

Notizen zu Open Source Software im Robot-Lab

Martin Sukale, TIS, HAW - Hamburg

11. Juni 2006

Inhaltsverzeichnis

1	Overview	3
2	Flowdesigner mit Robotflow	3
2.1	Beschreibung	3
2.2	screenshot	4
2.3	Installation	4
2.4	Links	4
2.5	getestete Versionen	4
2.6	Notizen zum Einsatz im RobotLab	4
3	player/stage	5
3.1	Beschreibung	5
3.2	screenshot	5
3.3	Installation	6
3.4	Links	6
3.5	getestete Versionen	6
3.6	Notizen zum Einsatz im RobotLab	6
4	gazebo	6
4.1	Beschreibung	6
4.2	screenshot	7
4.3	Installation	7
4.4	Links	7
4.5	getestete Versionen	7
4.6	Notizen zum Einsatz im RobotLab	7
5	mapviewer	7
5.1	Beschreibung	7
5.2	screenshot	8
5.3	Installation	8
5.4	Links	8
5.5	getestete Versionen	8
5.6	Notizen zum Einsatz im RobotLab	8

6	HAW World	8
6.1	3ds CAD-Objekte	9
6.2	Wände bauen	9
6.3	Terrain Files	11
6.4	Die HAW-Pio2s	12
7	Simulationsaufbau	13
7.1	connecting things	14
8	Ausblick	16

Zusammenfassung

Ich habe mich im SS2006 mit Alternativen zu den bestehenden Robot-Lab SDKs (e.g. Saphira, Lego) beschäftigt. Dabei bin ich auf interessante Projekte anderer Hochschulen gestossen, die - meist unter Linux laufende - relativ weit entwickelte Lösungen zur Programmierung und Simulation von Robots verwenden und frei zur Verfügung stellen. Die Zielsetzung der vorgestellten Projekte dreht sich vor allem um die Modularisierung und Wiederverwendbarkeit von Entwicklungen auf dem Gebiet der Robotic durch plattformunabhängige und erweiterbare Lösungen. Ich habe es mir nicht nehmen lassen, diese auszuprobieren. Hier meine gesammelten Erfahrungen.

1 Overview

Es sollen einige verbreitete Open Source Robotic Tools vorgestellt werden. Als wichtig erschienen mir:

- Flowdesigner mit Robotflow
- player/stage
- gazebo
- MARIE
- MapViewer

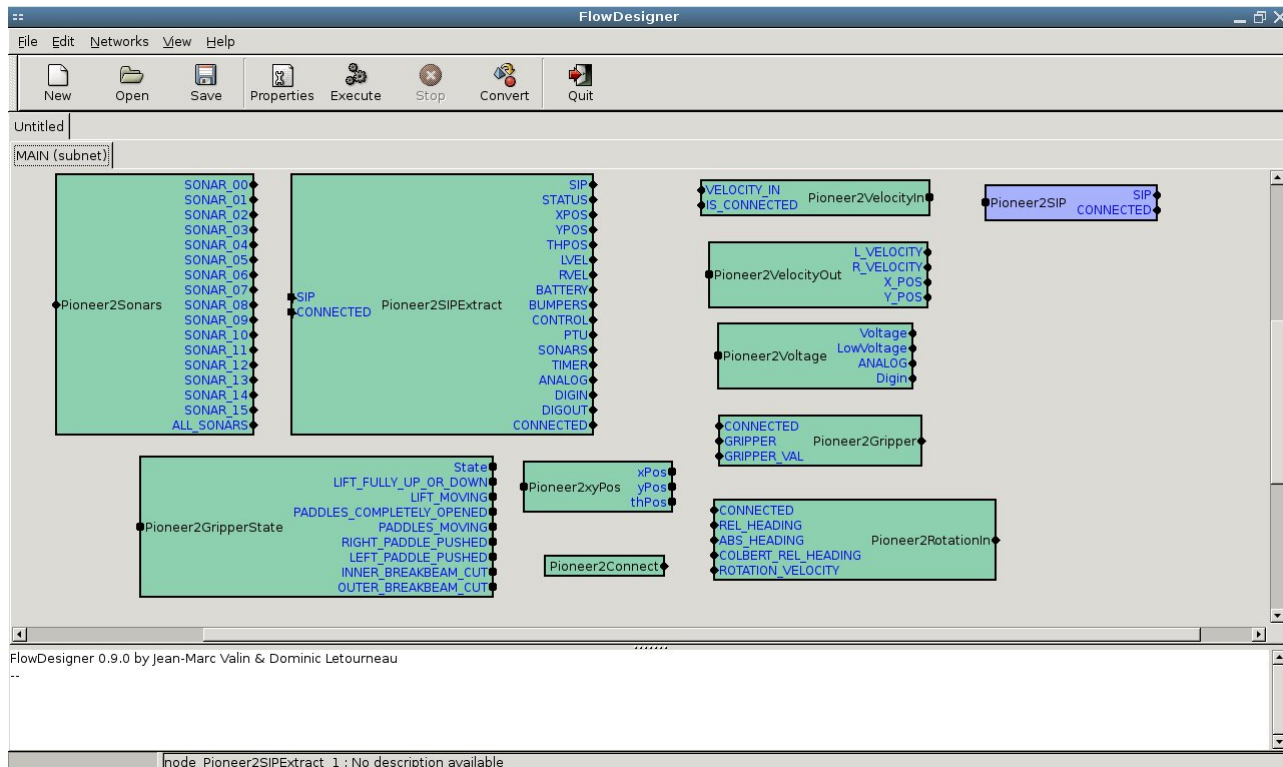
Im folgenden ein paar Notizen zu den einzelnen Tools.

