t.nextToken());
         if(v!=null) m_LocalVars.append(v);
      }
   }

/** Vermerkt, wieviele Zeitscheiben der aktuell in
* Verarbeitung befindliche Befehl insgesamt laufen soll.
* Diese Information muß gespeichert sein, wenn es zu
* einer Unterbrechung der Prozedur durch ein Callback
* kommt und der Zustand der laufenden Prozedur
* auf dem Stack abgelegt wird.
*@param steps Anzahl der Zeitscheiben, die der aktuelle Befehl
* insgesamt verbrauchen soll.
*/
   public synchronized void SetSteps(int steps)
   {
      m_Steps=steps;
   }

/** Vermerkt, wieviele Zeitscheiben der aktuell in
* Verarbeitung befindliche Befehl schon gelaufen ist.
* Diese Information muß gespeichert sein, wenn es zu
* einer Unterbrechung der Prozedur durch ein Callback
* kommt und der Zustand der laufenden Prozedur
* auf dem Stack abgelegt wird.
*@param steps Anzahl der Zeitscheiben, die der aktuelle Befehl
* schon verbraucht hat.
*/
   public synchronized void SetStep(int step)
   {
      m_Step=step;
   }

/** Erhöht die Anzahl der Zeitscheiben, die der aktuell in
* Verarbeitung befindliche Befehl schon gelaufen ist, um 1.
* @see SetStep
*/
   public synchronized void IncStep()
   {
      m_Step++;
   }

/** Gibt zurück, wieviele Zeitscheiben der aktuell in
* Verarbeitung befindliche Befehl insgesamt laufen soll.
* @see SetSteps
*@return Anzahl der Zeitscheiben, die der aktuelle Befehl
* insgesamt verbrauchen soll.
*/
   public int GetSteps()
   {
      return m_Steps;
   }

/** Gibt zurück, wieviele Zeitscheiben der aktuell in
* Verarbeitung befindliche Befehl schon gelaufen ist.
* @see SetStep
*@return Anzahl der Zeitscheiben, die der aktuelle Befehl
* schon verbraucht hat.
```

```
*/
   public int GetStep()
   {
      return m_Step;
   }

/** Gibt die Liste der lokalen Variablen zurück.
* @return Liste der lokalen Variablen
*/
   public LinkedList GetVars()
   {
      return m_LocalVars;
   }

/** Sucht die benannte Variable in der Liste der lokalen Variablen
* und gibt diese zurück.
* @param name Name der gesuchten Variable
* @ return die gefundene Variable
*/
   public Variable GetVar(String name)
   {
      Variable v=null;
       LinkedList li=(LinkedList) GetVars().clone();
      li.reset();
      while(li.hasMoreElements()) {
         v = (Variable) li.currentElement();
         if(v.GetName().equalsIgnoreCase(name)) {
            return v;
         }
         else li.nextElement();
      }
      return v;
   }

/** Gibt den Namen der Prozedur zurück.
* @return der Name der Prozedur
*/
   public String GetName()
   {
      return m_Name;
   }

/** Gibt die Parameterliste als String zurück.
* @return die durch Kommata getrennte Liste der Parameter.
*/
   public String GetParam()
   {
      return m_Param;
   }

/** Gibt die Liste der Befehle der Prozedur zurück.
* @return Die Liste der Befehle der Prozedur.
*/
   public LinkedList GetCom()
   {
      return m_Com;
   }

/** Trägt die übergebene Variable in die Liste der lokalen
* Variablen ein oder weist einer gleichnamigen, schon vorhandenen
* Variable einen neuen Wert zu.
* @see #Assign(Variable,LinkedList)
* @param varnew die Variable, die den Namen und den neuen Wert enthält.
* @return false im Fehlerfall
*/
   public boolean Assign(Variable varnew)
   {
      return Assign(varnew,m_LocalVars);
   }

/** Trägt die übergebene Variable in eine Liste von
* Variablen ein oder weist einer gleichnamigen, schon vorhandenen
* Variable einen neuen Wert zu.
* @param varnew die Variable, die den Namen und den neuen Wert enthält.
* @param li Liste der Variablen
* @return false im Fehlerfall
*/
   static public synchronized boolean Assign(Variable varnew,LinkedList li)
   {
```

```
      boolean found = false;
      Variable varold=null;
      li.reset();
      while(li.hasMoreElements()) {
         varold = (Variable) li.currentElement();
         if(varold!=null) {
            if(varold.GetName().equalsIgnoreCase(varnew.GetName())) {
               varold.Assign(varnew.GetValue());
               found=true;
            }
         }
         li.nextElement();
      }
      if(!found) {
         li.append(varnew);
         found=true;
      }
      return found;
   }

/** Ersetzt in einem Ausdruck alle Referenzen auf bekannte Variablen
* durch deren Werte.
* @param expr Der zu bearbeitene Ausdruck
* @param li Liste der Variablen
* @return Der bearbeitete Ausdruck
*/
   static public synchronized String AssignVars(String expr , LinkedList li)
   {
      if(!Expression.Contains(expr,'%')) return expr;
      String ret=new String(expr);
      if(li==null) return ret;
      li.reset();
      Variable v=null;
      while(li.hasMoreElements()) {
      v = (Variable) li.currentElement();
         if(v!=null) ret=Expression.ReplaceSubExpr(ret,"%"+v.GetName(),v.GetValue());
         li.nextElement();
      }
      return ret;
   }

/** Ersetzt in einem Ausdruck alle Referenzen auf die lokalen
* Variablen der Prozedur durch deren Werte.
* @param expr Der zu bearbeitene Ausdruck
* @return Der bearbeitete Ausdruck
*/
   public String AssignLocalVars(String expr)
   {
      return AssignVars(expr,m_LocalVars);
   }

/** Gibt die Liste der Befehle der Prozedur unter Berücksichtigung
* einer Parameterliste zurück.
* @param param die durch Kommata getrennte Parameterliste.
*        Jeder Parameter besteht aus einem Parameternamen und einen
*        Parameterwert.
*/
   public LinkedList GetCom(String param)
   {
      String tok;
      StringTokenizer t= new StringTokenizer(param,",");
      Variable v=null;
      while(t.hasMoreElements()) {
         v=new Variable(t.nextToken());
         if(v!=null) Assign(v);
      }
      return m_Com;
   }
}
```

3.6 Die Klasse Variable

```
/*****************************************************************
* @(#)Variable.java1.0 97/06/23 Hauke Ernst
*/
```

```
package VRRobot;

import java.awt.*;
import java.util.*;
import java.io.*;
import java.lang.*;

import VRMLInterface.LinkedList;


/** Klasse zur Verwaltung von Variablen im Rahmen der
* Skriptsprache, die im wesentlichen von der Klasse VRRobot
* implementiert wird.
* @see vrrobot
*
* @author  Hauke Ernst
* @version 1.0
*/
public class Variable
{
   String m_Name;
   String m_Value;

/** Konstruktor
* @param s Beschreibung der Variablen: Das erste Wort wird als Name,
*          der Rest als Wert interpretiert.
*/
   public Variable(String s)
   {
      m_Name=new String();
      m_Value=new String();
      String tok;
      StringTokenizer t= new StringTokenizer(s," ");
      if(t.hasMoreElements()) m_Name=t.nextToken();
      while(t.hasMoreElements()) {
         m_Value+=t.nextToken()+" ";
      }
      m_Value=m_Value.trim();
      m_Name=m_Name.trim();
   }

/** Gibt den Namen der Variablen zurück.
* @return Der Namen der Variablen.
*/
   public String GetName()
   {
      return m_Name;
   }

/** Weist der Variablen einen neuen Wert zu.
* @param val Der neue Wert.
*/
   public void Assign(String val)
   {
      m_Value=val;
   }

/** Gibt den Wert der Variablen als String zurück.
* Die Auswertung geschieht mit Hilfe von Funktionen
* der Klasse Expression.
* @see Expression
* @return Der Wert der Variablen als String.
*/
   public String GetValue()
   {
      return Expression.stringValue(m_Value);
   }

/** Interpretiert den Wert der Variablen als Gleitkommazahl vom Typ 'float'.
* Die Auswertung geschieht mit Hilfe von Funktionen
* der Klasse Expression.
* @see Expression
* @return Der Wert der Variablen als Zahl vom Typ 'float'.
*/
   public float floatValue()
   {
      return Expression.floatValue(m_Value);
   }
```

```
/** Interpretiert den Wert der Variablen als Liste von Gleitkommazahlen ('float').
* Dabei dient das Leerzeichen als Trennzeichen; mit runden Klammern
* eingeschlossene Teilausdrücke werden jedoch zuerst ausgewertet.
* Die Auswertung geschieht mit Hilfe von Funktionen
* der Klasse Expression.
* @see Expression
* @return Der Wert der Variablen als Liste von 'float'-Werten.
*/
   public LinkedList floatListValue()
   {
      LinkedList b=new LinkedList();
      String str=m_Value.trim();
      int lbrac=0;
      int rbrac=0;
      int i=0;
      String repl,trigger;
      while(lbrac != -1 && rbrac!=-1) {
      int len=str.length();
        lbrac=-1;
        rbrac=-1;
        for(i=0;i<len;i++) if(str.charAt(i)=='(') lbrac=i;
        if(lbrac!=-1) for(i=lbrac;i<len&&rbrac==-1;i++) if(str.charAt(i)==')') rbrac=i;
        if(lbrac != -1 && rbrac!=-1) {
           trigger=str.substring(lbrac,rbrac+1);
           repl = Float.toString(Expression.EvalFloatExpression(trigger));
           str = Expression.ReplaceSubString(str,trigger,repl);
        }
      }
      StringTokenizer t = new StringTokenizer(str," \t\n\r");
      while(t.hasMoreElements()) {
         b.append(new Float(Expression.floatValue(t.nextToken())));
      }
      return b;
   }

/** Interpretiert den Wert der Variablen als Zahl vom Typ 'int'.
* Die Auswertung geschieht mit Hilfe von Funktionen
* der Klasse Expression.
* @see Expression
* @return Der Wert der Variablen als Zahl vom Typ 'int'.
*/
   public int intValue()
   {
      return Expression.intValue(m_Value);
   }

/** Interpretiert den Wert der Variablen als Zahl vom Typ 'double'.
* Die Auswertung geschieht mit Hilfe von Funktionen
* der Klasse Expression.
* @see Expression
* @return Der Wert der Variablen als Zahl vom Typ 'double'.
*/
   public double doubleValue()
   {
      return Expression.doubleValue(m_Value);
   }

/** Interpretiert den Wert der Variablen als 'boolean'.
* Die Auswertung geschieht mit Hilfe von Funktionen
* der Klasse Expression.
* @see Expression
* @return Der Wert der Variablen als 'boolean'.
*/
   public boolean boolValue()
   {
      return Expression.boolValue(m_Value);
   }
}
```

3.7 Die Klasse Expression

```
/*****************************************************************
* @(#)Expression.java1.0 97/06/23 Hauke Ernst
*/
package VRRobot;
```

```
import java.awt.*;
import java.util.*;
import java.io.*;
import java.lang.*;

import VRMLInterface.LinkedList;
import VRMLInterface.ResultProcessor;
import VRMLInterface.TextOutput;

/**
* Die Klasse Expression enthält einige statische Funktionen zur
* Auswertung von Ausdrücken, die auf den Typen boolean, float und string.
* basieren.
*
*
*
* @author  Hauke Ernst
* @version 1.0
*/
public class Expression
{
   static double aRad = (double) (Math.PI/180.0);

/** Prüft auf eine Teilstring-Beziehung.
* @param str1 Der zu durchsuchene String
* @param str2 Der Such-String, nach dem in str1 gesucht werden soll
* @return true, wenn str2 Teilstring von str1 ist.
*/
   static public boolean SubStringCheck(String str1, String str2)
   {
      return FindSubString(str1,str2) != (-1);
   }

/** Durchsucht einen Audruck nach einem Teilausdruck.
* Ein Teilsausdruck kann gefunden werden, wenn str2
* Teilstring von str1 ist und durch Trennzeichen
* von anderen Teilausdrücken getrennnt ist.
* Als Trennzeichen sind felsgelegt:
* " <>=,.:;+-*()[]{}&|%$§?!'#~@/\\\n\t\r\""
* So ist "a" Teilausdruck von "10-a<4",
* aber nicht Teilausdruck von "Haus".
* @param str1 Der zu durchsuchene String
* @param str2 Der Such-Ausdruck, nach dem in str1 gesucht werden soll
* @return Die Position des Teilausdruckes.
*/
   static public int FindSubExpr(String str1, String str2)
   {
      return FindSubExpr(str1,str2," <>=,.:;+-*()[]{}&|%$§?!'#~@/\\\n\t\r\"");
   }

/** Durchsucht einen Audruck nach einem Teilausdruck.
* Ein Teilsausdruck kann gefunden werden, wenn str2
* Teilstring von str1 ist und durch Trennzeichen
* von anderen Teilausdrücken getrennnt ist.
* @param s1 Der zu durchsuchene String
* @param s2 Der Such-Ausdruck, nach dem in str1 gesucht werden soll
* @param delimit Die Trennzeichen
* @return Die Position des Teilausdruckes.
*/
   static public int FindSubExpr(String s1, String s2,String delimit)
   {
      if(s1.equalsIgnoreCase("")) return -1;
      if(s2.equalsIgnoreCase("")) return -1;
      int l1 = s1.length();
      int l2 = s2.length();
      if(l2 > l1) return -1;
      s1= s1.toLowerCase();
      s2= s2.toLowerCase();
    int lastchk=l1-l2;
      int i=0;
      while((i=s1.indexOf(s2,i))>=0)
        {
      if(i>0) if(!Contains(delimit,s1.charAt(i-1))) return -1;
       if(i+l2<l1) if(!Contains(delimit,s1.charAt(i+l2))) return -1;
         return i;
        }
      return -1;
   }
```

```
/**
* Durchsucht einen String nach einem bestimmten Zeichen.
* @param c das gesuchte Zeichen.
* @param str der zu durchsuchene String
* @return true, wenn das Zeichen c in str enthalten ist.
*/
   static public boolean Contains(String str, char c)
   {
      return str.indexOf(c)>=0;
   }

/**
* Durchsucht einen String nach bestimmten Zeichen.
* @param chars die gesuchten Zeichen.
* @param str der zu durchsuchene String
* @return true, wenn eines der Zeichen in chars in str enthalten ist.
*/
   static public boolean Contains(String str, String chars)
   {
      for (int i=0; i < chars.length(); i++)
         if(Contains(str,chars.charAt(i))) return true;
      return false;
   }

/**
* Durchsucht einen String nach einem Teilstring.
* Groß/Kleinschreibung spielt dabei keine Rolle.
* @param str1 der zu durchsuchene String
* @param str2 der gesuchte Teilstring.
* @return die Position des Teilstrings. Ist str2 nicht in str1
* enthalten, wird -1 zurückgegeben.
*/
   static public int FindSubString(String str1, String str2)
   {
      if(str1==null || str2==null) return -1;
      if(str1.equalsIgnoreCase("")) return -1;
      if(str2.equalsIgnoreCase("")) return -1;
       int l1 = str1.length();
       int l2 = str2.length();
      if(l2 > l1) return -1;

      String s1= str1.toLowerCase();
      String s2= str2.toLowerCase();
    return s1.indexOf(s2);
   }

/**
* Durchsucht einen String nach einem Teilstring und ersetzt diesen
* durch einen anderen. Es wird solange weitergesucht, bis keine Fundstelle
* mehr vorhanden ist.
* Groß/Kleinschreibung spielt dabei keine Rolle.
* @see #FindSubString
* @param str der zu durchsuchene String
* @param trigger der gesuchte Teilstring.
* @param repl Der String, der anstelle von trigger eingesetzt werden soll.
* @return der überarbeitete String
*/
   static public String ReplaceSubString(String str, String trigger, String repl)
   {
      int i,pos;
      if(trigger == null || str == null) return str;
      if(trigger.equalsIgnoreCase(repl)) return str;
      while((pos=FindSubString(str,trigger)) != (-1)) {
       str=str.substring(0,pos)+
            repl+
            str.substring(pos+trigger.length(),str.length());
      }
      return str;
   }

/**
* Durchsucht einen Ausdruck nach einem Teilausdruck und ersetzt diesen
* durch einen anderen. Es wird solange weitergesucht, bis keine Fundstelle
* mehr vorhanden ist.
* Groß/Kleinschreibung spielt dabei keine Rolle.
* @see #FindSubExpr
* @param str der zu durchsuchene String
* @param trigger der gesuchte Teilstring.
```

```
* @param repl Der String, der anstelle von trigger eingesetzt werden soll.
* @return der überarbeitete String
*/
   static public String ReplaceSubExpr(String str, String trigger, String repl)
   {
      int i,pos;
      if(trigger == null || str == null) return str;
      if(trigger.equalsIgnoreCase(repl)) return str;
      while((pos=FindSubExpr(str,trigger)) != (-1)) {
       str=str.substring(0,pos)+
            repl+
            str.substring(pos+trigger.length(),str.length());
      }
      return str;
   }

/**
* Wertet einen Ausdruck als Gleitkommazahl aus. Teilausdrücke können
* mit runden Klammern zusammengefaßt werden. Neben den Operatoren
* +,-, * und / stehen die Funktionen sin, cos, tan, asin, acos, atan
* und sqrt zur Verfügung.
* Beispiel: "5*asin (3-2)".
* @param str Der auszuwertene Ausdruck.
* @return der ermittelte Wert des Ausdruckes.
*/
   static public float EvalFloatExpression(String str)
   {
      float ret = 0;
      int i,pos;
      if(str == null) throw new NumberFormatException ("cannot evaluate null expression");
      int lbrac=-1;
      int rbrac=-1;
      int op=-1;
      str=str.trim();
      int len = str.length();
      if(len==0) throw new NumberFormatException ("cannot evaluate empty expression");
      for(i=0;i<len;i++) if(str.charAt(i)=='(') lbrac=i;
      if(lbrac!=-1) for(i=lbrac;i<len&&rbrac==-1;i++) if(str.charAt(i)==')') rbrac=i;
      if(lbrac!= -1 && rbrac != -1)
         return EvalFloatExpression(str.substring(0,lbrac)

      +EvalExpression(str.substring(lbrac+1,rbrac)).toString().trim()
                                    +str.substring(rbrac+1,len));
      if(lbrac!= -1 && rbrac == -1)
         TextOutput.TextOut("')' fehlt in Ausdruck -> "+str);

      for(i=1,op=-1;i<len;i++) if(str.charAt(i)=='+' && str.charAt(i-1)!='e') op=i;
      if(op!=-1)
         return EvalFloatExpression(str.substring(0,op))
               +EvalFloatExpression(str.substring(op+1,len));

      if(len>1 && str.charAt(0)=='-'&&str.charAt(1)=='-') return EvalFloatExpressi-
      on(str.substring(2,len));
      for(i=1,op=-1;i<len&&op==-1;i++) if(str.charAt(i)=='-') op=i;

      if(op!=-1) {
         try {
            return EvalFloatExpression(str.substring(0,op))
                  -EvalFloatExpression(str.substring(op+1,len));
         }
          catch(Exception e)
         {
         }
      }
      for(i=1,op=-1;i<len;i++) if(str.charAt(i)=='/') op=i;
      if(op!=-1)
         return EvalFloatExpression(str.substring(0,op))
               /EvalFloatExpression(str.substring(op+1,len));

      for(i=1,op=-1;i<len;i++) if(str.charAt(i)=='*') op=i;
      if(op!=-1)
         return EvalFloatExpression(str.substring(0,op))
               *EvalFloatExpression(str.substring(op+1,len));

      op=Expression.FindSubExpr(str,"asin");
      if(op==0)
         return (float)(Math.asin((double)EvalFloatExpression(str.substring(4,len)))/aRad);
      op=Expression.FindSubExpr(str,"acos");
      if(op==0)
```

```
         return (float)(Math.acos((double)EvalFloatExpression(str.substring(4,len)))/aRad);
      op=Expression.FindSubExpr(str,"atan");
      if(op==0)
         return (float)(Math.atan((double)EvalFloatExpression(str.substring(4,len)))/aRad);
      op=Expression.FindSubExpr(str,"sin");
      if(op==0)
         return (float) Math.sin(aRad*(double)EvalFloatExpression(str.substring(3,len)));
      op=Expression.FindSubExpr(str,"cos");
      if(op==0)
         return (float) Math.cos(aRad*(double)EvalFloatExpression(str.substring(3,len)));
      op=Expression.FindSubExpr(str,"tan");
      if(op==0)
         return (float) Math.tan(aRad*(double)EvalFloatExpression(str.substring(3,len)));
      op=Expression.FindSubExpr(str,"sqrt");
      if(op==0)
         return (float) Math.sqrt((double)EvalFloatExpression(str.substring(4,len)));

       ret = (new Float(str)).floatValue();
      return ret;
   }

/**
* Wertet einen Ausdruck als boolschen Wert aus. Teilausdrücke können
* mit runden Klammern zusammengefaßt werden. Neben den Operatoren
* <, >, =, <=, >= &, | und ! steht die Funktion
* contains (prüft auf Teilstring-Beziehung) zur Verfügung.
* Beispiel: "(b=3) | (a <= 5) & !(das Haus brennt contains Haus)".
* @param str Der auszuwertene Ausdruck.
* @return der ermittelte Wert des Ausdruckes.
*/
   static public boolean EvalBoolExpression(String str)
   {
      boolean ret = false;
      int i,pos;
      if(str == null) return false;
      int lbrac=-1;
      int rbrac=-1;
      int op=-1;
      str=str.trim();
      int len = str.length();
      for(i=0;i<len;i++) if(str.charAt(i)=='(') lbrac=i;
      if(lbrac!=-1) for(i=lbrac;i<len&&rbrac==-1;i++) if(str.charAt(i)==')') rbrac=i;
      if(lbrac!= -1 && rbrac != -1)
         return EvalBoolExpression(str.substring(0,lbrac)
                              +EvalExpression(str.substring(lbrac+1,rbrac)).toString()
                              +str.substring(rbrac+1,len));
      if(lbrac!= -1 && rbrac == -1) TextOutput.TextOut("')' fehlt in Ausdruck -> "+str);

      for(i=0,op=-1;i<len;i++) if(str.charAt(i)=='|') op=i;
      if(op!=-1)
         return EvalBoolExpression(str.substring(0,op))
               ||EvalBoolExpression(str.substring(op+1,len));

      for(i=0,op=-1;i<len;i++) if(str.charAt(i)=='&') op=i;
      if(op!=-1)
         return EvalBoolExpression(str.substring(0,op))
              &&EvalBoolExpression(str.substring(op+1,len));


      if(str.charAt(0)=='!')return !EvalBoolExpression(str.substring(1,len));

      op=Expression.FindSubString(str,"<=");
      if(op!=-1)
         return EvalFloatExpression(str.substring(0,op))
              <=EvalFloatExpression(str.substring(op+2,len));

      op=Expression.FindSubString(str,">=");
      if(op!=-1)
         return EvalFloatExpression(str.substring(0,op))
              >=EvalFloatExpression(str.substring(op+2,len));

      for(i=0,op=-1;i<len;i++) if(str.charAt(i)=='>') op=i;
      if(op!=-1)
         return EvalFloatExpression(str.substring(0,op))
               >EvalFloatExpression(str.substring(op+1,len));

      for(i=0,op=-1;i<len;i++) if(str.charAt(i)=='<') op=i;
      if(op!=-1)
         return EvalFloatExpression(str.substring(0,op))
```

```
            <EvalFloatExpression(str.substring(op+1,len));

      for(i=0,op=-1;i<len;i++) if(str.charAt(i)=='=') op=i;
      if(op!=-1) {
         try {
            return EvalFloatExpression(str.substring(0,op))
                 ==EvalFloatExpression(str.substring(op+1,len));
         }
         catch (Exception e) {
            return EvalBoolExpression(str.substring(0,op))
                 ==EvalBoolExpression(str.substring(op+1,len));
         }
      }
      op=Expression.FindSubString(str,"contains");
      if(op!=-1)
         return FindSubString(str.substring(0,op).trim(),
                              str.substring(op+8,len).trim())!=-1;

      if(str.equalsIgnoreCase("true")) ret=true;
      else if(str.equalsIgnoreCase("false")) ret=false;
      else throw new NumberFormatException ("Cannot convert to boolean -> "+str);
      return ret;
   }

/**
* Wertet einen Ausdruck als String aus. Dabei werden alle Vorkommen
* von "\n" durch einen Zeilenumbruch ersetzt. Die Funktion
* "Evaluate" kann angewendet werden, um einen Teilstring als
* Gleitkommazahl bzw. boolschen Wert auszuwerten.
* Teilstrings, die durch eckige Klammern ("[" und "]") gekennzeichnet
* sind, werden von jeder Verarbeitung ausgeschlossen.
* Beispiel: "[Evaluate a]=Evaluate a".
* @param str Der auszuwertene Ausdruck.
* @return der ermittelte Wert des Ausdruckes.
*/
   static public String EvalStringExpression(String str)
   {
      String ret = str;
      int i,pos;
      if(str == null) return null;
      int lbrac=-1;
      int rbrac=-1;
      int op=-1;
      str=str.trim();
      int len = str.length();
      for(i=0;i<len;i++) if(str.charAt(i)=='[') lbrac=i;
      if(lbrac!=-1) for(i=lbrac;i<len&&rbrac==-1;i++) if(str.charAt(i)==']') rbrac=i;
      if(lbrac!= -1 && rbrac != -1)
         return EvalStringExpression(str.substring(0,lbrac))
               +str.substring(lbrac+1,rbrac)
               +EvalStringExpression(str.substring(rbrac+1,len));
      if(lbrac!= -1 && rbrac == -1) TextOutput.TextOut("']' fehlt in Ausdruck -> "+str);
      op=Expression.FindSubString(str,"Evaluate");
      if(op!=-1)
         return EvalStringExpressi-
      on(str.substring(0,op))+EvalExpression(str.substring(op+8,len).trim()).toString();
      op=Expression.FindSubString(str,"\\n");
      if(op!=-1)
         return EvalStringExpression(str.substring(0,op)+"\n"+str.substring(op+2,len));
      return ret;
   }

/**
* Wertet einen Ausdruck als boolschen Wert oder als Gleitkommazahl aus.
* Wenn die Interpretation als boolscher Wert fehlschlägt (-> Exception),
* wird als Gleitkommazahl ausgewertet.
* @param str Der auszuwertene Ausdruck.
* @return der ermittelte Wert des Ausdruckes.
*/
   static public Object EvalExpression(String str)
   {
      try {
         return new Boolean(EvalBoolExpression(str));
      }
      catch(Exception e)
      {
      }
      Float f= new Float(EvalFloatExpression(str));
      return f;
```

```
   }

/**
* Wertet einen Ausdruck als double-Wert.
* @see #EvalFloatExpression
*/
   static public double doubleValue(String expr)
   {
      return (double) floatValue(expr);
   }

/**
* Wertet einen Ausdruck als int-Wert.
* @see #EvalFloatExpression
*/
   static public int intValue(String expr)
   {
      return (int) floatValue(expr);
   }

/**
* Wertet einen Ausdruck als float-Wert.
* @see #EvalFloatExpression
*/
   static public float floatValue(String expr)
   {
      float ret=0;
      try {
         ret = EvalFloatExpression(expr);
      }
      catch(Exception e)
      {
         TextOutput.TextOut("Caught Exception in Expression.floatValue: " + e);
      }
     return ret;
   }

/**
* Wertet einen Ausdruck als String-Wert.
* @see #EvalStringExpression
*/
   static public String stringValue(String expr)
   {
     return EvalStringExpression(expr);
   }

/**
* Wertet einen Ausdruck als boolean-Wert.
* @see #EvalBoolExpression
*/
   static public boolean boolValue(String expr)
   {
      boolean ret = false;
      try {
         ret = EvalBoolExpression(expr);
      }
      catch(Exception e)
      {
         TextOutput.TextOut("Caught Exception in Expression.boolValue: " + e);
      }
      return ret;
   }
}
```

3.8 Die Klasse Answer

```
/*****************************************************************
* @(#)Answer.java1.0 97/06/23 Hauke Ernst
*/
package VRRobot;


import java.awt.*;
import java.io.*;
import java.util.*;
```

```
import java.lang.*;

import VRMLInterface.LinkedList;
import VRMLInterface.TextOutput;
import VRMLInterface.ResultProcessor;
import VRRobot.Answering;


/**
* Die Klasse Answer sorgt mit Hilfe eines Pattern-Matching
* Algorithmus für die Abbildung von textlichen Benutzereingaben auf
* Antwortsätze. Jede Instanz der Klasse repräsentiert eine Regel,
* wie diese Abbildung durchgeführt werden kann. Solche Regeln
* enthalten jeweils ein Suchmuster mit statischen und variablen
* Anteilen sowie einen Antwortsatz, in dem die variablen Teile des
* Suchmusters referenziert sein können.
*
* @author  Hauke Ernst
* @version 1.0
*/
public class Answer
{
   String m_String;
   String m_RequestString;
   String m_AnswerString;
   String m_RestString;
   LinkedList m_RequestTokens;
   LinkedList m_AnswerTokens;
   LinkedList m_MatchTokens;
   LinkedList m_RestTokens;

/** Standard-Konstruktor
* @param s Die Pattern-Matching-Regel.
*              Suchmuster und Antwort sind jeweils
*          durch Anführungszeichen eingeschlossen.
*              Variable Anteile sind durch %x gekennzeichnet (1 <= x =>9)
*           Beispiel: "Wo befindet sich %1" "Keine Ahnung, wo %1 ist"
*/
    public Answer(String s){
      try {
         int i=0,requeststart,requestend,answerstart,answerend;
         m_RequestTokens=new LinkedList();
         m_AnswerTokens=new LinkedList();
         m_MatchTokens=new LinkedList();
         m_RestTokens=new LinkedList();

         m_String=s;
         while((i < s.length()) && (s.charAt(i)!= '\"'))i++;
         requeststart=++i;
         while((i < s.length()) && (s.charAt(i)!= '\"'))i++;
         requestend=i++;
         while((i < s.length()) && (s.charAt(i)!= '\"'))i++;
         answerstart=++i;
         while((i < s.length()) && (s.charAt(i)!= '\"'))i++;
         answerend=i++;
         if(answerend<=answerstart) {
            TextOutput.TextOut("Keine Antwort in Dialogeintrag "+m_String);
            return;
         }
         if(requestend<=requeststart) {
            TextOutput.TextOut("Keine Frage in Dialogeintrag "+m_String);
            return;
         }
         m_RequestString = Preprocess(s.substring(requeststart,requestend));
         m_AnswerString  = Preprocess(s.substring(answerstart,answerend));
         if(s.length()>answerend+1)
            m_RestString    = s.substring(answerend+1,s.length());

         StringTokenizer t;
         t = new StringTokenizer(m_RequestString," \t\n\r");
         String tok;
         MatchEntry me;
         int num;
         while(t.hasMoreElements()) {
            tok=t.nextToken();
            if(tok.charAt(0) == '%' && tok.charAt(1) >= '0'  && tok.charAt(1) <= '9') {
               num = (new Integer(tok.substring(1,2))).intValue();
               me = new MatchEntry(num);
               m_MatchTokens.append(me);
```

```
            m_RequestTokens.append(me);
         }
         else m_RequestTokens.append(tok);
      }
      if(m_AnswerString!= null) {
         t = new StringTokenizer(m_AnswerString," \t\n\r");
         while(t.hasMoreElements()) {
            tok=t.nextToken();
            if(tok.charAt(0) == '%' && tok.charAt(1) >= '0'  && tok.charAt(1) <= '9') {
               num = (new Integer(tok.substring(1,2))).intValue();
               m_MatchTokens.reset();
               me=null;
               MatchEntry metemp=null;
               while(m_MatchTokens.hasMoreElements()) {
                  metemp = ((MatchEntry)m_MatchTokens.currentElement());
                  if(metemp.GetNumber() == num) me = metemp;
                  m_MatchTokens.nextElement();
               }
               if(me == null) {
                  TextOutput.TextOut("Referenzierte Variable nicht belegt");
               }
               else m_AnswerTokens.append(me);
            }
            else m_AnswerTokens.append(tok);
         }
      }
      if(m_RestString!=null) {
         t = new StringTokenizer(m_RestString," \t\n\r");
         while(t.hasMoreElements()) {
            tok=t.nextToken();
            if(tok.charAt(0) == '%' && tok.charAt(1) >= '0'  && tok.charAt(1) <= '9') {
               num = (new Integer(tok.substring(1,2))).intValue();
               m_MatchTokens.reset();
               me=null;
               MatchEntry metemp=null;
               while(m_MatchTokens.hasMoreElements()) {
                  metemp = ((MatchEntry)m_MatchTokens.currentElement());
                  if(metemp.GetNumber() == num) me = metemp;
                  m_MatchTokens.nextElement();
               }
               if(me == null) {
                  TextOutput.TextOut("Referenzierte Variable nicht belegt");
               }
               else m_RestTokens.append(me);
            }
            else m_RestTokens.append(tok);
         }
      }
     }
     catch(Exception e)
     {
      TextOutput.TextOut("Caught Exception in Answer constructor: " + e);
     }

    }


/** Prüft eine Benutzeingabe auf die Anwendbarkeit der Regel.
* Dabei wird die als Parameter übergebene Benutzereingabe
* zunächst in eine Liste von Worten zerlegt, die mit Hilfe der Funktion
* MatchList auf Übereinstimmung geprüft wird.
* @param s Die zu überprüfende Benutzereingabe.
* @return Falls das Suchmuster der Regel zur Benutzeingabe passt,
* wird die Antwort zurückgegeben, sonst null;
*/
    public synchronized String Match(String s) {
      String answ=null;
      try {
         s=Preprocess(s);
         StringTokenizer t;

         t = new StringTokenizer(s," \t\n\r");
         String tok;
         LinkedList requestTokens=new LinkedList();
         while(t.hasMoreElements()) {
            tok=t.nextToken();
            requestTokens.append(tok);
         }
```

```
         if(MatchList(requestTokens)) {
            answ=new String();
            m_AnswerTokens.reset();
            while(m_AnswerTokens.hasMoreElements())
            {
               answ += (m_AnswerTokens.currentElement()).toString()+" ";
               m_AnswerTokens.nextElement();
            }
            m_RestString=new String();
            m_RestTokens.reset();
            while(m_RestTokens.hasMoreElements())
            {
               m_RestString += (m_RestTokens.currentElement()).toString()+" ";
               m_RestTokens.nextElement();
            }
         }
       }
       catch(Exception e)
       {
        TextOutput.TextOut("Caught Exception in matching answer: " + e);
       }
       return (answ);
    }

/** Ermittelt eine Bewertung für die Regel.
* Diese Bewertung wird benötigt, wenn zu einer Benutzereingabe
* mehrere Regeln passen und davon eine ausgewählt werden soll.
* @param match_cnt Die Anzahl der insgesamt passenden Regeln. Dieser
* Parameter wird in die Bewertung einbezogen.
* @return Die Bewertung der Regel.
*/
    public synchronized int GetMatchRating(int match_cnt)
    {
       int me_cnt=0;
       int str_cnt=0;
       m_RequestTokens.reset();
       while(m_RequestTokens.hasMoreElements()) {
          if(m_RequestTokens.currentElement() instanceof MatchEntry){
             me_cnt++;
          }
          else {
             str_cnt++;
          }
          m_RequestTokens.nextElement();
       }

       return me_cnt+match_cnt*str_cnt;
    }

/** Prüft eine Benutzereingabe, die als Wortliste vorliegt, auf die
* Anwendbarkeit der Regel.
* @param l Die zu überprüfende Benutzereingabe als Wortliste.
* @return Falls das Suchmuster der Regel zur Benutzeingabe passt,
* wird true zurückgegeben, sonst false;
*/
    public synchronized boolean MatchList(LinkedList l)
    {
       boolean matching=true;
       try {
          MatchEntry me=null;
          String mstr,str;
          m_RequestTokens.reset();
          l.reset();
          while(m_RequestTokens.hasMoreElements() && l.hasMoreElements() && matching)
          {
             str = (String) l.currentElement();
             if(m_RequestTokens.currentElement() instanceof MatchEntry)
             {
                me = (MatchEntry) m_RequestTokens.currentElement();
                me.SetString("");
                m_RequestTokens.nextElement();
             }
             else {
                mstr = (String) m_RequestTokens.currentElement();
                if(mstr.equalsIgnoreCase(str)) {
                   l.nextElement();
                   m_RequestTokens.nextElement();
                   me=null;
                }
```

```
               else {
                  if(me != null) {
                     me.AddString(" " + str);
                     l.nextElement();
                  }
                  else matching = false;
               }
            }
         }
         while(m_RequestTokens.hasMoreElements()) {
            if(!(m_RequestTokens.currentElement() instanceof MatchEntry))
                 matching = false;
            m_RequestTokens.nextElement();
         }
         if(l.hasMoreElements() && me==null) matching = false;

         while(matching && l.hasMoreElements() && me!=null)
         {
            str = (String) l.currentElement();
            me.AddString(" " +str);
            l.nextElement();
         }
       }
      catch(Exception e)
      {
       TextOutput.TextOut("Caught Exception in MatchList: " + e);
      }
      return matching;
   }

/** Gibt denjenigen Teil der Regel zurück, der hinter dem Suchmuster und
* dem Antwortsatz steht.
* Dieser wird als Prozeduraufruf der von VRRobot implementierten
* Skriptsprache aufgefaßt.
* @return Der letzte Teil des Regel-Tripels
*/
    public String GetRestString()
    {
      return m_RestString;
    }

/** Beschreibt das Regelobjekt als String.
* @return Die im Konstruktor als Text übergebene Regel-Definition.
*/
    public String toString()
    {
      return m_String;
    }

/** Führt eine Vorverarbeitung der Regel-Definition durch.
* @param input Die Original-Regel
* @return Die verarbeitete Regel
*/
   public String Preprocess(String input)
   {
     if(input.endsWith("\n")) input=input.substring(0,input.length()-1);

     if(input.endsWith("?")) input=input.substring(0,input.length()-1)+" ?";
     if(input.endsWith("!")) input=input.substring(0,input.length()-1)+" !";
     if(input.endsWith(".")) input=input.substring(0,input.length()-1)+" .";
     return input;
   }
}

////////////////////////////////////////////////////////////////////
/** Die Klasse MatchEntry ist für die interne Repräsentation einzelner
* Worte oder auch Wildcards zuständig.
*/
class MatchEntry
{
   public MatchEntry(int num)
   {
      m_Number=num;
      m_MatchString=new String("");
   }
   public String toString()
   {
      return m_MatchString;
   }
```