2 Flowdesigner mit Robotflow

2.1 Beschreibung

Datenflussorientierte Programmierumgebung. Visuelle Erstellung von komplexen Netzen aus sog. Nodes. Zustands- (FSM-Generator) und Verhaltensorientiert. Erweiterbar für Robots mit Robotflow-Modul. Generiert c++ Code. Unterstützt p2os - alle wichtigen Pioneer-Funktionen. Spielt mit "player/stage" zusammen...

2.2 screenshot



2.3 Installation

tricky... Hab ich erst aus einer veralteten debian quelle versucht zu installieren: negativ. Also latest source-tarball von sourceforge. Funktioniert nur mit player-1.5..also auch den "per Hand" kompiliert. 1. Schritt: FlowDesigner Quellen laden und kompilieren. (the usual `./configure && make && make install`) 2. Schritt:

```
tar -xzf RobotFlow...
cd RobotFlow...
./configure --with-player=PREFIX (für PREFIX player-Verzeichnis eintragen)
make && make install
```

2.4 Links

- <http://robotflow.sourceforge.net/> Downloads, Docs, etc...
- <http://robotflow.sourceforge.net/nodes.html> Beschreibung der Nodes, interne Hilfe gibt wenig Infos...

2.5 getestete Versionen

robotflow 0.2.6, flowdesigner 0.9.0

2.6 Notizen zum Einsatz im RobotLab

Dieses Tool scheint sehr mächtig zu sein. Es erinnert an Simulink/Labview/MAX-MSP. Das mitgelieferte Beispiel: Schrifterkennung mit neuronalem Netzwerk und automatischer

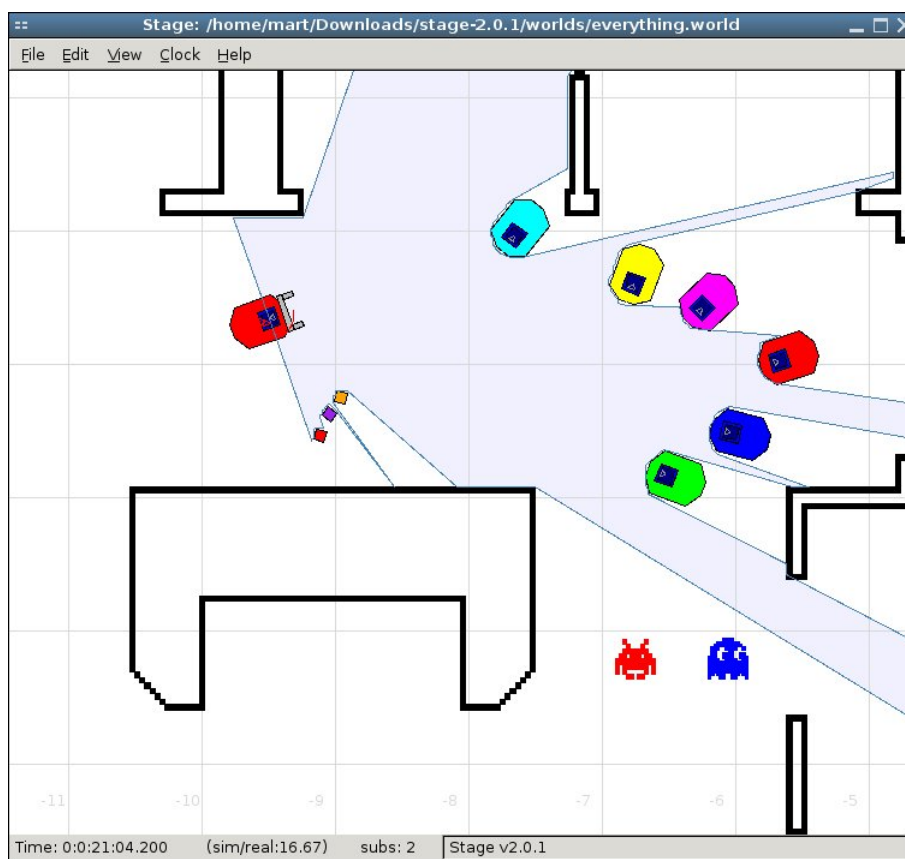
Pan/Tilt-steuerung ist komplexer als alles, was wir von Saphira aus gemacht haben... Vom Ansatz her ist eine datenflussorientierte visuelle Sprache meiner Meinung nach sehr geeignet für Robotics. Von den Sensoren erhalten wir kontinuierlich Werte die wir in geeigneter Reihenfolge durch bestimmte Knoten verarbeiten lassen wollen. Filtering, Windowing, Analyse usw. Diese Blöcke lassen sich einfach aus dem Menu zusammenklicken und verbinden... Auch das Generieren von c++-Code aus diesen Netzen ist ein feature, was nicht bei allen solchen Dataflow-IDEs selbstverständlich ist... Fazit: Wir sollten Flowdesigner/RobotFlow im Lab installieren und ein paar Tests machen.

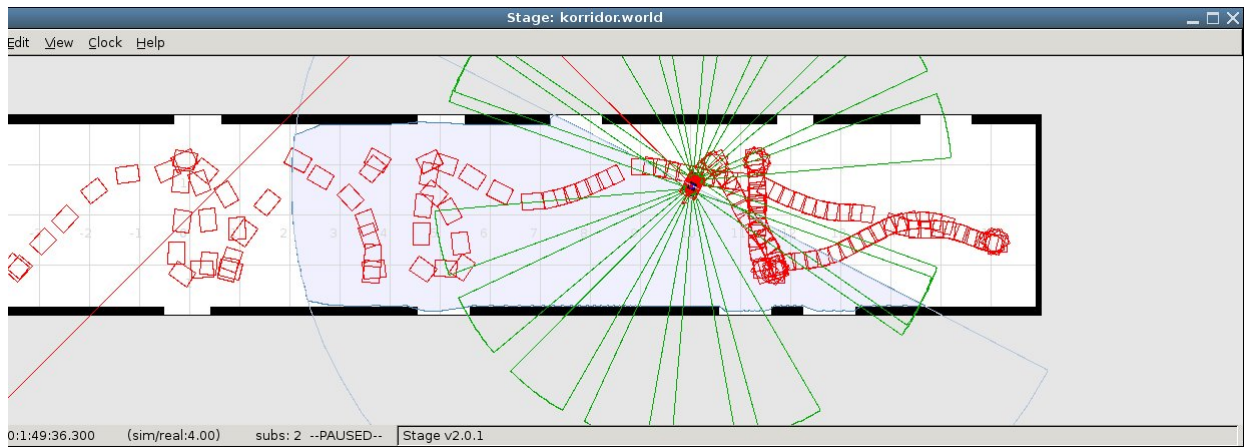
3 player/stage

3.1 Beschreibung

Player is the software package that most students use to interface with the pioneer and can also be used to interface with other robots. In addition to providing a standard interface to different robots, Player also ships with two simulators: Stage, a 2D simulator good for office environments and laser simulation, and Gazebo, a 3D simulation that uses a 3D physics engine and can be used for generating simple camera images. Stage is by far more stable.

3.2 screenshot





Pio2 mit Laser und Kollisionsvermeidung im Korridor der HAW

3.3 Installation

siehe Projektpage

3.4 Links

<http://playerstage.sourceforge.net/>

3.5 getestete Versionen

stage 2.0.1

3.6 Notizen zum Einsatz im RobotLab

Das neue stage, startet mit `stest worldfile robotname`. Sieht wesentlich besser aus als die Vorgänger. Simulation der Robots (`fahr-nicht-gegen-die-wand-behavior`), Verschieben der Objekte, Verschiedene Views sind möglich. Player wird von Stage erkannt und automatisch gestartet.

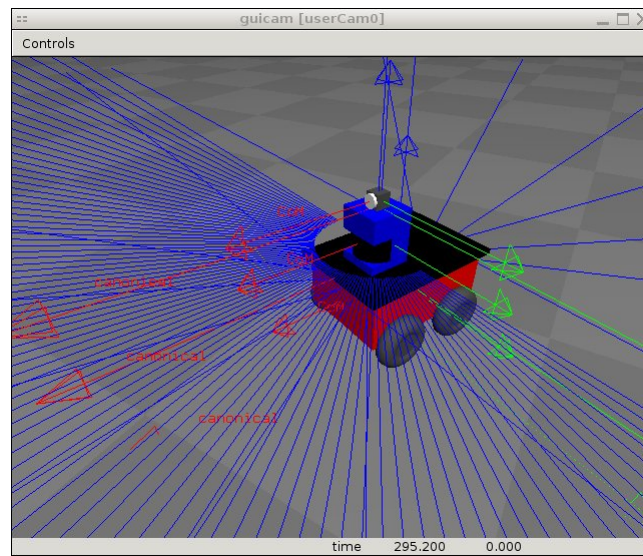
```
stest korridor.world robot1
```

4 gazebo

4.1 Beschreibung

Gazebo ist eine 3d-Simulation für Robots und deren "worlds". Baut auf `player/stage` auf. Pio2 ist schon dabei und einige andere Modelle (`shrimp`, `heli`, etc.).