```
   public synchronized void SetString(String s)
   {
      m_MatchString = Preprocess(s);
   }
   public synchronized void AddString(String s)
   {
      m_MatchString += Preprocess(s);
   }
   public String Preprocess(String s)
   {
      String ret=new String();
      String tok;
      StringTokenizer t = new StringTokenizer(s," ");
      while(t.hasMoreElements())
      {
         tok=t.nextToken();
         tok=tok.replace('!',' ');
         tok=tok.replace('?',' ');
         tok=tok.replace(',',' ');
         tok=tok.replace(';',' ');
         tok=tok.replace(':',' ');
         tok=tok.replace('.',' ').trim();
         if(     tok.equalsIgnoreCase("mein")) ret += "dein ";
         else if(tok.equalsIgnoreCase("meine")) ret += "deine ";
         else if(tok.equalsIgnoreCase("meinen")) ret += "deinen ";
         else if(tok.equalsIgnoreCase("meins")) ret += "deins ";
         else if(tok.equalsIgnoreCase("meines")) ret += "deines ";
         else if(tok.equalsIgnoreCase("dein")) ret += "mein ";
         else if(tok.equalsIgnoreCase("deine")) ret += "meine ";
         else if(tok.equalsIgnoreCase("deinen")) ret += "meinen ";
         else if(tok.equalsIgnoreCase("deins")) ret += "meins ";
         else if(tok.equalsIgnoreCase("deines")) ret += "meines ";
         else if(tok.equalsIgnoreCase("mich")) ret += "dich ";
         else if(tok.equalsIgnoreCase("dich")) ret += "mich ";
         else if(tok.equalsIgnoreCase("dir")) ret += "mir ";
         else if(tok.equalsIgnoreCase("mir")) ret += "dir ";
         else if(tok.equalsIgnoreCase("ich")) ret += "du ";
         else if(tok.equalsIgnoreCase("du")) ret += "ich ";
         else if(tok.equalsIgnoreCase("meinetwegen")) ret += "deinetwegen ";
         else if(tok.equalsIgnoreCase("deinetwegen")) ret += "meinetwegen ";

         else ret += tok+" ";
      }
      return ret;
   }

   public int GetNumber()
   {
      return m_Number;
   }
   int m_Number;
   String m_MatchString;
}
```

3.9 Die Klasse FileIO

```
/*********************************************************************
* @(#)FileIO.java1.0 97/06/23 Hauke Ernst
*/
package VRRobot;

import java.awt.*;
import java.util.*;
import java.io.*;
import java.lang.*;

import VRMLInterface.TextOutput;
import VRMLInterface.ResultProcessor;

/**
* Klasse zur Bereitstellung elementarer Dateioperationen.
*
*
* @author  Hauke Ernst
* @version 1.0
```

```
*/
public class FileIO
{
/** Schreibt einen String in eine Datei (überschreibend).
* @param filename Der Dateiname.
* @param s Der zu schreibende String.
* @see #AppendToFile
* @see #ReadFromFile
*/
   static public void SaveToFile(String filename, String s)
   {
      try
      {
       filename=Expression.ReplaceSubString(filename,"/",File.separator);
           filename=Expression.ReplaceSubString(filename,"\\",File.separator);
         FileOutputStream fout=null;
         fout = new FileOutputStream(filename);
         if(fout == null) return;
         PrintStream pout = new PrintStream(fout);
         pout.print(s);
         pout.close();
         fout.close();
      }
      catch(Exception e)
      {
         TextOutput.TextOut("Caught Exception in SaveToFile: " + e);
      }
   }

/** Schreibt einen String in eine Datei (überschreibend).
* @param filename Der Dateiname.
* @param path Verzeichnis, in dem die Datei gesucht werden soll, wenn
* sie nicht im aktuellen Verzeichnis gefunden werden konnte
* @param s Der zu schreibende String.
* @see #AppendToFile
* @see #ReadFromFile
*/
   static public void SaveToFile(String path,String filename, String s)
   {
      try
      {
         FileOutputStream fout=null;
         try {
               filename=Expression.ReplaceSubString(filename,"/",File.separator);
               filename=Expression.ReplaceSubString(filename,"\\",File.separator);
            fout = new FileOutputStream(filename);
         }
         catch(Exception e1)
         {
               System.out.println("Speichern in "+filename+" fehlgeschlagen");
            String url=(path+ "\\"+filename).replace(' ','\\');
            try {
               fout = new FileOutputStream(url);
            }
            catch(Exception e2)
            {
                  System.out.println("Speichern in "+url+" fehlgeschlagen");
               if(url.startsWith("file:")) {
                  int j=0;
                  for(j = 6; j<url.length()&&(url.charAt(j)=='\\'||url.charAt(j)==' '); j++);
                  url = url.substring(j,url.length());
               }
               try {
                 fout = new FileOutputStream(url);
               }
               catch(Exception e3)
               {
                      System.out.println("Speichern in "+url+" fehlgeschlagen");
                  url=url.replace('|',':');
                  fout = new FileOutputStream(url);
               }
            }
         }
         if(fout == null) return;
         PrintStream pout = new PrintStream(fout);
         pout.print(s);
         pout.close();
         fout.close();
      }
```

```
      catch(Exception e)
      {
         TextOutput.TextOut("Caught Exception in SaveToFile of file "+filename+": " + e);
      }
   }

/** Schreibt einen String in eine Datei (anhängend).
* @param filename Der Dateiname.
* @param s Der zu schreibende String.
* @see #SaveToFile
* @see #ReadFromFile
*/
   static public void AppendToFile(String filename, String s)
   {
      try
      {
         String oldcontent=ReadFromFile(filename);
         if(oldcontent!=null) s= oldcontent+s;
         SaveToFile(filename,s);
      }
      catch(Exception e)
      {
         TextOutput.TextOut("Caught Exception in AppendToFile: " + e);
      }
   }

/** Liest einen String aus einer Datei.
* @param filename Der Dateiname.
* @return Der gelesene String
* @see #AppendToFile
* @see #SaveToFile
*/
   static public String ReadFromFile(String filename)
   {
      try
      {
          filename=Expression.ReplaceSubString(filename,"/",File.separator);
          filename=Expression.ReplaceSubString(filename,"\\",File.separator);
         FileInputStream fin=null;
         fin = new FileInputStream(filename);
         if(fin == null) return null;
         DataInputStream in = new DataInputStream(fin);
         if(in==null) return null;
         String s= new String();
         String line;
         for(line=in.readLine(); line != null;line=in.readLine())
            s+=line+"\n";
         in.close();
         fin.close();
         return s;
      }
      catch(Exception e4)
      {
         TextOutput.TextOut("Caught Exception in ReadFromFile: " + e4);
      }
      return null;
   }

/** Liest einen String aus einer Datei.
* @param filename Der Dateiname.
* @param path Verzeichnis, in dem die Datei gesucht werden soll, wenn
* sie nicht im aktuellen Verzeichnis gefunden werden konnte
* @return Der gelesene String
* @see #AppendToFile
* @see #SaveToFile
*/
   static public String ReadFromFile(String path, String filename)
   {
      try
      {
         FileInputStream fin=null;
         try {
               filename=Expression.ReplaceSubString(filename,"/",File.separator);
               filename=Expression.ReplaceSubString(filename,"\\",File.separator);
            fin = new FileInputStream(filename);
         }
         catch(Exception e1)
         {
```

```
            String url = (path+File.separator +filename).replace(' ','\\' );
            try {
               fin = new FileInputStream(url);
            }
            catch(Exception e2)
            {
               if(url.startsWith("file:")) {
                  int j=0;
                  for(j = 6; j<url.length()&&(url.charAt(j)=='\\'||url.charAt(j)==' '); j++);
                  url = url.substring(j,url.length());
               }
               try {
                 fin = new FileInputStream(url);
               }
               catch(Exception e3)
               {
                  url=url.replace('|',':');
                  fin = new FileInputStream(url);
               }
            }
         }
         if(fin == null) return null;
         DataInputStream in = new DataInputStream(fin);
         if(in==null) return null;
         String s= new String();
         String line;
         for(line=in.readLine(); line != null;line=in.readLine())
            s+=line+"\n";
         in.close();
         fin.close();
         return s;
      }
      catch(Exception e4)
      {
         TextOutput.TextOut("Caught Exception in ReadFromFile: " + e4);
      }
      return null;
   }

/** Verkürzt die übergebene URL auf die Pfadangabe.
* @param url Eine URL
* @return Der Teilstring der URL bis zum letzten '/'-Zeichen.
*/
   static public String GetPath(String url)
   {
      try {
         int i=0;
         for(i = url.length()-1; i>0&&url.charAt(i)!='/'; i--);
          url = url.substring(0,i);
          url=Expression.ReplaceSubString(url,"/",File.separator);
          url=Expression.ReplaceSubString(url,"\\",File.separator);
         return url;
      }
      catch(Exception e)
      {
         TextOutput.TextOut("Caught Exception in GetPath: " + e);
      }
      return null;
   }
}
```

3.10 Die Klasse MainDialog

```
/**********************************************************************
* @(#)MainDialog.java1.0 97/06/23 Hauke Ernst
*/
package VRRobot;

import java.awt.*;
import VRMLInterface.ResultProcessor;
import VRRobot.Expression;
/**
* Die Klasse MainDialog stellt den Hauptdialog für
```

```
* die Interaktion mit VRRobots zur Verfügung.
* Dieser enthält zwei getrennte Textbereiche: in einem
* werden die Aussprüche des VRRobots ausgegeben, der andere
* gibt dem Benutzer die Möglichkeit, selber Texte einzugeben und
* diese dem VRRobot zu senden. Zu diesem Zweck ist eine Schaltfläche
* ("Antwort") vorhanden. Das Betätigen der EINGABE-Taste
* hat dieselbe Wirkung.
* Zusätzlich gibt es die Schaltflächen "Konfigurieren" und "Schliessen".
* Die Behandlung der einzelnen Dialogereignisse wird dem
* ResultProcessor - in diesem Fall der VRRobot - überlassen, indem
* dessen processResult-Funktion aufgerufen wird.
* @see vrrobot
*
* @author  Hauke Ernst
* @version 1.0
*/
public class MainDialog extends Frame
{
  String m_RobotTextOut=new String();
  String m_UserTextOut=new String();
private TextAreaWithLineBreaks m_TextAreaIn;
  private TextAreaWithLineBreaks m_TextAreaOut;
  private ResultProcessor m_ResultProcessor;
  private Font m_Font;

  public MainDialog(ResultProcessor creator)
   {
      m_Font = new Font("Helvetica", Font.BOLD, 14);
      setFont(m_Font);
      setTitle("VRRobot Dialog");
      Panel p = new Panel();
      p.setLayout(new FlowLayout());
      p.add(new Button("Antwort"));
      p.add(new Button("Konfigurieren"));
      p.add(new Button("Schließen"));
      add("North", p);
      m_TextAreaIn = new TextAreaWithLineBreaks(3,3);
      m_TextAreaIn.setBackground(Color.orange);
      add("South", m_TextAreaIn);
      m_TextAreaIn.requestFocus();
      m_TextAreaOut = new TextAreaWithLineBreaks(3,3);
      m_TextAreaOut.setEditable(false);
      m_TextAreaOut.setBackground(Color.lightGray);
      add("Center", m_TextAreaOut);
      m_ResultProcessor=creator;
   }

   public boolean action(Event evt, Object arg)
   {
      ((ResultProcessor)m_ResultProcessor).processResult(this,arg);
      return true;
   }

   public boolean keyDown(Event evt,int key)
   {
      if(key=='\n')
         ((ResultProcessor)m_ResultProcessor).processResult(this,"Antwort");
      return super.keyDown(evt,key);;
   }

   public String TextIn()
   {
      return m_TextAreaIn.getText();
   }

   public synchronized void RobotTextOut(String name, String s)
   {
      m_RobotTextOut+=s+"\n";
      TextOut(name+" sagt :> "+s);
   }

   public synchronized void UserTextOut(String s)
   {
      s = Expression.ReplaceSubString(s,"\n","");
      m_UserTextOut+=s+"\n";
      TextOut("Du sagst :> "+s);
   }

   public void TextOut(String s)
```

```
   {
      m_TextAreaOut.appendText(s+"\n");
      m_TextAreaIn.requestFocus();
   }

   public String RobotTextOut()
   {
      return m_RobotTextOut;
   }

   public String UserTextOut()
   {
      return m_UserTextOut;
   }

   public void TextIn(String s)
   {
      m_TextAreaIn.setText(s);
   }
}

class TextAreaWithLineBreaks extends TextArea
{
int m_cols;
int m_rows;
FontMetrics m_FontMetrics;
int m_Width;

 public TextAreaWithLineBreaks(int rows,int cols)
  {
      super(rows,cols);
      m_cols=cols;
      m_rows=rows;
      m_Width=cols;
  }

 public String InsertLineBreaks(String orig_str)
  {
      m_FontMetrics = getFontMetrics(getFont());
      m_Width=size().width-minimumSize().width;
      String new_str = new String();
      String str = new String();
      for(int i=0; i<orig_str.length(); i++) {
         if(orig_str.charAt(i)=='\n') str= new String();
         else if(m_FontMetrics.stringWidth(str) > m_Width) {new_str += '\n'; str=new String(); }
         else str += orig_str.charAt(i);
         new_str += orig_str.charAt(i);
        }
      return new_str;
  }

 public void appendText(String text)
  {
    String old_text=super.getText();
    if( old_text ==null
     || old_text.length()<=0
     || old_text.charAt(old_text.length()-1)=='\n'
     || text == null
     || text.length()<=0
     || text.charAt(0) =='\n')
       super.appendText(InsertLineBreaks(text));
       else {
         super.setText(InsertLineBreaks(old_text+text));
        }
      select(super.getText().length(), super.getText().length());
  }

 public void setText(String text)
  {
     super.setText(InsertLineBreaks(text));
  }

/* zu langsam
   public boolean keyDown(Event evt,int key)
   {
      boolean ret=super.keyDown(evt,key);
      if(evt.id==401) {
         int sel_start=getSelectionStart();
         int sel_end=getSelectionEnd();
```

```
         String orig_str=super.getText();
         String new_str =InsertLineBreaks(orig_str);
         super.setText(new_str);
         int more_chars= new_str.length()-orig_str.length();
         select(sel_start+more_chars,sel_end+more_chars);
        }
      return ret;
   }
*/
}
```

3.11 Die Klasse KonfigDialog

```
/*********************************************************************
* @(#)KonfigDialog.java1.0 97/06/23 Hauke Ernst
*/
package VRRobot;


import java.awt.*;
import VRMLInterface.ResultProcessor;

/** Dialogklasse zur Konfiguration des Verhaltens von
* VRRobots. Er erlaubt das Laden, Speichern von
* Definitions-Dateien (*.rcf) sowie das Aufrufen
* der Dialoge KonfigMotion und KonfigAnswers.
* @see KonfigMotion
* @see KonfigAnswers
*
* @author  Hauke Ernst
* @version 1.0
*/
public class KonfigDialog extends Frame
{
   public KonfigDialog(ResultProcessor creator)
   {
      Font myfont = new Font("Helvetica", Font.BOLD, 14);
      setFont(myfont);
      setTitle("Konfigurieren");
      Panel p = new Panel();
      p.setLayout(new FlowLayout());
      p.add(new Button("Schließen"));
      p.add(new Button("Laden"));
      p.add(new Button("Speichern"));
      add("South", p);
      p = new Panel();
      p.setLayout(new FlowLayout());
      p.add(new Button("Bewegungseinstellungen"));
      p.add(new Button("Dialogeinstellungen"));
      add("North", p);
      m_ResultProcessor=creator;
   }

   public boolean handleEvent(Event evt)
   {
      return super.handleEvent(evt);
   }

   public boolean action(Event evt, Object arg)
   {
      ((ResultProcessor)m_ResultProcessor).processResult(this,arg);
      return true;
   }

      private ResultProcessor m_ResultProcessor;


}
```

3.12 Die Klasse KonfigMotion

```
/*****************************************************************
* @(#)KonfigMotion.java1.0 97/06/23 Hauke Ernst
*/
package VRRobot;


import java.awt.*;
import VRMLInterface.ResultProcessor;

/** Dialogklasse zur Eingabe von Handlungsplänen, die auf der
* Scriptsprache von VRRobot beruhen.
* Da der eingegebene Text durch den Preprozessor von VRRobot
* vorverarbeitet wird, können Kommentare und "import"-Anweisungen
* verwendet werden.
* @see vrrobot
*
* @author  Hauke Ernst
* @version 1.0
*/
public class KonfigMotion extends Frame
{  public KonfigMotion(ResultProcessor creator)
   {
      Font myfont = new Font("Helvetica", Font.BOLD, 14);
      setFont(myfont);
      setTitle("Bewegung konfigurieren");
      Panel p = new Panel();
      p.setLayout(new FlowLayout());
      p.add(new Button("Schließen"));

      add("South", p);
      ta = new TextArea(8, 40);
      add("Center", ta);
      m_ResultProcessor=creator;
   }

   public boolean handleEvent(Event evt)
   {
      return super.handleEvent(evt);
   }

   public boolean action(Event evt, Object arg)
   {
      ((ResultProcessor)m_ResultProcessor).processResult(this,arg);
      return true;
   }
   public String getText()
   {
      return ta.getText();
   }
   public void setText(String s)
   {
      ta.setText(s);
   }


   private TextArea ta;
   private ResultProcessor m_ResultProcessor;


}
```

3.13 Die Klasse KonfigAnswers

```
/*****************************************************************
* @(#)KonfigAnswers.java1.0 97/06/23 Hauke Ernst
*/
package VRRobot;

import java.awt.*;
import VRMLInterface.ResultProcessor;