4.2 screenshot



4.3 Installation

Für Gazebo habe ich keine prekompilierten Pakete gefunden. Die Installation per Hand dauert lange. Es werden viele spezielle Versionen von Bibliotheken benötigt (gdal, grass, ode, opengl, lib3ds etc...siehe README) Auf Debian/unstable hat es aber funktioniert. Ich habe versucht die meisten libs aus den Debian-Quellen zu verwenden, nur gdal und grass habe ich aus den aktuellen sourcen kompiliert. Wichtig: man braucht unbedingt Hardware-3d-Beschleunigung, funktioniert nicht mit den mesa-libs sondern z.B. nur mit dem proprietären ATI-Treibern des Herstellers. Man kann das überprüfen mit: glxinfo (direct rendering: yes), bzw. wenn es hakt, dann sind es die falschen Treiber. xorg.conf modifizieren! Oder z.B. aticonfig benutzen.

4.4 Links

<http://playerstage.sourceforge.net/index.php?src=gazebo>

4.5 getestete Versionen

gazebo 0.6.0

4.6 Notizen zum Einsatz im RobotLab

Gazebo started man als erstes am besten mit dem GUI: wxgazebo. Man muss dem Programm auf der Kommandozeile ein World-file übergeben (siehe z.B. in /usr/share/gazebo/worlds), gazebo startet dann und man kann ein bisschen mit den Robots rumspielen (velocity, angle etc..). TODO: Korridor der HAW in gazebo laden und mit dem Pioneer rumfahren...

5 mapviewer

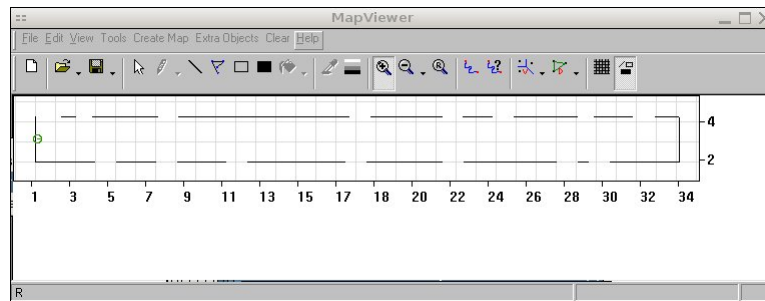
5.1 Beschreibung

Map-Editor - Konvertiert Map/World files und zeigt sie an z.B. stage -> spahira und vice versa. Im visuellen Editor lassen sich auch Objekte mit der Maus verschieben, oder

komplett neue Maps zeichnen (oder sogar zufällig erzeugen). Unterstützt die gängigen Robotics-Formate. Oder besser:

Map Viewer is a application for the creation, editing and analysis of maps, primarily in the field of mobile robotics, but not restricted to that domain. It is used for the creation, viewing and editing of vector maps and occupancy grids. It can load and save maps in a number of different formats, from many of the leading robot-simulation and control applications, such as CARMEN (<http://www-2.cs.cmu.edu/~carmen/>), Saphira (<http://robots.activmedia.com>), Player/Stage and BeeSoft, as well as having the ability to import/export images. Map Viewer can be used to edit maps, either by painting on them, or by drawing vectors on the map. It can translate between grid and vector maps, perform various statistical analysis functions on maps, average one or more maps, calculate Voronoi diagrams of maps and plan paths in a map, and randomly create new maps complete with obstacles, doors, rooms and corridors.

5.2 screenshot



5.3 Installation

Windows-Binary zum Download bereitgestellt. Läuft aber auch unter Linux mit WINE (getestet auf debian/unstable) Bibliothek ist OpenSource!

5.4 Links

<http://mapviewer.skynet.ie/main.html>

5.5 getestete Versionen

MapViewer 2.2

5.6 Notizen zum Einsatz im RobotLab

Ich habe ohne Probleme das Korridor wld.-file einlesen können. Konversion in stage-file (.world) dauert recht lange (bitmap muss gerendert werden). Und man muss die richtigen Einstellungen für resolution und threshold finden... Aber: scheint zu funktionieren! Fazit: nützlich.

6 HAW World

Das Interessanteste bzw. "präsentationswirksamste" Tool (Toy) ist sicher Gazebo. Um Gazebo in der HAW einzusetzen, war es jedoch zunächst notwendig eine geeignete Simulationsumgebung zu erstellen. Ausgehend von dem bereits vermessenen HAW-Korridor, der als Safira-World auf dem FTP Server vorliegt, kann man mit MapViewer ein Bitmap-File generieren (pnm):