/** Dialogklasse zur Konfiguration des Antwortverhaltens
```

```
* eines VRRobot. Es enthält ein Textfeld, in dem
* die Tripel ("<Suchmuster>" "<Antwort>" <Befehl>) für das
* Pattern-Matching eingegeben werden können.
* Da der eingegebene Text durch den Preprozessor von VRRobot
* vorverarbeitet wird, können Kommentare und "import"-Anweisungen
* verwendet werden.
* @see Answer
* @see vrrobot
*
*
* @author  Hauke Ernst
* @version 1.0
*/
public class KonfigAnswers extends Frame
{
   private TextArea ta;
   private ResultProcessor m_ResultProcessor;

   public KonfigAnswers(ResultProcessor creator)
   {
      Font myfont = new Font("Helvetica", Font.BOLD, 14);
      setFont(myfont);
      setTitle("Dialog konfigurieren");
      Panel p = new Panel();
      p.setLayout(new FlowLayout());
      p.add(new Button("Schließen"));
      add("South", p);
      ta = new TextArea(8, 40);
      add("Center", ta);
      m_ResultProcessor=creator;
   }

   public boolean handleEvent(Event evt)
   {
      return super.handleEvent(evt);
   }

   public boolean action(Event evt, Object arg)
   {
      ((ResultProcessor)m_ResultProcessor).processResult(this,arg);
      return true;
   }

   public String getText()
   {
      return ta.getText();
   }

   public void setText(String s)
   {
      ta.setText(s);
   }
}
```

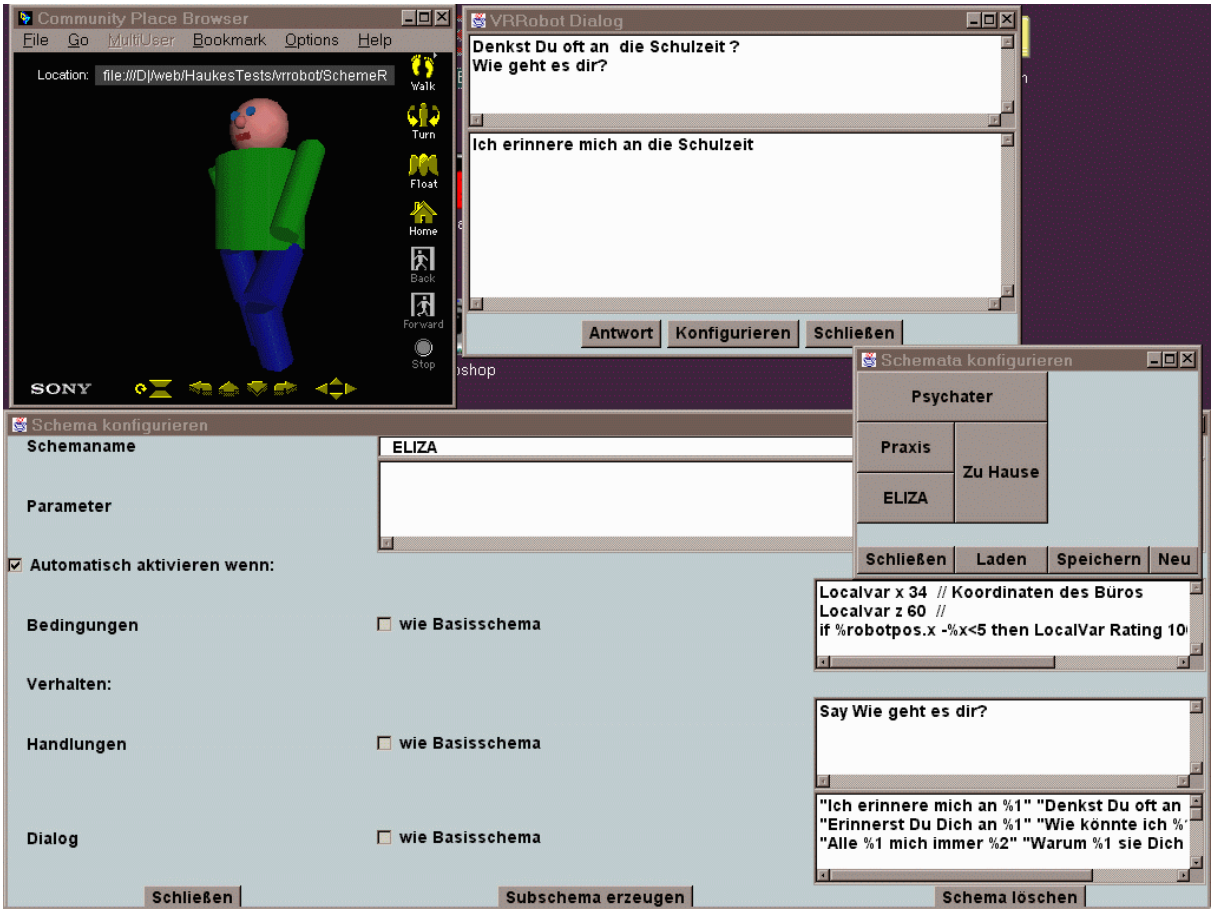
3.14 Die Schnittstelle Answering

```
/**********************************************************************
* @(#)Answering.java1.0 97/06/23 Hauke Ernst
*/
package VRRobot;

/** Umfaßt Klassen, die einer Texteingabe einen
* Antwortsatz zuordnen können.
*
*
*
* @author  Hauke Ernst
* @version 1.0
*/
public interface Answering
{
   public String Answer(String s);
}
```

4 Das Modul SchemeRobot

4.1 Aufgabenstellung

Die Klasse SchemeRobot erweitert die Oberklasse VRRobot um die Möglichkeit, weitaus komplexere Verhaltensweisen auf der Grundlage der Schematheorie zu beschreiben. Zu diesem Zweck wird insbesondere eine Verwaltung von Schemata mit Aktivierungsreizen und Handlungsplänen implementiert. Es wird immer dasjenige Schema automatisch aktiviert und ausgeführt, dessen Aktivierungsbedingungen am besten zu den Umgebungsvariablen wie Benutzernähe, Sichtbarkeit durch den Benutzer, aktueller Standort und dem bisherigen Gesprächsverlauf passen. Ein Schema kann jedoch die Aktivierung eines anderen Schemas explizit herbeiführen, auch wenn dessen Aktivierungsbedingungen in dieser Situation nicht zu einer automatischen Aktivierung geführt hätten. In einem solchen Fall wird nach Abarbeitung des Handlungsplanes des aufgerufenen Schemas das aufrufende Schema reaktiviert und dessen Handlungsplan weiter ausgeführt.
Jedem Schema ist eine Hemmung zugeordnet. Diese kann im Verlauf des Handlungsplanes variieren und bestimmt, inwieweit eine Unterbrechung des Handlungsplanes durch die Aktivierung eines anderen Schemas erfolgen kann. Durch Anheben der Hemmung werden andere Schemata ausgegrenzt, durch Absenken bevorzugt.



4.2 Funktionsbeschreibung

Die Klasse SchemeRobot besitzt zunächst einen von der Oberklasse abweichenden Konfigurationsdialog. In diesem wird dem Benutzer insbesondere eine hierarchische Darstellung der aktuell geladenen Schemata präsentiert. Aus dieser Hierarchie kann ein konkretes Schema ausgewählt werden, woraufhin sich ein weiterer Konfigurationsdialog öffnet, in dem das Schema editiert werden kann. Hier können der Schemaname, mögliche Parameter und die Aktivierungsbedingungen benannt, sowie das schemaspezifische Verhalten definiert werden. Außerdem kann dieses Schema (inklusive Unterschemata) gelöscht und Unterschemata können erzeugt werden.

Die in der aktuellen Implementierung verwendeten Aktivierungsbedingungen werden durch eine Bewertungsprozedur festgelegt. Nach der Ausführung dieser Funktion, die im Konfigurationsdialog jedes Schemas eingegeben wird, soll die Variable *Rating* die Bewertung der Situation enthalten. Die Aktivierung eines Schemas erfolgt dann, wenn dieses Schema aufgrund der Aktivierungsbedingungen als das zur aktuellen Situation am besten passende erkannt wurde (Rating ist am höchsten) oder wenn es explizit von einem anderen Schema aufgerufen wurde. Für solche Aufrufe wird in dieser

Klasse der Befehlsumfang der Skriptsprache zur Beschreibung von Handlungsplänen um den im Folgenden beschriebenen Befehl „call“ erweitert:

Call <Schemaname> [Parameterliste]

⇒ Aktiviert das Schema <Schemaname> und führt seinen Handlungsplan aus, unabhängig von dessen Aktivierungsbedingungen. Nach dem Ende des Handlungsplanes von <Schemaname> wird das aktuelle Schema reaktiviert. Optional können dem aufzurufenden Schema Parameter übergeben werden, und zwar in der Form:

<Parametername1> <Wert1> <Parametername2> <Wert2> ....

Für nicht übergebene Parameter wird der im aufzurufenden Schema definierte Standardwert angenommen.

Beispiel:

```
Call Bestellen Bestellung Ein Bier mit Honig bitte
Call LaufeimRechteck Breite 5 Länge 6
```

Der Befehl „Inhibit“ verändert die Hemmung bzgl. anderer Schemata:

Inhibit <Wert> [<Schemaname>]

⇒ Hebt bzw. senkt den Hemmungswert des Schemas <Schemaname> um <Wert>. Eine Erhöhung des Wertes bedeutet für ein Schema, daß jeweils andere Schemata bei der Aktivierung benachteiligt werden. Wird kein Schemaname angegeben, wird als Bezugsschema das gerade aktive angenommen.

Beispiel:

```
Inhibit 10 LaufeimRechteck
Inhibit -1
```

Die einzelnen Eigenschaften des Schemas können jeweils auch von der Basisklasse geerbt werden. Bei Schemata, die nur für die explizite Aktivierung durch andere Schemata konzipiert werden sollen, kann die automatische Aktivierung durch den Algorithmus zur Schemaauswahl abgeschaltet werden.

4.3 Modularisierung

Das Modul SchemeRobot besteht aus mehreren Klassen und Schnittstellen, die im folgenden einzeln beschrieben werden.

4.4 Die Klasse SchemeRobot

```
/*********************************************************************
* @(#)SchemeRobot.java 1.0 97/06/23 Hauke Ernst
*/
package SchemeRobot;


import java.awt.*;
import java.util.*;
import java.io.*;
import java.lang.*;

import VRMLInterface.*;
import VRRobot.*;
import SchemeRobot.*;

/**
* Klasse zur Steuerung von automatisierten Charakteren.
* Sie erweitert die Möglichkeiten zur Verhaltenssteuerung
* der Klasse VRRobot auf der Basis des Schema-Begriffes und
* demonstriert die Programmierschnittstelle von VRRobot.
* Mit Hilfe eines Konfigurationsfensters kann eine Hierarchie
* von Schemata hergestellt werden. Jedem dieser Schemata ist
* eigenes Verhalten (definiert in der Skriptsprache von VRRobot)
* sowie eine Bewertungsfunktion der aktuellen Situation zugeordnet.
* Diese Aktivierungbedingung sowie das Bewegungsverhalten und
* das Dialogverhalten können seperat vom "Elternschema" geerbt werden.
* Ein Thread führt ständig die Bewertungfunktionen aller Schemata
* aus und aktiviert das am höchsten bewertete.
* Außerdem wird die Skriptsprache um Mechanismen erweitert, um
* die Bewertung von Schemata positiv bzw. negativ zu manipulieren
* und um Schemata explizit zu aktivieren. Dabei ist die
* Übergabe von Parametern möglich.
*
*
*
*
* @author  Hauke Ernst
* @version 1.0
* @see     vrrobot
*/
public class SchemeRobot extends vrrobot
 implements ResultProcessor, Runnable, SchemeContainer
 {
   /** Der Thread, der im Hintergrund mit niedrigster Priorität
   * die Aktivierungbedingungen der Schemata überprüft */
   public Thread runner=null;
   Scheme m_BaseScheme;
   KonfigSchemeDialog m_KonfigSchemeDialog;
   SchemeDialog m_SchemeDialog;

/** Standard-Konstruktor */
   public SchemeRobot()   {
      super();
   }

 /**
 * Wird von der Basisklasse nach dem Erzeugen der Klasse aufgerufen.
 * Aufgaben sind
 *    Dateninitialisierung und
 *    Einlesen von Konfigurationsdaten aus der Datei, deren Namen in dem Feld
 *    "KonfigFile" festgelegt sein sollte. Existiert dieses Feld nicht, wird
 *    als Name "Klassenname.rcf" angenommen. Außerdem wird die
 *    zyklische Abarbeitung der Schema-Bewertungsfunktionen mit Hilfe der
 *    Funktion "CheckScheme" angestoßen.
 *    @see #CheckScheme
 */
   public synchronized void initialize()
   {
      super.initialize();
      try {
         m_KonfigSchemeDialog = null;
         m_SchemeDialog=null;
         CheckScheme();
      }
      catch(Exception e)
      {
```

```
      TextOutput.TextOut("Caught Exception in SchemaRobot script initialisation: " + e);
    }
  }

/** Wird aufgerufen, wenn der Browser das Java-Objekt freigibt.
 */
public void shutdown()
  {
   stop();
  }

 /** Beginnt die zyklische Abarbeitung der Schema-Bewertungsfunktionen durch
 *  einen Aufruf der Funktion "start".
 *    @see #start
 */
 public void CheckScheme()
 {
   try {
      start();
   }
  catch(Exception ex)
  {
   TextOutput.TextOut("Caught Exception in SchemeRobot.CheckScheme: " + ex);
  }
 }

 /** Der Thread "runner" wird erzeugt und gestartet.
 */
 public void start()
 {
   if(runner==null) {
      runner=new Thread(this);
      runner.start();
   }
 }

 /** Der Thread "runner" wird gestoppt.
 * @see #start
 */
 public void stop()
 {
   if(runner!=null && runner.isAlive())
      runner.stop();
   runner=null;
 }

 /** Wird vom Thread "runner" nach seinem Start aufgerufen.
 * Diese Funktion ist Teil des Interface "Runnable" und
 * enthält die Arbeit des Threads, in diesem Fall die ständige
 * Bewertung aller Schemata und die Aktivierung des am höchsten
 * bewerteten Schemas. Zu diesem Zweck wird die Funktion
 * "CheckScheme" der Klasse "Scheme" aufgerufen.
 * @see Scheme#CheckScheme
 *
 */
 public void run()
 {
   try {
         while(runner!=null) {
       m_BaseScheme.CheckScheme();
            try { Thread.sleep(10); }
            catch(InterruptedException e) {}
           }
   }
  catch(Exception ex)
  {
   TextOutput.TextOut("Caught Exception in SchemeRobot.run: " + ex);
  }
   runner=null;
 }

 /** Ruft einen speziellen Dialog zur Schemakonfiguration auf.
 * Dieser stellt die Schemahierarchie dar und erlaubt die
 * Auswahl einzelner Schemata, die dann in einem weiteren Dialog
 * im Detail angezeigt werden.
 * @see KonfigSchemeDialog
 * @see SchemeDialog
 */
public void KonfigDialog()
```

```
   {
      try {
         if(m_KonfigSchemeDialog==null)
         {
             m_KonfigSchemeDialog = new KonfigSchemeDialog(this,m_BaseScheme);
             m_KonfigSchemeDialog.resize(400,200);
         }
         else m_KonfigSchemeDialog.SetSchemes(m_BaseScheme);
         m_KonfigSchemeDialog.requestFocus();
         m_KonfigSchemeDialog.show();
      }
     catch(Exception e)
     {
      TextOutput.TextOut("Caught Exception in SchemeRobot.KonfigDialog: " + e);
     }
   }

    /** Sucht in der Schemahierarchie nach dem benannten Schema.
    * @see Scheme#GetNode
    * @param name Name des gesuchten Schemas
    * @return Das gesuchte Schema. Falls es kein Schema des
    * angegebenen Namens existiert, ist das Ergebnis "null".
    */
   public Scheme FindScheme(String name)
   {
      return m_BaseScheme.GetNode(name);
   }

    /** Ruft einen speziellen Dialog für das benannte Schema auf.
    * @see SchemeDialog
    * @param name Name des darzustellenden Schemas
    */
   public void SchemeDialog(String name)
   {
      try {
         Scheme s = FindScheme(name);
         if(s != null) {
            if(m_SchemeDialog == null) {
               m_SchemeDialog = new SchemeDialog(this,s);
               m_SchemeDialog.resize(700, 500);
            }
            else m_SchemeDialog.SetScheme(s);
            m_SchemeDialog.requestFocus();
            m_SchemeDialog.show();
         }
         else TextOutput.TextOut("Schema nicht gefunden");
      }
     catch(Exception e)
     {
      TextOutput.TextOut("Caught Exception in SchemeDialog: " + e);
     }
   }

/** Behandlungsfunktion für Dialogereignisse.
* @param source Der aufrufende Dialog.
* @param obj Das betreffende Dialogelement.
* @see #HandleKonfigSchemeDialog
* @see #HandleSchemeDialog
*/
   public void HandleDialog(Frame source, Object obj)
   {
     try {
      if(source instanceof KonfigSchemeDialog)
      {
         HandleKonfigSchemeDialog(source,obj);
      }
      else if(source instanceof SchemeDialog)
      {
         HandleSchemeDialog(source,obj);
      }
      else super.HandleDialog(source,obj);
    }
     catch(Exception e)
     {
      TextOutput.TextOut("Caught Exception in HandleDialog: " + e);
     }
   }

/** Behandlungsfunktion für Dialogereignisse aus der Dialogklasse KonfigSchemeDialog.
```

```
* @param source Der aufrufende Dialog.
* @param obj Das betreffende Dialogelement.
* @see #KonfigDialog
* @see KonfigSchemeDialog
*/
   public void HandleKonfigSchemeDialog(Frame source, Object obj)
   {
      try {
         if(((String)obj).equalsIgnoreCase("Schließen"))
         {
            source.hide();
         }
         else if(((String)obj).equalsIgnoreCase("Laden"))
         {
            FileDialog d = new FileDialog(source,"Konfiguration laden",
                           FileDialog.LOAD);
//            d.setFilenameFilter((FilenameFilter)"*.rcf");
            d.setDirectory(".");
            d.setFile(getClass().getName()+".rcf");
            d.show();
            String filename = d.getFile();
            if(filename!=null)
               ReadKonfig(d.getDirectory()+File.separator+filename);
               KonfigDialog();
               CheckScheme();
         }
         else if(((String)obj).equalsIgnoreCase("Neu"))
         {
               m_BaseScheme=new Scheme(this,null,"Basisschema");
               KonfigDialog();
               CheckScheme();
         }
         else if(((String)obj).equalsIgnoreCase("Speichern"))
         {
            FileDialog d = new FileDialog(source,"Konfiguration speichern",
                           FileDialog.SAVE);
//            d.setFileNameFilter("*.rcf");
            d.setDirectory(GetWorldPath());
            d.setFile(getClass().getName()+".rcf");
            d.show();
            String filename = d.getFile();
            if(filename!=null)
               SaveKonfig(d.getDirectory()+File.separator+filename);
         }
         else {
            SchemeDialog((String)obj);
         }
      }
      catch (Exception e){
         TextOutput.TextOut("Caught Exception in HandleKonfigDialog: " + e);
      }
   }

/** Behandlungsfunktion für Dialogereignisse aus der Dialogklasse SchemeDialog.
* @param source Der aufrufende Dialog.
* @param obj Das betreffende Dialogelement.
* @see #SchemeDialog
* @see SchemeDialog
*/
   public void HandleSchemeDialog(Frame source, Object obj)
   {
      try {
         Scheme scheme = ((SchemeDialog)source).GetScheme();
         if(obj instanceof String) {
            if(((String)obj).equalsIgnoreCase("Schließen"))
            {
               source.hide();
            }
            else if(((String)obj).equalsIgnoreCase("Schema löschen"))
            {
               Scheme parent=scheme.GetParent();
               if(parent != null && scheme != null) {
                  source.hide();
                  LinkedList l = parent.GetChildren();
                  l.reset();
                  while(l.hasMoreElements()) {
                     if(((Scheme) l.currentElement()).GetName().equals(scheme.GetName()))
                        l.remove();
                     else l.nextElement();
```

```
                  }
               }
               KonfigDialog();
               CheckScheme();
            }
            else if(((String)obj).equalsIgnoreCase("Subschema erzeugen"))
            {
               int nr=1;
               String newschemename;
               do {
                   newschemename=new String("Sub"+(nr++) +" "+scheme.toString());
               } while(FindScheme(newschemename)!=null);
               Scheme newscheme = new  Scheme(this,scheme,newschemename);
               scheme.AddSubScheme(newscheme);
//               SchemeDialog("Subschema von " + scheme.toString());
            }
            KonfigDialog();
            CheckScheme();
         }
      }
      catch (Exception e){
         TextOutput.TextOut("Caught Exception in HandleSchemeDialog: " + e);
      }
   }

/** Speichern des aktuellen Handlungsplanes in eine Datei.
* @param filename Der Dateiname
* @see Scheme#GetSaveString
* @see FileIO#SaveToFile
*/
   public void SaveKonfig(String filename)
   {
      try {
      FileIO.SaveToFile(GetWorldPath(),filename,m_BaseScheme.GetSaveString());
      }
      catch(Exception e)
      {
         TextOutput.TextOut("Caught Exception in SchemeRobot.SaveKonfig: " + e);
      }
   }

/** Lesen eines Handlungsplanes aus einer Datei.
* @param filename Der Dateiname
* @see Scheme#ReadSaveString
* @see FileIO#ReadFromFile
*/
   public synchronized void ReadKonfig(String filename)
   {
      try {
          stop();
          Scheme.Reset();
         String str = FileIO.ReadFromFile(GetWorldPath(),filename);
         m_BaseScheme = new Scheme(this,null);
         m_BaseScheme.ReadSaveString(str);
          CallInitProc();
         CheckScheme();
      }
      catch(Exception e)
      {
         TextOutput.TextOut("Caught Exception in SchemeRobot.ReadKonfig: " + e);
      }
   }