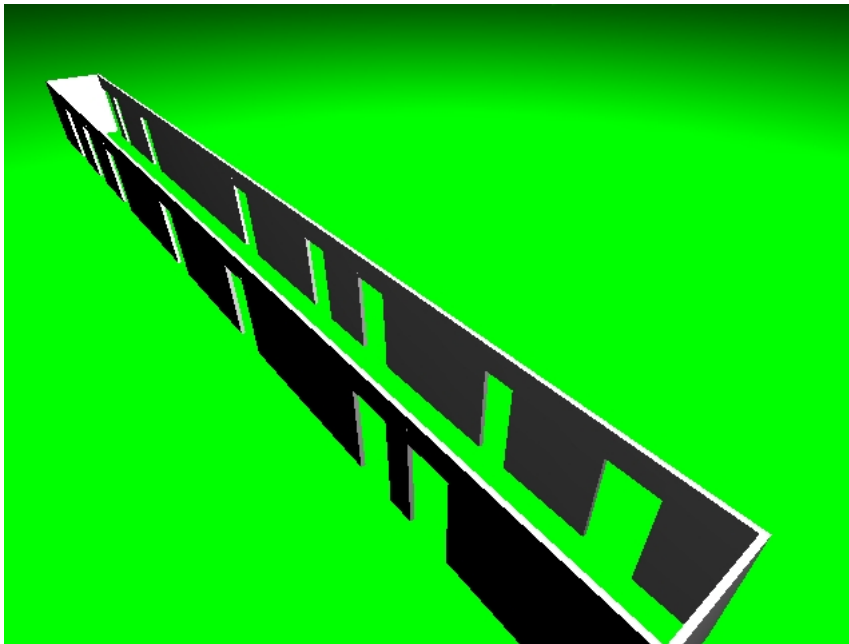
Dieses beinhaltet alle Informationen, die notwendig sind eine geeignete 2d-World z.B. für Stage zu erzeugen (Bitmaps können direkt eingebunden werden). Für Gazebo muß jedoch eine 3. Dimension hinzugefügt werden. Dieses habe ich auf unterschiedliche Arten versucht:

6.1 3ds CAD-Objekte

Es gibt die Möglichkeit .3ds-Files (The 3D Studio File Format) in Gazebo als "simple solids" einzubinden. Ich habe also ein 3ds File erstellt (geeignete Werkzeuge wären hier z.B. Blender oder Rhinoceros, SolidWorks, 3dStudio etc...). Das Bitmap kann z.B. "Extrudiert" werden... 3ds lässt sich in Gazebo wie folgt einbinden:

```
<model:SimpleSolid>
  <xyz>0 0 0</xyz>
  <color>0.5 0.5 0.1</color>
  <fiducial>0.5</fiducial>
  <skinFile>11thfloor.3ds</skinFile>
  <mass>20000</mass>
  <retro>0.5</retro>
</model:SimpleSolid>
```

Das sieht dann so aus:



Es macht allerdings keinen großen Sinn, ein so großes (20to schweres) Objekt zu simulieren, da Gazebo Performanz-Probleme bekommt. Zu dem habe ich keine Möglichkeit gefunden, die Wände "undurchdringbar" zu machen.

6.2 Wände bauen

2. Versuch eine Welt aufzubauen: Wände aus "simple solids" mit Box-Shape. Für jedes Wand Element wurde eine 3d-Box erstellt:

```

<model:SimpleSolid>
  <xyz>1.51 0 0</xyz>
  <size>3.020 0.15 2.1</size>
  <shape>box</shape>
  <color>1 1 0</color>
  <mass>10000</mass>
</model:SimpleSolid>

```

```

<model:SimpleSolid>
  <xyz>5.08 0 0</xyz>
  <size>2.03 0.15 2.1</size>
  <shape>box</shape>
  <color>1 1 0</color>
  <mass>10000</mass>
</model:SimpleSolid>

```

usw...

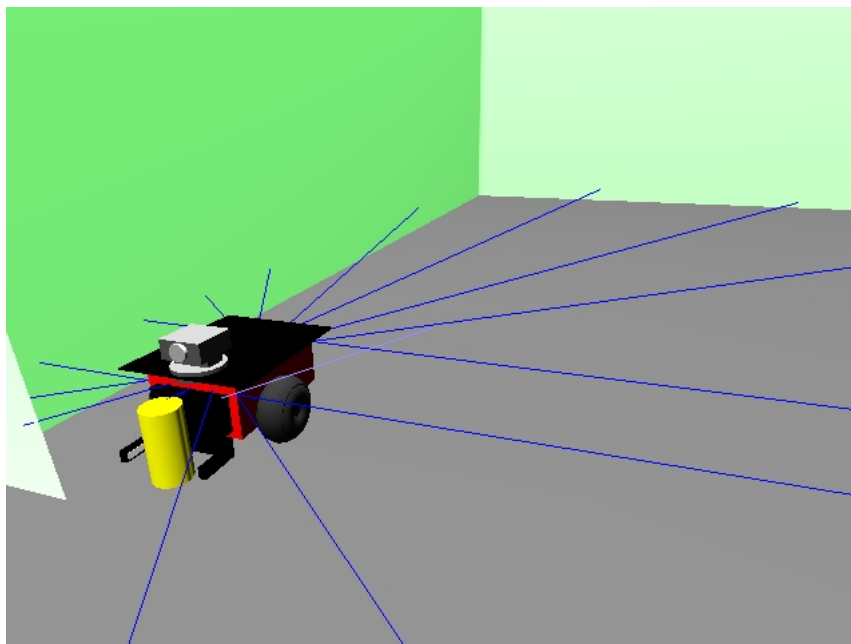
Macht viel Arbeit und bringt nicht das gewünschte Ergebnis: Die Wände werden durch die Open Dynamics Engine simuliert, d.h. Sie können aufgrund der Gravitation (die sich übrigens in Gazebo gesondert einstellen lässt) fallen oder umkippen und wackeln permanent. Es gibt einige Parameter (z.B. Gravitation ausschalten etc, oder Körpermasse), die das "Wandverhalten" verbessern. Die Simulation ist aber trotzdem sehr rechenintensiv.