/** Interpretation und Ausführung eines Befehls der Skriptsprache.
* Der Befehlsumfang der Elternklasse "VRRobot" wird dabei um einige
* schemaspezifische Befehle erweitert.
* @param s Der auszuführende Befehl
* @return Gibt zurück, wieviele Ticks der Befehl zur Ausführung braucht.
* Nach Ablauf dieser Ticks (EventIn "move") kann der nächste Befehl ausgeführt
* werden.
*/
   public int Apply(String s)
   {
      int ret=0;
      try {
         if(s==null || s.length()<=0) return 0;
         int supret = super.Apply(s);
         if(supret>0) return supret;
         ret = supret;
```

```
         StringTokenizer t = new StringTokenizer(s," \t\n\r");
         String Param;
         String Command;
         if(t.hasMoreElements()) {
            Param=new String();
            Command = t.nextToken();
            while(t.hasMoreElements()) {
               Param += t.nextToken() + " ";
            }
            if(Command.equalsIgnoreCase("Call")) ret=Call_Scheme(Param);
            else if(Command.equalsIgnoreCase("Inhibit")) ret = Inhibit(Param);
         }
      }
      catch(Exception e)
      {
         TextOutput.TextOut("Caught Exception in SchemeRobot.Apply: " + e);
      }
      return ret;
   }

/** Realisiert den Befehl "call" zur direkten Aktivierung eines anderen Schemas.
* Ein so aktiviertes Schema kann nicht durch den normalen Bewertungsmechanismus
* unterbrochen werden und wird in jedem Fall bis zum Ende dessen Hauptfunktion
* "VRRobotMain" ausgeführt. Daher sollte diese Funktion nur für sehr kurze
* Handlungen verwendet werden. In den meisten Fällen ist die Verwendung von
* "Inhibit" sauberer.
* <pre>
* Skript-Parameter:
*         (String) -> Schemaname
*         (String) -> Name des 1. Parameter
*         (String) -> Wert des 1. Parameter
*         (String) -> Name des 2. Parameter
*         (String) -> Wert des 2. Parameter
*         ..............................
*
* Für nicht übergebene Parameter wird der im aufzurufenden Schema definierte Standardwert ange-
      nommen.
* Beispiel:
* Call Bestellen Bestellung Ein Bier mit Honig, bitte!
* Call LaufeimRechteck Breite 5 Länge 6
* </pre>
* @see #Inhibit
* @param param Die Parameterliste des Befehls als String
* @return Gibt zurück, wieviele Ticks der Befehl zur Ausführung braucht.
* Nach Ablauf dieser Ticks (EventIn "move") kann der nächste Befehl ausgeführt
* werden.
*/
   public int Call_Scheme(String s)
   {
      StringTokenizer t = new StringTokenizer(s,"| \t\n\r(){}[]");
      String param;
      String name=null;
      param=new String("");
      if(t.hasMoreElements()) {
         name = t.nextToken();
         while(t.hasMoreElements())
            param += t.nextToken() + " ";
         Scheme scheme = FindScheme(name);
         if(scheme!=null) {
            scheme.SetRecallScheme(Scheme.GetAktScheme());
            scheme.SetRecallMotionPos(GetMotionPos());

            scheme.Realize(param);
         }
      }
      return 0;
   }

/** Realisiert den Befehl "Inhibit" zur positiven bzw. negativen Beeinflussung
* eines Schemas.
* <pre>
* Skript-Parameter:
*         (Float)  -> Inhibition, wird zur normalen Bewertung des Schemas addiert
*         (String) -> Names des betreffenden Schemas. Wird kein Name angegeben,
*                     ist das aktuelle Schema gemeint.
* Beispiel:
*   Inhibit 20 ELIZA_Scheme
*
*   oder
```

```
*
*   Tag DelayedDeakivation
*     Wait slides 5
*     Inhibit -1
*   Goto DelayedDeakivation
* </pre>

* @see #Call_Scheme
* @see Scheme#SetInhibition
* @param param Die Parameterliste des Befehls als String
* @return Gibt zurück, wieviele Ticks der Befehl zur Ausführung braucht.
* Nach Ablauf dieser Ticks (EventIn "move") kann der nächste Befehl ausgeführt
* werden.
*/
   public int Inhibit(String s)
   {
      try {
         StringTokenizer t = new StringTokenizer(s,"| \t\n\r");
         Scheme scheme=Scheme.GetAktScheme();
         String schemename=new String();
         float inh = 0;
         if(t.hasMoreElements()) inh = (new Float(t.nextToken())).floatValue();
         int pcount=0;
         while(t.hasMoreElements()) {
            schemename += t.nextToken();
            if(t.hasMoreElements()) schemename+=" ";
            pcount++;
         }
         if(pcount>0) scheme=FindScheme(schemename);
         if(scheme !=null) scheme.SetInhibition(scheme.GetInhibition()+
               inh);
         else TextOutput.TextOut("Fehler in Inhibit Anweisung: Kein Schemabezug");
      }
      catch(Exception e)
      {
         TextOutput.TextOut("Caught Exception in SchemeRobot.Inhibit: " + e);
      }
      return 0;
   }
}
```

4.5 Die Klasse Scheme

```
/*********************************************************************
* @(#)Scheme.java1.0 97/06/23 Hauke Ernst
*/
package SchemeRobot;


import java.awt.*;
import java.io.*;
import java.util.*;
import java.lang.*;
import VRMLInterface.*;
import VRRobot.*;
import SchemeRobot.SchemeContainer;

/** Die Klasse Scheme enthält die Daten und Funktionen,
* die zur Repräsentation eines Schemas im Rahmen
* des SchemeRobots gebraucht werden.
* Ein Schema besteht im wesentlichen aus einer Bewertungsfunktion
* (in der Skriptsprache des VRRobot), die bestimmt,
* inwieweit das Schema zur aktuellen Umwelt/Situation passt,
* einem Handlungsplan, einer Dialogkonfiguration und einigen
* Verwaltungsdaten. So enthält beispielsweise jedes Schema eine
* Liste von Subschemata. Ein Schema kann mit Hilfe des SchemeDialogs
* bearbeitet werden.
* @see SchemeDialog
* @see SchemeRobot
*
* @author  Hauke Ernst
* @version 1.0
*/
public class Scheme
{
```

```
   LinkedList m_Children;
   Scheme m_Parent;
   Scheme m_recScheme;
   String m_Name;
   String m_Condition;
   String m_Params;
   boolean m_AutoActivate;
   String m_MotionCommands;
   String m_DialogCommands;
   boolean m_likebase_condition;
   boolean m_likebase_motion;
   boolean m_likebase_dialog;
   boolean m_IsReady;
   int m_recMotionPos;
   float m_Inhibition;
   SchemeContainer m_Robot;
 float m_Match;
   static Scheme m_AktScheme=null;

/** Konstruktor, initializiert das Schema mit Hilfe der Funktion initialize.
* @param sc An dieser Stelle wird eine Referenz auf den SchemeRobot übegeben.
* Sie wird z.B. benötigt, um bei Schemawechseln dessen Handlungsplan auszutauschen.
* @param parent Das Elternschema.
* @param name Name des Schemas.
* @see #initialize
*/
    public Scheme(SchemeContainer sc,Scheme parent,String name){
      initialize(sc,parent,name);
    }

/** Konstruktor, initializiert das Schema mit Hilfe der Funktion initialize.
* @param sc An dieser Stelle wird eine Referenz auf den SchemeRobot übegeben.
* Sie wird z.B. benötigt, um bei Schemawechseln dessen Handlungsplan auszutauschen.
* @param parent Das Elternschema.
* Der Name des Schemas sollte später mit SetName nachgetragen werden.
* @see #SetName
* @see #initialize
*/
    public Scheme(SchemeContainer sc,Scheme parent){
      initialize(sc,parent,"Kein Name");
    }

/** Initializiert das Schema.
* @param sc An dieser Stelle wird eine Referenz auf den SchemeRobot übegeben.
* Sie wird z.B. benötigt, um bei Schemawechseln dessen Handlungsplan auszutauschen.
* @param parent Das Elternschema.
* @param name Name des Schemas.
*/
    public synchronized void initialize(SchemeContainer sc,Scheme parent,String name)
    {
      try {
         int i;
         m_recScheme=null;
         m_recMotionPos=0;
         m_Parent=parent;
         m_Name=name;
         m_Robot=sc;
         m_Match=0;
         m_Children= new LinkedList();
         if(parent != null) {
            m_Condition = new String(parent.GetCondition());
            m_MotionCommands=parent.GetMotionCommands();
            m_DialogCommands=parent.GetDialogCommands();
            m_AutoActivate=parent.AutoActivate();

         }
         else {
            m_Params = new String("");
            m_Condition=new String("");
            m_MotionCommands=new String("");
            m_DialogCommands=new String("");
            m_AutoActivate=false;
         }
         m_likebase_condition=false;
         m_likebase_motion=false;
         m_likebase_dialog=false;
         m_IsReady=true;
         m_Inhibition=0;
      }
```

```
    catch(Exception e)
    {
     TextOutput.TextOut("Caught Exception in Scheme constructor: " + e);
    }
   }

  /** Setzt die statischen Klassen-Variablen (i.B. das aktuelle Schema) zurück.
   */
  static public void Reset()
    {
     m_AktScheme=null;
    }

  /** @param b Legt fest, ob das Schema automatisch aktiviert werden soll,
  * wenn seine Bewertungsfunktion das beste Ergebnis liefert.
   */
  public void SetAutoActivate(boolean b) {  m_AutoActivate=b; }

  /** @return Gibt zurück, ob das Schema automatisch aktiviert werden soll,
  * wenn seine Bewertungsfunktion das beste Ergebnis liefert.
   */
  public  boolean AutoActivate() { return m_AutoActivate; }

  /** @param b Legt fest, ob das Schema seine Bewertungsfunktion
  * vom Elternschema erben soll.
   */
  public void SetLikeBase_Condition(boolean b) {  m_likebase_condition=b; }

  /** @return Gibt zurück, ob das Schema seine Bewertungsfunktion
  * vom Elternschema erben soll.
   */
  public boolean IsLikeBase_Condition() { return m_likebase_condition; }

  /** @param b Legt fest, ob das Schema seinen Handlungsplan
  * vom Elternschema erben soll.
   */
  public void SetLikeBase_Motion(boolean b) {  m_likebase_motion=b; }

  /** @return Gibt zurück, ob das Schema seinen Handlungsplan
  * vom Elternschema erben soll.
   */
  public boolean IsLikeBase_Motion() { return m_likebase_motion; }

  /** @param b Legt fest, ob das Schema seine Dialogkonfiguration
  * vom Elternschema erben soll.
   */
  public void SetLikeBase_Dialog(boolean b) {  m_likebase_dialog=b; }

  /** @return Gibt zurück, ob das Schema seine Dialogkonfiguration
  * vom Elternschema erben soll.
   */
  public boolean IsLikeBase_Dialog() { return m_likebase_dialog; }

/** @param i Setzt den Wert für die Unterdrückung anderer Schemata.
* Dies kann zur gezielten Verstärkung oder Abschwächung der
* Bewertung des Schemas verwendet werden.
*/
public void SetInhibition(float i) { m_Inhibition = i; }

/** @return Gibt den Wert für die Unterdrückung anderer Schemata zurück.
   */
  public float GetInhibition() { return m_Inhibition; }

/** @param s Definiert die Liste der erlaubten Parameter,
* die bei einem Aufruf mit dem Skriptbefehl CallScheme
* übergeben werden können.
* @see SchemeRobot#CallScheme
*/
  public void SetParams(String s) {  m_Params=s; }

/** @return Gibt die Liste der erlaubten Parameter zurück.
* @see SchemeRobot#CallScheme
*/
  public String GetParams() { return m_Params; }

  /** @param s Setzt den Namen des Schemas.
   */
  public void SetName(String s) { m_Name=s; }
```

```
  /** @return Gibt den Namen des Schemas zurück.
   */
  public String GetName() { return m_Name;}

  /** @return Gibt den Namen des Schemas zurück.
   */
  public String toString() { return m_Name;}

/** @return Gibt eine Kopie der Liste aller Subschemata zurück.
*/
  public LinkedList GetChildren()
  {
     LinkedList l =(LinkedList) m_Children.clone();
     l.reset();
     return l;
  }

  /** @return Gibt das Elternschema zurück.
   */
  public Scheme GetParent() { return m_Parent;}

  /** @return Ermittelt, ob der SchemeRobot gerade mit der
  * Abarbeitung einer Prozedur beschäftigt ist.
   */
  public boolean IsReady() {
     if(m_Robot!=null) m_IsReady = m_Robot.Motion_Ended();
     return m_IsReady;
  }

  /** @param Legt fest, das der SchemeRobot gerade keine
  * Prozedur in Bearbeitung hat.
   */
  public void SetIsReady(boolean b) {  m_IsReady=b; }

/** @return Ermittelt, ob das Schema zur Zeit unterbrochen werden darf.
*/
  public boolean IsInteruptable(){return (m_recScheme==null);}

/** @param Gibt an, welches Schema nach Beendigung aktiviert werden soll.
*/
  public void SetRecallScheme(Scheme recScheme){ m_recScheme=recScheme;}

/** @return Gibt das Schema zurück, das nach
* Beendigung aktiviert werden soll.
*/
  public Scheme GetRecallScheme(){return m_recScheme;}

  /* @param Definiert, an welcher Stelle in der Befehlsliste
  * das mit SetRecallScheme bestimmte Schema mit der Ausführung
  * einsetzen soll.
  * @see #SetRecallScheme
   */
  public void SetRecallMotionPos(int recmotionpos) { m_recMotionPos=recmotionpos;}

  /* @param Gibt zurück, an welcher Stelle in der Befehlsliste
  * das mit SetRecallScheme bestimmte Schema mit der Ausführung
  * einsetzen soll.
  * @see #SetRecallScheme
   */
public int GetRecallMotionPos() {return m_recMotionPos;}

  /** @return Gibt das gerade aktive Schema zurück.
   */
 static public Scheme GetAktScheme() { return m_AktScheme; }

  /** Prüft die Parameterliste, die bei der expliziten Aktivierung
  * (Skriptsprache CallScheme) übergeben wird und ergänzt bei
  * Bedarf Defaultwerte für nicht gesetzte Parameter.
  * @param Die Parameterliste
  * @return Die überarbeitete Parameterliste
   */
   public String CheckParams(String params) {
   String ret=new String();
   try {
        if(params==null) return m_Params;
        String tok_param,tok_script;
        String param_name,param_val;
        StringTokenizer t_paramlist = new StringTokenizer(params,"\t\r\n ");
        StringTokenizer t_defaultparamlist = new StringTokenizer(GetParams(),"\n");
```

```
      LinkedList param_list = new LinkedList();
      boolean found=false;

      while(t_defaultparamlist.hasMoreElements())
         param_list.append(t_defaultparamlist.nextToken().trim());
      param_name=null;
      param_val=null;
      while(t_paramlist.hasMoreElements()) {
         tok_param=t_paramlist.nextToken();
         param_list.reset();
         found=false;
         while(param_list.hasMoreElements()&&!found) {

   if(((String)param_list.currentElement()).toLowerCase().startsWith(tok_param.toLowerCase())
   ){
               found=true;
               param_list.remove();
            }
            else param_list.nextElement();
         }
         if(found) {
            if(param_name!=null) {
               if(param_val!=null) {
                  param_list.append(param_name+ " " +param_val);
               }
               else TextOutput.TextOut("Für den Parameter "+param_name+" wurde kein Wert an-
   gegeben");
            }
            param_name=new String(tok_param);
            param_val=new String("");
         }
         else {
            if(param_val!=null) param_val += new String(tok_param)+" ";
            else TextOutput.TextOut("Unbekannter Parameter "+tok_param);
         }
      }
      if(param_name!=null){
         if(param_val!=null) {
            param_list.append(param_name+ " " +param_val);
         }
         else TextOutput.TextOut("Für den Parameter "+param_name+" wurde kein Wert angege-
   ben");
      }
      param_list.reset();
      while(param_list.hasMoreElements()) {
         ret+= (String) param_list.currentElement()+"\n";
         param_list.nextElement();
      }
   }
   catch(Exception e)
   {
     TextOutput.TextOut("Caught Exception in Scheme.CheckParams: " + e);
   }
   return ret;
 }

/** @param s Setzt den Handlungsplan des Schemas.
 */
 public  void SetMotionCommands(String s) {
   if(!m_likebase_motion) m_MotionCommands=s; }

/** @return Gibt den Handlungsplan des Schemas zurück
 */
 public String GetMotionCommands() {
   return GetMotionCommands(null);
 }

/** @return Gibt den Handlungplan des Schemas unter Berücksichtigung
* einer Parameterliste zurück.
* @param eine durch '\n' getrennte Parameterliste
* (wird von der Funktion CheckParam erzeugt).
 */
 public String GetMotionCommands(String params) {
    Scheme parent = GetParent();
   if(m_likebase_motion && parent!=null) return parent.GetMotionCommands(params);
   else {
      if(params==null) return m_MotionCommands;
      String coms=new String("");
      String tok_param,tok_script;
```

```
        String param_name,param_val;
        StringTokenizer t_paramlist = new StringTokenizer(params,"\n");
        String ret = new String(m_MotionCommands);
        while(t_paramlist.hasMoreElements())
        {
           tok_param = t_paramlist.nextToken();
           StringTokenizer t_param = new StringTokenizer(tok_param," \t\r\n");
           if(t_param.hasMoreElements()) param_name=t_param.nextToken();
           else param_name = new String();
           param_val=new String("");
           while(t_param.hasMoreElements())
              param_val += t_param.nextToken()+" ";
           ret = Expression.ReplaceSubExpr(ret,param_name,param_val.trim());
        }
        return ret;
     }
   }

  /** @param s Setzt die Dialogkonfiguration des Schemas.
   */
   public void SetDialogCommands(String s) {
     if(!m_likebase_dialog) m_DialogCommands=s; }

  /** @return Gibt die Dialogkonfiguration des Schemas zurück.
   */
   public String GetDialogCommands() {
     return GetDialogCommands(null);
   }

  /** @return Gibt die Dialogkonfiguration des Schemas unter Berücksichtigung
  * einer Parameterliste zurück.
  * @param eine durch '\n' getrennte Parameterliste
  * (wird von der Funktion CheckParam erzeugt).
   */
   public String GetDialogCommands(String params) {
      Scheme parent = GetParent();
     if(m_likebase_dialog && parent!=null) return parent.GetDialogCommands(params);
     else {
        if(params==null) return m_DialogCommands;
        String coms=new String("");
        String tok_param,tok_script;
        String param_name,param_val;
        StringTokenizer t_paramlist = new StringTokenizer(params,"\n");
        String ret = new String(m_DialogCommands);
        while(t_paramlist.hasMoreElements())
        {
           tok_param = t_paramlist.nextToken();
           StringTokenizer t_param = new StringTokenizer(tok_param," \t\r\n");
           if(t_param.hasMoreElements()) param_name=t_param.nextToken();
           else param_name = new String();
           param_val=new String("");
           while(t_param.hasMoreElements())
              param_val += t_param.nextToken()+" ";
           ret = Expression.ReplaceSubExpr(ret,param_name,param_val.trim());
        }
        return ret;
     }
   }

  /** @param s Setzt die Bewertungsfunktion des Schemas.
  * Diese besteht aus einer oder mehrerer Zeilen, die mit
  * Hilfe der Skriptsprache die globale Variable "Rating"
  * festlegen. Der Wert dieser Variable bestimmt - im Vergleich
  * mit anderen Schemata - welches Schema am besten zur Situation
  * passt und aktiviert wird.
   */
   public void SetCondition(String s) {
     if(!m_likebase_condition)  m_Condition=s; }

  /** @return Gibt die Bewertungsfunktion des Schemas zurück.
   */
   public String GetCondition() {
      Scheme parent = GetParent();
     if(m_likebase_condition && parent!=null) return parent.GetCondition();
     else return m_Condition;
   }

/** Nimmt ein Schema in die Liste der Subschemata auf.
* @param sub Das einzufügene Schema
```

```
*/
  public synchronized void AddSubScheme(Scheme sub)
  {
     try {
        if(sub instanceof Scheme)
           m_Children.append(sub);
        else TextOutput.TextOut("Nur Schemata können in die Schema-Hierarchie eingefügt wer-
     den");
     }
    catch(Exception e)
    {
     TextOutput.TextOut("Caught Exception in Scheme.AddSubScheme: " + e);
    }

  }

  /** @return Gibt die Hierarchieebene des Schemas zurück. Für das Basisschema
  * ist dieser Wert 0.
   */
  public int GetParentLevels()
  {
     int lev=0;
      Scheme parent = GetParent();
     try {
        if(parent!=null) lev = parent.GetParentLevels()+1;
     }
     catch(Exception e)
     {
        TextOutput.TextOut("Caught Exception in Scheme.GetParentLevels: " + e);
     }
     return lev;
  }

  /** @return Ermittelt, wieviele Hierarchieebenen von Subschemata vorhanden sind.
   */
  public int GetChildLevels()
  {
     int maxchildlevel=0,childlevel=0,mylevel =0;
     try {
        LinkedList children = GetChildren();
        children.reset();
        while(children.hasMoreElements()) {
           childlevel = ((Scheme) children.currentElement()).GetChildLevels();
           if(childlevel > maxchildlevel) maxchildlevel=childlevel;
           children.nextElement();
        }
     }
     catch(Exception e)
     {
       TextOutput.TextOut("Caught Exception in Scheme.GetChildLevels: " + e);
     }
     return maxchildlevel+1;
  }

/** @return Das Basisschema der Schemahierarchie.
*/
  public Scheme GetBaseScheme()
  {
     Scheme pa=this;
      Scheme parent = GetParent();
     try {
        if(parent!=null) pa= parent.GetBaseScheme();
     }
     catch(Exception e)
     {
        TextOutput.TextOut("Caught Exception in Scheme.GetParentLevels: " + e);
     }
     return pa;
  }

/** @return Ermittelt für dieses Schema die Anzahl der Blätter
* im Schemabaum seiner Subschemata. Wenn keine Subschemata vorhanden
* sind, ist dieser Wert 1.
*/
  public int GetLeafCount()
  {
     int sum =0;
     try {
        LinkedList children = GetChildren();
```

```
         children.reset();
         while(children.hasMoreElements()) {
            sum += ((Scheme) children.currentElement()).GetLeafCount();
            children.nextElement();
         }
         if(sum==0) sum = 1;
      }
      catch(Exception e)
      {
         TextOutput.TextOut("Caught Exception in Scheme.GetChildLevels: " + e);
      }
      return sum;
   }

 /** @return Ermittelt die horizontale Position des Schemas in der
 * Schemahierarchie für die Darstellung im KonfigSchemeDialog.
 * @see KonfigSchemeDialog
 */
   public int GetLeafNum()
   {
      int num=0;
       Scheme parent = GetParent();
      try {
         Scheme brother=null;
         if(parent!=null) {
            LinkedList brothers = parent.GetChildren();
            brothers.reset();
            while(brothers.hasMoreElements()&& brother != this) {
               brother = (Scheme) brothers.currentElement();
               if(brother != this) num += brother.GetLeafCount();
               brothers.nextElement();
            }
            num += parent.GetLeafNum();
         }
      }
      catch(Exception e)
      {
         TextOutput.TextOut("Caught Exception in Scheme.GetLeafNum: " + e);
      }
      return num;
   }

/** Durchsucht die Schemahierarchie nach einem benannten Schema.
* @param name Der Name des gesuchten Schemas
* @return Das gefundene Schema, sonst null.
 */
   public Scheme GetNode(String name)
   {
      Scheme scheme=null;
      try {
         if(toString().equalsIgnoreCase(name)) scheme=this;
         LinkedList children = GetChildren();
         children.reset();
         while(children.hasMoreElements() && scheme == null) {
            scheme = ((Scheme) children.currentElement()).GetNode(name);
            children.nextElement();
         }
      }
      catch(Exception e)
      {
        TextOutput.TextOut("Caught Exception in Scheme.GetNode: " + e);
      }
      return scheme;
   }

 /** Ersetzt Handlungsplan und Dialogkonfiguration des SchemeRobot
 * durch die im Schema gespeicherten.
 */
   public void Realize() { Realize(null,0); }

 /** Ersetzt Handlungsplan und Dialogkonfiguration des SchemeRobot
 * durch die im Schema gespeicherten.
 * @param params Eine durch '\n' getrennte Parameterliste
 * @see #Realize(String, int)
 */
   public void Realize(String params) { Realize(params,0); }

 /** Ersetzt Handlungsplan und Dialogkonfiguration des SchemeRobot
 * durch die im Schema gespeicherten.
```

```
* @param params Eine durch '\n' getrennte Parameterliste, deren Elemente
* mit Hilfe der Funktionen GetMotionCommands und GetDialogCommands
* ersetzt werden.
* @see #GetMotionCommands(String)
* @see #GetDialogCommands(String)
* @param motionpos Position im Handlungsplan, wo mit der Ausführung begonnen
* werden soll.
* @see #SetMotionCommands(String,int)
*/
  public void Realize(String params,int motionpos)
  {
     SetIsReady(false);
     String newparams = CheckParams(params);
     m_Robot.SetMotionCommands(GetMotionCommands(newparams),motionpos);
     m_Robot.SetDialogCommands(GetDialogCommands(newparams));
     System.out.println("Schema "+toString()+ " aktiviert");
  }

/** @return Prüft mit Hilfe der Funktion Matches, welches Schema im (Sub-)Schemabaum
* am besten zur aktuellen Situation passt und daher aktiviert werden sollte.
*/
  public Scheme FindMatchingChild()
  {
     Scheme scheme = null;
     Scheme matchingscheme=this;
     Scheme bestchildscheme=null;
     float bestchildmatch = (-999);
     float bestmatch = (-999);
     float mymatch = (-999);
      try {
       mymatch=Matches();
         matchingscheme=this;
     LinkedList l = GetChildren();
         if(l==null) return this;
     l.reset();
     while(l.hasMoreElements())
     {
        scheme = (Scheme) l.currentElement();
        bestchildscheme = scheme.FindMatchingChild();
              if(bestchildscheme!=null) {
              bestchildmatch = bestchildscheme.GetMatch();
           if(bestchildmatch > bestmatch) {
              matchingscheme=bestchildscheme;
              bestmatch=bestchildmatch;
           }
         }
        l.nextElement();
     }
       if(mymatch >= bestmatch) {
          matchingscheme=this;
          bestmatch=mymatch;
       }
     }
     catch(Exception e)
     {
       TextOutput.TextOut("Caught Exception in Scheme.FindMatchingChild: " + e);
     }
     return matchingscheme;
  }

  float GetMatch()
  {return m_Match;}

/** Führt die Befehle (Skriptsprache) der Bewertungsfunktion aus und
* gibt das Ergebnis zurück.
* Das Ergebis der Bewertungfunktion wird nach Abarbeitung der Befehle
* aus der globalen Variablen "Rating") gelesen und zum Wert für die Unterdrückung
* anderer Schemata (-> GetInhibition) addiert.
* @see GetInhibition
* @return Die Gesamtbewertung des Schemas
*/
  public float Matches()
  {
     try {
     if(AutoActivate()) {
        float inhibition = GetInhibition();
        String condition = GetCondition();
        if(condition==null) return 0;
        StringTokenizer t = new StringTokenizer(condition,"\n");
```

```
        while(t.hasMoreElements()) m_Robot.Apply(t.nextToken());
        Variable rating = m_Robot.GetVar("Rating");
              if(rating == null) return 0;
              float rate_val=rating.floatValue()+inhibition;
              m_Match=rate_val;

        return (  rate_val);
     }
     }
     catch(Exception e)
     {
       TextOutput.TextOut("Caught Exception in Scheme.Matches: " + e);
     }
     return (-1);
  }

/** Führt eine Bewertung aller Schemata durch und aktiviert das beste.
* @return Gibt das aktivierte Schema zurück.
*/
  public Scheme CheckScheme()
  {
     if(m_Robot==null) return null;
      Scheme aktScheme = m_AktScheme;
     Scheme newscheme = aktScheme;
     int motionpos=0;
     try {
        if(aktScheme!= null && aktScheme.IsReady()) {
           newscheme=aktScheme.GetRecallScheme();
           if(newscheme==null) {
           newscheme=GetBaseScheme().FindMatchingChild();
           }
           else motionpos=aktScheme.GetRecallMotionPos()+1;
           aktScheme.SetRecallMotionPos(0);
           aktScheme.SetRecallScheme(null);
        }
        else {
           if(aktScheme ==null || aktScheme.IsInteruptable())
              newscheme=GetBaseScheme().FindMatchingChild();
        }
        if(newscheme==null) newscheme=GetBaseScheme();
        if( (aktScheme == null) ||
           (newscheme!=aktScheme &&
          newscheme.GetMatch() > aktScheme.GetMatch())) {
           // Schemawechsel
           aktScheme=newscheme;
           aktScheme.Realize(null,motionpos);
        }
       m_AktScheme=aktScheme;
     }
     catch(Exception e)
     {
       TextOutput.TextOut("Caught Exception in Scheme.CheckScheme: " + e);
     }
     return newscheme;
  }

/** @return Serialisiert das Schema mit allen Subschemata
* und gibt das Ergebnis zurück.
*/
  public String GetSaveString()
  {
     String str=new String();
     str += "Begin_Scheme\n";
     str += "Name="+GetName()+"\n";
     str += "Begin_Params\n";
     str += GetParams()+"\n";
     str += "End_Params\n";
     str += "AutoActivate="+AutoActivate()+"\n";
     str += "Begin_Condition\n";
     str += GetCondition()+"\n";
     str += "End_Condition\n";
     str += "IsLikeBase_Condition="+IsLikeBase_Condition()+"\n";
     str += "IsLikeBase_Motion="+IsLikeBase_Motion()+"\n";
     str += "IsLikeBase_Dialog="+IsLikeBase_Dialog()+"\n";

     str += "Begin_Motion\n";
     str += GetMotionCommands()+"\n";
     str += "End_Motion\n";
     str += "Begin_Dialog\n";
```

```
      str += GetDialogCommands()+"\n";
      str += "End_Dialog\n";
      LinkedList l= GetChildren();
      l.reset();
      while(l.hasMoreElements()) {
         str += ((Scheme) l.currentElement()).GetSaveString();
         l.nextElement();
      }
      str += "End_Scheme\n";
      return str;
   }

/** Liesst ein serialisiertes Schema inclusive seiner Subschemata
* ein und baut eine entsprechende Schemahierarchie auf.
* @param str Das serialisierte Schema als String
*/
   public void ReadSaveString(String str) { ReadSaveString(null,str); }

   synchronized void ReadSaveString(StringTokenizer t,String str)
   {
      try {
         StringTokenizer t2=null;
         String tok;
         String motionstring;
         String dialogstring;
         String conditionstring;
         String paramstring;
         Scheme basescheme=null;
         boolean schemestarted=false;
         boolean schemeended=false;
         if(str!=null) {
            if(t==null)
               t = new StringTokenizer(str,"\n");
            else {
               basescheme=this;
               schemestarted=true;
            }
            while(t.hasMoreElements()&& !schemeended)
            {
               tok = t.nextToken();
               if(tok.equalsIgnoreCase("Begin_Scheme"))
               {
                  if(schemestarted) { // SubSchema erzeugen
                     Scheme subscheme=new Scheme(m_Robot,this);
                     AddSubScheme(subscheme);
                     subscheme.ReadSaveString(t,str);
                  }
                  else schemestarted=true;
               }
               if(schemestarted) {
                  if(tok.equalsIgnoreCase("End_Scheme"))
                  {
                     schemeended=true;
                  }
                  else if(tok.equalsIgnoreCase("Begin_Motion"))
                  {
                     motionstring=new String();
                     tok = t.nextToken();
                     while(t.hasMoreElements() && !tok.equalsIgnoreCase("End_Motion"))
                     {
                        motionstring += tok+"\n";
                        tok = t.nextToken();
                     }
                     SetMotionCommands(motionstring);
                  }
                  else if(tok.equalsIgnoreCase("Begin_Dialog"))
                  {
                     dialogstring=new String();
                     tok = t.nextToken();
                     while(t.hasMoreElements() && !tok.equalsIgnoreCase("End_Dialog"))
                     {
                        dialogstring += tok+"\n";
                        tok = t.nextToken();
                     }
                     SetDialogCommands(dialogstring);
                  }
                  else if(tok.equalsIgnoreCase("Begin_Condition"))
                  {
                     conditionstring=new String();
```

```
                    tok = t.nextToken();
                    while(t.hasMoreElements() && !tok.equalsIgnoreCase("End_Condition"))
                    {
                       conditionstring += tok+"\n";
                       tok = t.nextToken();
                    }
                    SetCondition(conditionstring);
                 }
                 else if(tok.equalsIgnoreCase("Begin_Params"))
                 {
                    paramstring=new String();
                    tok = t.nextToken();
                    while(t.hasMoreElements() && !tok.equalsIgnoreCase("End_Params"))
                    {
                       paramstring += tok+"\n";
                       tok = t.nextToken();
                    }
                    SetParams(paramstring);
                 }
                 else {
                    String line = new String(tok);
                    t2 = new StringTokenizer(line,"=");
                    if(t2.countTokens() >= 2)
                    {
                       String entry=t2.nextToken();
                       String value=t2.nextToken();
                       if(entry.equalsIgnoreCase("Name")) SetName(value);
                       else if(entry.equalsIgnoreCase("AutoActivate")) SetAutoActiva-
      te(GetBool(value));
                       else if(entry.equalsIgnoreCase("IsLikeBase_Condition")) SetLikeBa-
      se_Condition(GetBool(value));
                       else if(entry.equalsIgnoreCase("IsLikeBase_Motion")) SetLikeBa-
      se_Motion(GetBool(value));
                       else if(entry.equalsIgnoreCase("IsLikeBase_Dialog")) SetLikeBa-
      se_Dialog(GetBool(value));
                    }
                 }
              }
           }
        }
      }
      catch(Exception e)
      {
         TextOutput.TextOut("Caught Exception in ReadSaveString: " + e);
      }
   }

   static boolean GetBool(String s)
   {
      return (s.trim().equalsIgnoreCase("true"));
   }
}
```

4.6 Die Klasse SchemeDialog

```
/*****************************************************************
* @(#)SchemeDialog.java1.0 97/06/23 Hauke Ernst
*/
package SchemeRobot;

import java.awt.*;
import VRMLInterface.ResultProcessor;
import VRMLInterface.TextOutput;
import SchemeRobot.Scheme;

/** Die Dialogklasse SchemeDialog erlaubt das Bearbeiten
* eines Schemas in einer Eingabemaske Außerdem ist eine
* Schaltfläche zum Erzeugen eines Subschemas vorhanden.
*
*
* @author  Hauke Ernst
* @version 1.0
*/
public class SchemeDialog extends Frame
{
```

```
   private ResultProcessor m_ResultProcessor;
   Scheme m_Scheme;
   Checkbox m_likebase_condition;
   Checkbox m_likebase_motion;
   Checkbox m_likebase_dialog;
   Label    m_name_label;
   Checkbox    m_actif;
   Label    m_condition_label;
   Label    m_behav_label;
   Label    m_motion_label;
   Label    m_dialog_label;
   Label    m_param_label;

   TextArea m_param;
   TextField m_name;
   TextField m_place;
   TextArea m_condition;
   TextArea m_motion;
   TextArea m_dialog;

/** Konstruktor
* @param creator Benennt den ResultProcessor, der für die
* Behandlung von Dialogereignissen zuständig ist.
* @param scheme Das Schema, das in der Maske bearbeitet werden soll
* @see ResultProcessor
*/
   public SchemeDialog(ResultProcessor creator, Scheme scheme)
   {
      m_ResultProcessor=creator;
      SetScheme(scheme);
   }

/** Stellt ein anderes Schema in der Maske dar.
* @param scheme Das Schema, das in der Maske bearbeitet werden soll
*/
   public synchronized void SetScheme(Scheme scheme)
   {
      m_Scheme = scheme;
      removeAll();

      Font myfont = new Font("Helvetica", Font.BOLD, 14);
      setFont(myfont);
      setTitle("Schema konfigurieren");
      GridBagLayout layout = new GridBagLayout();
      setLayout(layout);
      GridBagConstraints constraints = new GridBagConstraints();
      constraints.fill = GridBagConstraints.HORIZONTAL;

      constraints.weightx = 100;
      constraints.weighty = 100;
      constraints.anchor = GridBagConstraints.WEST;

      String likebase_string = "wie Basisschema";
      int col=0;
      m_name_label = new Label("Schemaname");
      constraints.fill = GridBagConstraints.NONE;
      add(m_name_label,layout,constraints,0,col,1,1);
      m_name = new TextField(scheme.GetName());
      constraints.fill = GridBagConstraints.HORIZONTAL;
      add(m_name,layout,constraints,1,col,4,1);
      col++;
      if(scheme.GetParent()!=null)
      {
         m_param_label = new Label("Parameter");
         constraints.fill = GridBagConstraints.NONE;
         add(m_param_label,layout,constraints,0,col,1,1);
         m_param = new TextArea(scheme.GetParams(),4,10);
         constraints.fill = GridBagConstraints.HORIZONTAL;
         add(m_param,layout,constraints,1,col,4,1);
         col++;

         m_actif = new Checkbox("Automatisch aktivieren wenn:",null,scheme.AutoActivate());
         constraints.fill = GridBagConstraints.NONE;
         add(m_actif,layout,constraints,0,col,2,1);
         col++;

         m_condition_label = new Label("Bedingungen");
         constraints.fill = GridBagConstraints.NONE;
         add(m_condition_label,layout,constraints,0,col,1,1);
```

```
         m_condition = new TextArea(scheme.GetCondition(),4,20);
         constraints.fill = GridBagConstraints.HORIZONTAL;
//         constraints.gridwidth = GridBagConstraints.REMAINDER;
         add(m_condition,layout,constraints,2,col,3,1);
         m_likebase_condition = new Check-
      box(likebase_string,null,scheme.IsLikeBase_Condition());
         constraints.fill = GridBagConstraints.NONE;
         add(m_likebase_condition,layout,constraints,1,col,1,1);
         col++;
      }
      m_behav_label = new Label("Verhalten:");
      add(m_behav_label,layout,constraints,0,col,1,1);
      col++;

      m_motion_label = new Label("Handlungen");
      constraints.fill = GridBagConstraints.NONE;
      add(m_motion_label,layout,constraints,0,col,1,1);
      m_motion = new TextArea(scheme.GetMotionCommands(),4,20);
      constraints.fill = GridBagConstraints.HORIZONTAL;
      add(m_motion,layout,constraints,2,col,3,1);
      if(scheme.GetParent()!=null) {
         m_likebase_motion = new Checkbox(likebase_string,null,scheme.IsLikeBase_Motion());
         constraints.fill = GridBagConstraints.NONE;
         add(m_likebase_motion,layout,constraints,1,col,1,1);
      }
      col++;

      m_dialog_label = new Label("Dialog");
      constraints.fill = GridBagConstraints.NONE;
      add(m_dialog_label,layout,constraints,0,col,1,1);
      m_dialog = new TextArea(scheme.GetDialogCommands(),4,20);
      constraints.fill = GridBagConstraints.HORIZONTAL;
      add(m_dialog,layout,constraints,2,col,3,1);
      if(scheme.GetParent()!=null) {
         m_likebase_dialog = new Checkbox(likebase_string,null,scheme.IsLikeBase_Dialog());
         constraints.fill = GridBagConstraints.NONE;
         add(m_likebase_dialog,layout,constraints,1,col,1,1);
      }
      col++;

      constraints.fill = GridBagConstraints.NONE;
      constraints.anchor = GridBagConstraints.SOUTH;
      add(new Button("Schließen"),layout,constraints,0,col,2,1);
      add(new Button("Subschema erzeugen"),layout,constraints,2,col,2,1);
      add(new Button("Schema löschen"),layout,constraints,4,col,1,1);
   }