Simple Solid Objekte sollte man nur für einfache Hindernisse o.ä. verwenden. Z.B. für die gelbe Papprolle im Labor:

```

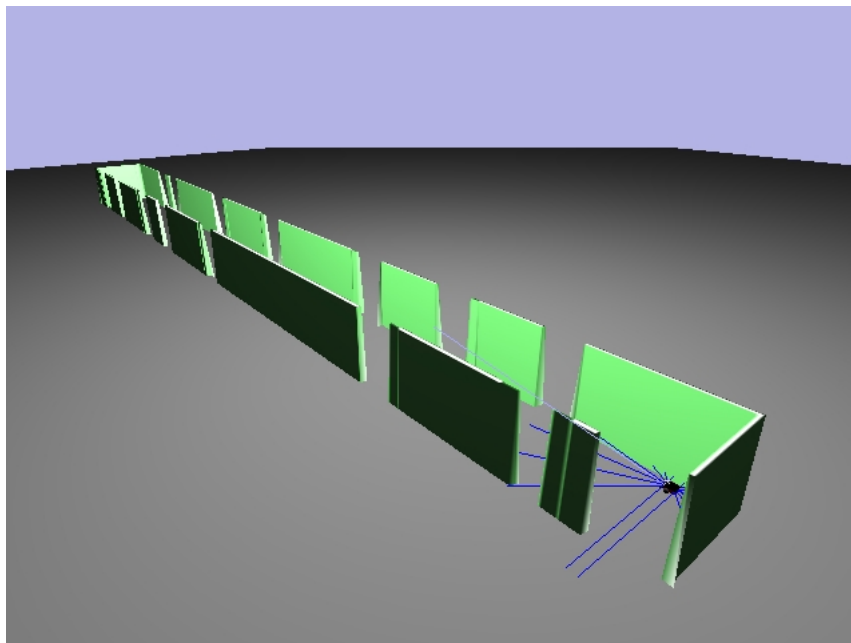
<model:SimpleSolid>
  <xyz>6 1 0</xyz>
  <size>0.1 0.2</size>
  <shape>cylinder</shape>
  <color>1 1 0</color>
  <mass>0.2</mass>
</model:SimpleSolid>

```



6.3 Terrain Files

Mit der GDAL (Geospatial Data Abstraction Library) steht uns in Gazebo ein Werkzeug zur Verfügung zum Erstellen von 2.5-D (keine Tunnel) - Objekten aus Raster-Formaten also z.B. aus .png Bitmaps. Eigentlich wohl nicht für Wände gedacht sondern für GEO-Daten (siehe auch www.maptools.org). Das Gazebo-Paket bietet mit gzbuilder ein tool zum Erstellen von Robotersimulationsumgebungen im gzb-Format (sog. Terrain files). Helle Pixel werden von diesem als Erhöhungen im Gelände dargestellt. Nimmt man das HAW-Korridor-Bitmap als Grundlage ergibt sich eine Karte aus Vektoren (Trimesh), die in das World-File eingebettet werden kann:



Die Kanten sind nicht ganz so scharf wie bei den Simple Solids. Es gibt aber einige command-line Parameter des gzbuilder-tools, mit denen hier noch Details (Wandhöhe, Fehlerauflösung, Skalierung, Offsets) optimiert werden könnten.

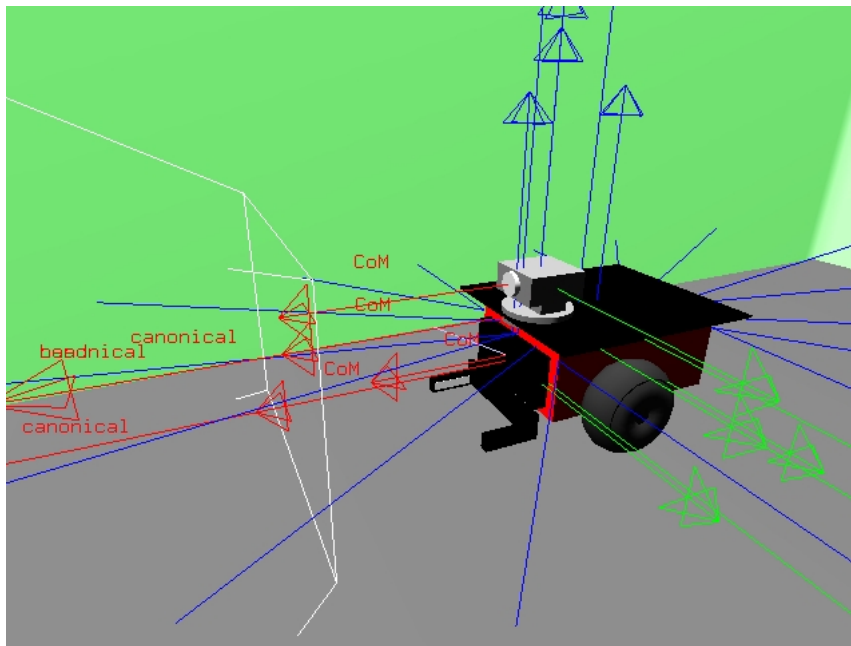
Als Vorlage zur Bildung des Terrains diente dieses aus dem 3ds-File gerenderte Bitmap (Renderengine: Rhinoceros von MacNeal, besser wäre evtl. gleich mit AutoCAD o.ä. ein 2D-File zu erzeugen)