/** Gibt Das gerade dargestellte Schema zurück.
* Das gerade dargestellte Schema
*/
   public Scheme GetScheme()
   {
      return m_Scheme;
   }

   private void add(Component c, GridBagLayout gbl,
      GridBagConstraints gbc,
      int x, int y, int w, int h)
   {  gbc.gridx = x;
      gbc.gridy = y;
      gbc.gridwidth = w;
      gbc.gridheight = h;
      gbl.setConstraints(c, gbc);
      add(c);
   }

/** Wird zur Behandlung von Dialogereignissen aufgerufen.
* Die Änderungen der Schemakonfiguration wirken sich sofort aus.
* Zwecks Benachrichtigung wird beim ResultProcessor die
* Funktion processResult aufgerufen.
* @param evt Der Event
* @param arg Das betreffene Objekt
* @return true
*/
   public synchronized boolean action(Event evt, Object arg)
   {
      try {
         if(arg instanceof Boolean) {
            if(m_Scheme.GetParent()!=null) {
```

```
               m_Scheme.SetAutoActivate(m_actif.getState());
               m_Scheme.SetLikeBase_Condition(m_likebase_condition.getState());
               m_Scheme.SetLikeBase_Motion(m_likebase_motion.getState());
               m_Scheme.SetLikeBase_Dialog(m_likebase_dialog.getState());
            }
         }
         else if(arg instanceof String) {
            m_Scheme.SetName(m_name.getText());
            if(m_Scheme.GetParent()!=null) {
               m_Scheme.SetCondition(m_condition.getText());
               m_Scheme.SetParams(m_param.getText());
            }
            m_Scheme.SetDialogCommands(m_dialog.getText());
            m_Scheme.SetMotionCommands(m_motion.getText());
            ((ResultProcessor)m_ResultProcessor).processResult(this,arg);
         }
      }
      catch(Exception e)
      {
         TextOutput.TextOut("Caught Exception in SchemeDialog.action: " + e);
      }
      return true;
   }
}
```

4.7 Die Klasse KonfigSchemeDialog

```
/*********************************************************************
* @(#)KonfigSchemeDialog.java 1.0 97/06/23 Hauke Ernst
*/

package SchemeRobot;

import java.awt.*;
import VRMLInterface.ResultProcessor;
import VRMLInterface.TextOutput;
import VRMLInterface.LinkedList;
import SchemeRobot.Scheme;

/** Dialogklasse zur Darstellung der Hierarchie von Schemata.
* Die einzelnen Schemata werden jeweils als Schaltfläche
* dargestellt, mit deren Hilfe ein SchemeDialog geöffnet werden
* kann, der das ausgewählte Schema zur Bearbeitung lädt.
*
* @see SchemeDialog
*
* @author  Hauke Ernst
* @version 1.0
*/
public class KonfigSchemeDialog extends Frame
{
   private ResultProcessor m_ResultProcessor;

/** Konstruktor.
* @param creator Benennt den ResultProcessor, der für die
* Behandlung von Dialogereignissen zuständig ist.
* @param basescheme Das Basisschema der Schema-Hierarchie
*/
   public KonfigSchemeDialog(ResultProcessor creator, Scheme basescheme)
   {
      try {
         m_ResultProcessor=creator;
         SetSchemes(basescheme);
      }
      catch(Exception e)
      {
        TextOutput.TextOut("Caught Exception in constructor KonfigSchemeDialog: " + e);
      }
   }
/** Aktualisiert die Darstellung.
* @param basescheme Das Basisschema der Schema-Hierarchie
*/
   public void SetSchemes(Scheme basescheme)
   {
      try {
```

```
         Font myfont = new Font("Helvetica", Font.BOLD, 14);
         setFont(myfont);
         setTitle("Schemata konfigurieren");
         removeAll();
         GridBagLayout layout = new GridBagLayout();
         setLayout(layout);
         GridBagConstraints constraints = new GridBagConstraints();
         constraints.fill = GridBagConstraints.BOTH;
         constraints.weightx = 100;
         constraints.weighty = 100;

         int rows=basescheme.GetChildLevels();
         int cols=basescheme.GetLeafCount();
         AddSchemeButtons(basescheme,rows,layout,constraints);
         constraints.fill = GridBagConstraints.HORIZONTAL;
         constraints.anchor = GridBagConstraints.SOUTH;
         add(new Button("Schließen"),layout,constraints,0,rows+1,1,1);
         add(new Button("Laden"),layout,constraints,1,rows+1,1,1);
         add(new Button("Speichern"),layout,constraints,2,rows+1,1,1);
         add(new Button("Neu"),layout,constraints,3,rows+1,1,1);
      }
      catch(Exception e)
      {
        TextOutput.TextOut("Caught Exception in KonfigSchemeDialog.SetSchemes: " + e);
      }

   }
   private  void AddSchemeButtons(Scheme basescheme, int rows,
      GridBagLayout gbl, GridBagConstraints gbc)
   {
      try {
         if(basescheme == null) return;
         Scheme scheme=null;
         int height = 1;
         int cl=basescheme.GetChildLevels();
         int pl=basescheme.GetParentLevels();
         if(cl<=1) height = (rows-pl);
         add(new Button(basescheme.toString()),gbl,gbc,
               basescheme.GetLeafNum(),
               pl,
               basescheme.GetLeafCount(),
               height);
         LinkedList children = basescheme.GetChildren();
         children.reset();
         while(children.hasMoreElements()) {
            if(children.currentElement()==null) TextOutput.TextOut("Leeres Element in Knotenli-
      ste");
               Object obj = children.currentElement();
               scheme=(Scheme)obj;
               AddSchemeButtons(scheme,rows,gbl,gbc);

            children.nextElement();
         }
      }
      catch(Exception e)
      {
        TextOutput.TextOut("Caught Exception in KonfigSchemeDialog.AddSchemeButtons: " + e);
      }
   }

private void add(Component c, GridBagLayout gbl,
      GridBagConstraints gbc,
      int x, int y, int w, int h)
   {  gbc.gridx = x;
      gbc.gridy = y;
      gbc.gridwidth = w;
      gbc.gridheight = h;
      gbl.setConstraints(c, gbc);
      add(c);
   }

/** Wird zur Behandlung von Dialogereignissen aufgerufen.
* Die Funktion processResult der Klasse ResultProcessor
* wird hierfür verwendet.
* @param evt Der Event
* @param arg Das betreffene Objekt
* @return true
*/
   public boolean action(Event evt, Object arg)
```

```
   {
      ((ResultProcessor)m_ResultProcessor).processResult(this,arg);
      return true;
   }
}
```

4.8 Die Schnittstelle SchemeContainer

```
/*****************************************************************
* @(#)SchemeContainer.java 1.0 97/06/23 Hauke Ernst
*/
package SchemeRobot;

import VRRobot.Procedure;
import VRRobot.Variable;

/** Das Interface SchemeContainer beschreibt die Funktionalität,
* die die Klasse Scheme von einem SchemeRobot erwartet. Dieses
* Konstrukt ist notwendig, da Scheme vor SchemeRobot kompiliert
* werden muß.
*/
public interface SchemeContainer
{
/** Ersetzt den aktuellen Handlungsplan durch einen neuen.
* @param motioncommands der neue Handlungsplan
* @param motionpos bezeichnet den Befehl, mit dem
* die Ausführung beginnen soll
*/
public void SetMotionCommands(String motioncommands,int motionpos);

/** Ersetzt die aktuelle Dialogkonfiguration durch eine neue.
* @param dialogcommands Die neue Dialogkonfiguration
*/
  public void SetDialogCommands(String dialogcommands);

/** Führt einen Befehl der Skriptsprache aus.
* @param command der auszuführende Befehl
*/
  public int Apply(String command);

/** Gibt die aktuell in Ausführung befindliche Prozedur zurück.
* @return die aktuell in Ausführung befindliche Prozedur
*/
  public Procedure AktProcedure();

/** Ermittelt, ob der SchemeRobot gerade eine Prozedur verarbeitet.
* @return true, wenn der SchemeRobot sich in einem Wartezustand befindet.
*/
  public boolean Motion_Ended();

/** Sucht die benannte Variable in der Liste der globalen Variablen
* und gibt diese zurück.
* @param name Name der gesuchten Variable
* @ return die gefundene Variable
*/
   public Variable GetVar(String name);

}
```

5 Das Beispiel Museumsführer

5.1 Szenenbeschreibung in VRML2

```
#VRML V2.0 utf8

#
# This file is generated by Community Place Conductor Version 1.0 Alpha 2.
# Sony Corporation 1997.
#

Background {
  skyColor  0 0.5 1
}

NavigationInfo {
  headlight TRUE
  type      [ "FLY" ]
#  avatarSize       [ 10, 10, 10 ]
}


DEF EntryView Viewpoint {

         position 0 0 50
#         orientation 1 0 0  3.14
         jump TRUE
#         fieldOfView 0.8
}

PROTO TextOut [
   field MFString string "Text"
   field SFVec3f translation 0 0 0
   field SFVec3f scale 4 4 4
   field SFRotation rotation 0 1 0 0
]
{
   Transform {
     translation IS translation
     scale IS scale
     rotation IS rotation
     children [
       Shape {
         geometry Text {
           string IS string
              fontStyle FontStyle {justify "MIDDLE"}
         }
       }
     ]
   }
}

PROTO ImageBox [
      field MFString file "images/matiss1.jpg"
      field MFString name "Test"
      field MFString im_name "Test"
      field SFVec3f scale 1 1 1
      field SFVec3f translation 0 0 0
      field SFRotation rotation 0 1 0 0
]
{
   Transform {
     rotation IS rotation
     translation IS translation
     scale IS scale
     children [
             Shape {
               appearance Appearance {
                 texture ImageTexture {
                   url IS file
                 }
```

```
              }
              geometry Box {}
             }
    ]
   }
}

EXTERNPROTO VRRobot [
      field SFString Name
    field SFVec3f translation
    field SFVec3f scale
     field SFRotation rotation
      field SFString ScriptFile
      field MFString URL
      field SFString Outs
      field SFString Ins
] "Proto.wrl#VRRobot"



PROTO Platform [
      field SFVec3f scale 50 1 50
      field SFVec2f texscale 25 25
      field SFVec3f translation 0 0 0
      field MFString texture "images/marmor.jpg"

]
{
   Transform {
     translation IS translation
     scale IS scale
     children [
       Shape {
         appearance Appearance {
           texture ImageTexture {
             url IS texture
           }
           textureTransform TextureTransform {
            scale IS texscale
           }
         }
         geometry Box {}
       }
     ]
   }
}


PROTO FoodContainer [
   field SFVec3f translation 0 0 0
   field SFVec3f scale 0.7 0.7 0.7
]
{
   Transform {
      translation IS translation
      scale IS scale
      children [
            Transform {
         children [
               Shape {
                  appearance Appearance {
               material Material { diffuseColor 0 0.5 1}
                 }
                 geometry Cylinder { radius 1 height 4 parts ALL}
                 }
           ]
              }
           Transform {
              rotation  1 0 0 3.14
              translation 0 2 0
              scale 2 2 2
         children [
                  Shape {
               appearance Appearance {
                  material Material { diffuseColor 1 0 0}
           }
             geometry Cone { }
             }
```

```
                ]
          }
          TextOut { translation 0 5 0 scale 1 1 1 string "Chicken Nuggets"}
         ]
     }
}


Platform {translation 65 -5 0 texscale 35 5 scale 35 1 5 texture "images/wood.jpg"}
Platform {translation 0 -5 65 texscale 5 35 scale 5 1 35 texture "images/wood.jpg"}


Platform {translation 0 -5 0 texscale 15 15 scale 30 1 30}
TextOut {string "Paul Klee" translation 0 10 0 }
TextOut {string "1" translation -20 6 0 }
TextOut {string "2" translation   0 6 0 }
TextOut {string "3" translation  20 6 0 }
ImageBox {file "images/klee1.jpg" scale 4 4 4 translation -20 0 0 rotation 0 1 0  0}
ImageBox {file "images/klee2.jpg" scale 4 4 4 translation 0 0 0 rotation 0 1 0  0}
ImageBox {file "images/klee3.jpg" scale 4 4 4 translation 20 0 0 rotation 0 1 0  0}
FoodContainer { translation 25 -4 25 }


Platform {translation 130 -5 0 texscale 15 15 scale 30 1 30}
TextOut {string "Henri Matisse" translation 130 10 0 rotation 0 1 0 -1.57 }
TextOut {string "1" translation  130 6 -20 rotation 0 1 0 -1.57 }
TextOut {string "2" translation  130 6   0 rotation 0 1 0 -1.57 }
TextOut {string "3" translation  130 6  20 rotation 0 1 0 -1.57 }
ImageBox {file "images/matiss1.jpg" scale 4 4 4 translation 130 0 -20 rotation 0 1 0  0}
ImageBox {file "images/matiss2.jpg" scale 4 4 4 translation 130 0 0 rotation 0 1 0  0}
ImageBox {file "images/matiss3.jpg" scale 4 4 4 translation 130 0 20 rotation 0 1 0  0}


Platform {translation 0 -5 130 texscale 15 15 scale 30 1 30}
TextOut {string "Wassily Kandinski" translation 0 10 130 rotation 0 1 0 3.14}
TextOut {string "1" translation  -20 6 130 rotation 0 1 0 3.14 }
TextOut {string "2" translation    0 6 130 rotation 0 1 0 3.14 }
TextOut {string "3" translation   20 6 130 rotation 0 1 0 3.14 }
ImageBox {file "images/kandin1.jpg" scale 4 4 4 translation -20 0 130 rotation 0 1 0  0}
ImageBox {file "images/kandin2.jpg" scale 4 4 4 translation 0 0 130 rotation 0 1 0  0}
ImageBox {file "images/kandin3.jpg" scale 4 4 4 translation 20 0 130 rotation 0 1 0  0}


DEF Robot VRRobot {
       Name "Museumsfuehrer"
       translation 0 -4 20
       scale 0.5 0.5 0.5
       ScriptFile "script/guide.rcf"
       URL "../classes/SchemeRobot/SchemeRobot.class"
}
```

5.2 Handlungsplan für den Museumsführer

```
// Funktionssammlung zur Ansteuerung von VRRobots
// Author: Hauke Ernst
// Stand : 17.07.1997


procedure VRRobotMain
begin
GlobalVar eatlock false
StopWalk
FollowArea
end

procedure FollowArea
begin
  Tag Loop
         LocalVar userx %userpos.x
         LocalVar userz %userpos.z

         if ((%userx > -65) & (%userx < 65)  & (%userz > -65) & (%userz < 65)) then GotoKleeArea
         if ((%userx > -65) & (%userx < 65)  & (%userz >  65) & (%userz < 195)) then Goto-
      KandinskyArea
```

```
        if ((%userx >  65) & (%userx < 195) & (%userz > -65) & (%userz < 65)) then GotoMatisse-
     Area
        TurnToUser
        Wait slides 10
  Goto Loop
end

procedure GotoKleeArea
begin
  if ((%robotpos.x > -65) & (%robotpos.x < 65) & (%robotpos.z > -65) & (%robotpos.z < 65)) then
     Goto ende
Say Warte, ich komme auch in den Bereich von Paul Klee!
Inhibit 100
GlobalVar eatlock true
StartWalk
  if ((%robotpos.x > 65) & (%robotpos.x < 195) & (%robotpos.z > -65) & (%robotpos.z < 65)) then
     Goto FromMatisse
  if ((%robotpos.x > -65) & (%robotpos.x < 65) & (%robotpos.z > 65) & (%robotpos.z < 195)) then
     Goto FromKandinsky
Goto talkpos
Tag FromMatisse
Go x 100, z 0, startwalk false, stopwalk false
Go x 30, z 0, startwalk false, stopwalk false
goto talkpos
Tag FromKandinsky
Go x 0, z 100, startwalk false, stopwalk false
Go x 0, z 30, startwalk false, stopwalk false
Tag talkpos
Go x -25, z 25, startwalk false, stopwalk true
Inhibit -100
GlobalVar eatlock false

Tag ende
end

procedure GotoKandinskyArea
begin
  if ((%robotpos.x > -65) & (%robotpos.x < 65) & (%robotpos.z > 65) & (%robotpos.z < 195)) then
     Goto ende
Say Warte, ich komme auch in den Bereich von Wassily Kandinsky!
Inhibit 100
GlobalVar eatlock true
StartWalk
  if ((%robotpos.x > -65) & (%robotpos.x < 65) & (%robotpos.z > -65) & (%robotpos.z < 65)) then
     Goto FromKlee
  if ((%robotpos.x > 65) & (%robotpos.x < 195) & (%robotpos.z > -65) & (%robotpos.z < 65)) then
     Goto FromMatisse
goto talkpos
Tag FromMatisse
Go x 100, z 0, startwalk false, stopwalk false
Go x 30, z 0, startwalk false, stopwalk false
Tag FromKlee
Go x 0, z 30, startwalk false, stopwalk false
Go x 0, z 100, startwalk false, stopwalk false
Tag talkpos
Go x 25, z 105, startwalk false, stopwalk true
Inhibit -100
GlobalVar eatlock false
Tag ende
end

procedure GotoMatisseArea
begin
  if ((%robotpos.x > 65) & (%robotpos.x < 195) & (%robotpos.z > -65) & (%robotpos.z < 65)) then
     Goto ende
Say Warte, ich komme auch in den Bereich von Henry Matisse!
Inhibit 100
GlobalVar eatlock true
StartWalk
  if ((%robotpos.x > -65) & (%robotpos.x < 65) & (%robotpos.z > -65) & (%robotpos.z < 65)) then
     Goto FromKlee
  if ((%robotpos.x > -65) & (%robotpos.x < 65) & (%robotpos.z > 65) & (%robotpos.z < 195)) then
     Goto FromKandinsky
goto talkpos
Tag FromKandinsky
Go x 0, z 100, startwalk false, stopwalk false
Go x 0, z 30, startwalk false, stopwalk false
Tag FromKlee
Go x 30, z 0, startwalk false, stopwalk false
```

```
Go x 100, z 0, startwalk false, stopwalk false
Tag talkpos
Go x 105, z -25, startwalk false, stopwalk true
Inhibit -100
GlobalVar eatlock false
Tag ende
end


procedure Go x 0, z 0, startwalk true, stopwalk true
begin
Inhibit 100
  Localvar xposdif (%x-%robotpos.x)
  Localvar zposdif (%z-%robotpos.z)
  GotoPlace xpos %x, zpos %z, ticks 0.5*sqrt (%xposdif*%xposdif+%zposdif*%zposdif), startwalk
      %startwalk, stopwalk %stopwalk
Inhibit -100
end

procedure TurntoUser
begin
  TurnToPos xpos %userpos.x, zpos %userpos.z
  Wait slides 5
end

procedure TurnToPos xpos 0 ,zpos 0
begin
Inhibit 100
  Localvar xposdif ((%xpos)-(%robotpos.x))
  Localvar zposdif ((%zpos)-(%robotpos.z))
  if (%zposdif=0)&(%xposdif>=0) then Localvar angle 90
  if (%zposdif=0)&(%xposdif< 0) then Localvar angle -90

  if !((%zposdif)=0) then Localvar angle Evaluate (atan (%xposdif/%zposdif))
if ((%zposdif)<0) then LocalVar angle 180+%angle
  Rotate vector robotrot,axisy 1,direction %angle, slides 5 , wait false
Inhibit -100
end


procedure GotoPlace xpos 0 ,zpos 0, ticks 30, startwalk true, stopwalk true
begin
  TurnToPos xpos (%xpos), zpos (%zpos), ticks 5
  if %startwalk then StartWalk
  Translate vector robotpos , x %xpos, y %robotpos.y, z %zpos ,slides %ticks
  if %stopwalk then StopWalk
end

procedure StartWalk
begin
  Rotate vector arm1rot , axisx 1, axisy 0, axisz 0 ,direction (-40), slides 5, wait false
  Rotate vector arm2rot , axisx 1, axisy 0, axisz 0 ,direction 40, slides 5, wait false
  Rotate vector thigh1rot , axisx 1, axisy 0, axisz 0 ,direction (-50), slides 5, wait false
  Rotate vector thigh2rot , axisx 1, axisy 0, axisz 0 ,direction 30, slides 5, wait false
  Rotate vector shank1rot , axisx 1, axisy 0, axisz 0 ,direction 80, slides 5, wait false
  Rotate vector shank2rot , axisx 1, axisy 0, axisz 0 ,direction 0, slides 5, wait false
  Wait slides 6
  Rotate vector arm1rot , axisx 1, axisy 0, axisz 0 ,direction 40 (-40), slides 10, wait fal-
      se,cyclic true
  Rotate vector arm2rot , axisx 1, axisy 0, axisz 0 ,direction (-40) 40, slides 10, wait fal-
      se,cyclic true
  Rotate vector thigh1rot , axisx 1, axisy 0, axisz 0 ,direction 30 (-50), slides 10, wait fal-
      se,cyclic true
  Rotate vector thigh2rot , axisx 1, axisy 0, axisz 0 ,direction (-50) 30 , slides 10, wait fal-
      se, cyclic true
  Rotate vector shank1rot , axisx 1, axisy 0, axisz 0 ,direction 0 80, slides 10, wait fal-
      se,cyclic true
  Rotate vector shank2rot , axisx 1, axisy 0, axisz 0 ,direction 80 0, slides 10, wait false,
      cyclic true
end

procedure StopWalk
begin
  Rotate vector arm1rot , axisx 1, axisy 0, axisz 0 ,direction 0, slides 5,wait false
  Rotate vector arm2rot , axisx 1, axisy 0, axisz 0 ,direction 0, slides 5,wait false
  Rotate vector thigh1rot , axisx 1, axisy 0, axisz 0 ,direction 0, slides 5,wait false
  Rotate vector thigh2rot , axisx 1, axisy 0, axisz 0 ,direction 0, slides 5,wait false
  Rotate vector shank1rot , axisx 1, axisy 0, axisz 0 ,direction 0, slides 5,wait false
  Rotate vector shank2rot , axisx 1, axisy 0, axisz 0 ,direction 0, slides 5,wait false
```

```
  Wait slides 6
end


procedure Wave
begin
  Rotate vector arm1rot , axisx 0, axisy 0, axisz 1 , direction 150 190 150 190 150 0, slides 5
end

procedure Oh
begin
  Rotate vector arm1rot , axisx 1.6, axisy 1 axisz 0 , direction -150 -150 0, slides 5
end

procedure Shrug
begin
  Rotate vector arm1rot , axisx 0, axisy 0, axisz 1 , direction 50 50 0, slides 5, wait false
  Rotate vector arm2rot , axisx 0, axisy 0, axisz 1 , direction -50 -50 0, slides 5, wait false
  Wait slides 15
end

procedure Point
begin
  Rotate vector arm1rot, axisx 1, axisy 0, axisz 0 ,direction -90 -90 0, slides 5
end

procedure PointUser
begin
  Turntouser
  Rotate vector arm1rot, axisx 1, axisy 0, axisz 0 ,direction -90 -90 0, slides 5
end

procedure PointPos x 0, z 0
begin
  TurntoPos xpos %x, zpos %z
  Rotate vector arm1rot, axisx 1, axisy 0, axisz 0 ,direction -90 -90 0, slides 5
end


procedure Nod wait true
begin
  Rotate vector headrot, axisx 1, axisy 0, axisz 0 , direction 20 -20 20 -20 0, slides 5
  if %wait then Wait slides 25
end

procedure ShakeHead
begin
  Rotate vector headrot, axisx 0, axisy 1, axisz 0 , direction 20 -20 20 -20 0, slides 5
end

procedure eat value 5
begin
Inhibit 100
LocalVar posx %robotpos.x
LocalVar posz %robotpos.z
GlobalVar eatlock true
StartWalk
  if ((%robotpos.x > 65) & (%robotpos.x < 195) & (%robotpos.z > -65) & (%robotpos.z < 65)) then
      Goto FromMatisse
  if ((%robotpos.x > -65) & (%robotpos.x < 65) & (%robotpos.z > 65) & (%robotpos.z < 195)) then
      Goto FromKandinsky
Goto eatpos
Tag FromMatisse
Go x 100, z 0, startwalk false, stopwalk false
Go x 30, z 0, startwalk false, stopwalk false
goto eatpos
Tag FromKandinsky
Go x 0, z 100, startwalk false, stopwalk false
Go x 0, z 30, startwalk false, stopwalk false
Tag eatpos
Go x 23, z 23, startwalk false, stopwalk true
TurnToPos x 25, z 25
Wait slides 5
 Oh
  Globalvar food Evaluate (%food+%value)
Say Schon besser
Go x %posx, z %posz, startwalk true, stopwalk true
GlobalVar eatlock false
Inhibit -100
```

```
end

// Einige Callbacks (EventIns)

procedure initialize
begin
GlobalVar eatlock false
GlobalVar food 14
end

procedure hunger_tick
begin
   GlobalVar food Evaluate (%food-1)
   if ((%food <= 10) & !%eatlock)  then [Say Einen Moment bitte. Ich gehe schnell was es-
      sen\nEat]
end
```

5.3 Dialogkonfiguration für das Klee-Schema

```
/**************************************
** Dialogverhalten zum Thema Paul Klee **
**************************************/

"%1 Matisse %2" "Wir sollten zuerst zum Matisse-Bereich gehen."
"%1 Kandinsky %2" "Wir sollten zuerst zum Kandinsky-Bereich gehen."

"%1 wer %2 Kuenstler %3" "Diese Werke stammen von Paul Klee"
"%1 wer %2 Paul %3" "Diese Werke stammen von Paul Klee"
"%1 wer %2 Klee %3" "Diese Werke stammen von Paul Klee"

"%1 wie heisst %2 Kuenstler %3" "Der Name des Künstlers ist Paul Klee"
"Wann hat %1 Klee gelebt %2" "Paul Klee ist am 18.12.1879 in Muenchenbuchsee (bei Bern) geboren.
      Gestorben ist er am 29.6.1940 in Locarno-Muralto."
"Wann hat %1 der Kuenstler gelebt %2" "Paul Klee ist am 18.12.1879 in Muenchenbuchsee (bei Bern)
      geboren. Gestorben ist er am 29.6.1940 in Locarno-Muralto."
"Wann hat er gelebt %2" "Paul Klee ist am 18.12.1879 in Muenchenbuchsee (bei Bern) geboren. Ge-
      storben ist er am 29.6.1940 in Locarno-Muralto."
"Welche Art %1 er %2" "Anfangs war Klee Zeichner, seit 1914 auch Maler. Seine Bilder mit phanta-
      stischen Gestalten und Formen knuepfen fast immer an Gegenstaendliches an und machen
      traumhafte Vorgaenge sichtbar."
"Welche Art %1 Klee %2" "Anfangs war Klee Zeichner, seit 1914 auch Maler. Seine Bilder mit phan-
      tastischen Gestalten und Formen knuepfen fast immer an Gegenstaendliches an und machen
      traumhafte Vorgaenge sichtbar."
"Was fuer einen Stil %1 er %2" "Anfangs war Klee Zeichner, seit 1914 auch Maler. Seine Bilder
      mit phantastischen Gestalten und Formen knuepfen fast immer an Gegenstaendliches an und
      machen traumhafte Vorgaenge sichtbar."
"Was fuer einen Stil %1 Klee %2" "Anfangs war Klee Zeichner, seit 1914 auch Maler. Seine Bilder
      mit phantastischen Gestalten und Formen knuepfen fast immer an Gegenstaendliches an und
      machen traumhafte Vorgaenge sichtbar."
"%1 Ausdrucksmittel %2" "Klee formuliert seine sensible Bildsprache oft durch Chiffren und Buch-
      staben, in Bewegungsmotiven durch Pfeile, und macht so seine hohe Musikalitaet spuerbar."
"%1 Bildsprache %2" "Klee formuliert seine sensible Bildsprache oft durch Chiffren und Buchsta-
      ben, in Bewegungsmotiven durch Pfeile, und macht so seine hohe Musikalitaet spuerbar."
"%1 anderen Kuenstler %2" "Seit seiner Muenchener Zeit (1906) war Klee mit Kuenstlern des Blauen
      Reiters befreundet, spaeter (ab 1912) in Paris erhielt Klee Anregungen von Picasso,
      Rousseau und Delaunay. 1914 war er mit Macke in Tunis."
"%1 andere Kuenstler %2" "Seit seiner Muenchener Zeit (1906) war Klee mit Kuenstlern des Blauen
      Reiters befreundet, spaeter (ab 1912) in Paris erhielt Klee Anregungen von Picasso,
      Rousseau und Delaunay. 1914 war er mit Macke in Tunis."
"%1 Bauhaus %2" "1921-1931 war Paul Klee als Meister am Bauhaus in Weimar und Dessau."
"%1 1 %2" "Das Bild mit der Nummer 1 hat den Titel 'Embrace'. Klee malte dieses Bild 1939 mit
      Wasserfarbe und Oel auf Papier." PointPos x -20, z 0
"%1 2 %2" "Das Bild mit der Nummer 2 hat den Titel 'Insula Dulcamara'. Klee malte dieses Bild
      1938 mit Oel auf Zeitungspapier, aufgezogen auf Sackleinwand." PointPos x 0, z 0
"%1 3 %2" "Das Bild mit der Nummer 3 hat den Titel 'Red and White Domes'. Klee malte dieses Bild
      1914 mit Wasserfarbe und Koerperfarbe auf japanisches Pergament, aufgezogen auf Sackklein-
      wand." PointPos x 20, z 0
```

5.4 Dialogkonfiguration für das Kandinsky-Schema

```
/**************************************
```

```
** Dialogverhalten zum Thema Wassily Kandinsky **
****************************************/

"%1 Matisse %2" "Wir sollten zuerst zum Matisse-Bereich gehen."
"%1 Klee %2" "Wir sollten zuerst zum Klee-Bereich gehen."

"%1 wer %2 Kuenstler %3" "Diese Werke stammen von Wassily Kandinsky"
"%1 wer %2 Wassily %3" "Diese Werke stammen von Wassily Kandinsky"
"%1 wer %2 Kandinsky %3" "Diese Werke stammen von Wassily Kandinsky"

"%1 wie heisst %2 Kuenstler %3" "Der Name des Kuenstlers ist Wassily Kandinsky"
"Wann hat %1 Kandinsky gelebt %2" "Wassily Kandinsky ist am 4.12.1866 in Moskau geboren. Gestor-
      ben ist er am 13.12.1944 in Neuilly-sur-Seine."
"Wann hat %1 der Kuenstler gelebt %2" "Wassily Kandinsky ist am 4.12.1866 in Moskau geboren. Ge-
      storben ist er am 13.12.1944 in Neuilly-sur-Seine."
"Wann hat er gelebt %2" "Wassily Kandinsky ist am 4.12.1866 in Moskau geboren. Gestorben ist er
      am 13.12.1944 in Neuilly-sur-Seine."
"Welche Art %1 er %2" "Kandinsky war 1910 der erste, der gegenstandslose Bilder zu malen begann.
      Auf heftig bewegte Farbphantasien folgten Kompositionen aus geometrischen Farbflaechen und
      richtungsbetonenden Linien, die Raum- und Bewegungsvorstellungen vermitteln."
"Welche Art %1 Kandinsky %2" "Kandinsky war 1910 der erste, der gegenstandslose Bilder zu malen
      begann. Auf heftig bewegte Farbphantasien folgten Kompositionen aus geometrischen Farb-
      flaechen und richtungsbetonenden Linien, die Raum- und Bewegungsvorstellungen vermitteln."
"Was fuer einen Stil %1 er %2" "Kandinsky war 1910 der erste, der gegenstandslose Bilder zu ma-
      len begann. Auf heftig bewegte Farbphantasien folgten Kompositionen aus geometrischen
      Farbflaechen und richtungsbetonenden Linien, die Raum- und Bewegungsvorstellungen vermit-
      teln."
"Was fuer einen Stil %1 Kandinsky %2" "Kandinsky war 1910 der erste, der gegenstandslose Bilder
      zu malen begann. Auf heftig bewegte Farbphantasien folgten Kompositionen aus geometrischen
      Farbflaechen und richtungsbetonenden Linien, die Raum- und Bewegungsvorstellungen vermit-
      teln."
"%1 anderen Kuenstler %2" "Kandinsky kam 1896 als Schueler von F.v. Stuck nach Muenchen, war
      Mitbegruender des blauen Reiters, hatte von 1918-1921 diverse Kontakte in Russland und
      ging schliesslich 1933 nach Neuilly-sur-Seine bei Paris."
"%1 andere Kuenstler %2" "Kandinsky kam 1896 als Schueler von F.v. Stuck nach Muenchen, war Mit-
      begruender des blauen Reiters, hatte von 1918-1921 diverse Kontakte in Russland und ging
      schliesslich 1933 nach Neuilly-sur-Seine bei Paris."
"%1 Schriften %2" "Kandinsky schrieb 1912 'Ueber das Geistige in der Kunst' und 1926 'Punkt und
      Linie zur Flaeche'."
"%1 Bauhaus %2" "Seit 1922 war Kandinsky am Bauhaus in Weimar und Dessau."
"%1 1 %2" "Das Bild mit der Nummer 1 hat den Titel 'Autumn in Bavaria'. Kandinsky malte dieses
      Bild 1908 mit Oel auf Kartonpapier." PointPos x -20, z 130
"%1 2 %2" "Das Bild mit der Nummer 2 hat den Titel 'On White II'. Kandinsky malte dieses Bild
      1923 mit Oel auf Leinwand." PointPos x 0, z 130
"%1 3 %2" "Das Bild mit der Nummer 3 hat den Titel 'Yellow, Red, Blue'. Kandinsky malte dieses
      Bild 1925 mit Oel auf Leinwand." PointPos x 20, z 130
```

5.5 Dialogkonfiguration für das Matisse-Schema

```
/**************************************
** Dialogverhalten zum Thema Henri Matisse **
**************************************/

"%1 Klee %2" "Wir sollten zuerst zum Klee-Bereich gehen."
"%1 Kandinsky %2" "Wir sollten zuerst zum Kandinsky-Bereich gehen."


"%1 wer %2 Kuenstler %3" "Diese Werke stammen von Henri Matisse"
"%1 wer %2 Henri %3" "Diese Werke stammen von Henri Matisse"
"%1 wer %2 Matisse %3" "Diese Werke stammen von Henri Matisse"

"%1 wie heisst %2 Kuenstler %3" "Der Name des Kuenstlers ist Henri Émile Benoît Matisse, ein
      franszoesischer Maler und Graphiker."
"Wann hat %1 Matisse gelebt %2" "Henri Matisse ist am 31.12.1869 in Le Cateau geboren. Er starb
      am 3.11.1954 in Cimiez bei Nizza."
"Wann hat %1 der Kuenstler gelebt %2" "Henri Matisse ist am 31.12.1869 in Le Cateau geboren. Er
      starb am 3.11.1954 in Cimiez bei Nizza."
"Wann hat er gelebt %2" "Henri Matisse ist am 31.12.1869 in Le Cateau geboren. Er starb am
      3.11.1954 in Cimiez bei Nizza."
"Welche Art %1 er %2" "Als sich Matisse 1890 der Malerei zuwandte, malte er impressionistisch.
      Spaeter begann er unter dem Einfluss der Neoimpressionisten seit 1905 starke, reine Farb-
      flaechen gegeneinander zu stellen. Matisse blieb gegenstaendlich und deformierte nur zu-
      gunsten des Ausdrucks. Skulpturen ergaenzen das Werk."
"Welche Art %1 Matisse %2" "Als sich Matisse 1890 der Malerei zuwandte, malte er impressioni-
      stisch. Spaeter begann er unter dem Einfluss der Neoimpressionisten seit 1905 starke, rei-
```

```
      ne Farbflaechen gegeneinander zu stellen. Matisse blieb gegenstaendlich und deformierte
      nur zugunsten des Ausdrucks. Skulpturen ergaenzen das Werk."
"Was für einen Stil %1 er %2" "Als sich Matisse 1890 der Malerei zuwandte, malte er impressioni-
      stisch. Spaeter begann er unter dem Einfluss der Neoimpressionisten seit 1905 starke, rei-
      ne Farbflaechen gegeneinander zu stellen. Matisse blieb gegenstaendlich und deformierte
      nur zugunsten des Ausdrucks. Skulpturen ergaenzen das Werk."
"Was für einen Stil %1 Matisse %2" "Als sich Matisse 1890 der Malerei zuwandte, malte er impres-
      sionistisch. Spaeter begann er unter dem Einfluss der Neoimpressionisten seit 1905 starke,
      reine Farbflaechen gegeneinander zu stellen. Matisse blieb gegenstaendlich und deformierte
      nur zugunsten des Ausdrucks. Skulpturen ergaenzen das Werk."
"%1 anderen Kuenstler %2" "1893 wurde Matisse Schueler von G. Moreau. Spaeter wurde er besonders
      angeregt von P. Cézanne und ueberwandt den Impressionismus seiner Fruehzeit. Matisse und
      die sich um ihn gruppierenden Maler des Neoimpressionismus wurden Fauves (Wilde) genannt."
"%1 andere Kuenstler %2" "1893 wurde Matisse Schueler von G. Moreau. Spaeter wurde er besonders
      angeregt von P. Cézanne und ueberwandt den Impressionismus seiner Fruehzeit. Matisse und
      die sich um ihn gruppierenden Maler des Neoimpressionismus wurden Fauves (Wilde) genannt."
"%1 1 %2" "Das Bild mit der Nummer 1 hat den Titel 'La Musique'. Matisse malte dieses Bild 1939
      mit Oel auf Leinwand." PointPos x 130, z -20
"%1 2 %2" "Das Bild mit der Nummer 2 hat den Titel 'Le bonheur de vivre'. Matisse malte dieses
      Bild 1905-1906 mit Oel auf Leinwand." PointPos x 130, z 0
"%1 3 %2" "Das Bild mit der Nummer 3 hat den Titel 'Notre-Dame, une fin d'après-midi'. Matisse
      malte dieses Bild 1902 mit Oel auf Papier aufgezogen auf Leinwand." PointPos x 130, z 20
```