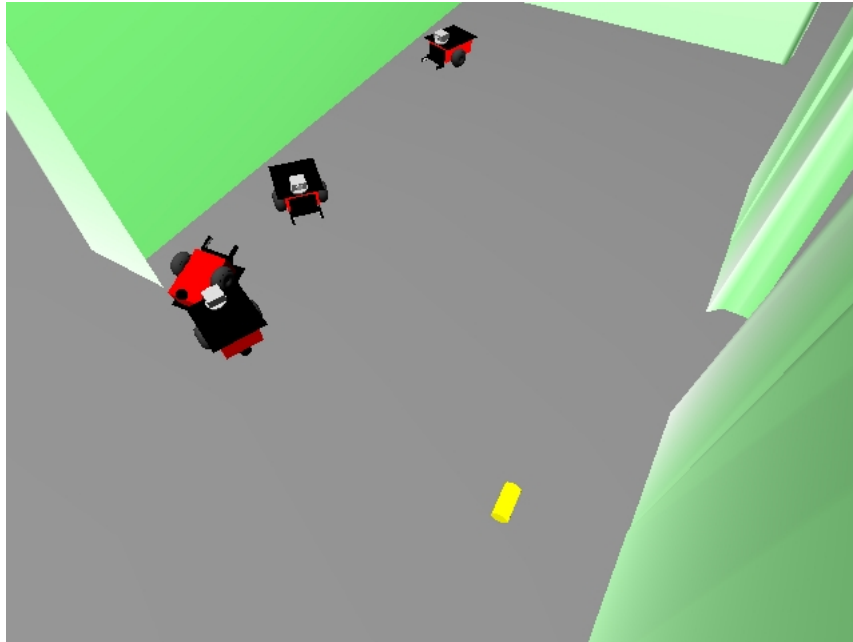
6.4 Die HAW-Pio2s

Die Gazebo-Entwickler haben Pio2 Modelle implementiert. Diese können an unsere Realität angepasst werden. Ein Pio mit Sony-Camera und Gripper an der vorderen Plattform lässt sich einfach im World-File erzeugen:

```
<model:Pioneer2DX>
  <id>keule</id>
  <xyz>2 1.000 0.234</xyz>
  <rpy>-0 0 0</rpy>
  <model:SonyVID30>
    <id>ptz1</id>
    <xyz>0.1 0.0 0</xyz>
  </model:SonyVID30>
  <model:Pioneer2Gripper>
    <id>gripper1</id>
    <xyz>0.14003 0 -0.0975</xyz>
  </model:Pioneer2Gripper>
</model:Pioneer2DX>
```



Roboter lassen sich auch kopieren (mit einer anderen ID versehen), so daß auch Multi-Robot Umgebungen dargestellt werden können:

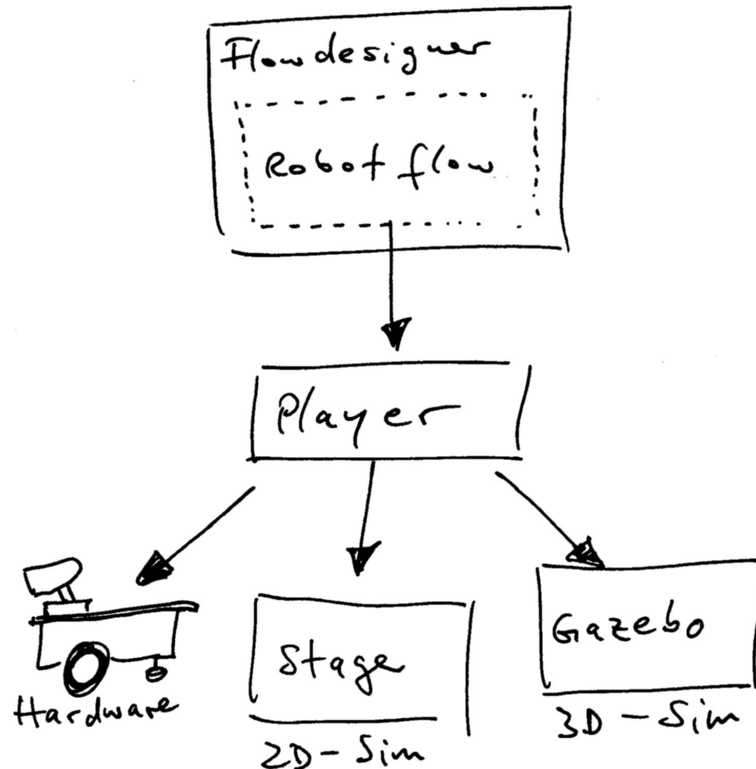


Man sollte darauf achten, dass man nicht Objekte bzw. Robots übereinander plaziert, wie auf dem obigen Bild. Sie kippen dann um (wie in der Realität) oder verhaken sich. Leider ist bei Multibotsimulationen viel Rechenleistung nötig. 2d-Stage-Simulation wären hier evtl. geeigneter.

Damit ist der Korridor des 11. Stockwerks im Groben nachgebildet.

7 Simulationsaufbau

Nachdem nun die Programme installiert sind und der 11.Stock simuliert werden kann (sowohl von Stage als auch von Gazebo), muss eine geeignete Verschaltung der einzelnen Komponenten vorgenommen werden. Grundsätzliches Schema:



Flowdesigner agiert hier als Programmierumgebung, Player als Robot-Server, der die darunterliegenden Schichten abstrahiert (Hardware Abstraction Layer) und Schnittstellen zu den gängigen Simulationsumgebungen und Hardwareszenarios bietet. Dies ist die Grundidee von "MARIE"(Mobile and Autonomous Robotics Integration Environment):

...a programming environment allowing multiple applications, programs and tools, to operate on one or multiple machines/OS and work together on a mobile robot implementation...

7.1 connecting things

Um die einzelnen Applikationen zu verbinden (dies geschieht hier über TCP/IP und shared libraries). Muss zunächst die Abstraktionsschicht, der Player-Server für die entsprechende Umgebung konfiguriert werden. Dies geschieht über das Player-.cfg-File. Hier werden die Treiber für die echte Hardware...

```

driver
(
  name "p2os"
  provides ["odometry::position2d:0"]
  port "/dev/ttyS0"
)
driver
(
  name "sicklms200"
  provides ["laser:0"]
  port "/dev/ttyS1"

```

...oder die simulierte Hardware geladen:

```
# load the Stage plugin simulation driver
driver
(
  name "stage"
  provides ["simulation:0" ]
  plugin "libstageplugin"

  # load the named file into the simulator
  worldfile "korridor.world"
)

# Create a Stage driver and attach position2d and laser interfaces
# to the model "robot1"
driver
(
  name "stage"
  provides ["position2d:0" "laser:0" ]
  model "robot1"
)
```

Um eine einfache Simulation zu starten, kann man z.B. stage aufrufen mit dem file korridor_old.world (old, da es sich um eine Version für das alte stage 1.3.5 handelt) Stage startet automatisch den player. Nun ist es möglich sich über den IP-Port 6665 des player-servers mit dem stage-robot zu verbinden. Clients sind z.B. playerv (einfacher Viewer mit Steuerungsfunktionen) oder playerjoy (Tastatur-Steuerung des Robots) oder FlowDesigner mit Robotflow:

The image shows two windows side-by-side. The left window is titled 'FlowDesigner' and displays a block diagram. A 'PlayerConnect' block is connected to a 'CLIENT' block. The 'CLIENT' block has three output ports: 'XPOS', 'YPOS', and 'THETA'. These are connected to 'Constant' blocks. The 'CLIENT' block also has three input ports: 'VELOCITY', 'YPOS', and 'THETA'. These are connected to 'Constant' blocks. The 'CLIENT' block is labeled 'PlayerVelocity'. The right window is titled 'Terminal' and shows the output of the simulation. The output includes the following text:

```
ket 10 **
** Player [port 6665] client accepted from
127.0.0.1 on socket 10 **
warning : not allowing subscription to unkn
own device "6665:gripper:0"
warning : not allowing subscription to unkn
own device "6665:ptz:0"
warning : not allowing subscription to unkn
own device "6665:sonar:0"
warning : not allowing subscription to unkn
own device "6665:blobfinder:0"
warning : not allowing subscription to unkn
own device "6665:laser:1"
Warning: positiondevice motor power request
not implemented
Warning: positiondevice velocity mode reques
t not implemented
** Player [port 6665] killing client on soc
ket 10 **
```

Robotflow verbindet sich via PlayerConnect-Box mit dem Server und man kann nun die entsprechenden Nodes aus der Robotflow-Bibliothek einsetzen...

8 Ausblick

Eine Linux-Installation auf einer Wechselplatte mit den genannten Tools ist im HAW Robot-Lab verfügbar. Eine Webseite und diese Einführung stehen zur Verfügung. Wer sich mit Robot-Simulation beschäftigen möchte sollte sich Robotflow/Player/Stage/Gazebo anschauen.

...it's an open source project. Your contribution is welcomed!